CMS upgrade – DTTF inputs Asynchronous Links

J.Ero, C.Foudas, N.Loukas,

N.Manthos, I.Papadopoulos

S.Sotiropoulos

Vienna HEPHY

University of Athens

University of Ioannina

21 March 2014 - Workshop on FPGAs for High Energy Physics

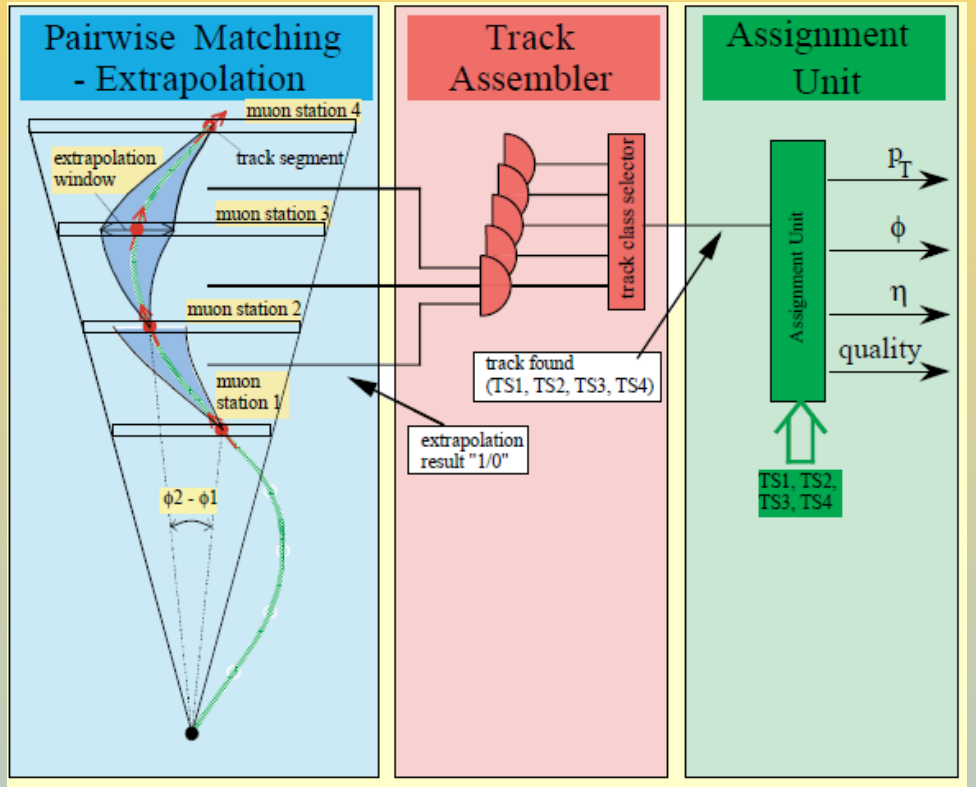
Outline

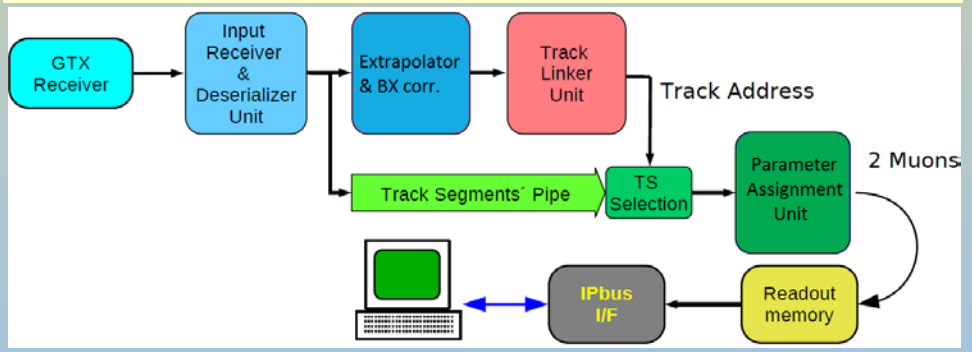
- Drift Tube Track Finder overview
- Trigger Inputs needs
- Optical links
  - Synchronous to the LHC clock
  - Self-synchronous to the LHC clock
  - Asynchonous to the LHC clock
- Asynchronous Protocol
- Results
- Future plans

Drift Tube Track Finder overview

Vienna HEPHY | University of Athens | University of Ioannina

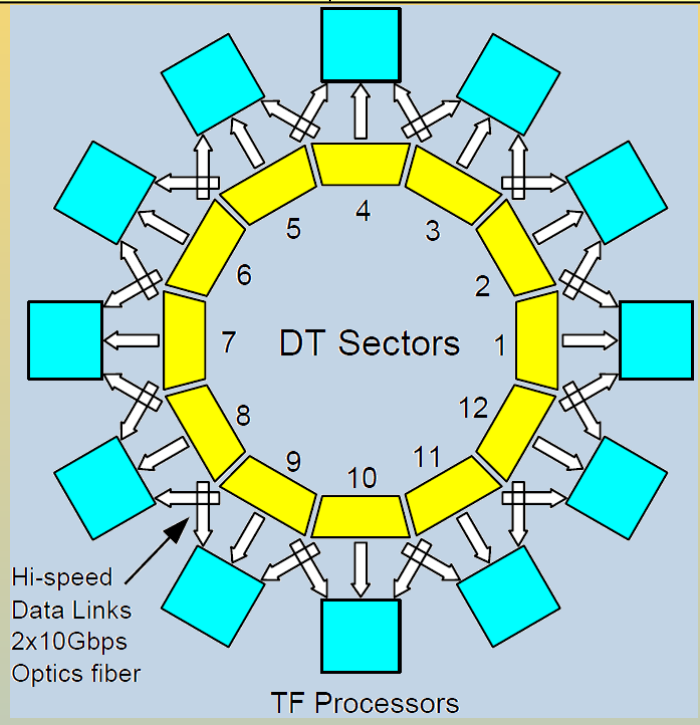
- At the CMS TDR for the L1 trigger upgrade (1.Aug.2013) have been decided that the DTTF will be upgraded as follows
  - 12 VME Crates will be replace by 3 uTCA
  - Old trigger cards will be replaced by new which hosts the state of art of Xilinx FPGAs (Imperial MP7)
  - The input bandwidth will be increased by **6** from 1.6Gb/s of GOL links to 9.6Gb/s
  - At the first state of the DTTF upgrade until the and of 2015 the same algorithms will run in the new FPGA framework
  - At the second state RPC hits will be included in the track finding algorithms





Trigger input needs

- Drift Tube trigger data will run on 9.6Gb/s (192bits@40MHz).
- Hence two DT links are enough for one Sector (384 bits).
- Hence 10 DT links will be used for one full wedge
- The new sector collector (TwinMux) will fan-out data from neighbor wedges
- Hence every DTTF processor will receive data from 3 wedges (30 DT links at 9.6Gb/s)
- After 2015 RPC data will be also used by 23 links at 9.6Gb/s

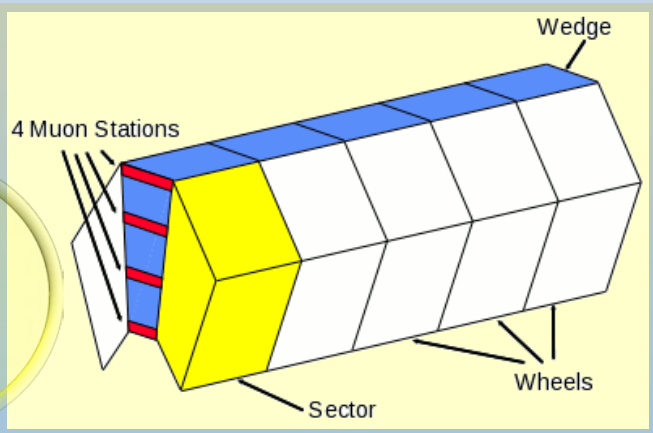


CMS L1 TDR 2000 - Drift Tube data

Data	Width (bits)	Type & comments
ϕ_r up track	12	signed, 1-compl.
ϕ_b up track	10	signed, 1-compl.
Quality up track	3	see table 3.4
$1^{st}/2^{nd}$ up track	1	
ϕ_r down track	12	signed, 1-compl.
ϕ_b down track	10	signed, 1-compl.
Quality down track	3	see table 3.4
$1^{st}/2^{nd}$ down track	1	
θ triggers	8	
θ quality	8	H/L for each trigger bit
Bunch Crossing 0	3	From up, down and θ
Bunch Crossing count	2	2 LSbits of the bunch counter
Parity ϕ data	2	Up and down
CCB info	4	Minicrate Control Board status
Trigger output	1	Chamber autotrigger

80 bits for one Station

320 bits for one Sector



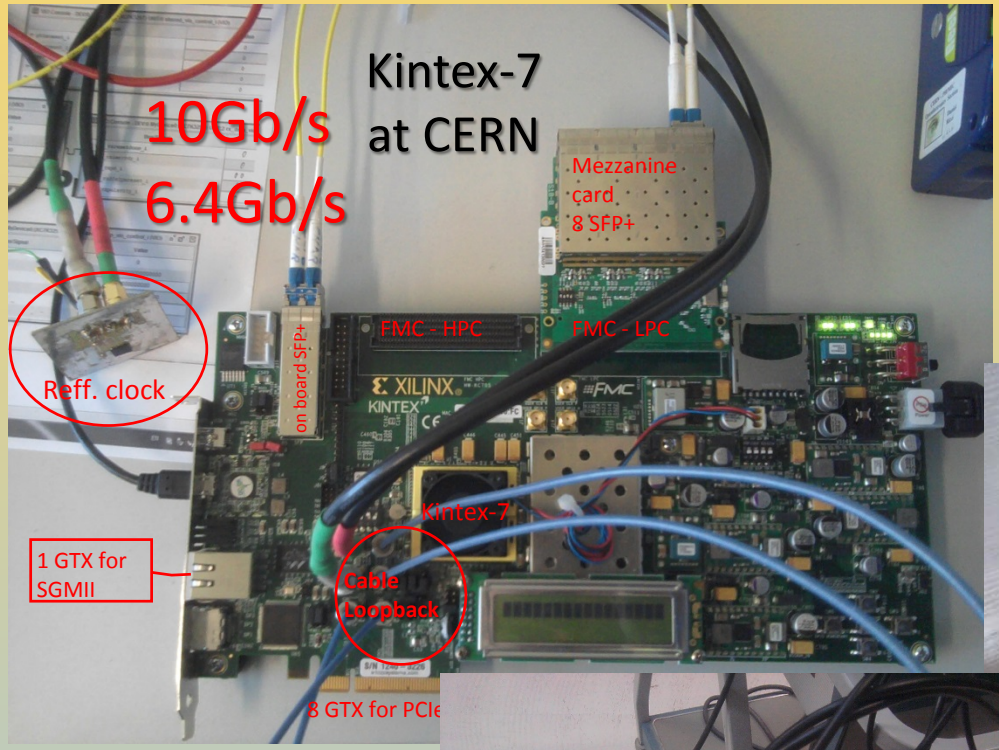
Optical links

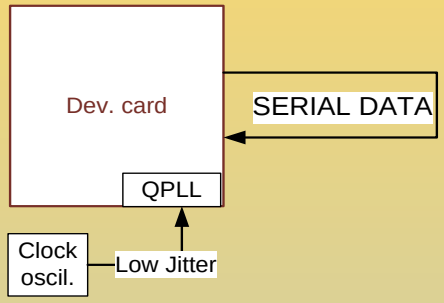
Some of the most suitable rates in CERN experiments

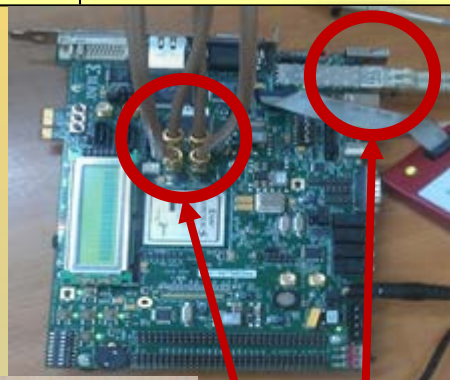
- Synchronous – all suitable with the 40 MHz
  - 1.6Gb/s(GOL), 3.2Gb/s(Spartan6-Virtex5), 4.8Gb/s(Virtex6), 6.4Gb/s(all 7series), 9.6Gb/s(only Virtex7 with GTH), 11.2Gb/s(7series with speed grade 3)
- Self-synchronous – clock domains bad for the 40 MHz
  - 3.125Gb/s(Spartan6-Virtex5), 5Gb/s(Virtex6), 8Gb/s and 10Gb/s (All 7series FPGAs) 12Gb/s(7series with speed grade 3)
- Asynchronous (channel rate with a lower payload)
  - In the past rates of ~ 2 Gb/s have been used in CMS
  - 10Gb/s with a payload of 9.6Gb/s (All 7series FPGAs) Done!
  - 12Gb/s with a payload of 11.2Gb/s (7series with speed grade 3)

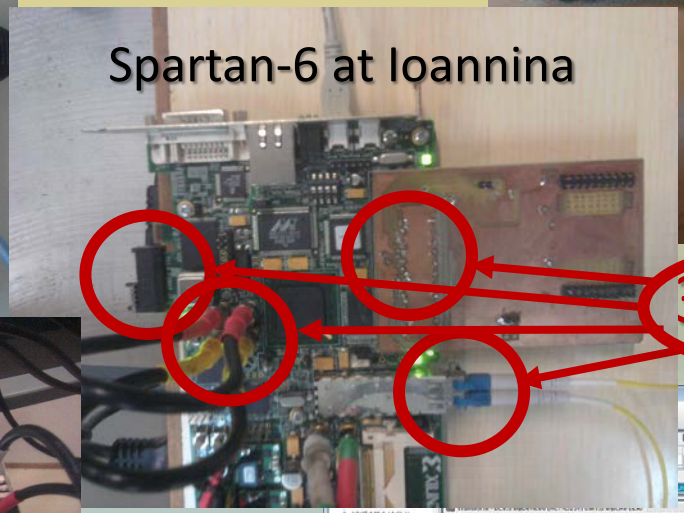
Loopback tests

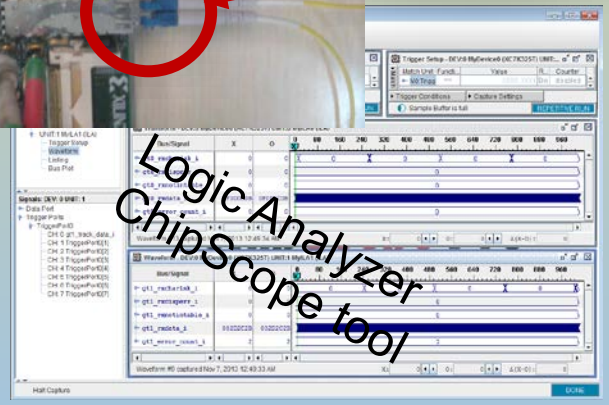
Vienna HEPHY | University of Athens | University of Ioannina









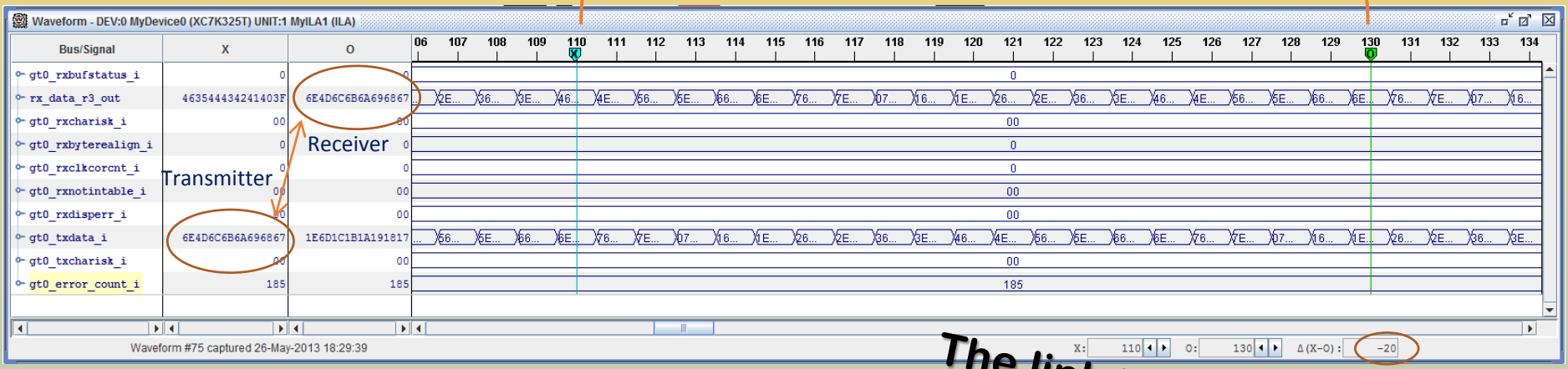


Loopback tests for Kintex7 (6.4Gb/s, 64 bit width on 80MHz)

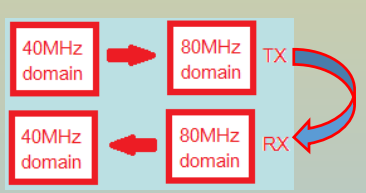
Vienna HEPHY | University of Athens | University of Ioannina

64 bit bus @ 80 MHz

Channel latency 20 clock cycles 12.5ns (250ns)

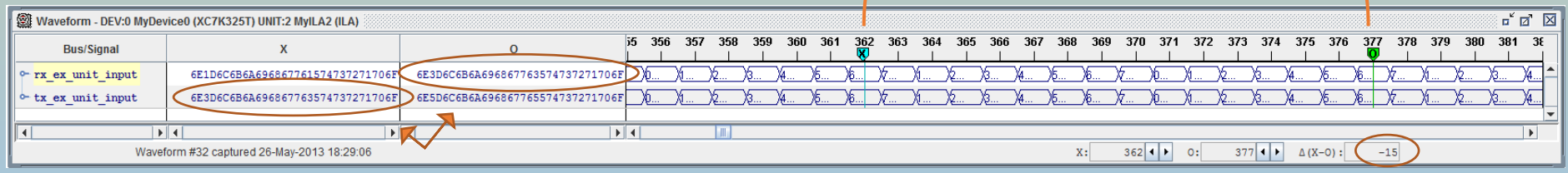


The link is stable, no error.



128 bit bus @ 40 MHz

Total Channel latency 15 bx (375ns)

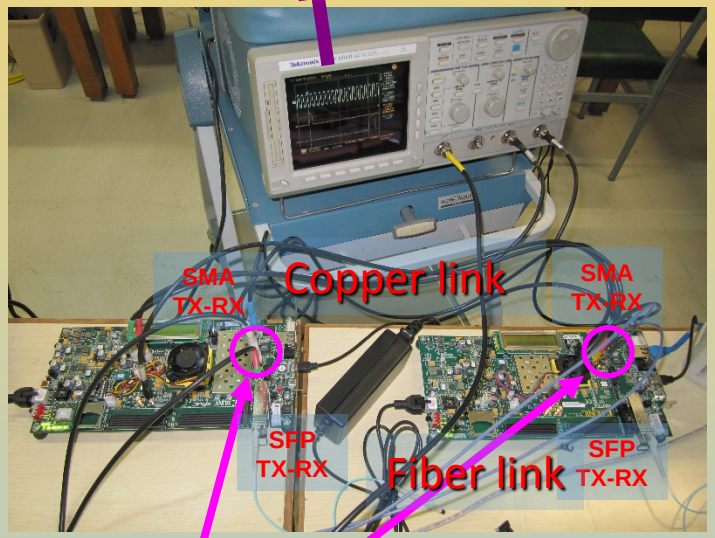


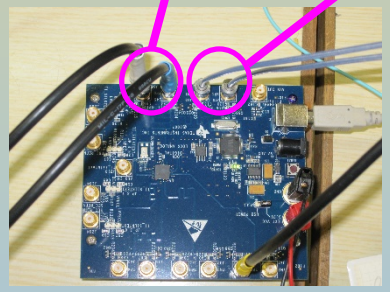
Transmitted word Received word

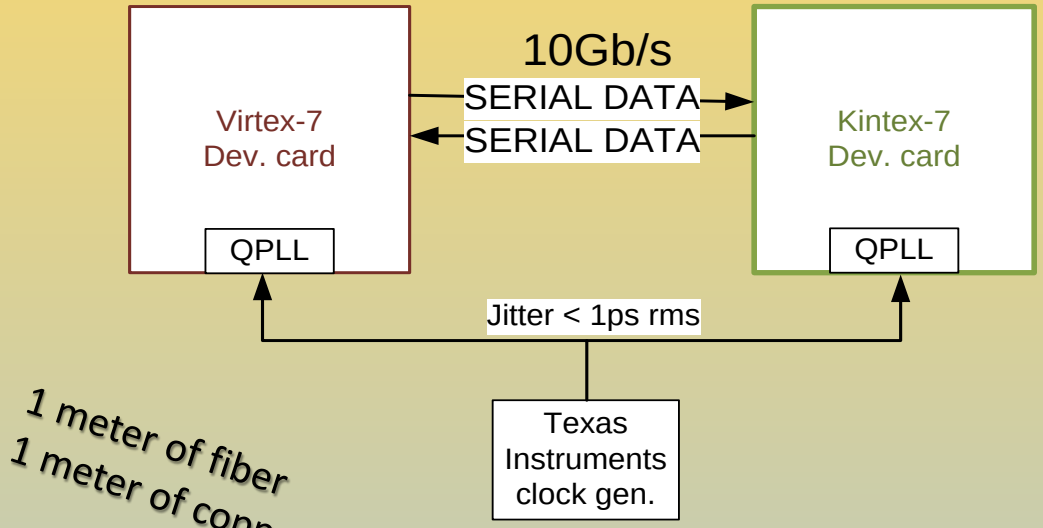
Synchronous links

Vienna HEPHY | University of Athens | University of Ioannina

Total FPGA to FPGA latency of 32bits 72 ns (< 3bx)







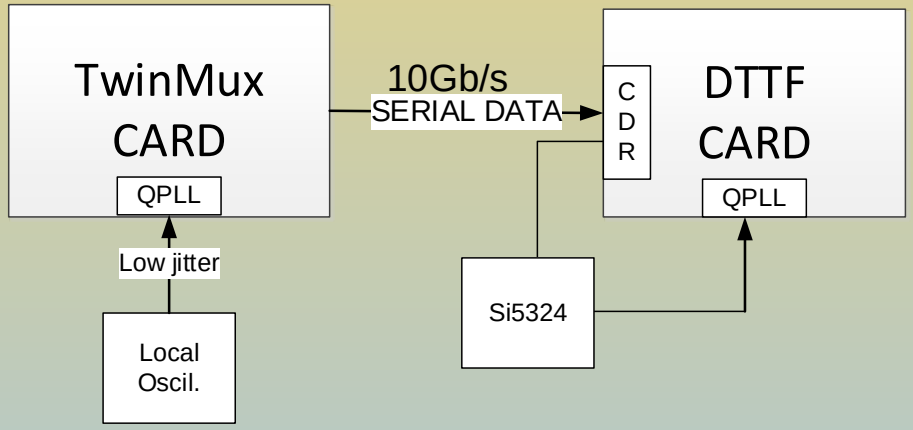
1 meter of fiber
1 meter of copper

- Implementation of 10 Gb/s links for Kintex-7 and Virtex-7 FPGAs
- Tx, Rx buffers bypassed
- One common clock source
- 32bit bus at 250 MHz
- 2 Days running, no data error

- Synchronous links at 6.4Gb/s have been used in order to test DTTF algorithms last year
- 32bit bus at 160MHz
- 128bit bus at 40MHz
- PHTF need 110 bits

Self - synchronous links

- We cannot just use one oscillator in each card because they wont have exactly the same frequency (best case ±50ppm).
- The LHC clock is not always stable to drive transceivers running at 10Gb/s.
- To avoid losing the links we need to drive the QPLLs of the GTX with very good Jitter Performance clocks (about ~1ps RMS)
- There is a need for a common clock source.

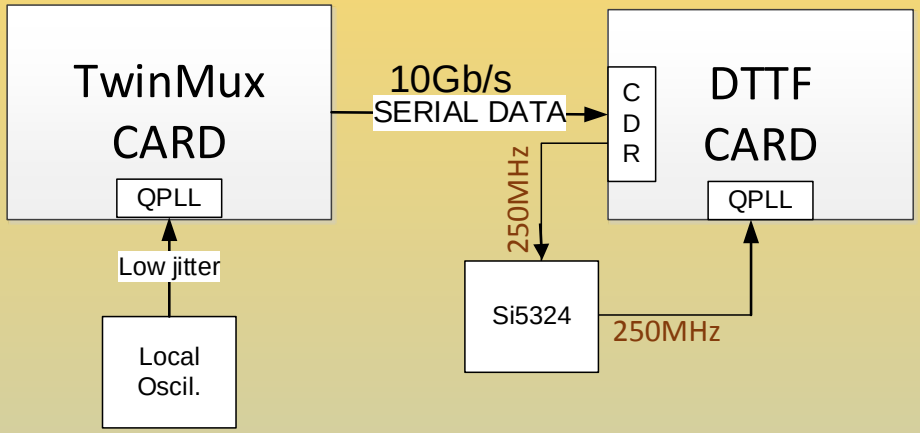


Running for a weekend
NO ERROR

- In self - synchronous communication we are using a local oscillator which drives the Transmitter of the first board.
- The receiver of the second board recovers the clock from serial data.
- The clock has to be cleaned in order to driver the QPLLs.

Self - synchronous links

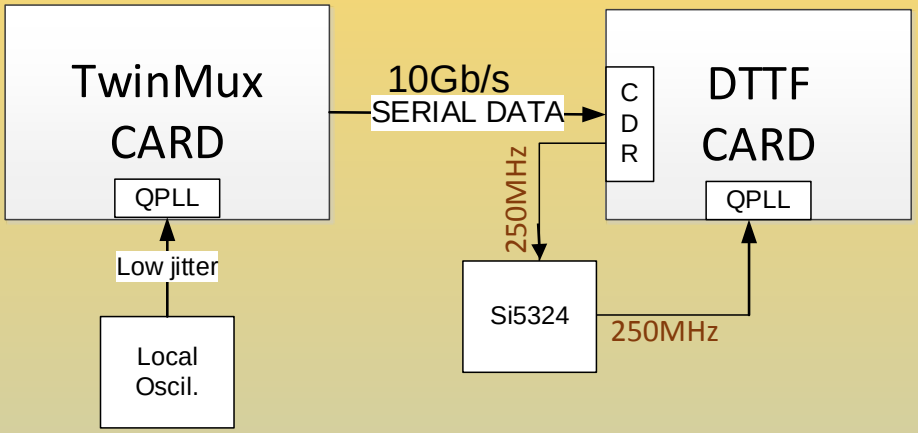
- The receiver starts from a local oscillator and when detects a CDR lock switch to the recovered clock.
- The clock from CDR goes to a jitter cleaner module. The Si5324 is fully configurable by a vhdl package.
- A bash script generates code with registers values to be written to the Jitter cleaner device.
- An I2C interface does the rest.
- The recovered and cleaned clock can drive 12 receivers without errors
- As a result the receiver FPGA is synchronous the transmitter.



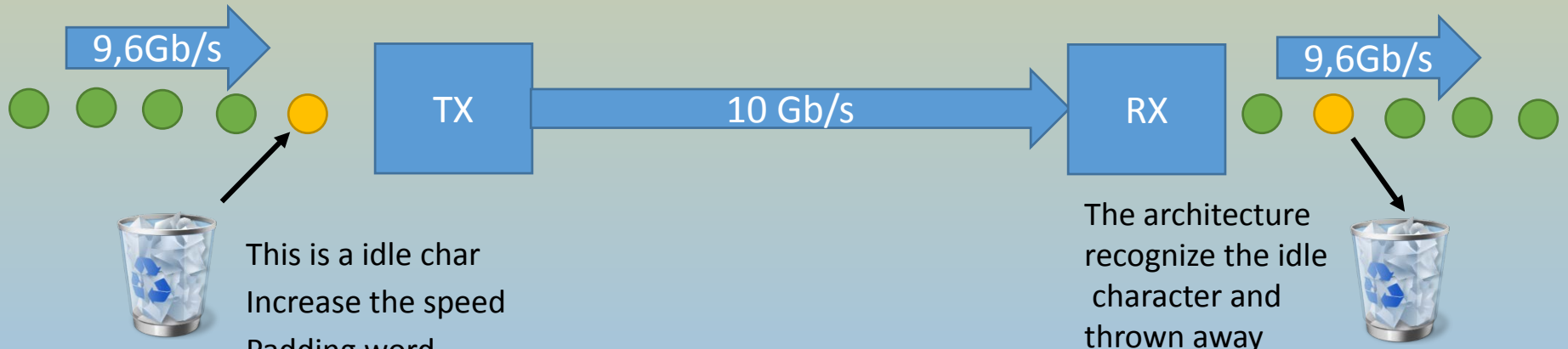

```
-- This Package declares the values registers to be written to the Si5324 jitter cleaner
--
-- This file has been AUTOGENERATED from the package_builder.sh (on Linux) - do not hand edit
--
-- First you have to use DSPLLsim tool inserting the preferred clock configuration.
-- Then save the Register Map File (txt) and copy it to the this directory.
-- Run the script. Done. Now you have the package. Just import it the your ISE project.
--
-- Nikitas.LOUKAS@cern.ch, November 2013

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

-----Package with functions:-----------------------------------------------
package register_pkg is

   constant fsm_cycles : integer;
   function reg_data(signal j : integer range 0 to (fsm_cycles-1)) return std_logic_vector;
   function reg_addr(signal j : integer range 0 to (fsm_cycles-1)) return std_logic_vector;

end register_pkg;
-------------------------------------------------------------------------------

package body register_pkg is

   constant fsm_cycles : integer := 43;

   function reg_data(signal j : integer range 0 to (fsm_cycles-1)) return std_logic_vector is
      variable byte : std_logic_vector(7 downto 0);
   begin
      case j is
         when 0      => byte := X"14";
         when 1      => byte := X"E4";
         when 2      => byte := X"A2";
```

Asynchronous links

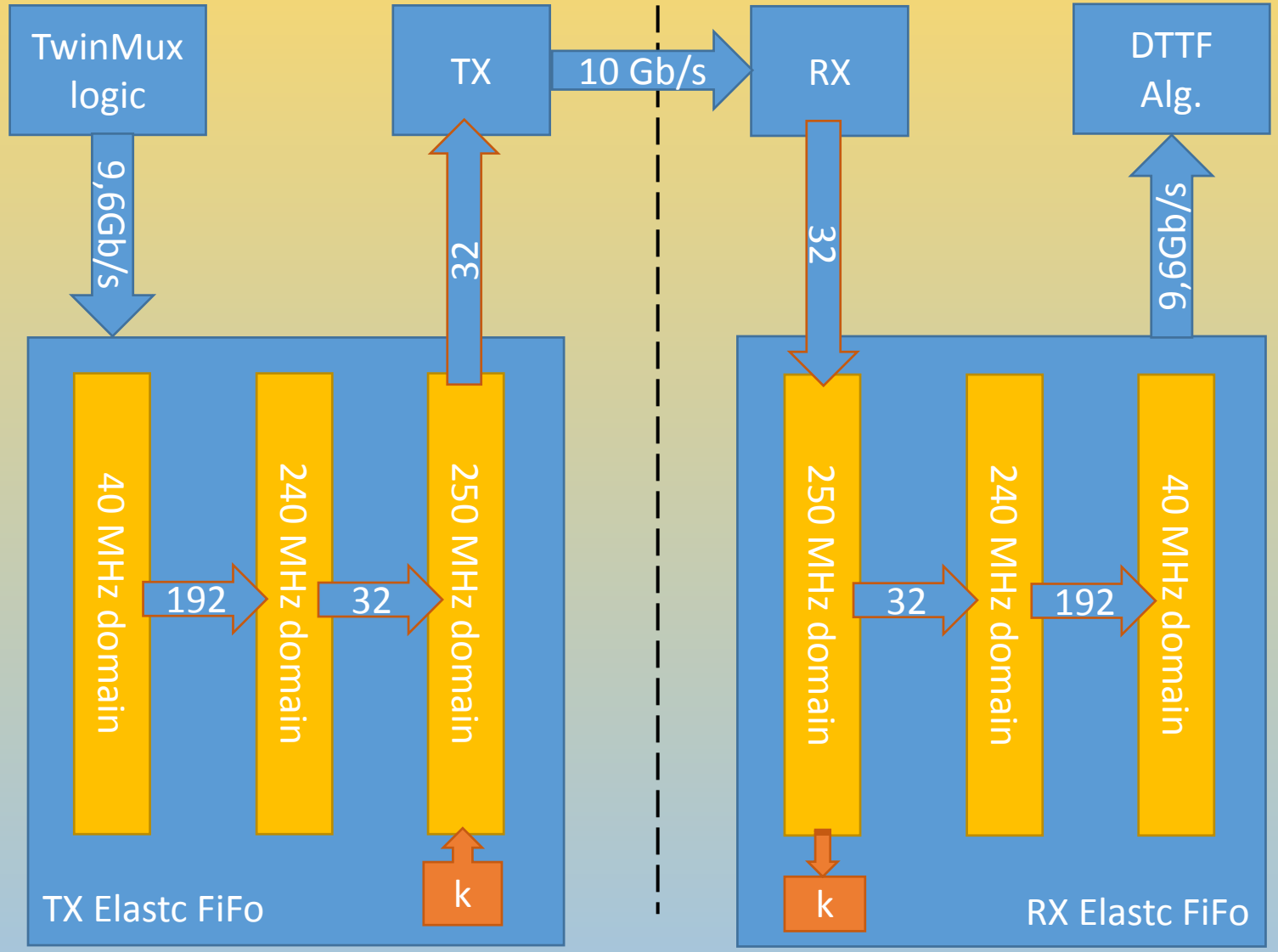
- In asynchronous communication the GTX and fiber runs in a differenced speed than the rest FPGA logic.
- For instance the GTX runs at 10Gb/s while the FPGA is processing at 9.6Gb/s.
- As in case of the self-synchronous the receiver use the CDR to get synchronized with the transmitter
- Every FPGA use additional logic to merge deferent clock domains



- RX and TX elastic FIFOs are used and they implement a padding method



9.6Gb/s Asynchronous Protocol

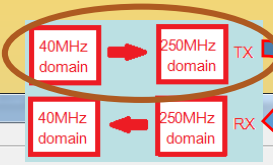


Simulation results

Vienna HEPHY | University of Athens | University of Ioannina

<2 bx time

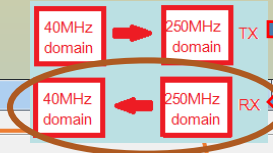
This word is padding

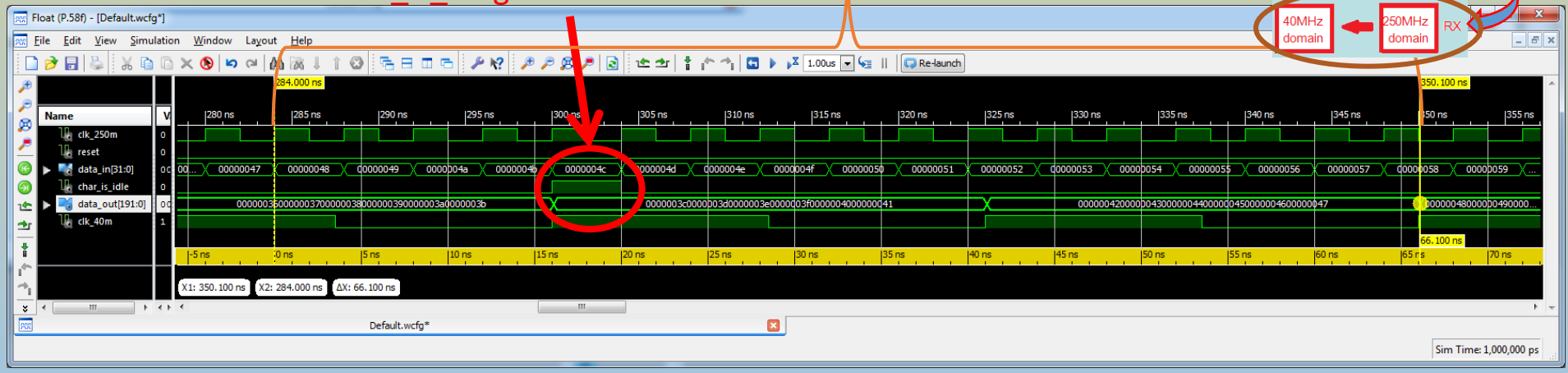


A similar code for rates of ~2Gb/s was kindly given by Greg Iles

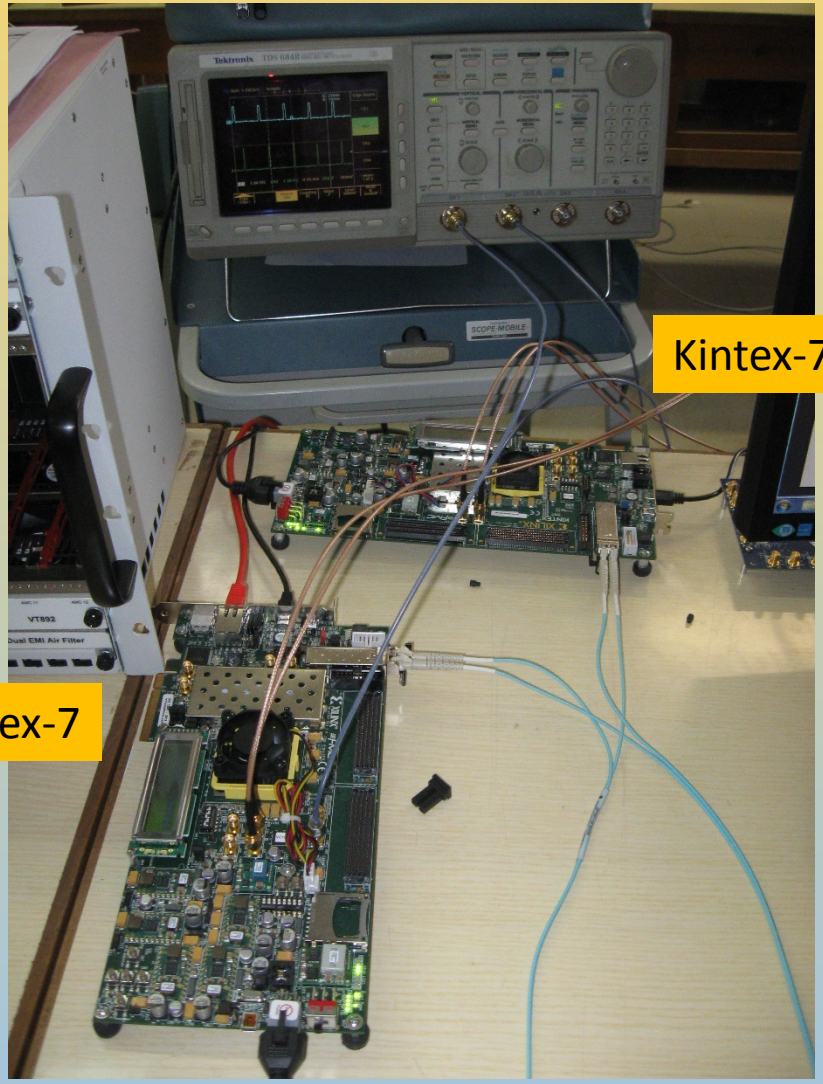
<3 bx time

Char_is_k signal from the receiver





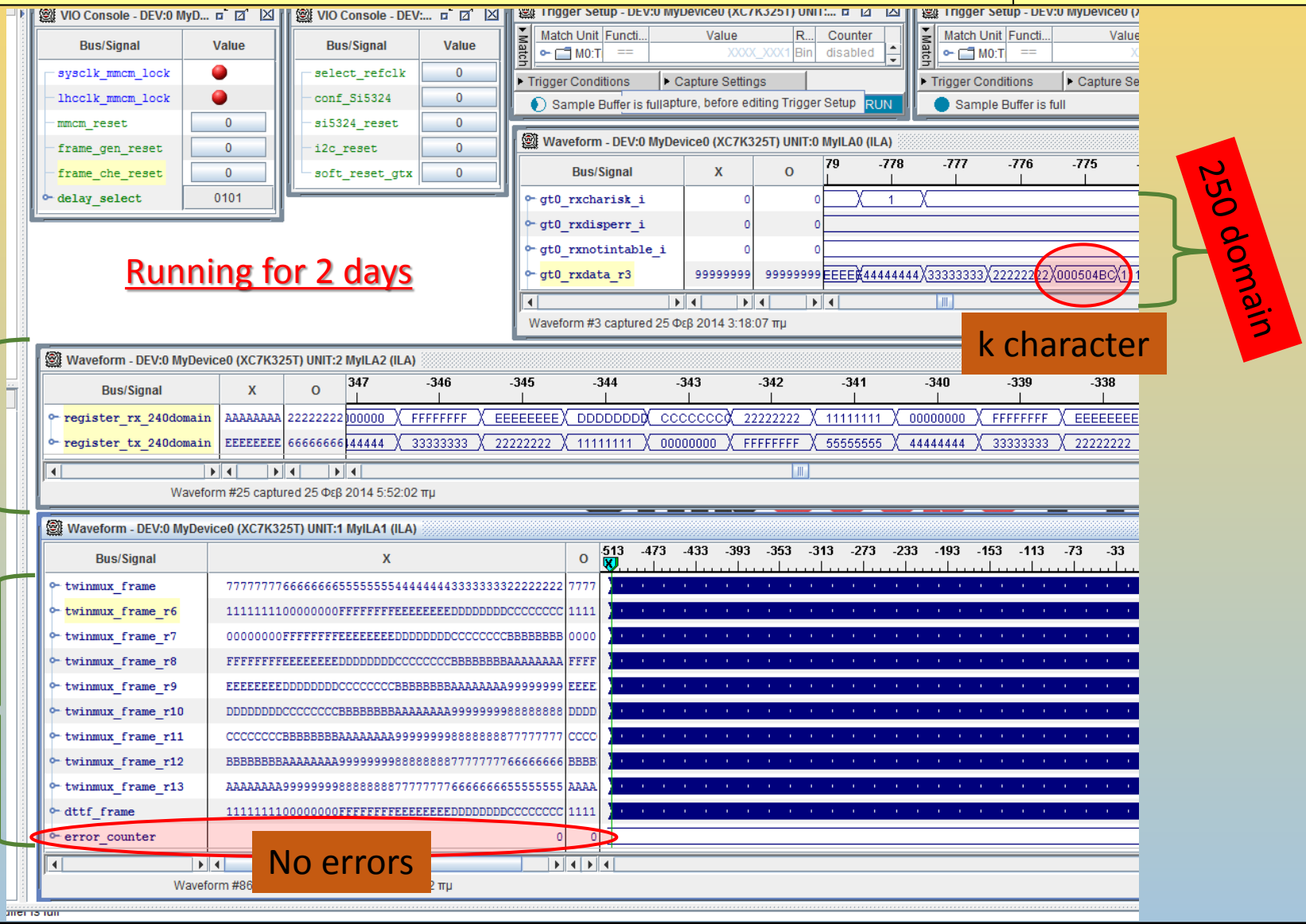
Real world – The setup



- Both the cards send and receives one optical link at 9,6Gb/s
- They use the on board oscillators to drive the GTX and a common 40 MHz source (represent the LHC clock)
- The design use on board jitter cleaner which “clean” the recovered clock
- IPbus has been implemented to spy data that have been received
- Oscilloscope shows commas that have been received

Testing the Async- links

Vienna HEPHY | University of Athens | University of Ioannina



How the logic works
Why to choose asynchronous links

Vienna HEPHY | University of Athens | University of Ioannina

All the complexity is in the TX side

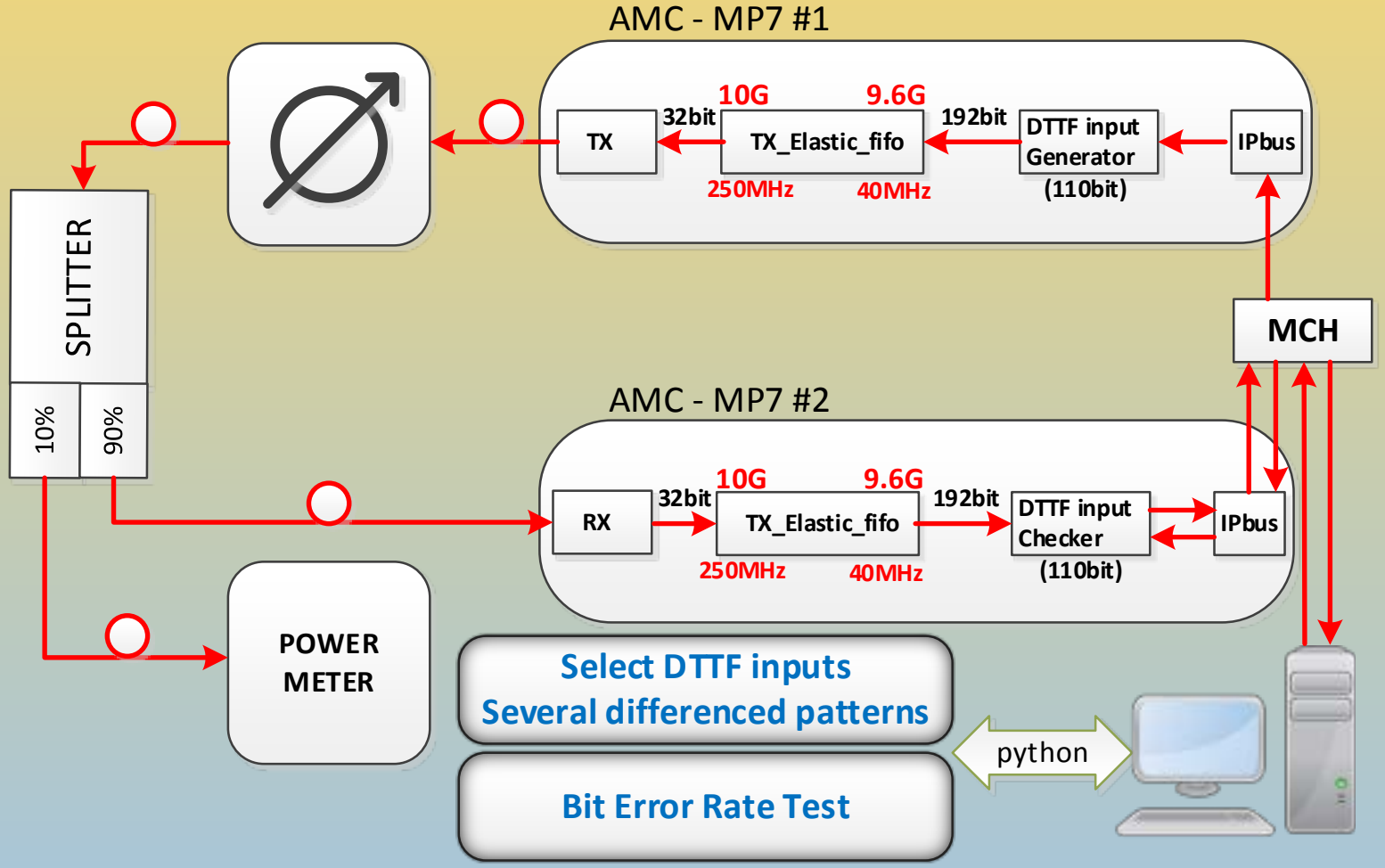
9.6Gb/s-->10Gb/s		
	240MHz	250MHz
1	4,2	4,0
2	8,3	8,0
3	12,5	12,0
4	16,7	16,0
5	20,8	20,0
6	25,0	24,0
7	29,2	28,0
8	33,3	32,0
9	37,5	36,0
10	41,7	40,0
11	45,8	44,0
12	50,0	48,0
13	54,2	52,0
14	58,3	56,0
15	62,5	60,0
16	66,7	64,0
17	70,8	68,0
18	75,0	72,0
19	79,2	76,0
20	83,3	80,0
21	87,5	84,0
22	91,7	88,0
23	95,8	92,0
24	100,0	96,0
25	PAD	100,0

- Asynchronous links has the advantage that it will keep running even if the LHC clock completely collapse
- To avoid delays of startup GTX after TTC problems
- The idle characters are used for padding and contain K characters
- The disadvantage is that the elastic FIFOs need to use additional resources
- Latency has to be checked!

Additional example
Asynchronous for 7series
with speed grade 3

11.2Gb/s-->12Gb/s		
	280MHz	300MHz
1	3,6	3,3
2	7,1	6,7
3	10,7	10,0
4	14,3	13,3
5	17,9	16,7
6	21,4	20,0
7	25,0	23,3
8	28,6	26,7
9	32,1	30,0
10	35,7	33,3
11	39,3	36,7
12	42,9	40,0
13	46,4	43,3
14	50,0	46,7
15	PAD	50,0

Future plans



The end

I thank Greg Iles for the help
given during the past year