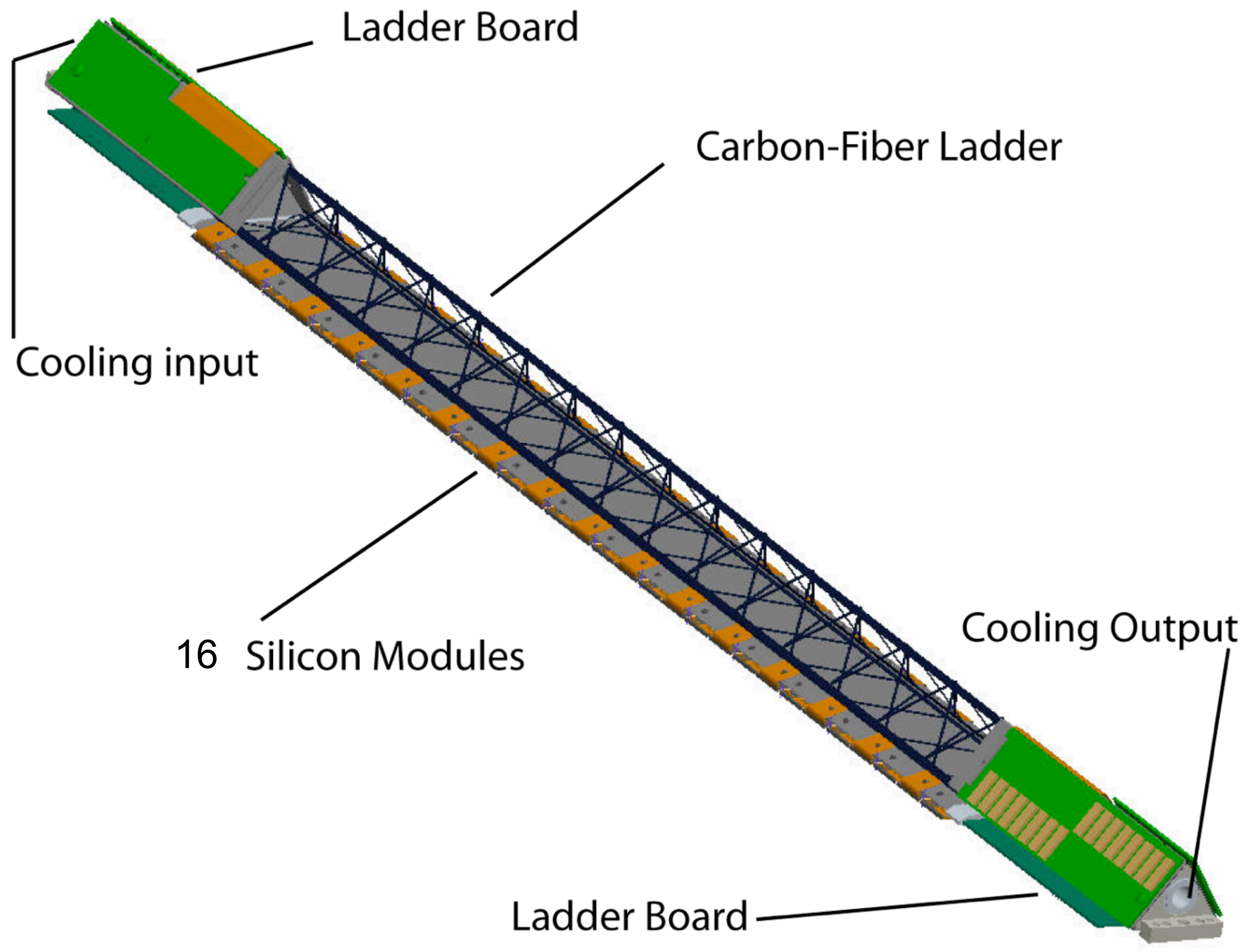
SSD electronics review

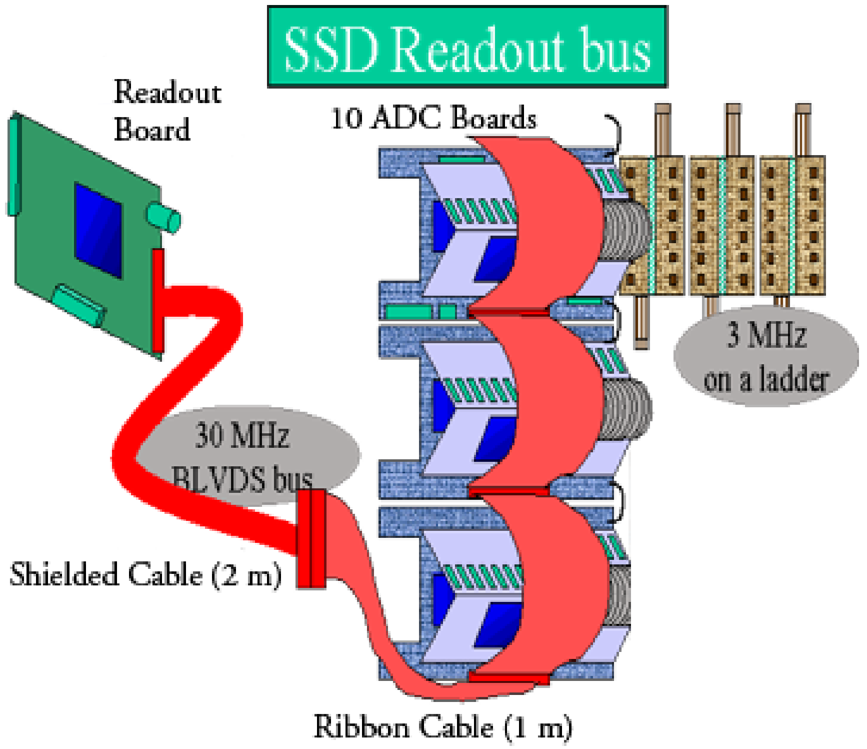
M. LeVine BNL

Quick overview of upgrade

SSD ladder


Ladder Board
Carbon-Fiber Ladder
Cooling input
16 Silicon Modules
Cooling Output
Ladder Board

Previous readout configuration



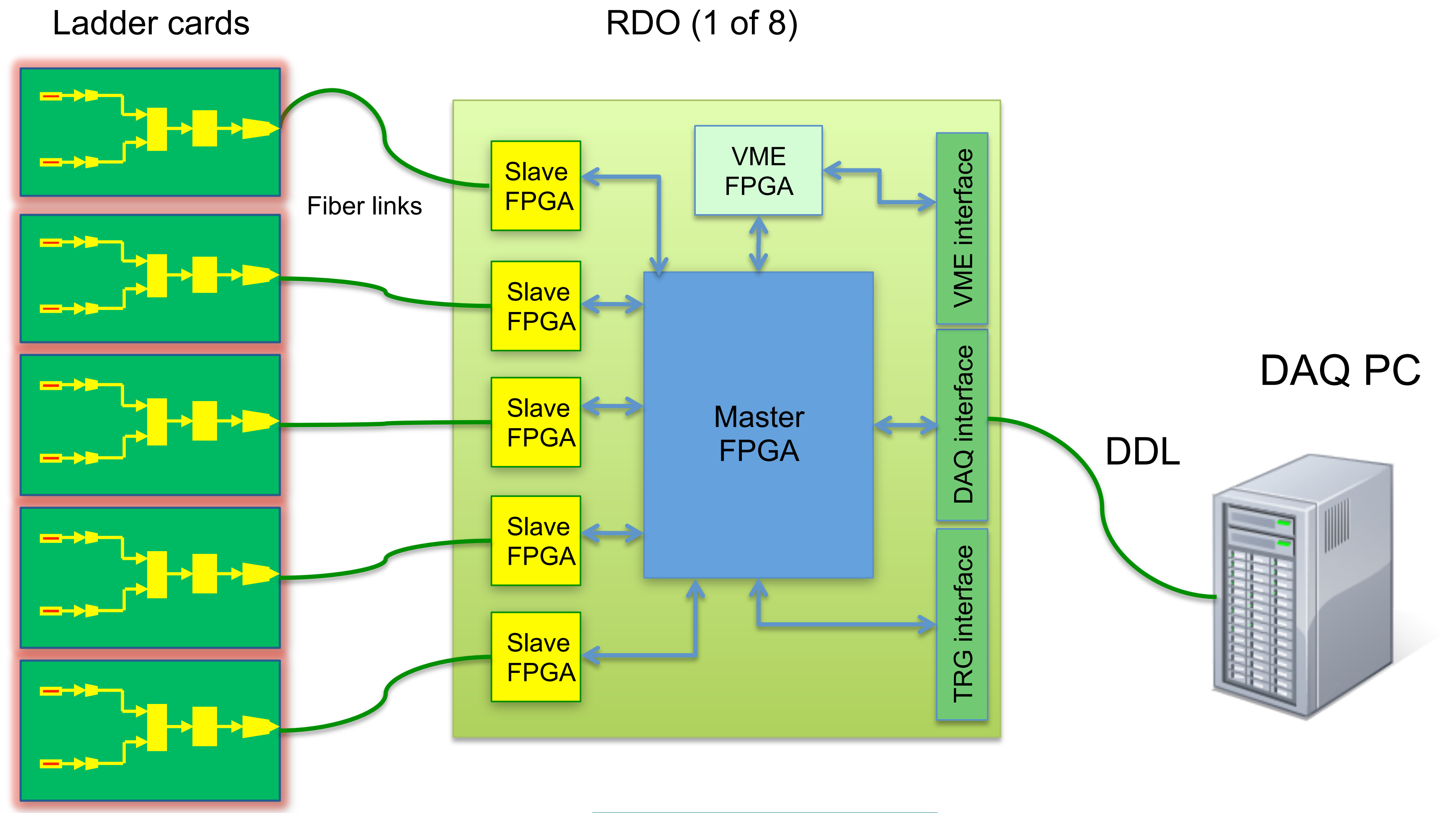
Readout upgrade concept

- Reading out front end:
  - Replace single ADC with 16 ADCs
    - digitize 16 modules in parallel
  - Increase sampling rate to 5.00 MHz
  - All ladders processed concurrently

$\left.\right\}$ 2.5 ms -> 163 μs

- Transferring data to PC
  - Increase link throughput to DAQ PC to 120 Mbyte/s per 5 ladders
    - 1850 μs -> 450 μs
  - Multiple (derandomizing) buffers effectively hides this time
- Dead time: 10%@750Hz, <2%@100Hz
- [*cf*. existing: >80%@750Hz,30% @ 100Hz]

Readout components



Outer support cone

South platform VME crate

DAQ room

Data formats

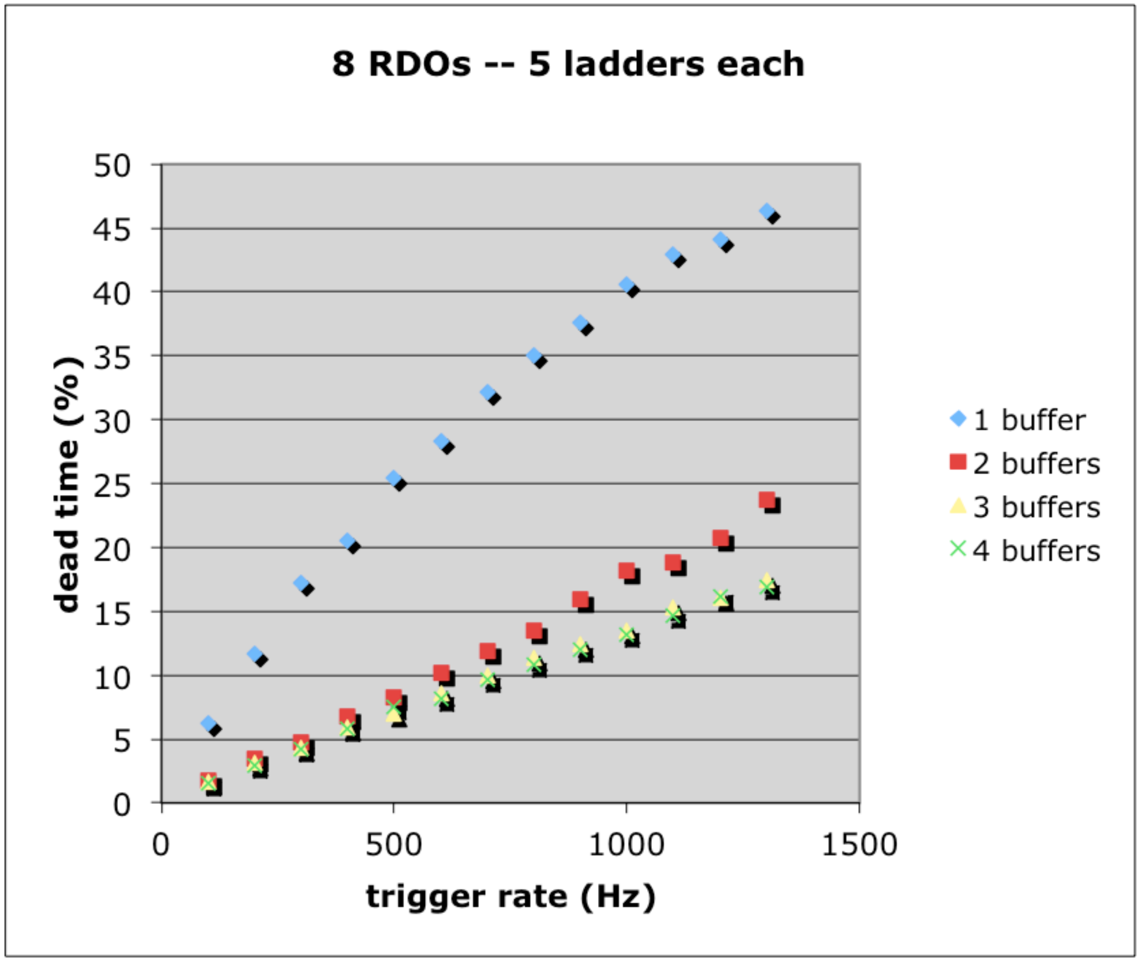
non zero-suppressed

- 3 10-bit ADC values to a 32-bit word
- Fixed order: position in buffer/word -> geographical position of strip

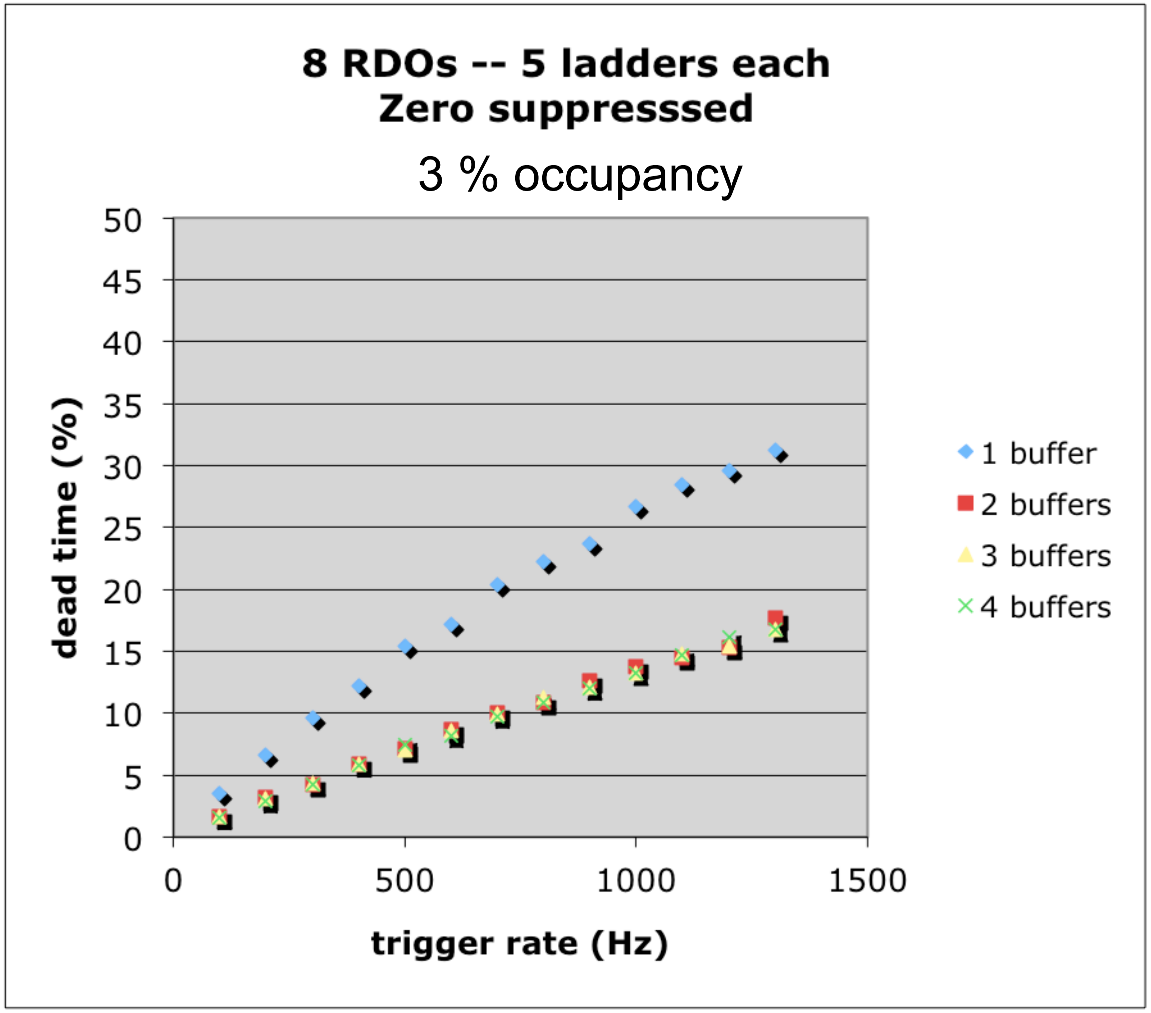
zero suppressed

- Only strips with ADC value above threshold are present
- ADC value (10 bits) + strip location (14 bits)
- One strip per 32-bit word
- Alleviates large memory access burden on DAQ PC
- Doing this in real time in FPGA is simple

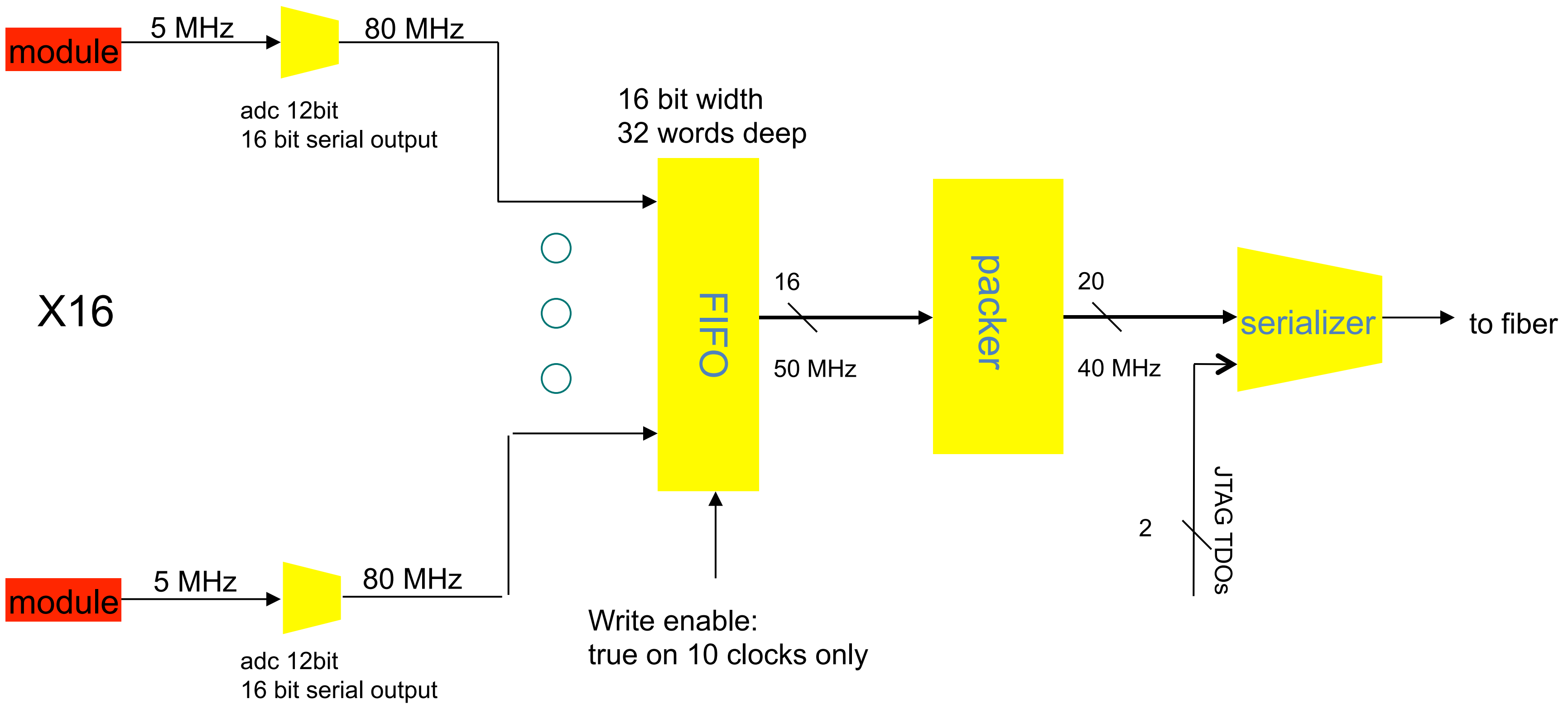
Dead time calculation - no zero suppression



Dead time calculations – zero suppression



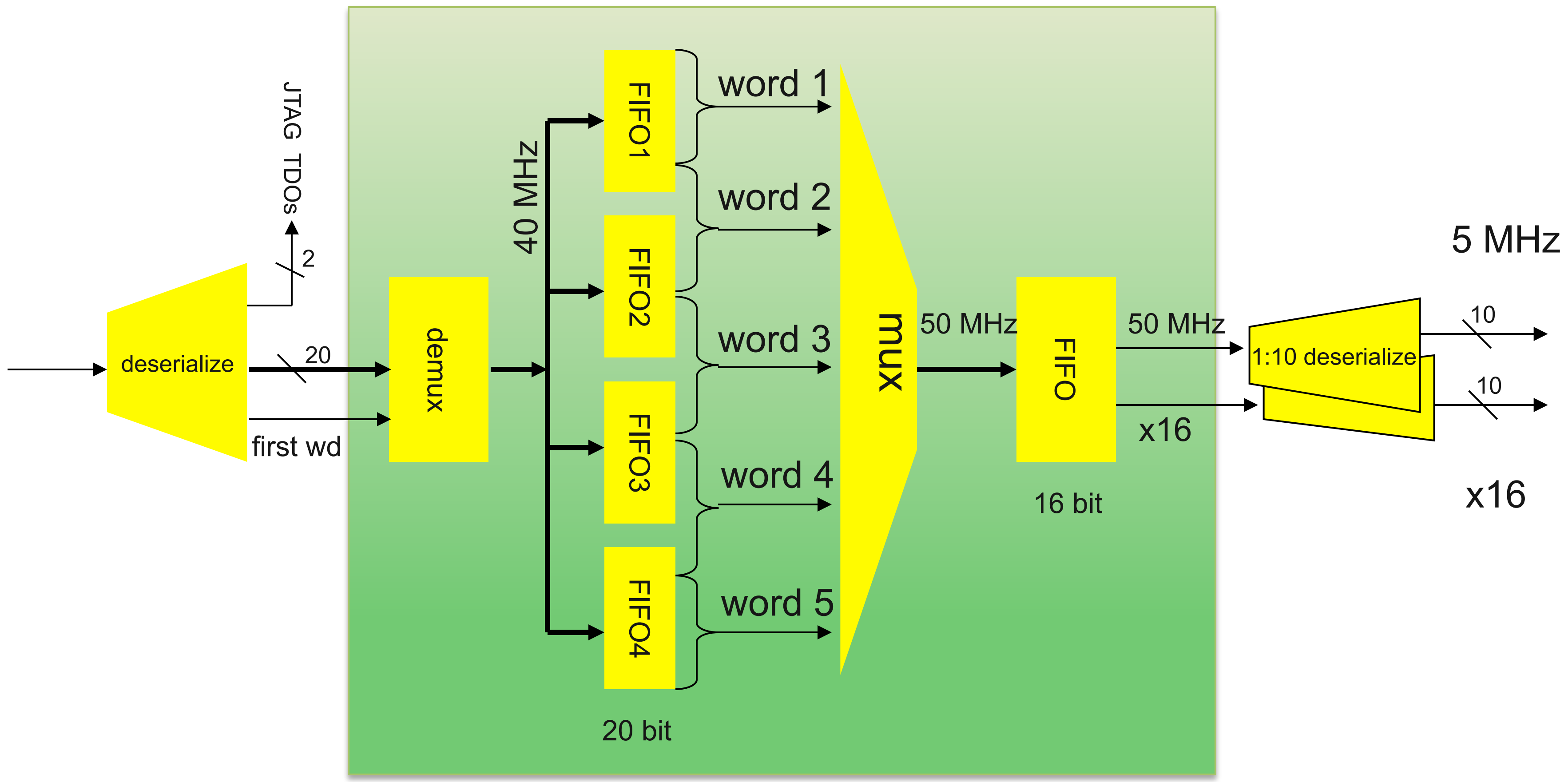
Ladder data path



1 set of adc samples: 10 *X* 16 bits => repacked into: 2 *X* 4 *X* 20 bits

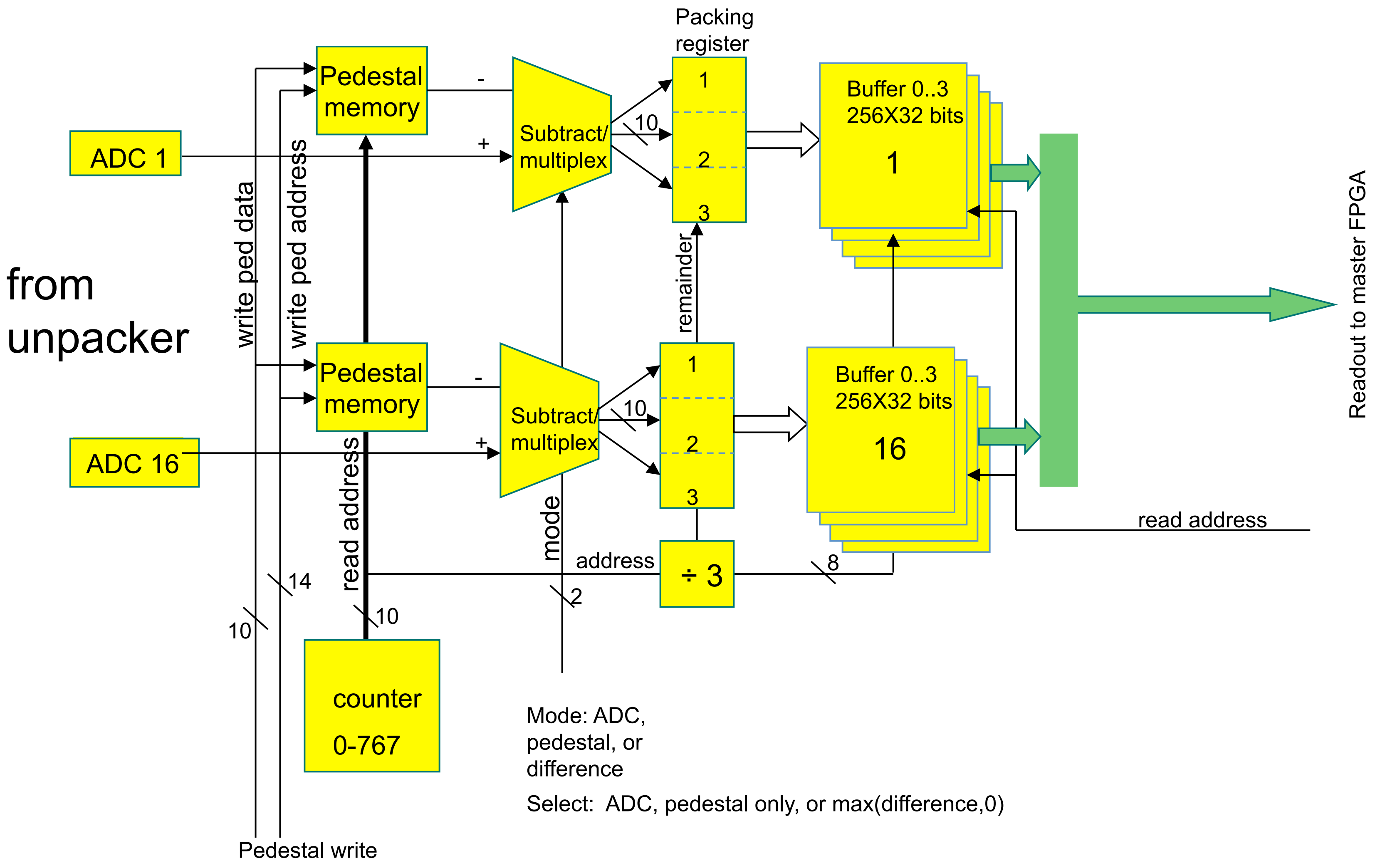
RDO slave - unpacker

unpacking 4 20-bit words to 5 16-bit words



Slave FPGA – ADC processing

(no zero suppression)



Mode: ADC, pedestal, or difference

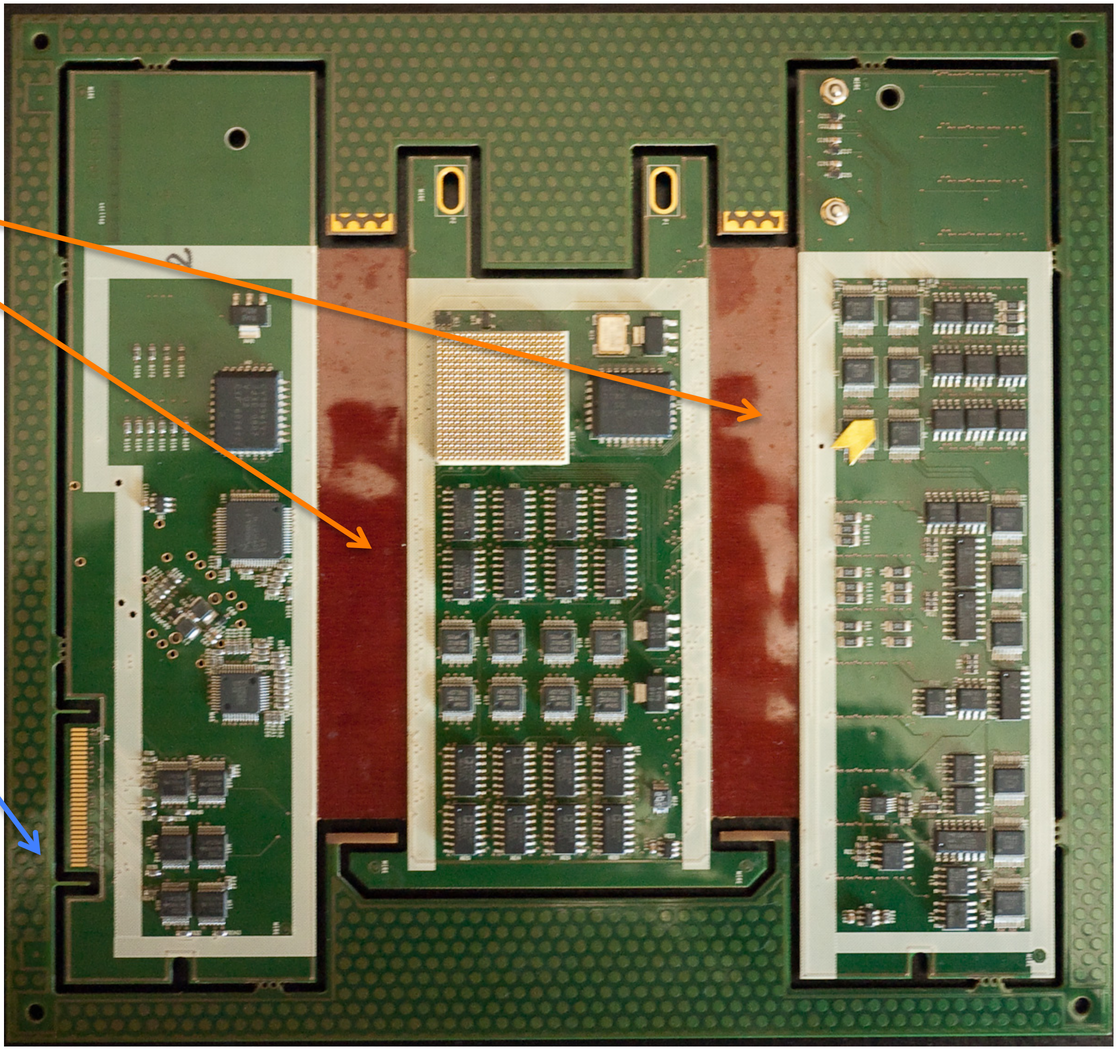
Select: ADC, pedestal only, or max(difference,0)

Prototype ladder card testing

Ladder board: (inside)

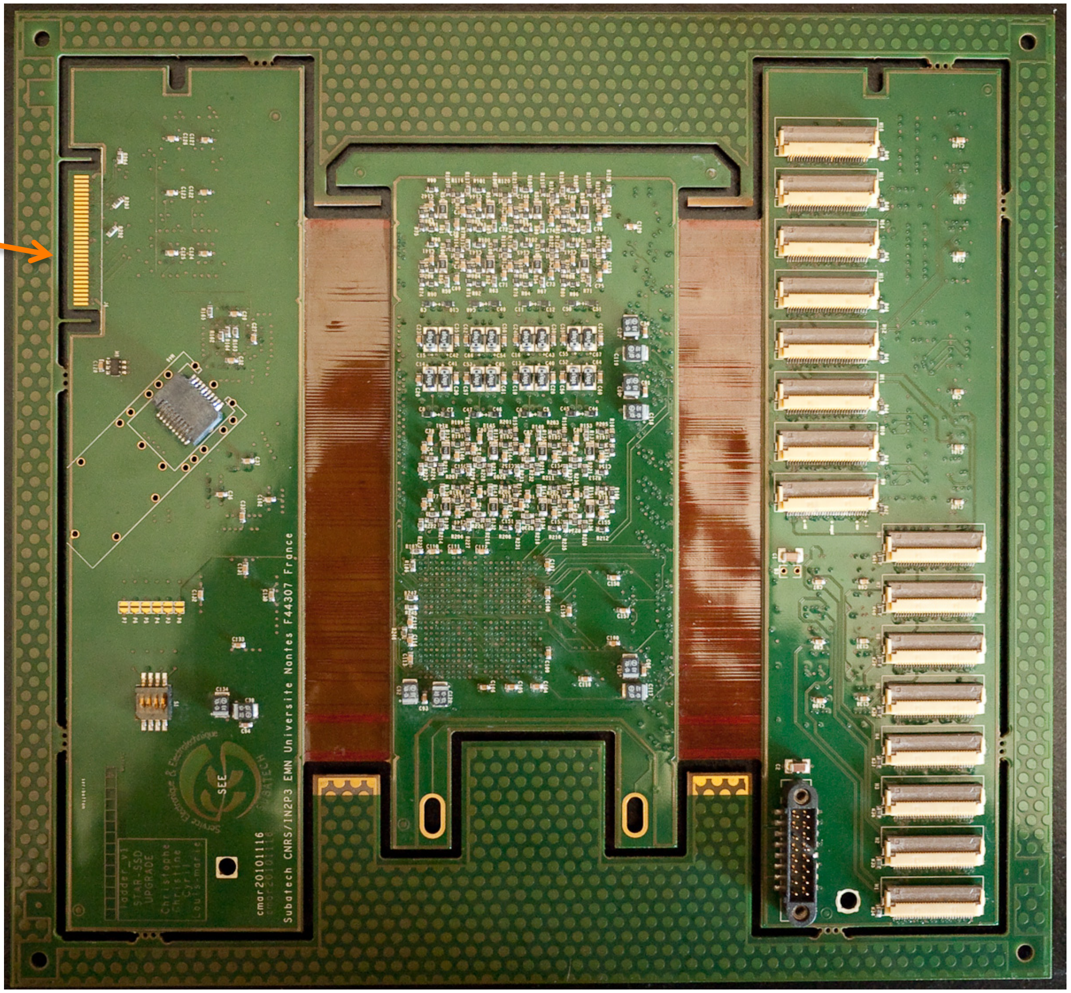
Flex circuit layer

Frame cut loose when board is ready for installation



Ladder board: (outside)

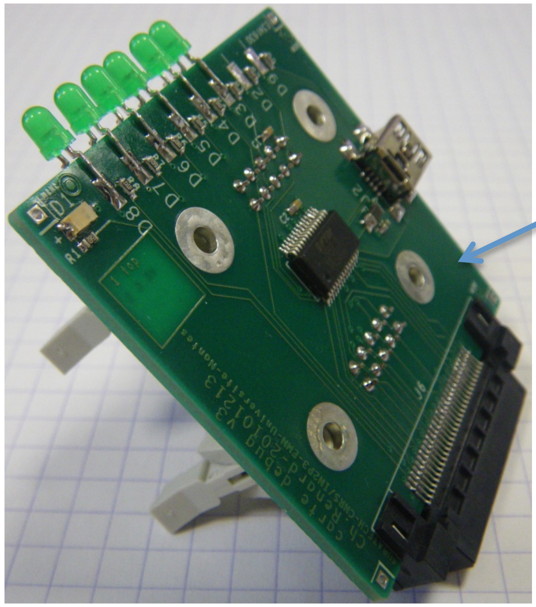
Edge connector for debug card



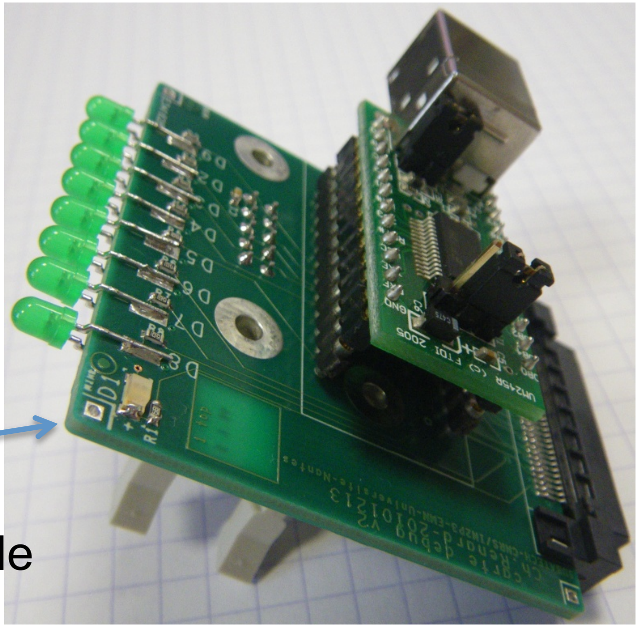
Debug card (via edge connector)

Provides:

- JTAG header to configure FPGA
- JTAG header for slow controls
    - Both will be provided via fiber
- USB access to FPGA



Version using
FTDI naked chip



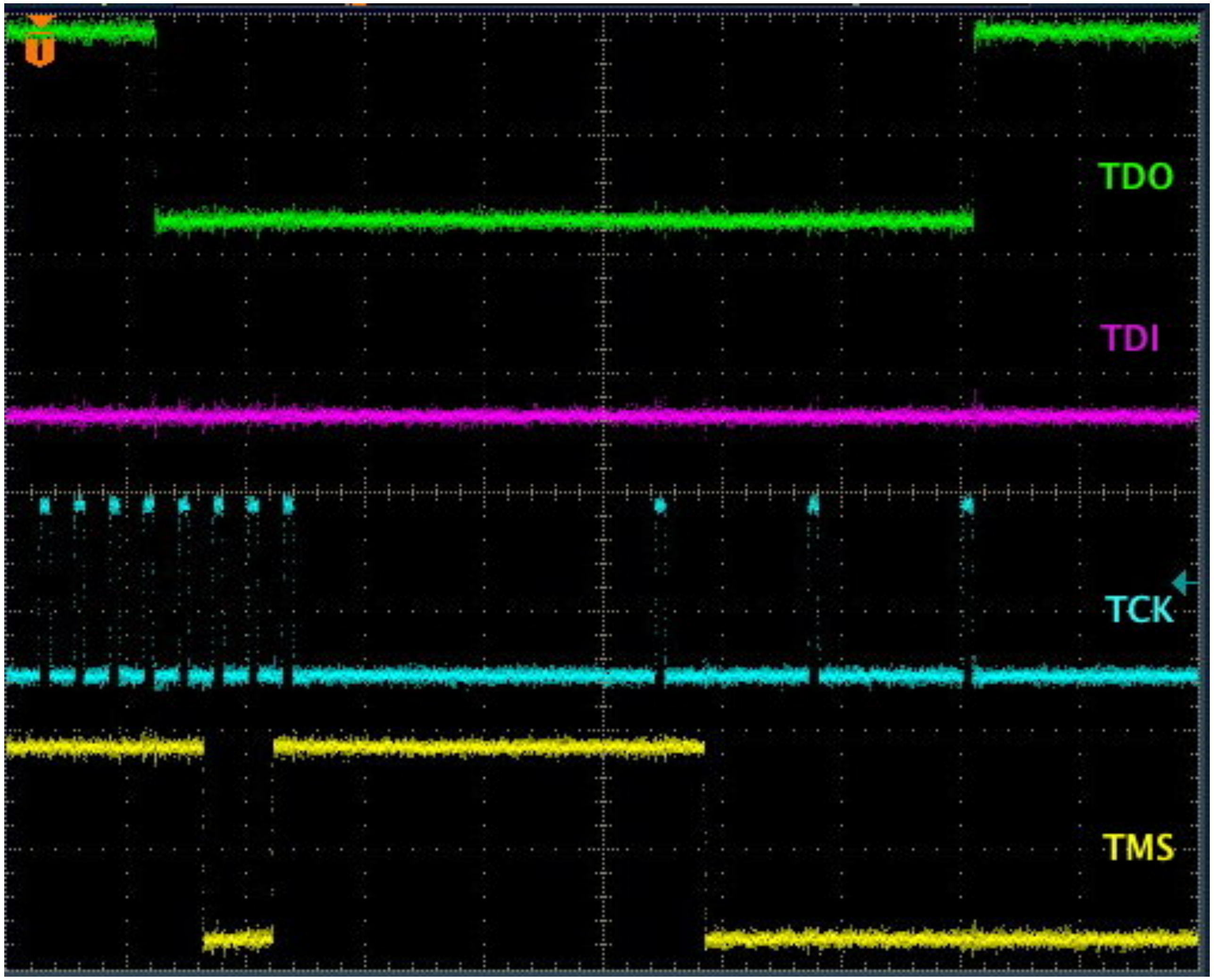
Version using
FTDI plugin module

USB test results

```
Number of devices is 2
==== Device 0 is Subatech DbgV3n1 ======
          Serial # A7U3EQBC

1) sending the following bytes
0x41 0x41 0xFF 0xFF 0x42 0x42 0x00 0x00 0x43 0x43
0xFF 0xFF 0x44 0x44 0x44 0x44

2) sending the following bytes
0x41 0x41 0xFF 0xFF 0x42 0x42 0x00 0x00 0x43 0x43
0xFF 0xFF 0x44 0x44 0x44 0x44

FT_Read = 32
pcBufRead[0] = 0x41
pcBufRead[1] = 0x41
pcBufRead[2] = 0xFF
pcBufRead[3] = 0xFF
pcBufRead[4] = 0x42
pcBufRead[5] = 0x42
pcBufRead[6] = 0x00
pcBufRead[7] = 0x00
pcBufRead[8] = 0x43
pcBufRead[9] = 0x43
pcBufRead[10] = 0xFF
pcBufRead[11] = 0xFF
pcBufRead[12] = 0x44
pcBufRead[13] = 0x44
pcBufRead[14] = 0x44
pcBufRead[15] = 0x44
```

```
pcBufRead[16] = 0x41
pcBufRead[17] = 0x41
pcBufRead[18] = 0xFF
pcBufRead[19] = 0xFF
pcBufRead[20] = 0x42
pcBufRead[21] = 0x42
pcBufRead[22] = 0x00
pcBufRead[23] = 0x00
pcBufRead[24] = 0x43
pcBufRead[25] = 0x43
pcBufRead[26] = 0xFF
pcBufRead[27] = 0xFF
pcBufRead[28] = 0x44
pcBufRead[29] = 0x44
pcBufRead[30] = 0x44
pcBufRead[31] = 0x44


Closed device A7U3EQBC
```

Slow controls JTAG signals



Slow controls – read registers

✔ = Register with known content at startup

```
reading register 0x01  ROBOCLKS: 0xaa 0xaa 0xaa   ✔
reading register 0x02  STATUS  : 0x3e 0x60 0x3f
reading register 0x03  CONFIG  : 0x00 0x00
reading register 0x04  DAC VALS: 0x0a 0xa9 0x55  ✔
reading register 0x07  HYBRIDS : 0x00 0x00
reading register 0x08  LATCHUP : 0x00
reading register 0x09  RALLUMAG: 0x00 0x00
reading register 0x0b  BYPASS  : 0x00 0x00
reading register 0x0c  VERSION : 0x26 0x01 0x20 0x11  ✔
reading register 0x0e  TEMPS   : 0x00 0x00 0x00 0x00 0x00
0x00
reading register 0x1b  IDENTITE: 0xb7    ✔
```

Ladder response *vs* clock frequency

4.3 MHz

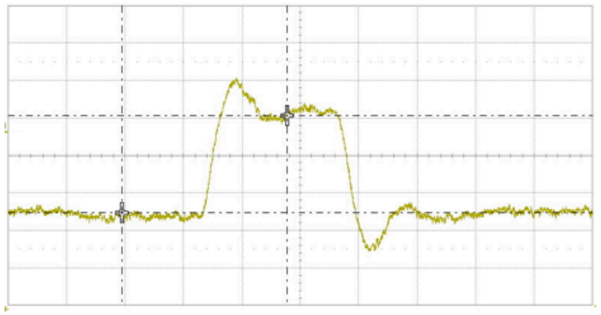


Fig 4: 4.29 MHz clock. Pedestal mean= - 441mV, S.D.= 18.0mV. Peak (cursor 2) mean= 88.3mV, S.D. = 10.4mV.

5.0 MHz

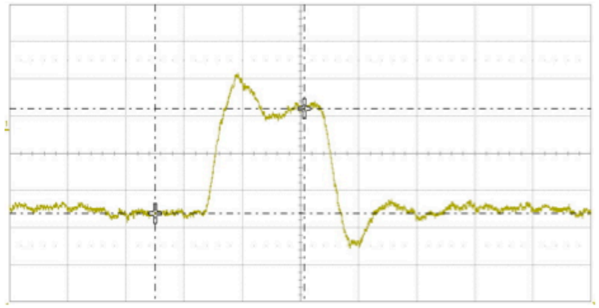


Fig 5: 5.00 MHz clock. Pedestal mean= - 443mV, S.D.= 8.6mV. Peak (cursor 2) mean= 112.6mV, S.D. = 8.6mV.

6.0 MHz

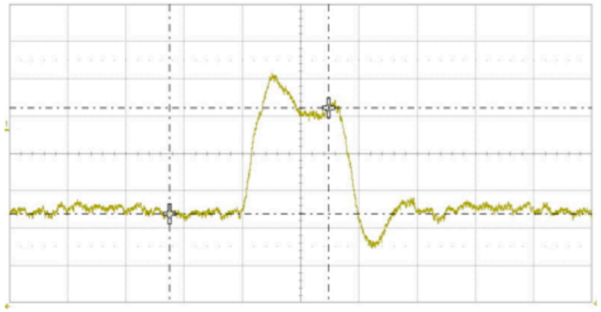
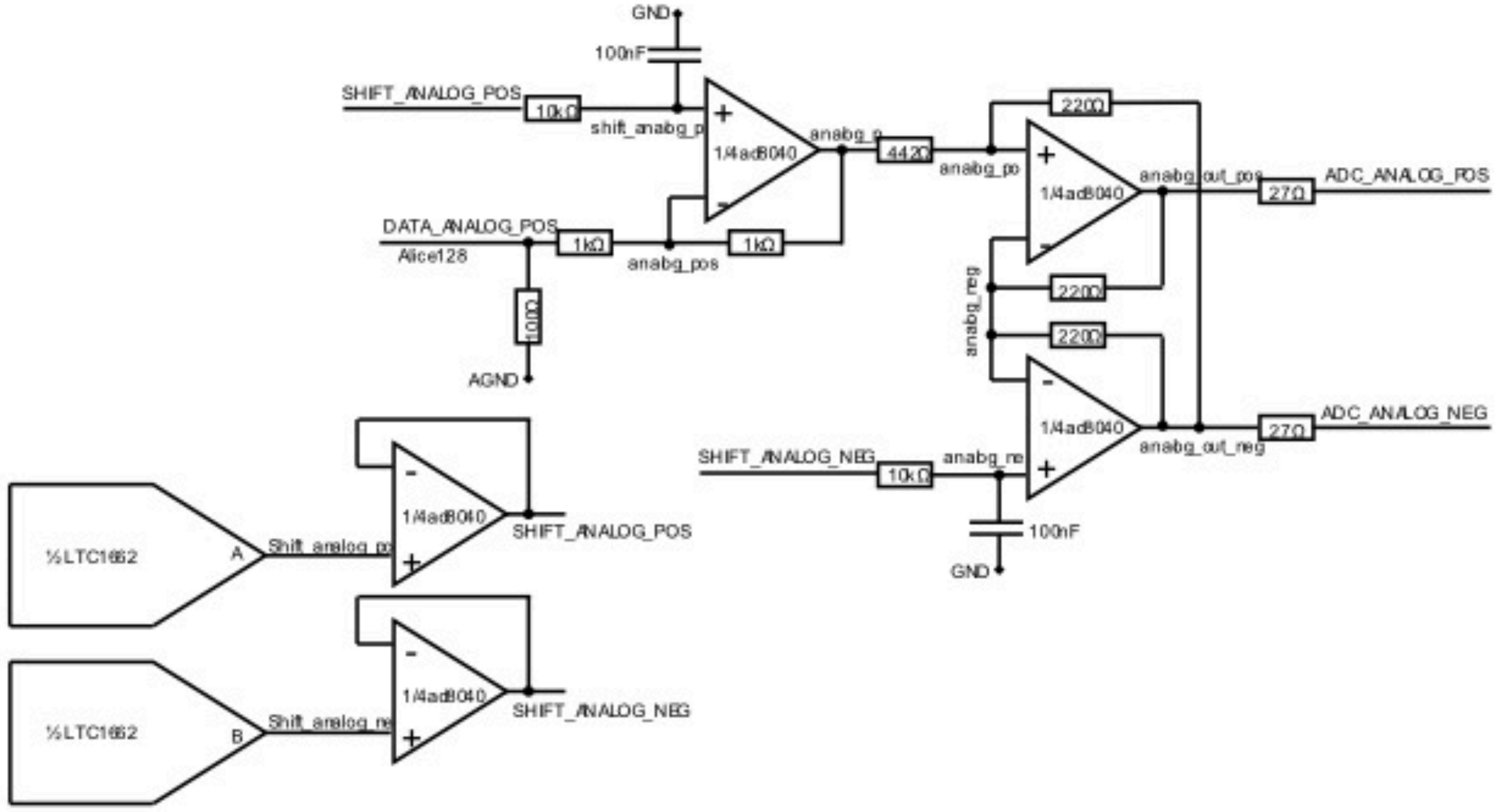


Fig 6: 6.0 MHz clock. Pedestal mean= - 447mV, S.D.= 8.8mV. Peak (cursor 2) mean= 114mV, S.D. = 8.8mV.

Horiz:100 ns/cm

Level-shifting for bias side

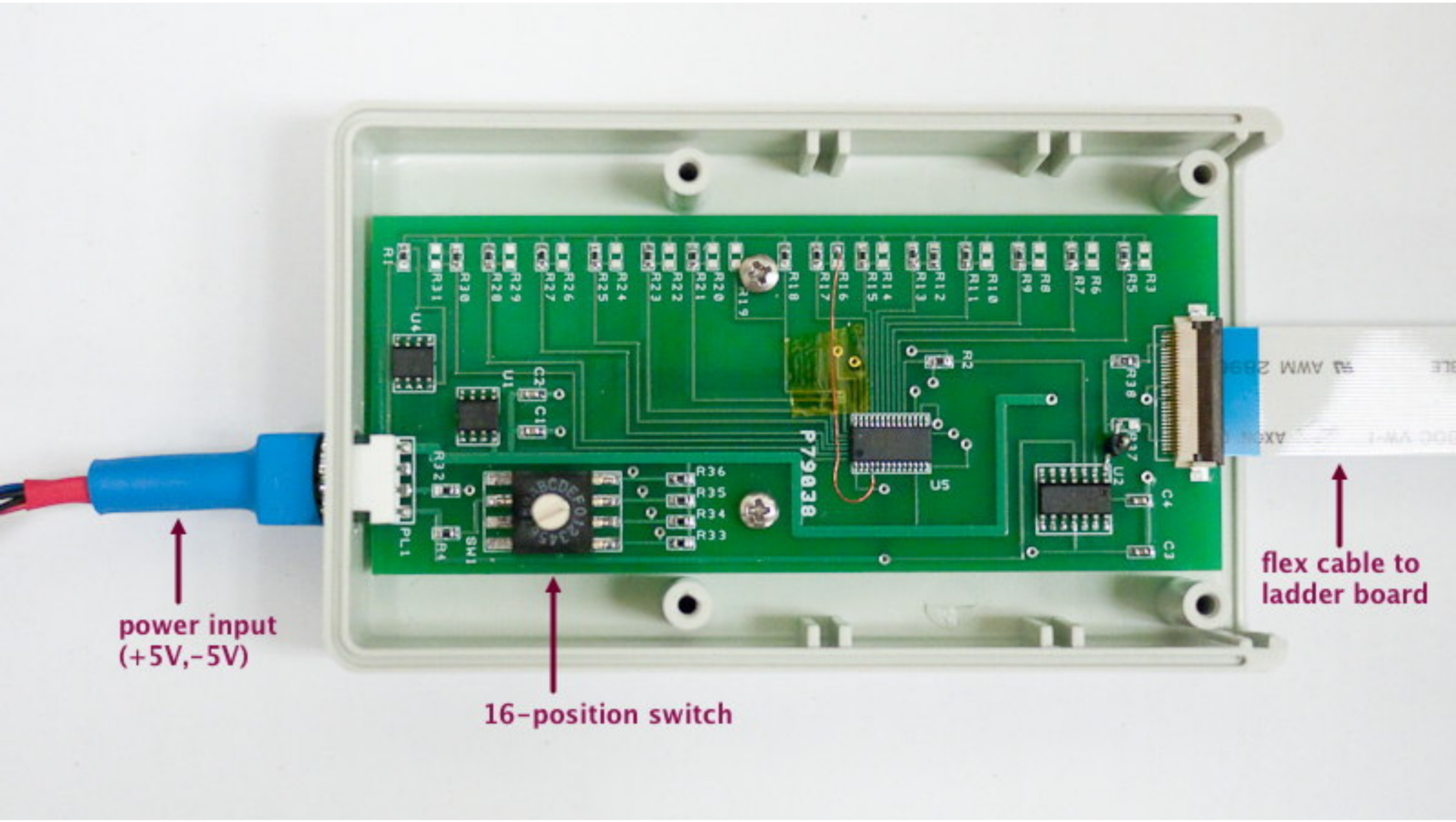


DOE HFT Review

Single event upsets

- Ionizing radiation causes single bit errors in configuration memory (internal to FPGA)
  - Change FPGA behavior
- Scale from observed error frequency in TOF
  - Estimate 1 error per 10 minutes in SSD
- Must pro-actively detect these errors by running CRC checks while acquiring data
  - Provided by Altera
- Time to reconfigure FPGA: < 1 sec

Fake static source



Ladder card testing road map

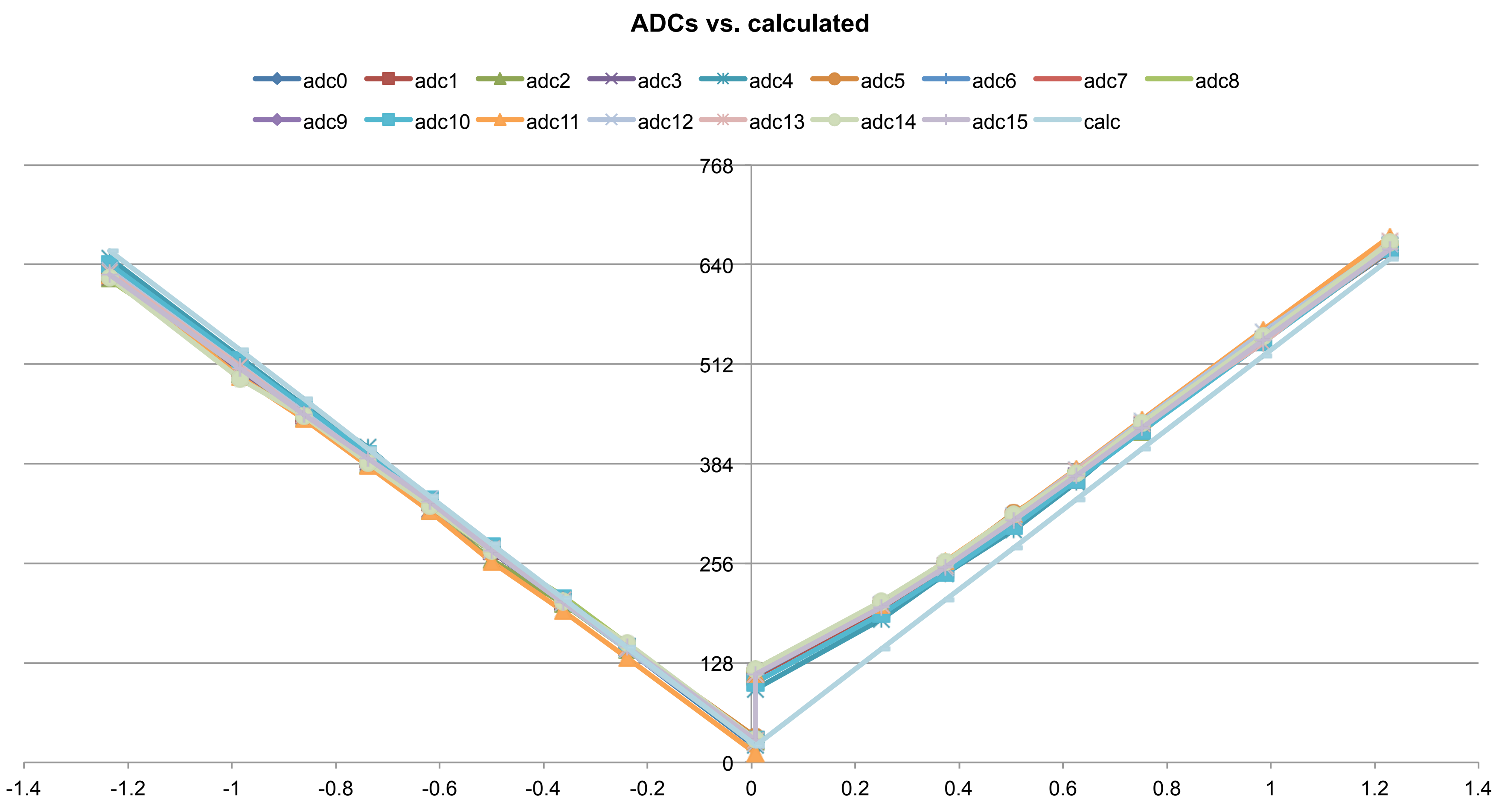
- Use USB to trigger ADC conversion, gather data
- Use “static fake ladder” to provide selectable DC level at each ladder input
- Allows verification of basic functionality of analog section
- Use “dynamic fake ladder” (in design) to verify ADC timing for each ladder.
- Use QRDO (in layout) to verify ladder card functionality up through fiber link

DOE HFT Review

Mapping analog response

- Software
  - Python script driving
  - Multiple .exe (C code)
- Time to map response for 1 ADC: 30 sec
- Time to map all 16 ADCs: 20 minutes
  - Disconnect/connect flex cable
- *Basis for future slow controls software*
- SC uses JTAG header on debug card
  - Will be replaced by fiber protocol

Analog response for all ADCs



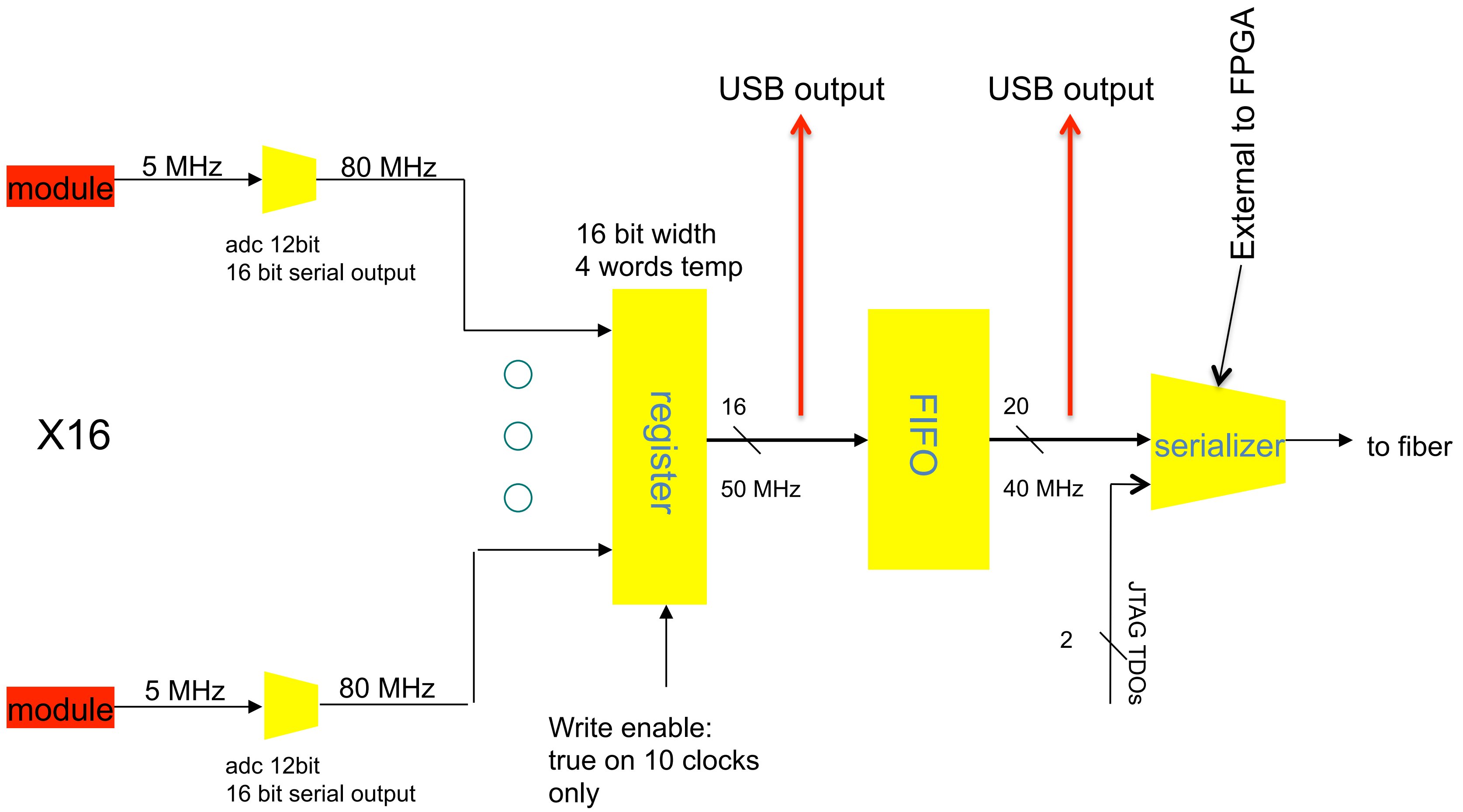
STAR

Analog response

- Non-linear behavior of N-face not yet understood
- Discovered we are sensitive to PS fluctuations via DAC
  - Will be separately regulated in production version (prototype in test)

Verification of packing code

- USB output for ADC data
- Install USB spy at output of FIFO



Ladder packer verification

- Following 4 slides are identical except for notation showing which bits are to be extracted for each word sent to the FIFO
- Lines FIFO[0]…FIFO[7] show the 21-bit word exiting the FIFO
- Comparison shows that the packer is functioning correctly

FIFO21 word 0

DAC A = 300, DAC B = 500

			bit 9									bit 0
ADC[15]:	529	0x211	1	0	0	0	0	1	0	0	0	1
ADC[14]:	885	0x375	1	1	0	1	1	1	0	1	0	1
ADC[13]:	889	0x379	1	1	0	1	1	1	1	0	0	1
ADC[12]:	891	0x37B	1	1	0	1	1	1	1	0	1	1
ADC[11]:	901	0x385	1	1	1	0	0	0	0	1	0	1
ADC[10]:	893	0x37D	1	1	0	1	1	1	1	1	0	1
ADC[9]:	888	0x378	1	1	0	1	1	1	1	0	0	0
ADC[8]:	136	0x088	0	0	1	0	0	0	1	0	0	0
ADC[7]:	887	0x377	1	1	0	1	1	1	0	1	1	1
ADC[6]:	890	0x37A	1	1	0	1	1	1	1	0	1	0
ADC[5]:	886	0x376	1	1	0	1	1	1	0	1	1	0
ADC[4]:	901	0x385	1	1	1	0	0	0	0	1	0	1
ADC[3]:	888	0x378	1	1	0	1	1	1	1	0	0	0
ADC[2]:	888	0x378	1	1	0	1	1	1	1	0	0	0
ADC[1]:	891	0x37B	1	1	0	1	1	1	1	0	1	1
ADC[0]:	892	0x37C	1	1	0	1	1	1	1	1	0	0

Bit 20 ("first word")

↓

FIFO[0]	0x1FFEFF
FIFO[1]	0x0107EF
FIFO[2]	0x06EF09
FIFO[3]	0x076EF7
FIFO[4]	0x1FF6EF
FIFO[5]	0x0B1374
FIFO[6]	0x00E24C
FIFO[7]	0x0FC921

Fifo 0 (19..16)   Fifo 0 (15..0)

FIFO21 word 1

DAC A = 300, DAC B = 500

```
                          bit          bit
                           9            0
                          =====================
ADC[15]:   529   0x211     1 0 0 0 0 1 0 0 0 1
ADC[14]:   885   0x375     1 1 0 1 1 1 0 1 0 1
ADC[13]:   889   0x379     1 1 0 1 1 1 1 0 0 1
ADC[12]:   891   0x37B     1 1 0 1 1 1 1 0 1 1
ADC[11]:   901   0x385     1 1 1 0 0 0 0 1 0 1
ADC[10]:   893   0x37D     1 1 0 1 1 1 1 1 0 1
ADC[9]:    888   0x378     1 1 0 1 1 1 1 0 0 0
ADC[8]:    136   0x088     0 0 1 0 0 0 1 0 0 0
ADC[7]:    887   0x377     1 1 0 1 1 1 0 1 1 1
ADC[6]:    890   0x37A     1 1 0 1 1 1 1 0 1 0
ADC[5]:    886   0x376     1 1 0 1 1 1 0 1 1 0
ADC[4]:    901   0x385     1 1 1 0 0 0 0 1 0 1
ADC[3]:    888   0x378     1 1 0 1 1 1 1 0 0 0
ADC[2]:    888   0x378     1 1 0 1 1 1 1 0 0 0
ADC[1]:    891   0x37B     1 1 0 1 1 1 1 0 1 1
ADC[0]:    892   0x37C     1 1 0 1 1 1 1 1 0 0
```

Bit 20 ("first word") ↓

FIFO[0]	0x1FFEFF
FIFO[1]	0x0107EF
FIFO[2]	0x06EF09
FIFO[3]	0x076EF7
FIFO[4]	0x1FF6EF
FIFO[5]	0x0B1374
FIFO[6]	0x00E24C
FIFO[7]	0x0FC921

Fifo 1 (19..12)   Fifo 1 (11..0)

FIFO21 word 2

DAC A = 300, DAC B = 500

			bit 9 ... bit 0
ADC[15]:	529	0x211	1 0 0 0 0 1 0 0 0 1
ADC[14]:	885	0x375	1 1 0 1 1 1 0 1 0 1
ADC[13]:	889	0x379	1 1 0 1 1 1 1 0 0 1
ADC[12]:	891	0x37B	1 1 0 1 1 1 1 0 1 1
ADC[11]:	901	0x385	1 1 1 0 0 0 0 1 0 1
ADC[10]:	893	0x37D	1 1 0 1 1 1 1 1 0 1
ADC[9]:	888	0x378	1 1 0 1 1 1 1 0 0 0
ADC[8]:	136	0x088	0 0 1 0 0 0 1 0 0 0
ADC[7]:	887	0x377	1 1 0 1 1 1 0 1 1 1
ADC[6]:	890	0x37A	1 1 0 1 1 1 1 0 1 0
ADC[5]:	886	0x376	1 1 0 1 1 1 0 1 1 0
ADC[4]:	901	0x385	1 1 1 0 0 0 0 1 0 1
ADC[3]:	888	0x378	1 1 0 1 1 1 1 0 0 0
ADC[2]:	888	0x378	1 1 0 1 1 1 1 0 0 0
ADC[1]:	891	0x37B	1 1 0 1 1 1 1 0 1 1
ADC[0]:	892	0x37C	1 1 0 1 1 1 1 1 0 0

Bit 20 ("first word") ↓

FIFO[0]	0x1FFEFF
FIFO[1]	0x0107EF
FIFO[2]	0x06EF09
FIFO[3]	0x076EF7
FIFO[4]	0x1FF6EF
FIFO[5]	0x0B1374
FIFO[6]	0x00E24C
FIFO[7]	0x0FC921

Fifo 2 (19..8)   Fifo 2 (7..0)

FIFO21 word 3

DAC A = 300, DAC B = 500

```
                                bit              bit
                                 9                0
                                =======================
ADC[15]:   529   0x211          1 0 0 0 0 1 0 0 0 1
ADC[14]:   885   0x375          1 1 0 1 1 1 0 1 0 1
ADC[13]:   889   0x379          1 1 0 1 1 1 1 0 0 1
ADC[12]:   891   0x37B          1 1 0 1 1 1 1 0 1 1
ADC[11]:   901   0x385          1 1 1 0 0 0 0 1 0 1
ADC[10]:   893   0x37D          1 1 0 1 1 1 1 1 0 1
ADC[9]:    888   0x378          1 1 0 1 1 1 1 0 0 0
ADC[8]:    136   0x088          0 0 1 0 0 0 1 0 0 0
ADC[7]:    887   0x377          1 1 0 1 1 1 0 1 1 1
ADC[6]:    890   0x37A          1 1 0 1 1 1 1 0 1 0
ADC[5]:    886   0x376          1 1 0 1 1 1 0 1 1 0
ADC[4]:    901   0x385          1 1 1 0 0 0 0 1 0 1
ADC[3]:    888   0x378          1 1 0 1 1 1 1 0 0 0
ADC[2]:    888   0x378          1 1 0 1 1 1 1 0 0 0
ADC[1]:    891   0x37B          1 1 0 1 1 1 1 0 1 1
ADC[0]:    892   0x37C          1 1 0 1 1 1 1 1 0 0
```

Bit 20 ("first word") ↓

FIFO[0]	0x1FFEFF
FIFO[1]	0x0107EF
FIFO[2]	0x06EF09
FIFO[3]	0x076EF7
FIFO[4]	0x1FF6EF
FIFO[5]	0x0B1374
FIFO[6]	0x00E24C
FIFO[7]	0x0FC921

Fifo 3 (19..4) Fifo 3 (3..0)

Ladder card testing status

- Routing error on FPGA discovered
  - Temporary fix using interposer
  - Continue testing using original PCB w/ interposer
- Configure FPGA via debug card JTAG header
- Communication with FPGA via USB verified
- Slow controls functionality (JTAG) verified

Ladder card hardware

- Testing of ladder card has exposed only 2 problems
  - FPGA orientation
    - Corrected on pre-production version
  - Susceptibility of analog section to PS variation
    - Can degrade ability to interpolate centroid
    - Alternate DAC design being tested
  - To be done
    - Verify interaction with ladder modules
      - Token passing
      - JTAG

Ladder card firmware

- All functions that have been tested are working correctly
- Remaining to verify –
  - JTAG to ladder components

Rad hardness of optical xcvr

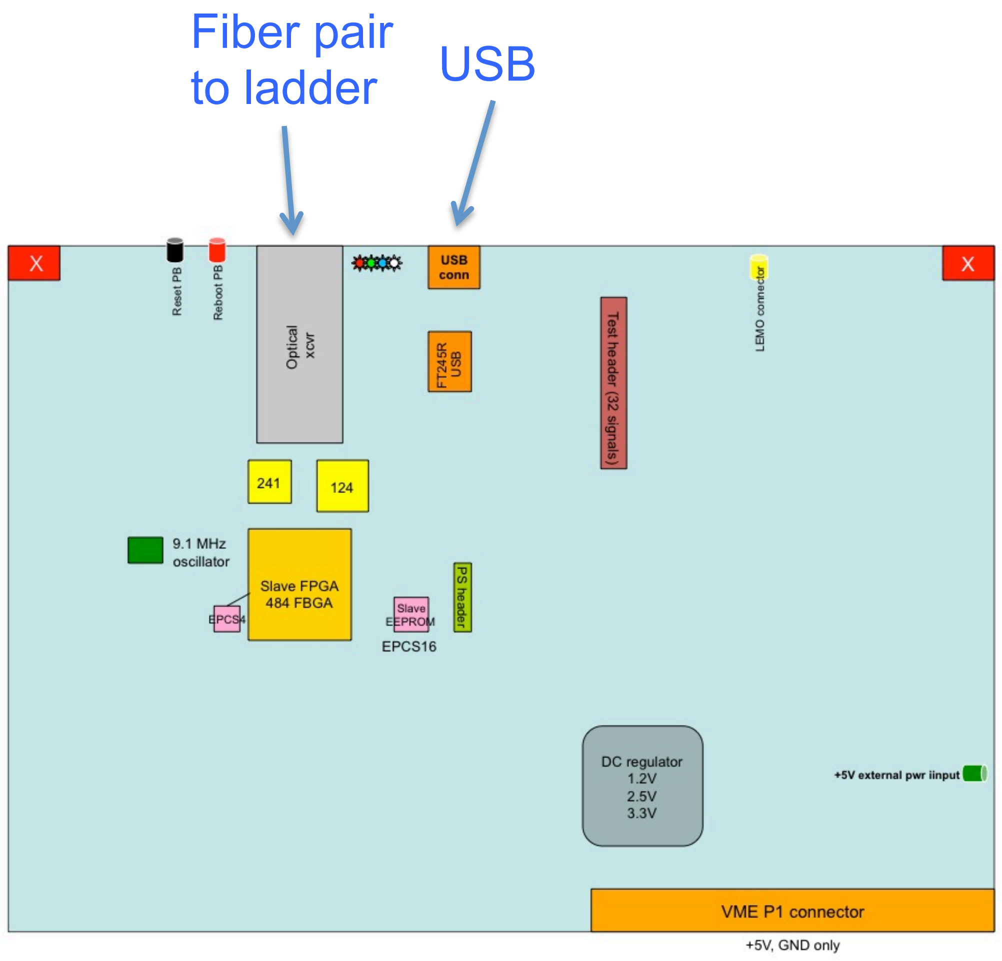
- Tests of Avago xcvr show that it dies around 100 kRad (Co^{60} source)
- A rad hard version has been developed for LHC
  - Working prototypes now available
- *Almost* plug compatible with Avago
  - Requires 2.5V instead of 3.3V
  - Jumper provided on ladder board to select 2.5V supply

QRDO -

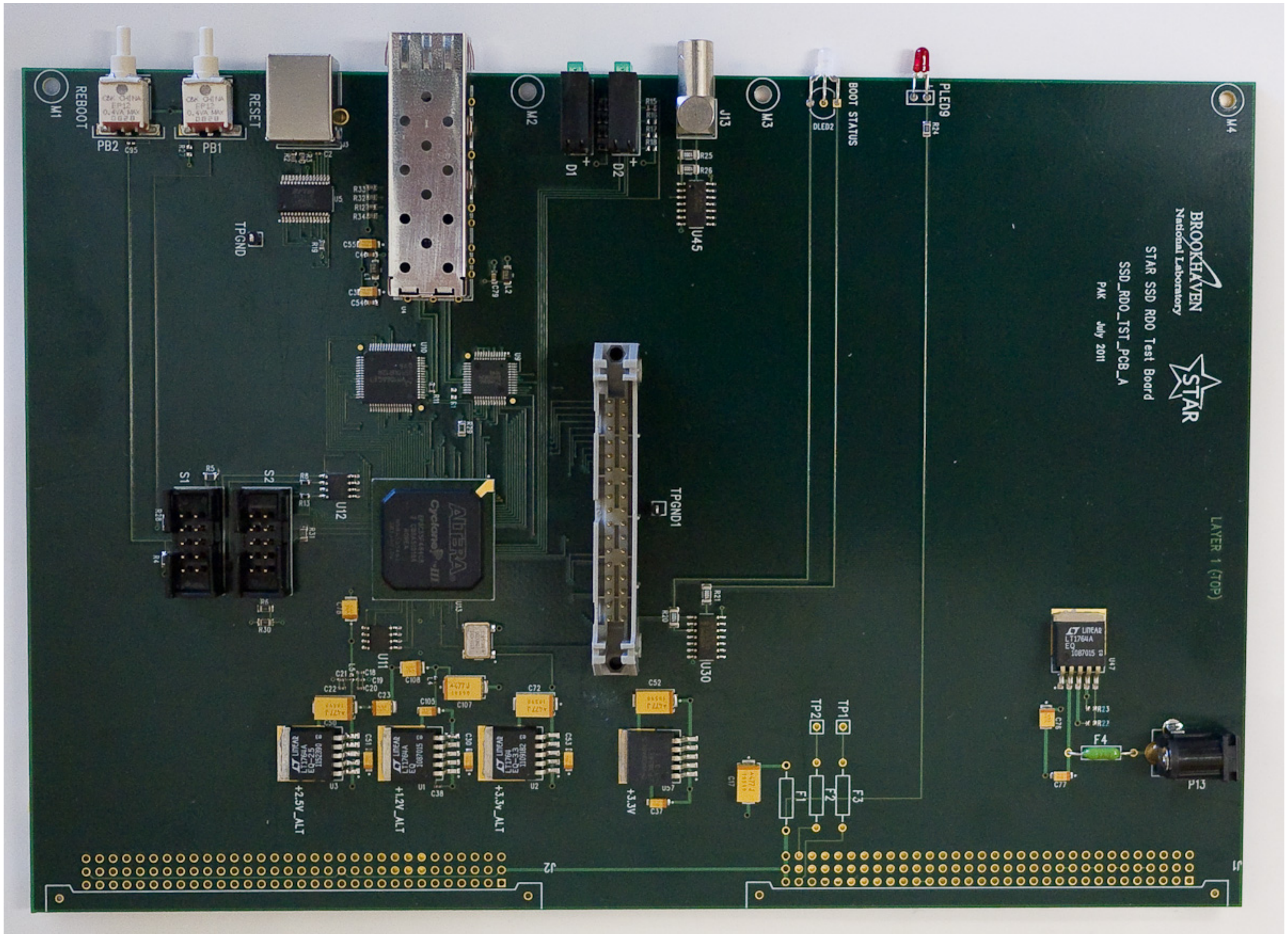
- prototype RDO slave
- ladder card test stand

QRDO (Fast-track version of RDO)

- Interfaces to one ladder board
- Implements one slave FPGA
- Only features required for testing ladder board
  - No TRG, DAQ
- All input/output via USB
- Can acquire up to 4 events at full speed



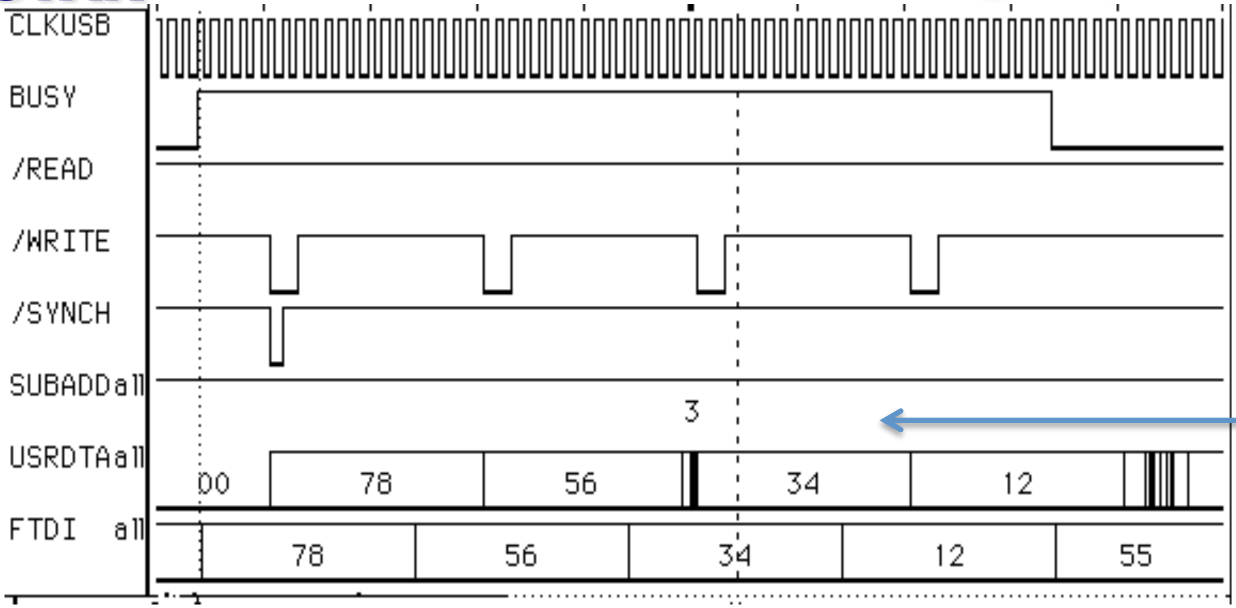
QRDO assembled



QRDO commissioning

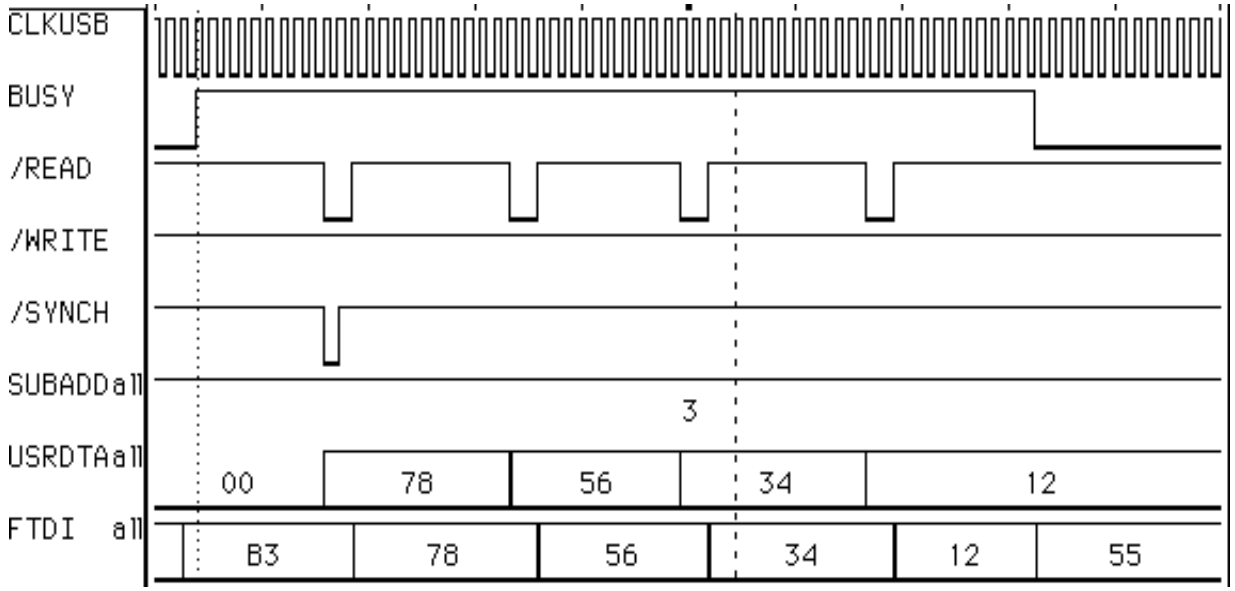
- Orsay USB protocol implemented
  - Message layer on top of byte pipe
  - Goal: replace VME (4-byte messages)
- Problems –
  - Bad synthesis by Synopsys tool !!
  - Now resolved
- Message protocol working

USB message protocol



4-byte write

6-bit subaddress used for register addressing



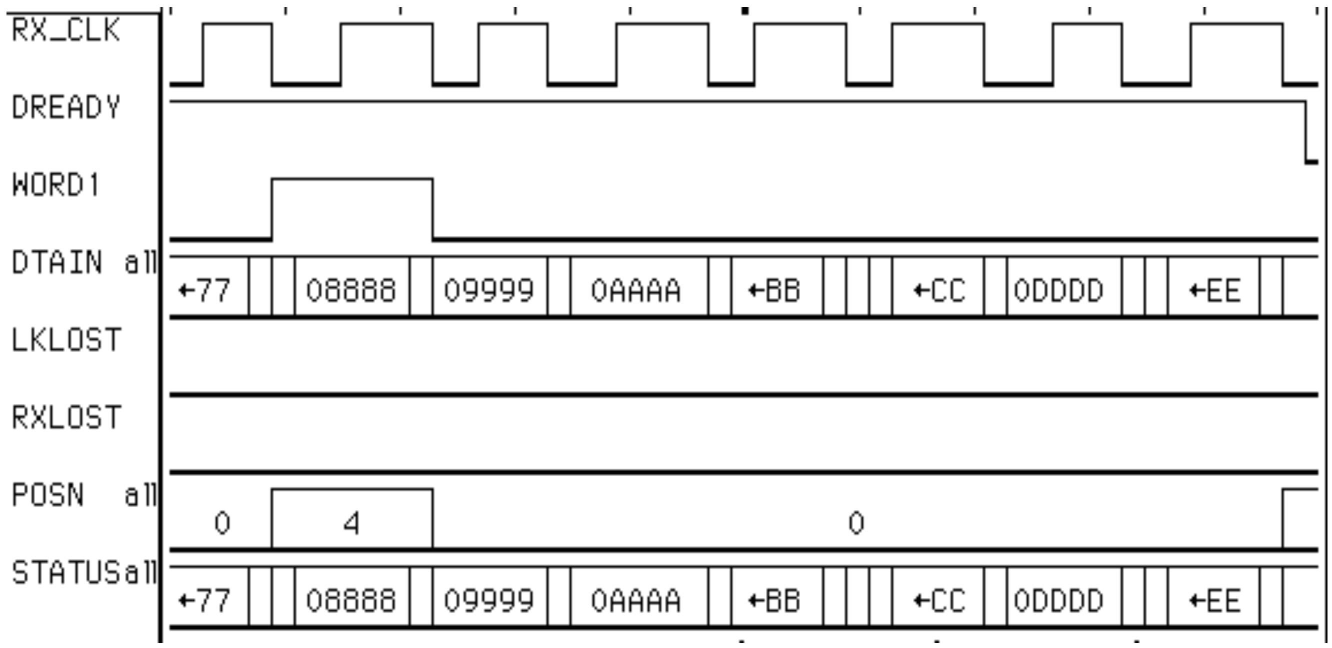
4-byte read

Testing ladder card with QRDO

Integration of ladder card/QRDO

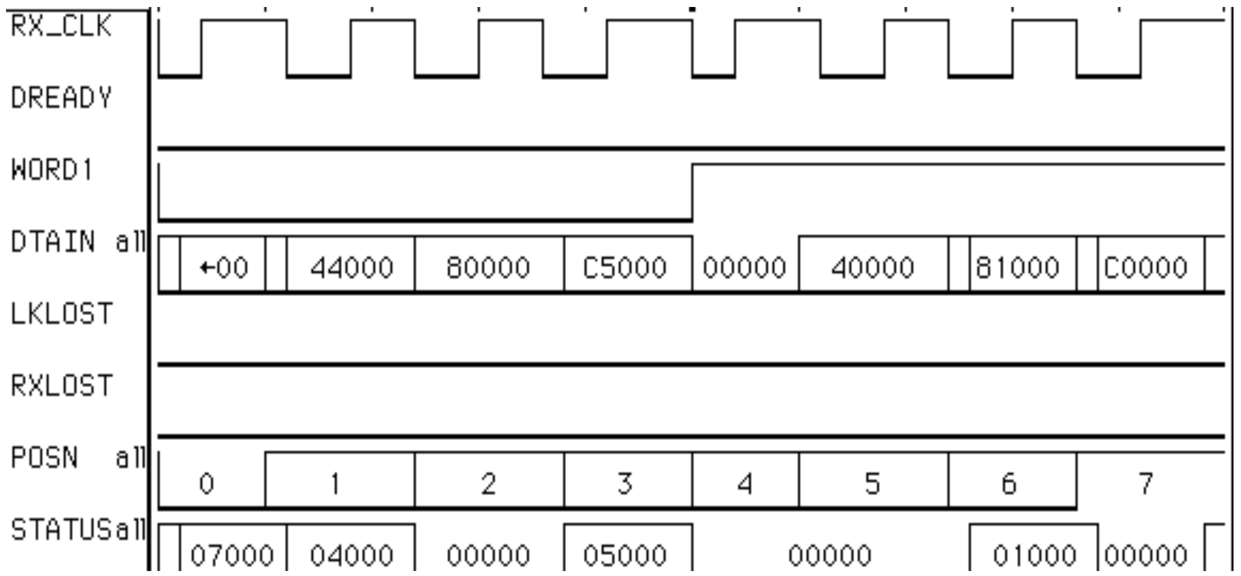
- Generate test patterns on ladder card
- Spy at QRDO on incoming data via fiber with logic analyzer

Data sequence received in QRDO



DREADY='1' signifies data phase
Data shown here are artificial for diagnostics

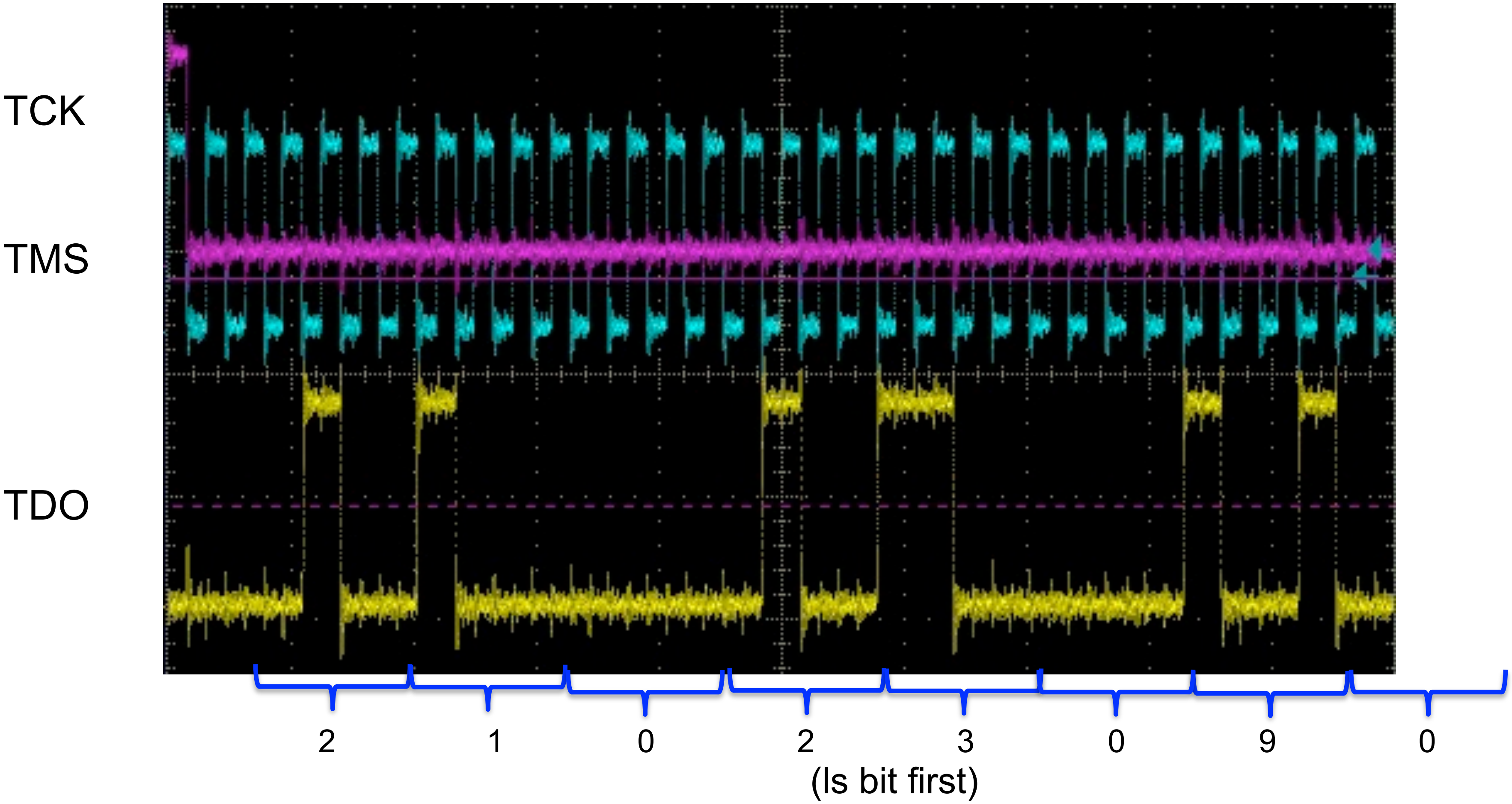
Status words received in QRDO



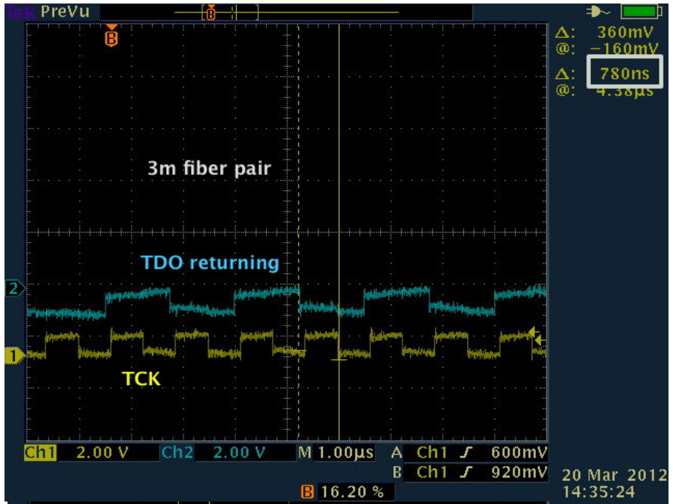
Word	Value	Comment (from Table 52, master document)
0	07000	configured, OK, serdes clock used
1	04000	deserializer lock OK
2	00000	(no optical transceiver problems)
3	05000	usb present, debug present
4	00000	
5	00000	serial #1 (agrees with hardware assignment on board)
6	01000	
7	00000	ladder 0 (not yet assigned by QRDO)

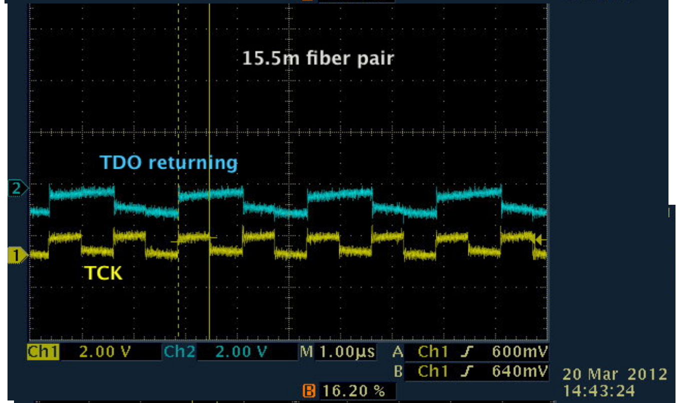
JTAG testing via fiber

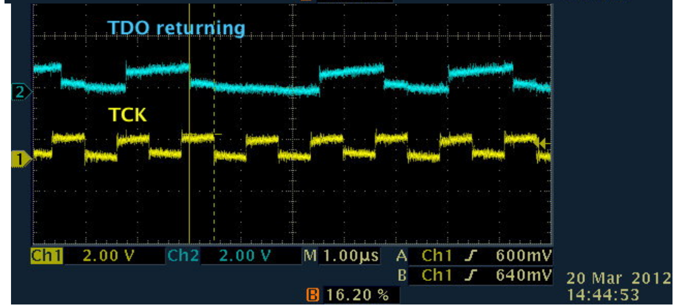
Read version date register (contents = 0x09032012)



Test JTAG with full fiber length







🡐 1m

🡐 15.5m

🡐 31m

Returning TDO is shifted on falling edge of TCK. ➔ There is still a safety margin of 480ns with 31m fiber cable.

Note Δ=780ns with 3m cable, 600ns with 15.5m cable, 480ns with 31m cable.

(scope probes were not properly grounded, thus shifting baselines)

QRDO status

- Hardware working as expected
  - PCB layout by Phil Kuczewski
- Firmware status:
  - JTAG to ladder via fiber
    - working
  - Configure ladder FPGA over fiber
    - working

QRDO (RDO prototype)

- Status
  - Have been working with assembled QRDO since Summer 2011
  - Uses USB as control port
    - Not without problems!
  - Allowed verification of data arriving via fiber
  - Allowed JTAG to be debugged and verified
  - Allowed configuration of ladder FPGA to be debugged and verified
    - Download FPGA code via fiber in 0.5s

RDO

RDO status

- Schematic complete
- PCB layout finished June 13, 2012
- Assembled prototypes expected mid July, 2012
- need VME master to test complete functionality
  - Use USB-VME bridge (Wiener)

RDO slave <-> master

- Design uses lvds serial lanes
  - 4 slave -> master (event data)
  - 2 master -> slave (commands)
- 5 slaves -> master required 5 PLLs in master
- Only 4 PLLs available in this chip family

Alternate designs considered

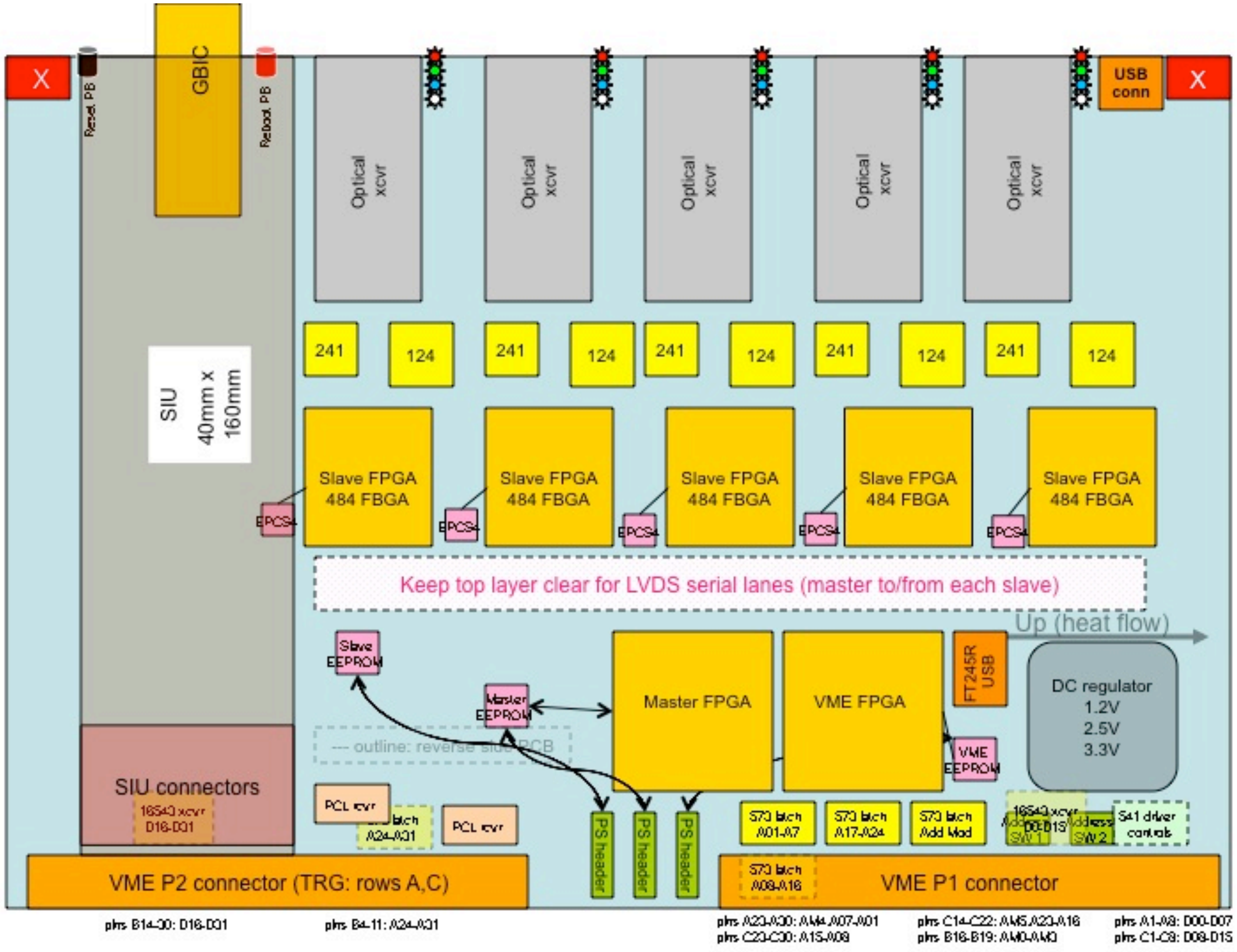
- Parallel bus
  - High freq clock to have required data xfer rate
- Daisy chained highway
  - Too many pins required on slaves

- Large enough FPGA to implement 5 slaves + master
  - Not an option until one month ago

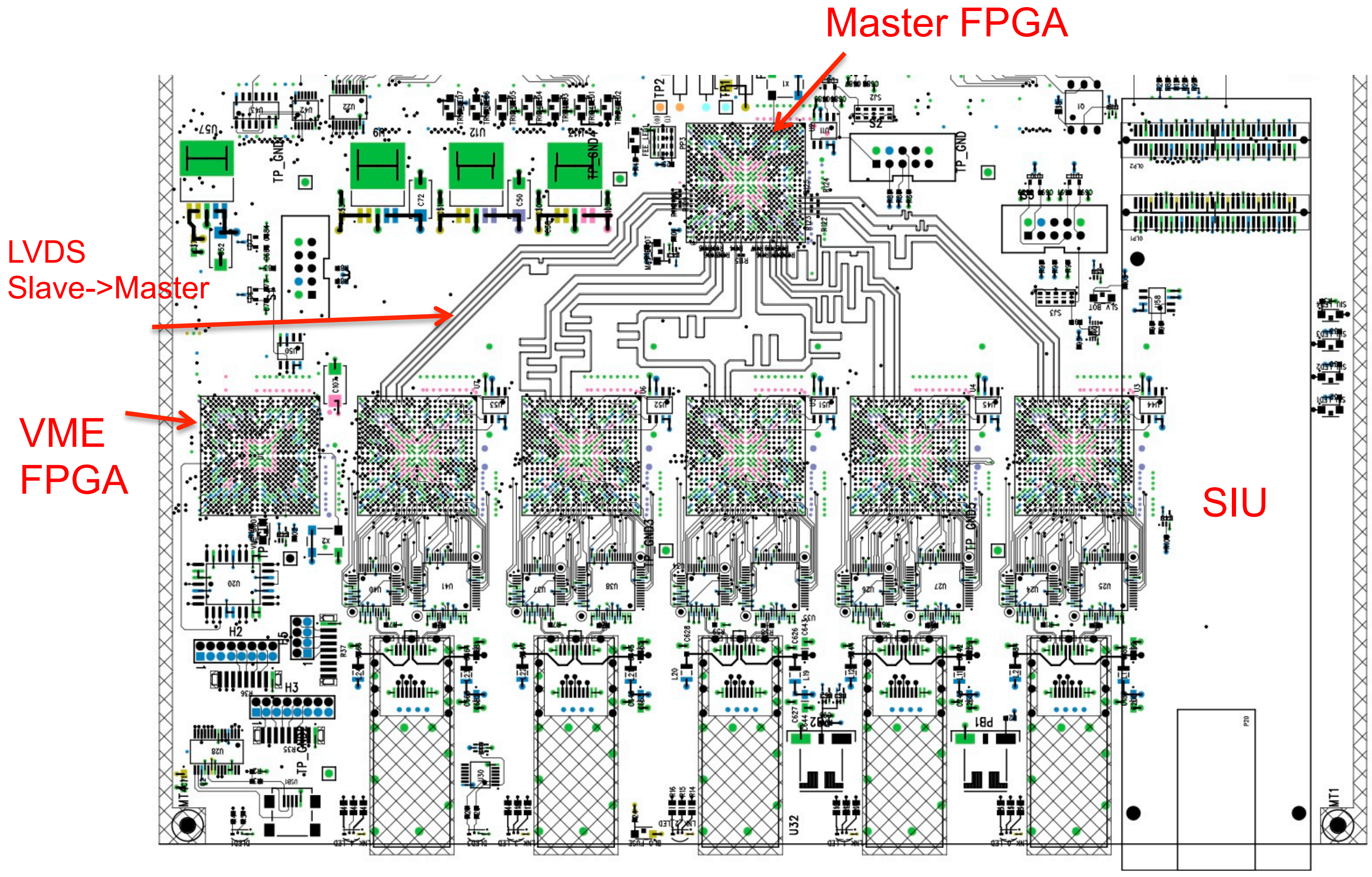
RDO workaround

- Workaround chosen:
  - Source synchronous using clock distributed by master
  - Clock captured by slave PLLs and used to transmit to master
  - Requires equal round-trip path lengths
  - LVDS 454 MHz (2.2ns)
  - Path lengths matched to 2.5mm (17 ps) for all 5 slaves

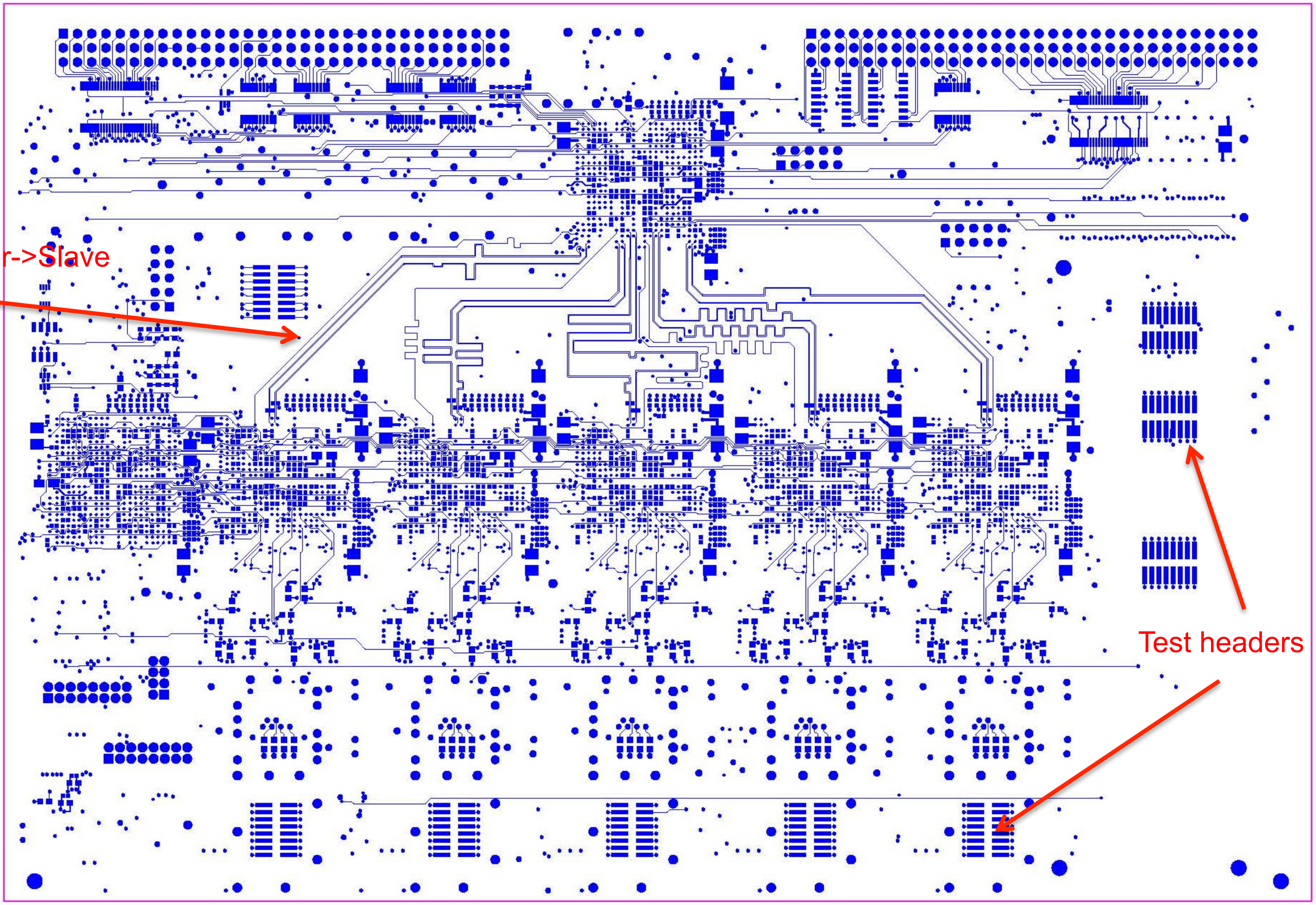
RDO conceptual layout



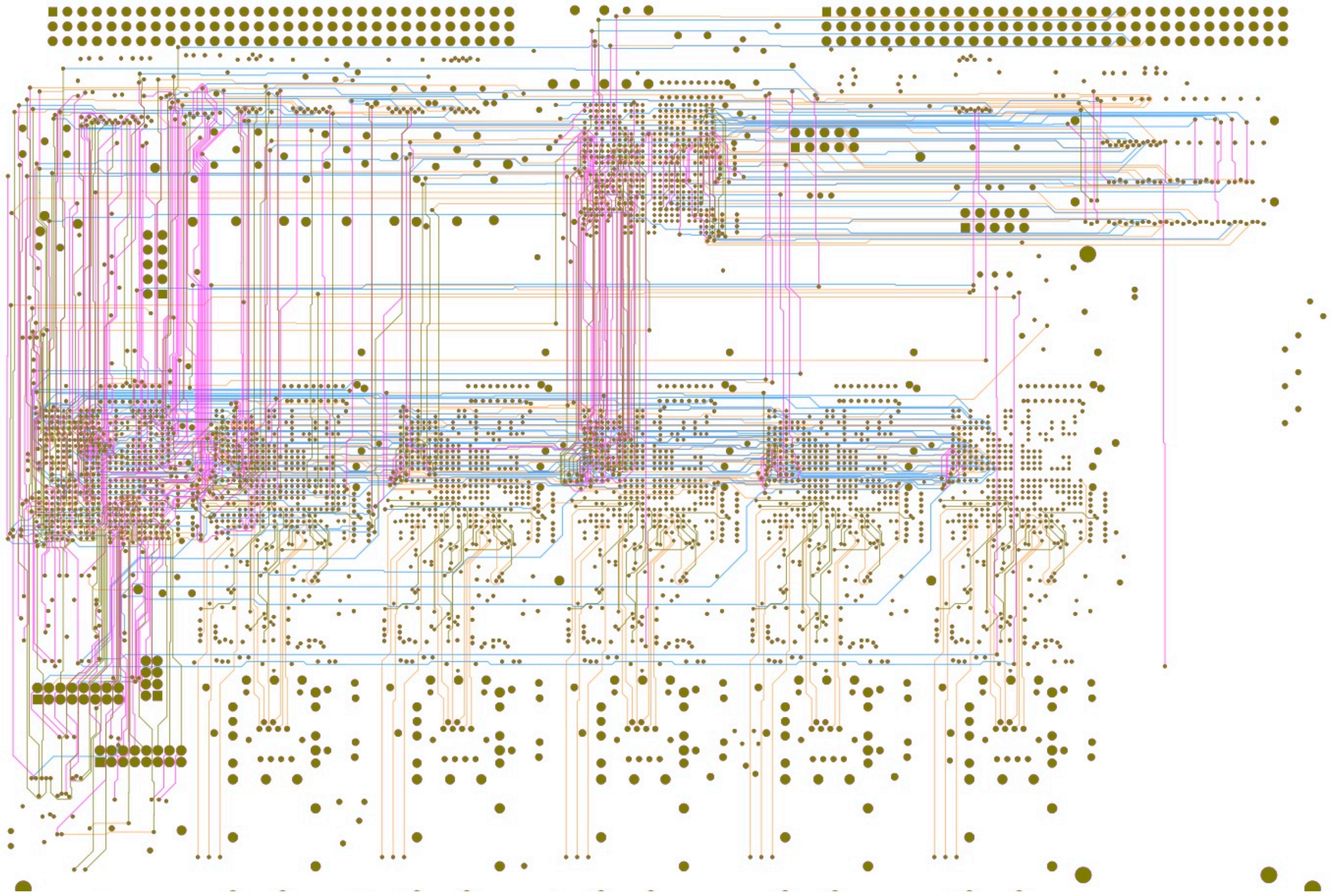
RDO top layer



RDO bottom layer



RDO internal layers



DAQ PC status

- 2 DAQ PCs delivered
- 2 D-RORCs installed
  - 1 PC only
- Scientific Linux 5.x installed (64 bit)
- DAQ software installed (Tonko) – 1 PC
- Currently using as a test bed for USB software
  - prototype slow controls platform (temporary)

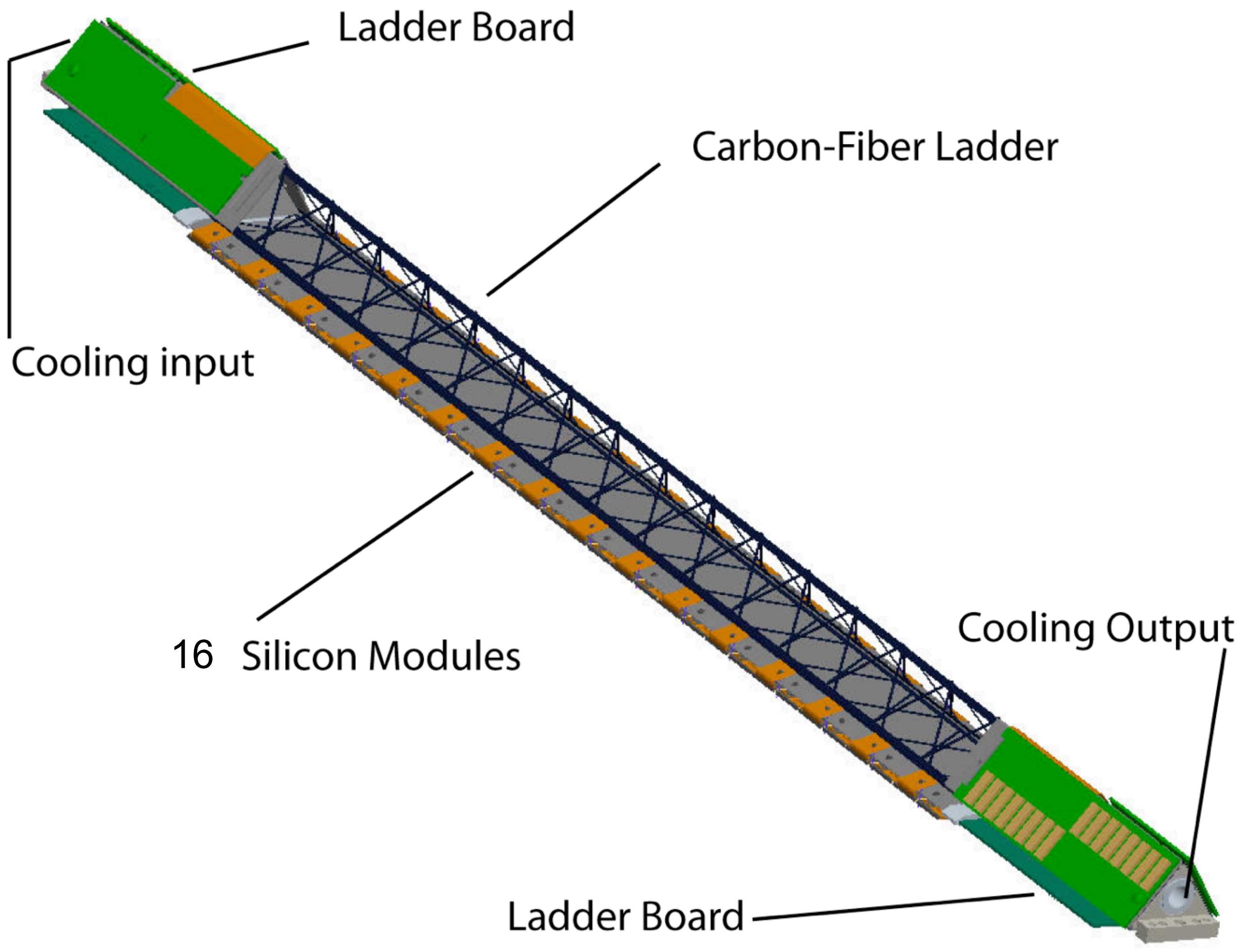
RDO

- Status
  - 6U VME board – very tight
  - Components in house
  - Schematic finished
  - Board layout finished
  - Schedule
    - Board ready for fabrication
    - Assembled by mid-July
    - Testing in July-August

SSD slow controls roadmap

STAR

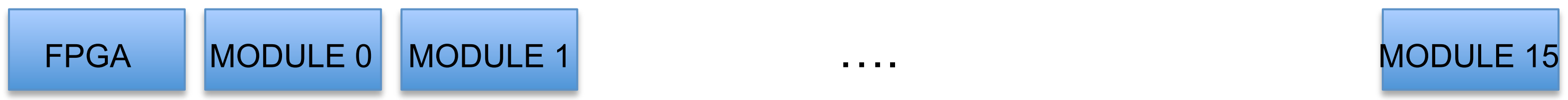
SSD ladder



JTAG chain possibilities

Legacy:

FPGA + 16 modules – 7 devices per module !!

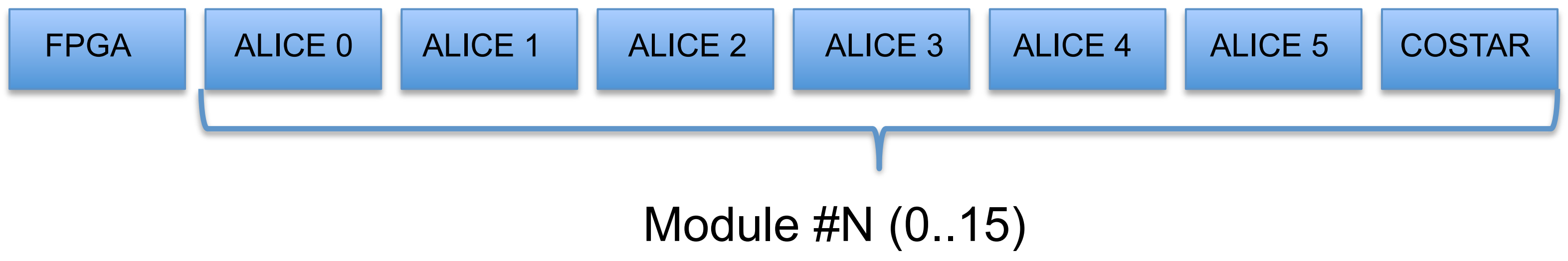


103 elements in chain. Not robust!

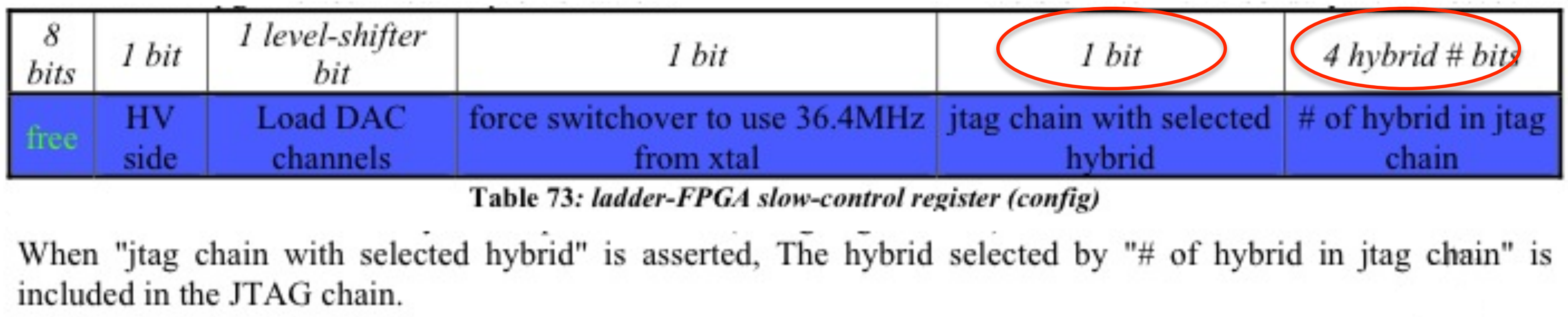
Upgrade:

FPGA only

FPGA + module #N



FPGA register configures chain



8 bits	*1 bit*	*1 level-shifter bit*	*1 bit*	*1 bit*	*4 hybrid # bits*
free	HV side	Load DAC channels	force switchover to use 36.4MHz from xtal	jtag chain with selected hybrid	# of hybrid in jtag chain

Table 73: ***ladder-FPGA slow-control register (config)***

When "jtag chain with selected hybrid" is asserted, The hybrid selected by "# of hybrid in jtag chain" is included in the JTAG chain.

Slow controls - layers

- Top level – runs on SC linux machine
  - Identical to previous implementation
- Lower level - now a hybrid
  - Part on linux VME master
  - Part in RDO slave FPGAs
- Interface between these defined 4/12
- Code for lower level written
  - Not yet debugged
    - Requires a ladder or one module of ladder

Changes required to slow controls

- Must reconfigure chain each time a module is included/excluded
  - Extra steps that were not part of legacy SC software
- Legacy implementation used Corelis VME module
  - Based on TI8990 which fills role of JTAG master engine
  - Output was multiplexed to 4 Readout boards
- Upgrade JTAG masters implemented in (40) slave FPGAs
  - Slow control software must be modified to speak directly to slave FPGAs via VME
  - Advantage: 40 macro instructions can be executed simultaneously

Existing JTAG implementation

Used for tests --

```
/**************************************************************************

      Function: void write_register(reg_no, value)

      Function: unsigned int read_register(reg_no)

      Function: void reset_TAP()

**************************************************************************/
```

Note – read register implemented as non-destructive (uses circulate data)

New interface defined (1)

Interface between existing slow controls software (upper level) and the implementation in the slave FPGAs has been negotiated between Weihua Yan and MJL as the following 3 functions which need to be implemented in VHDL and C++:

```
/**************************************************************************

       Function: scan_ir()

        Summary: Scans a bit stream into the TAP instruction register

          Usage: void scan_ir(output, length, input)
                 unsigned short *output;
                 unsigned short length;
                 unsigned short *input;

**************************************************************************/
void scan_ir(unsigned short *output, unsigned short length, unsigned short *input) {  }
```

New interface defined (2)

```
/************************************************************************

       Function: scan_dr()

        Summary: Scans a bit stream out the TAP data path

          Usage: void scan_dr(output, length, input)
                 unsigned short *output;
                 unsigned short length;
                 unsigned short *input;

************************************************************************/
void scan_dr(unsigned short *output, unsigned short length, unsigned short *input)  {  }
```

New interface defined (3)

```
/************************************************************************

        Function: circulate_dr()

         Summary: Circulates a bit stream thru the TAP controller data path

           Usage: void circulate_dr(length, data)
                  unsigned short length;
                  unsigned short *data;

************************************************************************/
void circulate_dr(unsigned short length, unsigned short *output)  {  }
```