Ultimate / Test data link - Normal readout = Not test modes

Strasbourg November 2010

OUTLINE

- Ultimate Steering & Readout
- Why & How to test data transmission ?
- Test with header, trailer & frame counter
- Test by emulating hits on matrix
- Conclusion

Steering & Readout : Ultimate ?

Ultimate ?

- **Matrix of 890 880 pixels : 960 columns x 928 lines – 19,872 mm x 19,210 mm – 20,7 µm pitch – 185,6 µs integration time**
  - **Normal digital data stream after Zero Suppression (ZS)**

Steering

- **Configuration (operating modes, bias etc …) by JTAG slow control**
- **Main clock 160 MHz**
  - **On board oscillator (single Mimosa 26 setup)**
  - **Provided by external oscillator (Telescope or ladder setup)**
- **Start command**
  - **By JTAG slow control (single Ultimate setup)**
  - **By external HW signal (Telescope or ladder setup)**

Readout

- **Normal ZS output** - **Full memory size → 2 Outputs at 160 MHz**

Steering & Readout : Signals ?

Readout configuration N° 3 : 2 outputs @ 160 MHz

Summary

- Data generated on rising edge of Ultimate output clock
- Synchronization signal MKD – Duration 4 clocks cycles
- Data LSB first
- First bit of frame = Header LSB

Readout configuration N° 3 : 2 outputs @ 160 MHz

- **Provides the whole states memory size : 1850 W16 (word of 16 bits) – 925 W16 / link**

Summary

- **Data generated on rising edge of Ultimate clock**
- **Header** → **16 bits / output**
- **Frame counter** → **16 bits / output**
- **Data length (useful part of data)** → **16 bits / output (Sum the 2 W16 to get matrix W16 size)**
- **Data (format on next slide)** → **Max = 1850 x 16 bits / output**
- **Trailer** → **16 bits / output**
- **Padding zero** → **32 bits / output**
- **Total stream size per output : 29696 bits = 1856 W16 = 3712 W8**

Readout configuration N° 3 : 2 outputs (D0, D1) @ 160 MHz

- **If there is at least one hit in one line, the following data stream is generated**
  - **Status/Line word**
    - **Address of line**
    - **Number of States (9 Max, overflow flag if > 9)**
  - **States list – One state = consecutive pixels at 1 in the line**
    - **Column address of the first pixel at 1**
    - **Number of pixels at 1 - Warning : 0 means one pixel at 1 (value → pixel nb : 0 → 1, 1 → 2, 2 → 3, 3 → 4)**
- **For each line with hit : one Status/line followed by up to 9 States**

Status/ line															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit(0-3)				Bit(0-9)											
number of *States*				The address of the line										Not used	OVF

State															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit(0 -1)		Bit(0-9)													
number of hit pixels		the address of the column										not used			

Steering & Readout : Bit stream Mux on D0, D1

Readout configuration N° 3 : 2 outputs (D0, D1) @ 160 MHz

- Each output has one Header, Frame counter, Data length, Trailer as W16 fields
  - It may be wise to build W32 fields
    - Low 16 bits = field on output D0
    - High 16 bits = field on output D1
  - Data length ?
    - The total data length is the sum of the two fields (on D0, D1)

- Useful data multiplexed over D0 and D1
  - The first W16 is on D0, next on D1, and so on
  - What happens if total W16 number is odd ?
    - Same W16 number / output – Total is always even !
    - A dummy W16 is added at end of D1 stream

Test data transmission : Why & How ?

Why ?

- We have a bus a bit difficult to handle → Because all signals (clk, sync, data) are not on the same wire
  - But this approach is still acceptable at 160 MHz
- Therefore all signals (clock, syncho, data) may not have the same delay from Ultimate / DAQ
- We must find the best sampling position
- Problem complexity will increase on a ladder → N x Ultimates = N data paths with different delays !

How ?

- Probe signals with an oscilloscope → Measure delays
  - Need to add test points to connect oscilloscope
  - Signal test path will add delay dispersion = difficult to see real timings → Clock period is 6,25 ns !!!
- Acquire data & process BER (Bit Error Rate)
  - Acquires frames – Count errors
  - Adjust timing via FPGA " deskewing " functionality (IODELAY)
  - Redo the procedure until you get lower BER

Test data transmission : How to perform the test ?

What do we need on Ultimate side ?

- Do we need to generate well known data stream ?
  - Test at physical level = eye diagram opening – jiiter → No
  - Count data transmission errors → Yes
- It means that we need a kind of pattern generator inside Ultimate

Do we have one ? … Yes & No …

- No because
  - We can't apply a pattern during the whole frame length = we can't replace Ultimate frame by a pattern
  - Because pattern registers "Line_pat0 " & "Line_pat1 " are before Zero Suppression logic
- Yes
  - We can use the Header & Trailer fields + The frame counter → They act as a pattern generator
  - We can define a hit pattern in the matrix
    - Define by line → "Line_pat0" even lines / "Line_pat1" odd lines
    - But this will not be a straight forward way → Need some SW
      - Because you need to decode data stream to extract a matrix " bit map" of emulated hits
      - Compare this bit map to the pattern configured via JTAG

Test data transmission : Test with header & trailer

Easiest way … but limited statistic

- Set hit emulation to 0 → Frame will contain only : Header, Frame counter, Trailer (Data length = 0)
- On each link we will have 16 (Header) + 16 (Frame counter) + 16 (Trailer) = 48 bits pattern
- 32 bits have same value (Header + Trailer) on all frames – 16 bits counts frames
- The frame period is 185,6 µs = 5398 Hz → 253 kb/s on each link
- If (???) we want to test a BER @ 10^{-12} with 10^{12} bits statistic → 1072 hours … 45 days !

Test data transmission : Test by emulating hits

A bit more complicated … but we have statistic

- Create a pattern to fill the matrix
- On each link we will have 1850 x 16 bits = 29600 bits
- The frame period is 185,6 µs = 5398 Hz → 152 Mb/s on each link
- If we want to test a BER @ 10^{-12} with 10^{12} bits statistic → 1,75 hours

Test data transmission : Conclusion

What is the best way to perform the test ?

- First tests → Header & Trailer → but limited statistic 10^{12} = 45 days – (10.10^{6} = 38 seconds)
- If we need a BER test with a stat of 10^{12} we need to fill frame to get statistic → Emulate hits

How to perform a test by emulating hits ?

- Physical level test → We don't need to acquire & check data with DAQ
  - We need only a pattern generator
  - We must take into account ZS algorithm to create patterns
- Test data transmission = count errors → We need to acquire & check data with DAQ
  - We must take into account ZS algorithm to create patterns in both directions
    - To create the pattern
    - To decode acquired data & compare to expected pattern

- Test automation ? → Yes
  - JTAG sw can be controlled via COM interface
  - It's possible to configure JTAG from DAQ sw making request on JTAG sw via COM interface