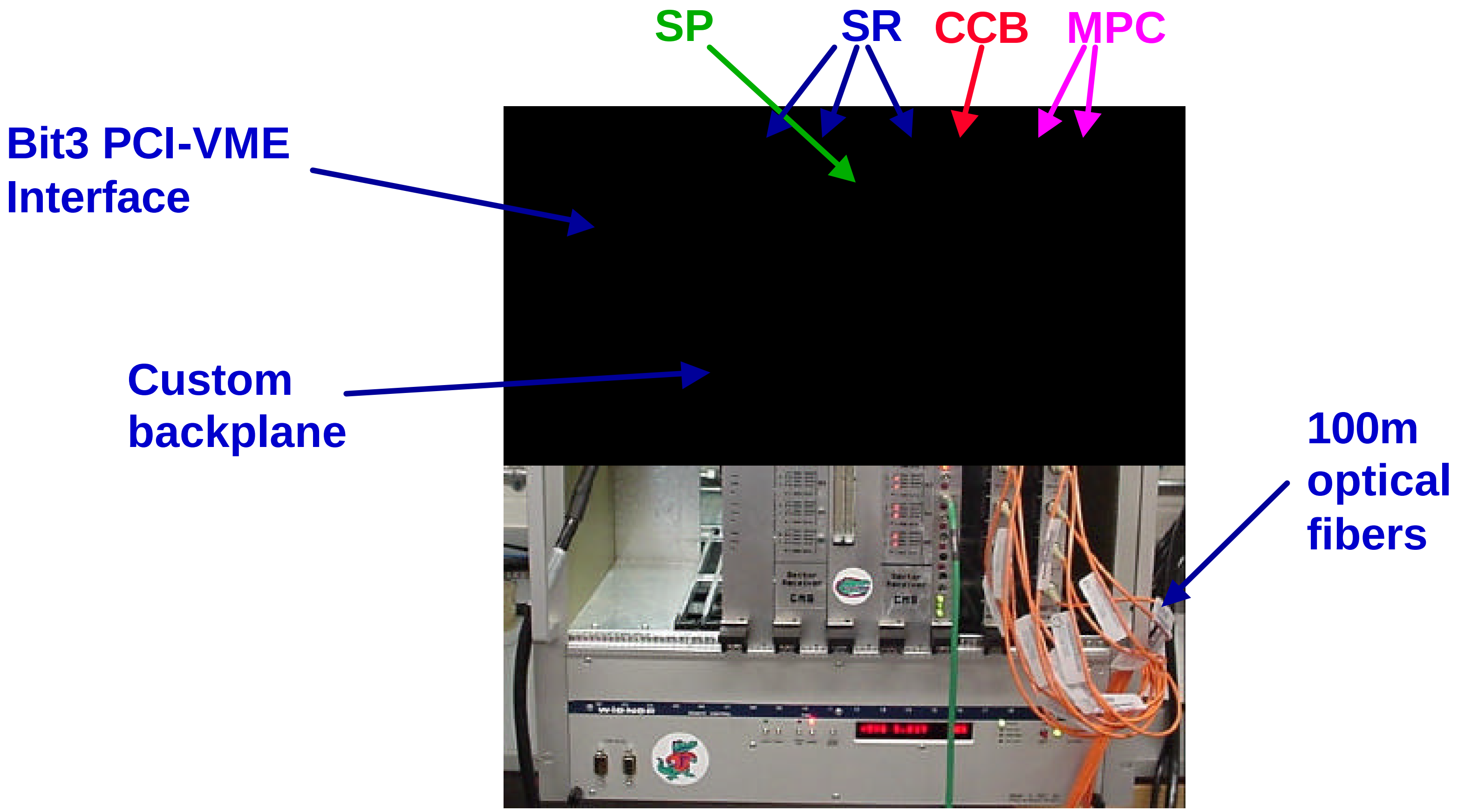
CSC Trigger Software Experience and Plans

D.Acosta

University of Florida

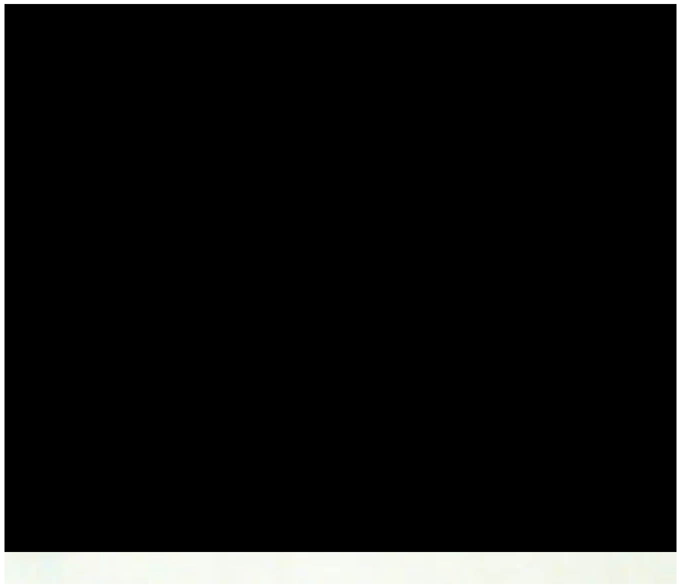
Track-Finder Crate Tests

- ➔ Last year, we successfully tested a complete chain of prototypes, yielding perfect agreement with the simulation for millions of events
- ➔ Documented in TDR, and reported at LEB’01 conference



Sector Receiver
UCLA

Port Card
Rice



Sector Processor
Florida, PNPI

Test Setup

VME Controller

- ➔ SBS Bit3 PCI-to-VME interface

Software environment

- ➔ PC running MS Windows
- ➔ Visual C++ compiler
- ➔ JAVA development kit

- ☐ Everything written in C or C++ for portability
- ☐ JAVA used for GUI rather than MS Windows

Board Configuration

Command line utilities were written to load the following

FPGAs

- ➔ Xilinx ".bit" files for each FPGA are converted to industry standard SVF format by the Xilinx JTAG programmer
- ➔ All files concatenated into one JTAG chain (17 FPGAs for SP)
- ➔ National Semiconductor software "SCANPSC", along with Bit3 VME driver, used to send JTAG data in parallel through VME to glue logic on the board, which then serializes the JTAG data at 25 MHz to configure the FPGAs (including VME interface)

SRAM

- ➔ SRAM is configured by writing directly to VME registers on the board (several MBs of data)
- ➔ Can be read back for verification

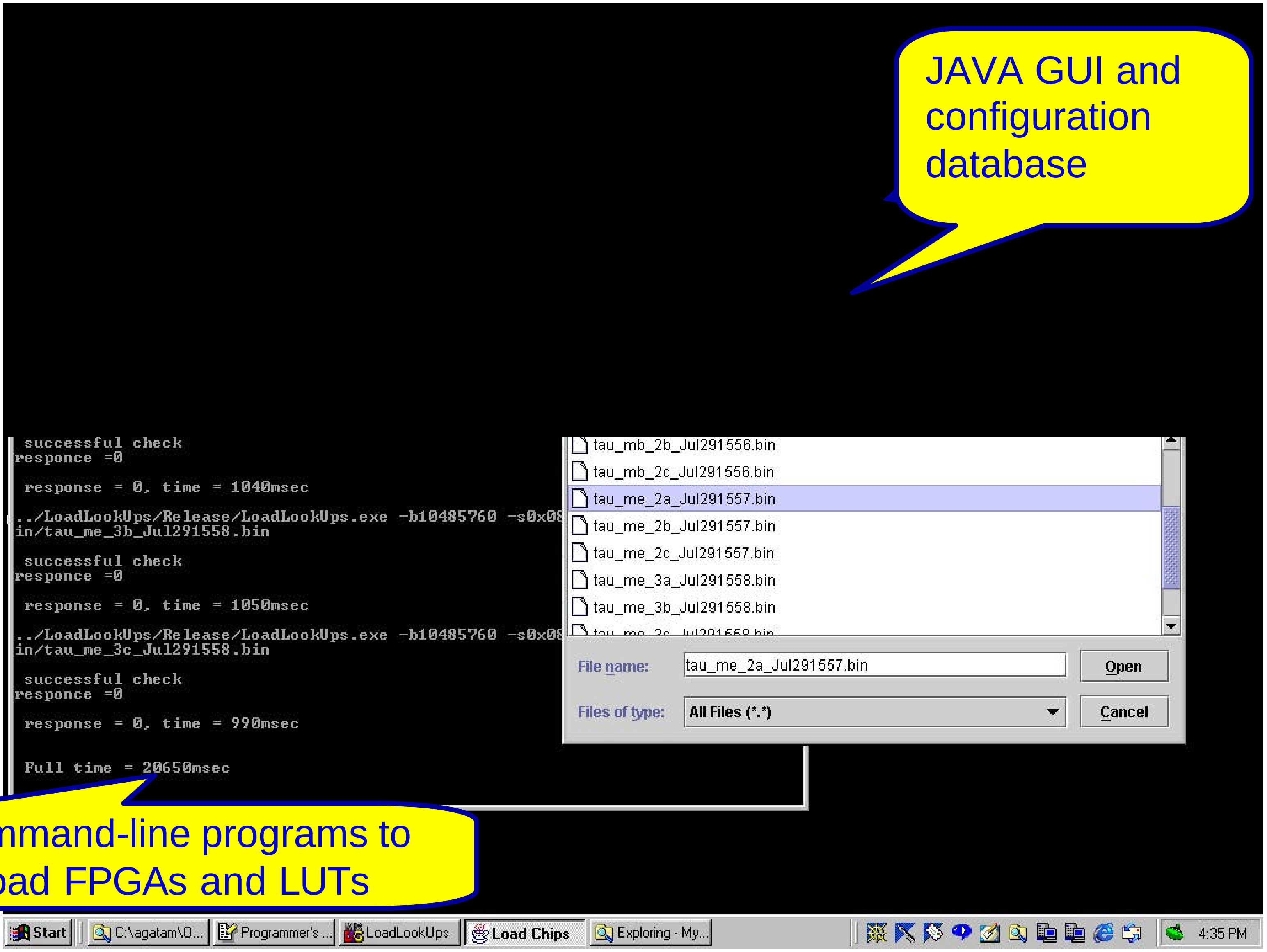
- ❑ Total configuration of SP took about 30 seconds (vs. 6 min using Xilinx serial JTAG interface through LPT port)

Configuration Database

A Board Configuration Database with GUI was written in JAVA

- ➔ Keeps track of the many configuration variants and provides a one-click selection of any one of them
- ➔ Each variant contains the complete information for FPGA and lookup memory configuration.
- ➔ Can be used for multiple boards.
- ➔ Also used to select the test configuration

Configuration Screenshot



```
 successful check
responce =0

 response = 0, time = 1040msec

../LoadLookUps/Release/LoadLookUps.exe -b10485760 -s0x08
in/tau_me_3b_Jul291558.bin

 successful check
responce =0

 response = 0, time = 1050msec

../LoadLookUps/Release/LoadLookUps.exe -b10485760 -s0x08
in/tau_me_3c_Jul291558.bin

 successful check
responce =0

 response = 0, time = 990msec

 Full time = 20650msec
```

tau_mb_2b_Jul291556.bin
tau_mb_2c_Jul291556.bin
tau_me_2a_Jul291557.bin
tau_me_2b_Jul291557.bin
tau_me_2c_Jul291557.bin
tau_me_3a_Jul291558.bin
tau_me_3b_Jul291558.bin

File name: tau_me_2a_Jul291557.bin
Files of type: All Files (*.*)

JAVA GUI and configuration database

Command-line programs to load FPGAs and LUTs

Test software

- C++ description of the boards extracted from ORCA to run standalone in Windows
  - ➔ Switches put in code so same class works in both environments
- C++ classes written for each board to control hardware
  - ➔ Inherit from common base class for VME access (okay, not all of us used it…)
- C++ module written to control tests
  - ➔ Reads ORCA data from an ASCII file
  - ➔ Calls appropriate methods depending on the test selected
    - ◻ Static test of a single board, dynamic test, or chain test
    - ◻ Loads input FIFOs, initiates test (BC0), reads output FIFO
  - ➔ Compares the output of the C++ model with the board output

Trigger Control

In addition to testing each board individually, we performed the following chain test:

- Port Card ® Sector Receiver ® Sector Processor
- Only data for a fixed number of BX (usually 256) was loaded into the input FIFO of a board through VME

The Clock and Control Board coordinated these tests

- Distributed clock and control signals with programmable delays
- Sent BC0 to initiate tests

Synchronization

- A known pattern was sent across several BX
- We looked for this pattern in the output FIFOs
  - Pipeline adjusted if a word is on the wrong BX
  - Clock edge adjusted if word is not reproducible

Lessons (1)

After much work, we were able to demonstrate exact agreement between the hardware and software

- ➔ ORCA data, pseudo-random data, known patterns

Part of the problem was that the C++ model of the hardware did not exactly match the hardware *a priori*

- ➔ Sorting algorithm, ghost cancellation, LUT contents, …
- ➔ We spent most of the time chasing these differences rather than debugging hardware

For this reason, we have changed the basis of the Track-Finder design from a mix of schematics and VHDL to a complete Verilog description

- ➔ Translated first C++ description into C-like Verilog
- ➔ Back-propagated to a new C++ description with line-by-line correspondence (which will go into ORCA eventually)
- ➔ So we essentially have just one description now for HW+SW
- ➔ By the way, this "second draft" of the Track-Finder design saved us 14 BX!

Lessons (2)

Developing trigger control software from the bottom up while trying to meet a deadline (TDR) is probably not the best way to proceed

- ➔ We lacked an overall guiding framework
- ➔ We also lacked a coordinator to ensure uniform style and functionality from all developers, solve disputes, and state requirements
- ➔ More "spaghetti code" was generated the closer the deadline approached

On the other hand, this chain test was extremely useful to understand the challenges we will face when trying to commission the final trigger system

- ➔ Many more boards, crates, and people to integrate

Plans

We are now designing a pre-production prototype of the Track-Finder

- ➔ Aside from the trigger logic improvements, it will now include a circular buffer and DAQ interface to a FED
- ➔ Additional VME registers implemented
  - ◻ Adjustable trigger parameters (h cuts, coincidence level…)
  - ◻ LUT and FPGA version numbers

We will build on the test software developed already

We would like to expand the chain test to include boards all the way down to the detector

- ➔ Lacked manpower to do this with previous prototypes
- ➔ Technically not possible with new prototypes until 2003

But our tests need to evolve from a fixed number of BX to a free-running system that responds to L1A

- ➔ i.e. we need a Local Trigger Control System as well as a monitoring system a

Conclusions

- Prototype tests were a success, but the software development was a lot of work
- It was a useful exercise to perform a chain test of the prototype trigger electronics
  - ➔ Gives us some guidance on how to commission the final system
- It would be nice to have a common CMS software environment for testing hardware
  - ➔ We have ORCA packages for the simulation, but where do we store our hardware classes, VME software, JTAG, etc. ?
  - ➔ For our prototype tests, we set up a CVS repository and *tried* to follow some basic coding standards