# An Asynchronous Level-1 Tracking Trigger for Future LHC Detector Upgrades

A. Madorsky, D. Acosta
University of Florida/Physics, POB 118440,
Gainesville, FL, USA, 32611

# Current Level-1 trigger systems

- ➔All boards work from system clock
- ➔Data are pushed through the entire trigger at system clock frequency, as in a pipeline
- ➔"Level-1 accept" timing is fixed
- ➔Front-end boards "count time back" from L1A, and find DAQ data to report

## Problems:

- ➔System-wide machine clock distribution
- ➔Complicated synchronization procedures
- ➔Higher clock frequencies in future systems
- ➔Multi-clock windows for data arrival
- ➔Multi-Gigabit serial link synchronization to system clock
- ➔Data link bandwidth is not fully used => extra cables

# Asynchronous design

- ➔Machine clock distributed only to front-end boards, for time marker assignment
- ➔All data from front-end boards sent asynchronously to trigger
- ➔Each track stub has time marker assigned by front-end
- ➔Trigger boards operate each on its own clock, for optimal performance
- ➔Trigger boards analyze time markers in input data to match track stubs
- ➔"Level-1 Accept" decision with time marker sent back to front-end board
- ➔Front-end boards report DAQ data corresponding to L1A time marker
- ➔L1A decision must be sent no later (but possibly earlier) than maximum latency.

**Current synchronous systems already have elements of asynchronous design – see next slides**

# Clock distribution

## Synchronous system:

- Machine clock distributed to all boards
- Requires complicated clock distribution system

➔ Asynchronous elements:

- Different clock and data delays require data re-alignment to clock on each board
- Data from different front-end boards still have to be aligned to each other

## Asynchronous system:

- Machine clock delivered only to front-end boards for data time marker assignment
- No machine clock on trigger boards
- Trigger algorithm is based on time markers, not on physical data arrival time.

# Synchronization procedures

## Synchronous system:

- Data links between any two boards have to be synchronized
  - Typically requires sending test signals through the entire trigger system
  - May be problematic at run time
  - Combined software-hardware procedure
  - In some cases, human intervention required
- Asynchronous elements:
  - Data links typically carry Bunch-Crossing Zero, or a few bits of Bunch Crossing Number for error detection – a rudimentary substitute for time marker.

## Asynchronous system:

- Only front-end boards have to be synchronized with each other, to assign accurate time markers
  - Can be as simple as one Reset signal
- Differences in cable lengths are not important, since physical data arrival time does not matter
  - Of course, there is a limit on cable lengths – latency still must be met!

# Multi-clock windows for data arrival

## Synchronous systems:

- Data for one event scattered in several bunch-crossings because of
  - Chamber drift time
  - Limited data link bandwidth
- Asynchronous elements:
  - Some trigger systems analyze data in several bunch-crossings to build a track

## Asynchronous system:

- Analysis based on time markers
- Makes it easy to analyze multiple bunch-crossings

# Multi-gigabit links

## Synchronous system:

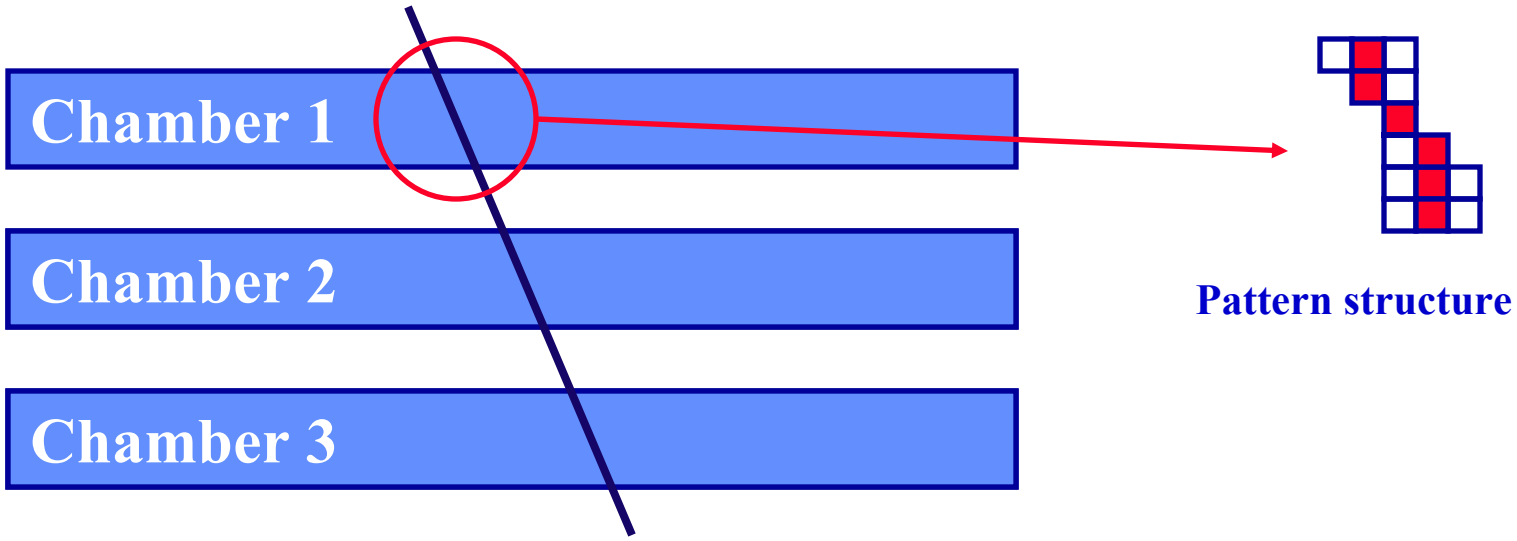
- Links run on multiple of machine clock frequency
- Links require very low jitter in input clock (~40 ps), and it may become even less for future faster links.
- Conventional clock multipliers give too much jitter (100 ps and more)
- PLL based on voltage-controlled crystal oscillator must be used
- Crystal oscillator for PLL must be custom-made
- Links we currently use are specified for 1.062 or 2.125 Gb/sec (Fiber Channel). We are lucky they work at 1.6 Gb/sec in our system (80 MHz input clock).
  - Future links may not be designed to work at arbitrary frequency
- Maximum bandwidth cannot be reached in most cases
  - Leads to extra cables

## Asynchronous system:

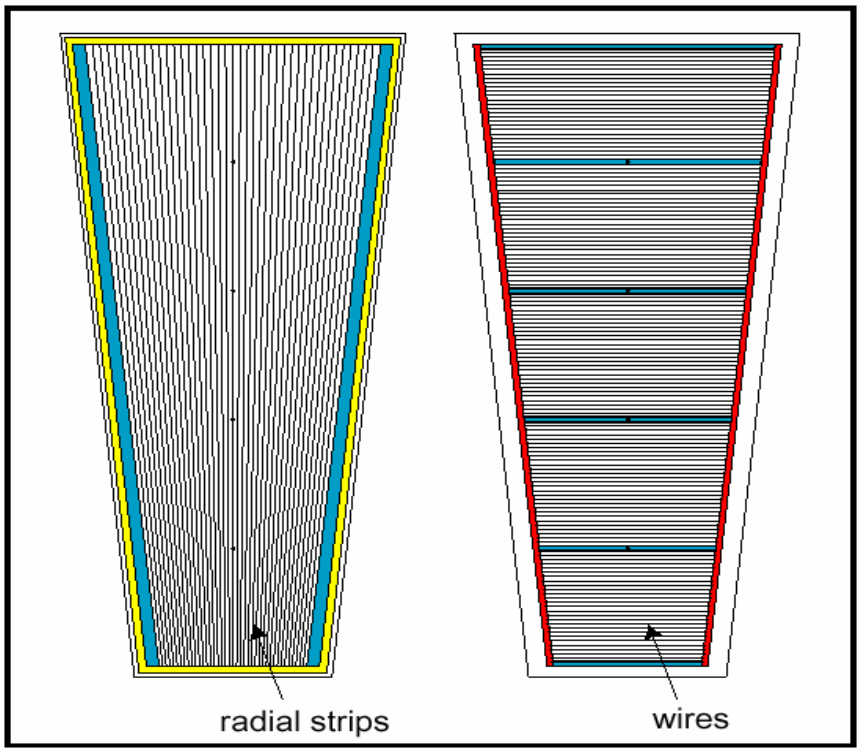
- Links run from their own oscillators
- Optimal frequency for maximum bandwidth
- Industry-standard low-jitter oscillators are inexpensive and easily available
- One can reasonably hope that oscillators for future faster links will be easily available too, for industry-standard frequencies.

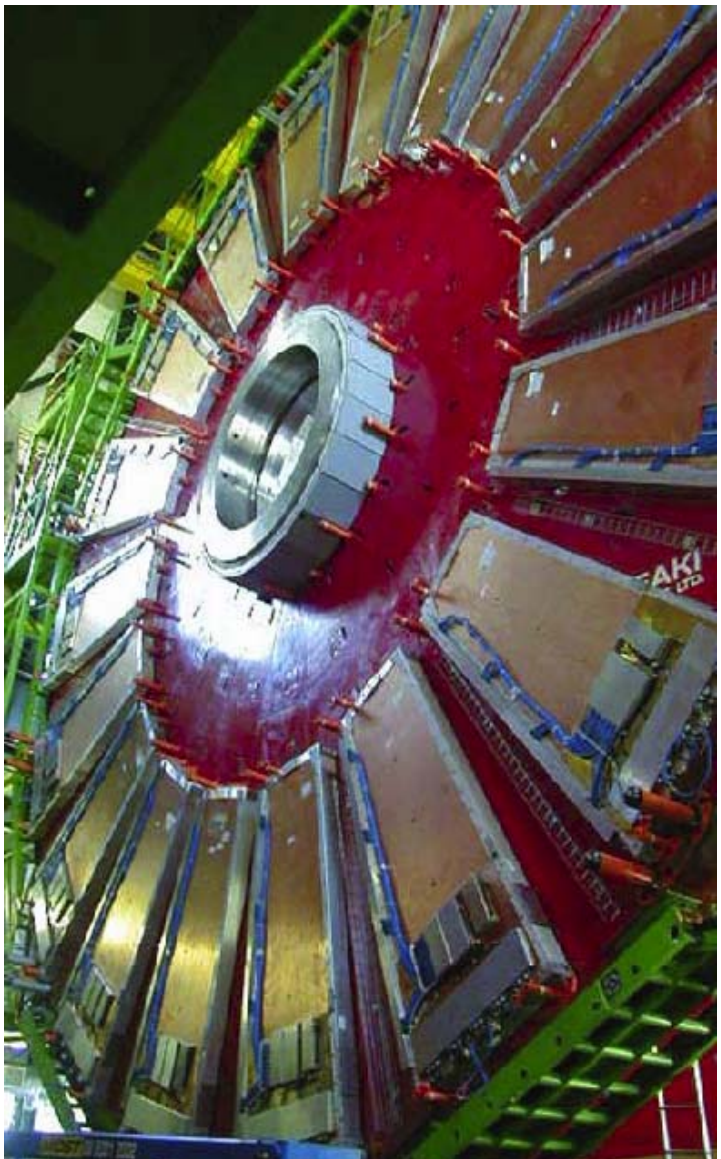
# Asynchronous trigger prototype

- Will be based on existing anode electronics on CMS Endcap Cathode Strip Chambers
- UF cosmic test stand setup will be used
- Track stub information is sent to Trigger board from each chamber's front-end board (ALCT)
  - Wiregroup number
  - Angle (pattern)
  - Time marker
- Trigger board sends back Level-1 decision with time marker
- Simulated machine clock of 80 MHz (targeting SLHC)



# CMS Endcap Cathode Strip chambers

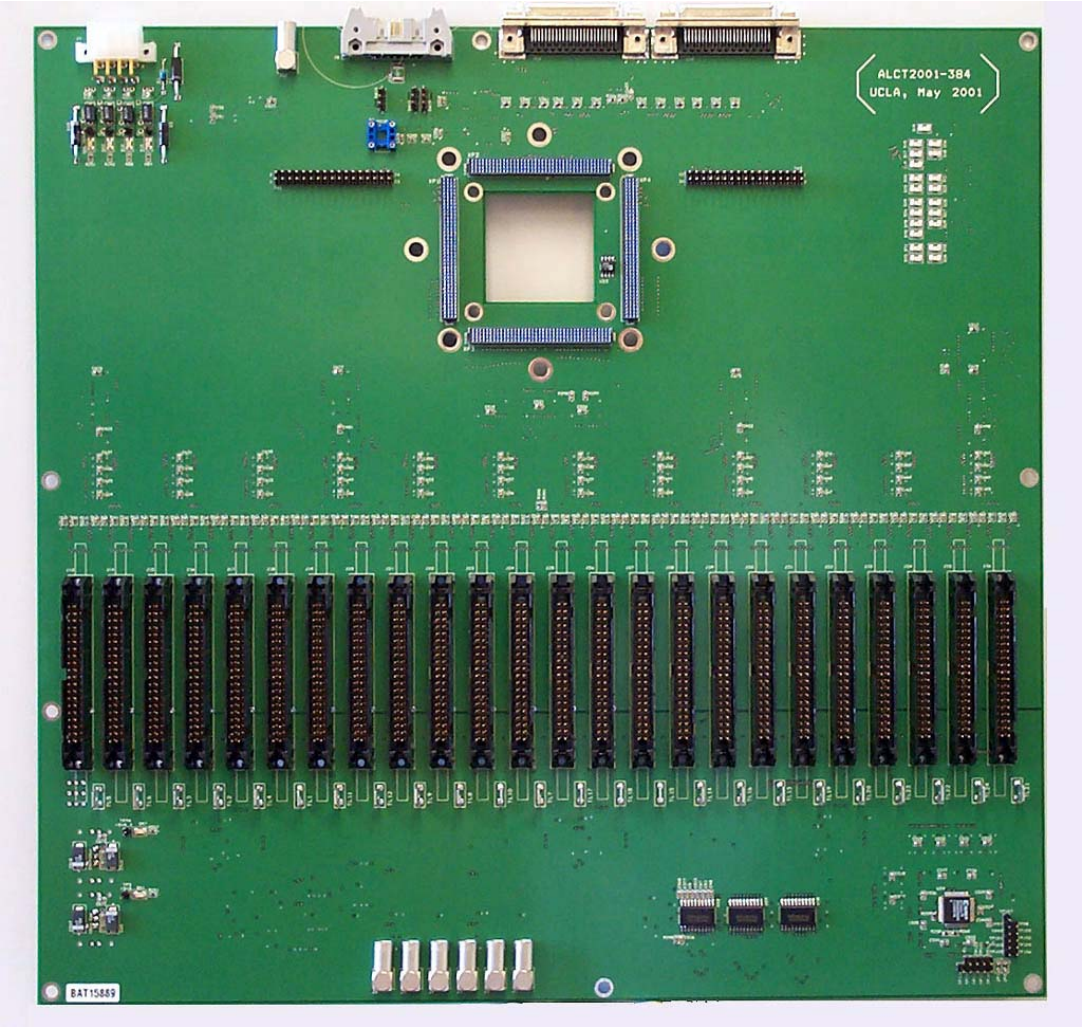


- **6-layer, 2-coordinate multi-wire proportional cathode strip chamber**
- **Anode wires in the azimuthal direction, cathode strips in the radial direction**

# UF cosmic rays test stand

# Anode Local Charged Track board (ALCT)

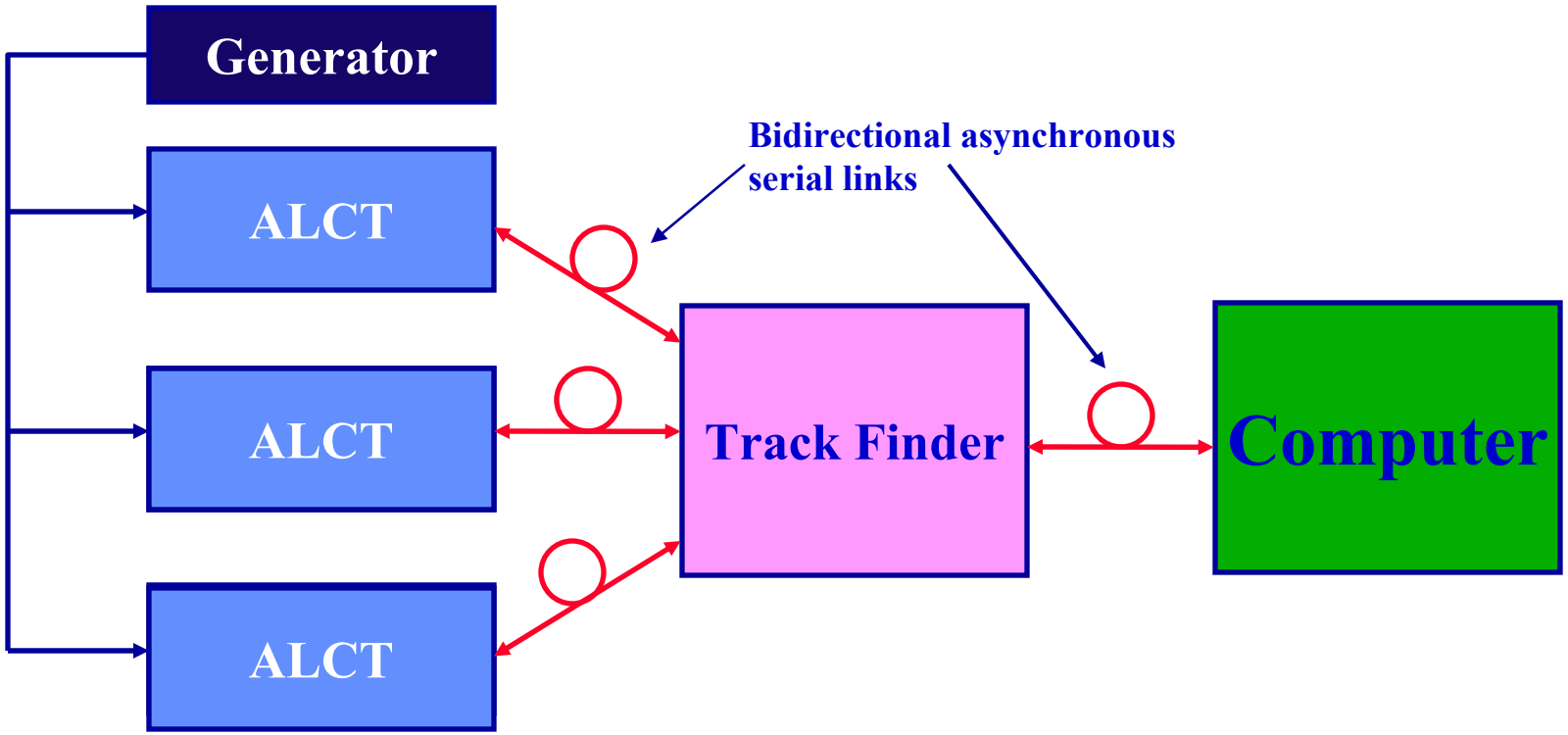




- **Board designed in UCLA**
- **Track-finding and DAQ firmware written by us**
- **New mezzanine board will be designed for asynchronous system prototype**

# Prototype structure

- Anode electronics upgrade – to simulate front-end boards
- New Asynchronous Track-Finder board
- Generator simulates clock and control system for front-end
  - Just 80 MHz clock + Reset signal
  - All data links and Track Finder run on their own async clocks



# Data exchange

**All data exchange between each front-end and trigger board via one link**

- ➔To maximize bandwidth use and avoid extra cables
- ➔Priority system must be established

**From front-end to trigger board, in order of priority:**

- ➔Track stub data for trigger decisions, with time markers.
- ➔Critical status information (buffer overflow, etc)
- ➔DAQ information (raw hit data)
- ➔Slow control data

**From trigger board to front-end, in order of priority:**

- ➔Level-1 accept, with time marker
- ➔Slow control commands and data

**Time-critical data don't wait for anything!**

# System level model

- Is being written in C++
- Based on VPP – Verilog HDL simulation and generation library
  - Library written in our group
  - Was used for two large projects:
    - CMS Endcap CSC ALCT firmware
    - CMS Endcap CSC Sector Processor firmware
  - Simulation completely matches hardware
  - http://www.phys.ufl.edu/~madorsky/vpp/
- We used GNU and Microsoft C++ compilers, should also work with most other C++ compilers
- Exactly simulates behavior of logic devices
- Can be incorporated into CERN trigger system modeling environment
- Generates valid Verilog HDL code for programmable logic
- Initial version of data analysis code for Track Finder board is ready, under tests now.
  - Analyzing data based on time markers is easier than expected
  - The entire system model to be finished by February '05 (tentative)

# Anode Mezzanine board

- Replaces the original mezzanine board on ALCT – Anode Local Charged Track (front-end) board
- Input raw hit data from the cathode chamber
- Finds best track segments
- Assign exact time marker for each track segment found
- Report track segments asynchronously via the serial link to the track processing board
- Store raw hit information in the circular memory buffer
- Retrieve the raw hit information upon a Level-1 decision and send it to the track processing board via the same serial link.
- To be finished by June ‘05

# Track Finder board

- Track must contain a certain number of track segments found by ALCTs
- Check that these segments have time markers matching within certain limits
- Log the complete track. A computer can later read out the information about this track.
- Generate a "Level-1 accept" decision and send it to the ALCT mezzanine boards, with a time marker
- Receive raw hit data for this track from the ALCTs, and send to DAQ computer
- Discard track segments that did not result into the complete track.
- To be finished by September '05