tifr
tata institute of
fundamental research

PCIe-IPBus Phase-2 Control Infrastructure Development

S. Dugad, K. Harder, G. Iles, D. Newbold, A. Rose, D. Sankey,

I. Mirza, R. Shukla, A. Thea, T. Williams

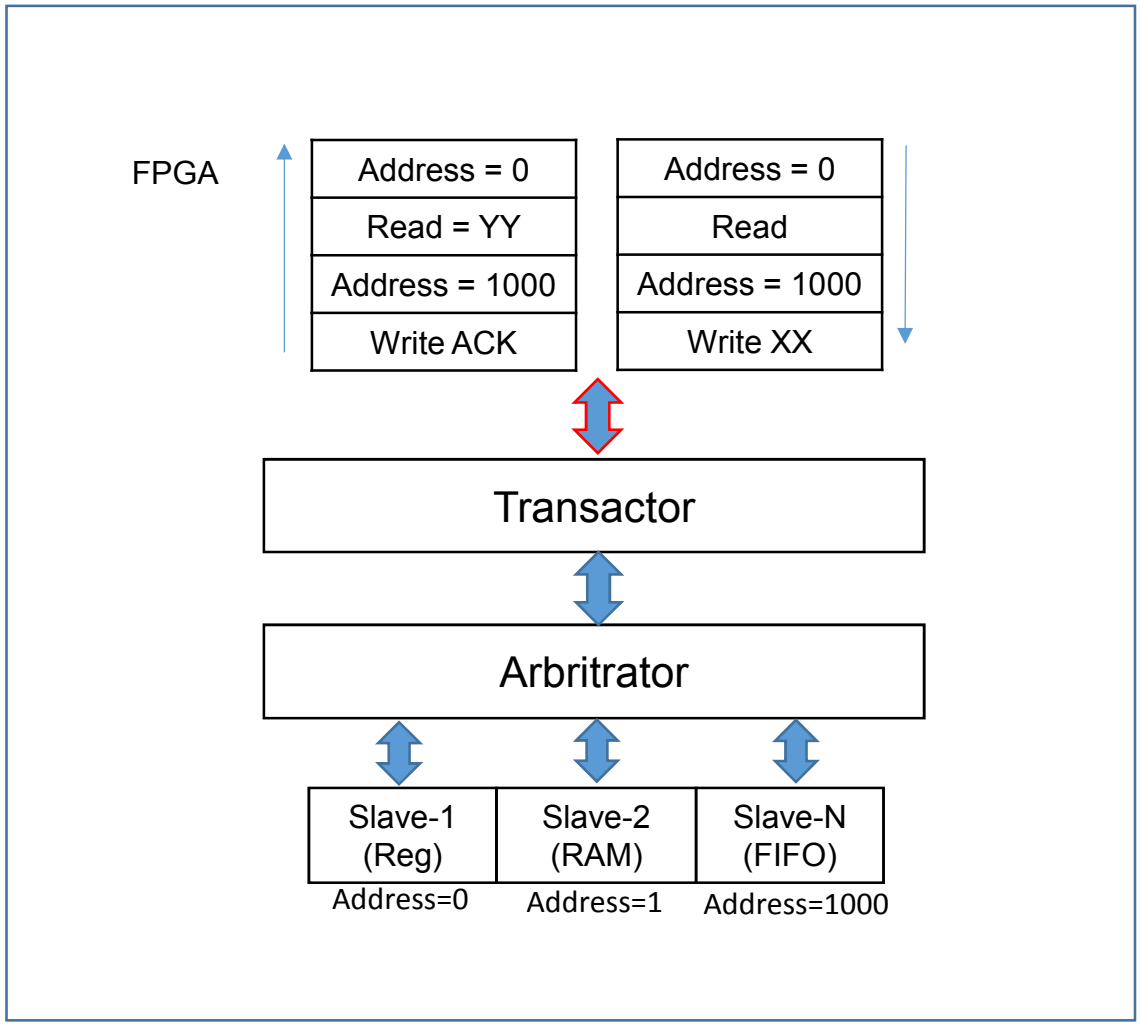
Overview

- Implementation of PCIe-IPbus framework on various FPGA development boards and its performance verification for the CMS phse-2 control Infrastructure development
- Design and development of the ultra high speed daughter board (Xilinx Ultrascare+ FPGA) for high performance Serenity Motherboard

IPBus

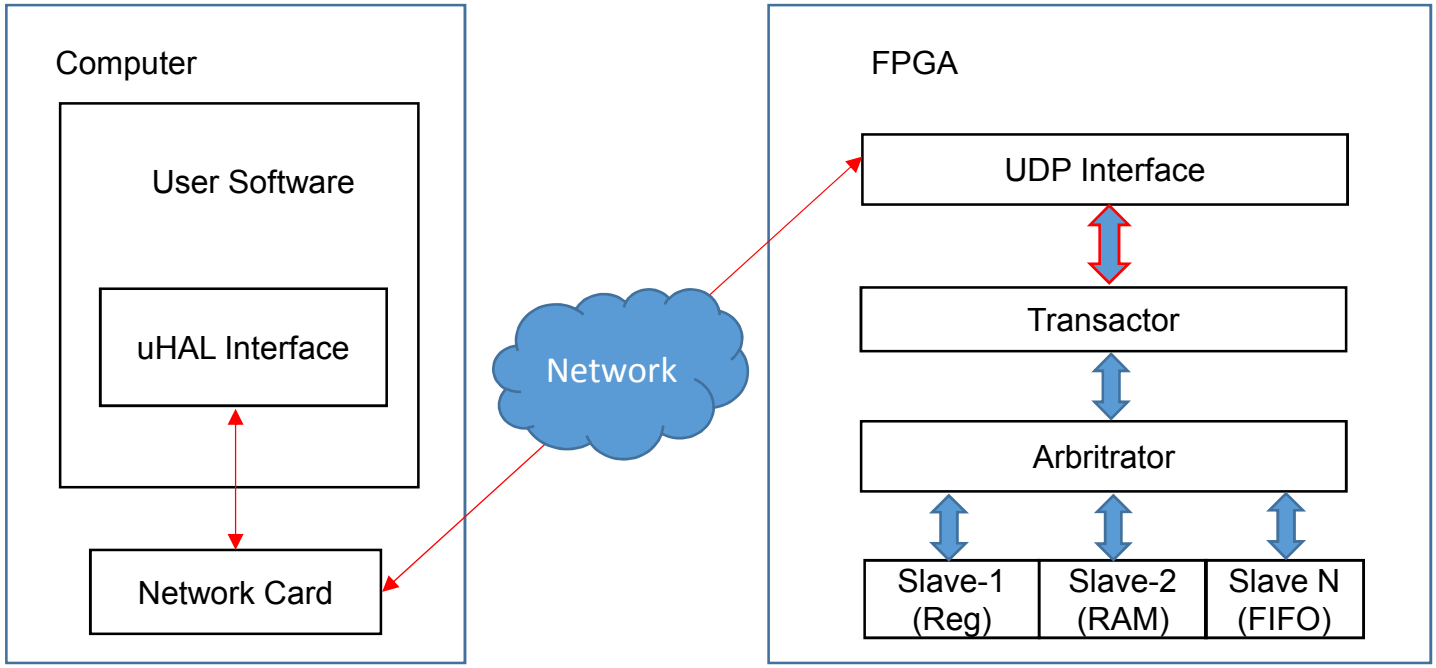
- What is IPBus ?
  - IPbus is a simple, reliable, IP-based protocol for controlling hardware devices.
  - It is a virtual bus (implemented in FPGA) based on 32-bit address/data lines (A32/D32) and control signals (similar to legacy RAM interface)
  - Data transactions are carried out between Master and Slave with predefined simple protocol
  - IPbus master (*the transactor*) carries out transaction with slaves as per IPBus packet submitted to it via external interfaces such as UDP
- Firmware + Software framework
  - IPBus core firmware (*transactor*), arbitrator and large number of example slaves such as Registers, RAM, FIFO are part of the firmware package
  - uHAL (uTCA Hardware Access Libraries) software package provides easy to use C++/Python APIs to access IPBus slaves
- Used is phase-1 CMS upgrades as well as outside CMS (ATLAS, ALICE etc.)

IPBus in Action



- Scatter all request
- Gather all responses

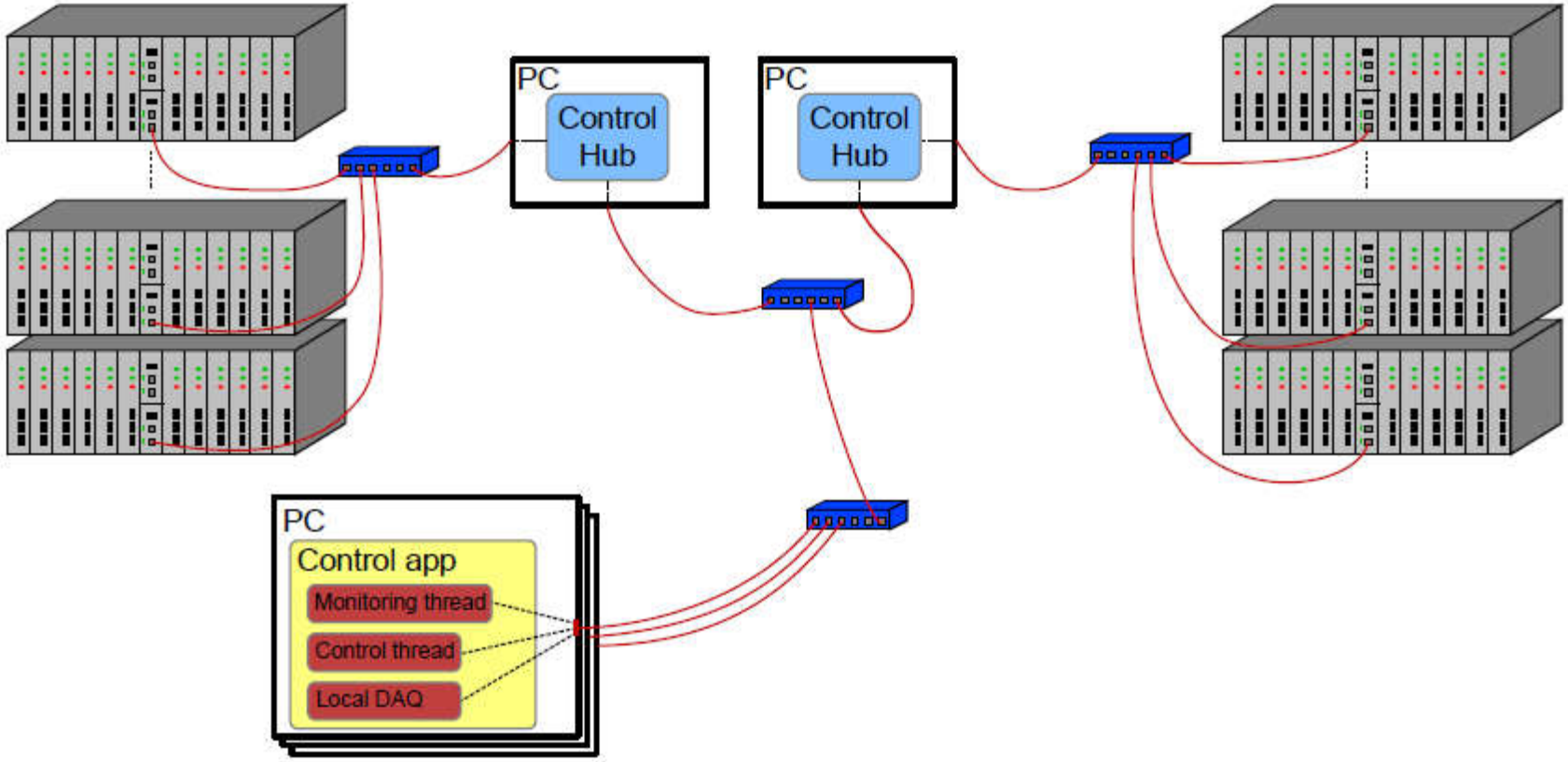
IPBus Architecture



- Arbitrator supports multiple masters, thus debug interfaces such as UART can be introduced
- IPbus packet protocol combines transactions for multiple slaves into single IPBus packet and hence provides high throughput

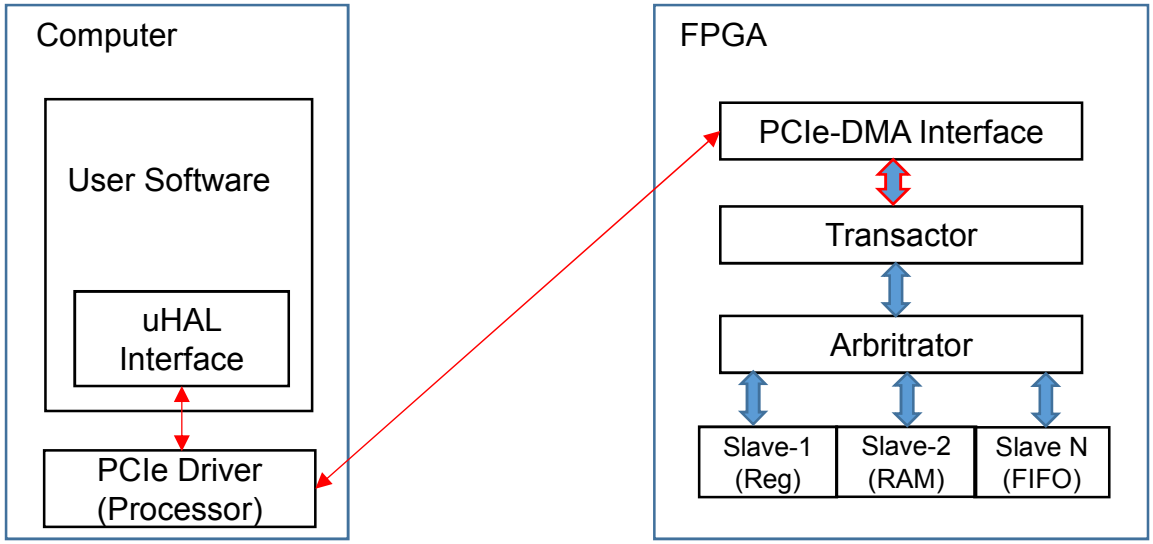
Ethernet based IPbus system

CMS phase-1



Towards PCIe

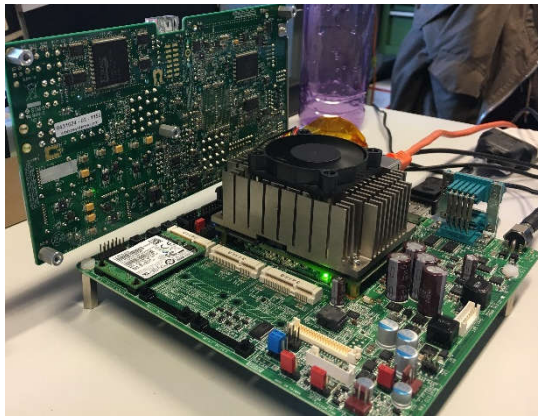
- With CMS upgrading to high speed uTCA/ATCA based hardware, dedicated high speed control system has become necessary
- uTCA/ATCA provides dedicated PCIe fabric on the backplane and thus it has become a natural choice for the upgrade
- The PCIe-IPbus framework is being upgraded to **add** PCIe support without any user level changes
- PCIe offers much larger transfer sizes and high transfer rates (Gen3: 8 Gbps/lane)



* DMA = Direct Memory Access

PCIe Implementation

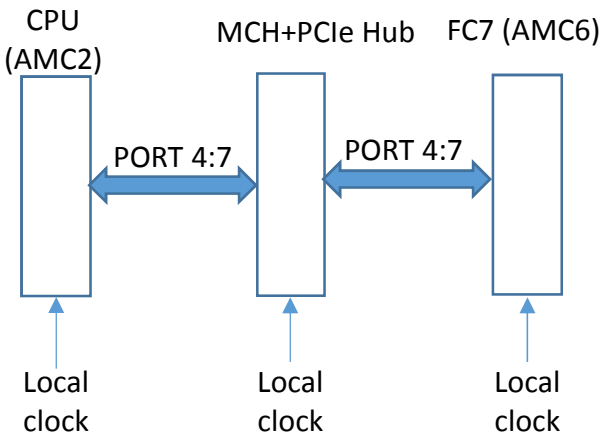
- We started with Spartan-6 FPGA Dev Board (PCIe Gen1 x1)
- ATCA service card like system demonstrated with Com-Express Mini



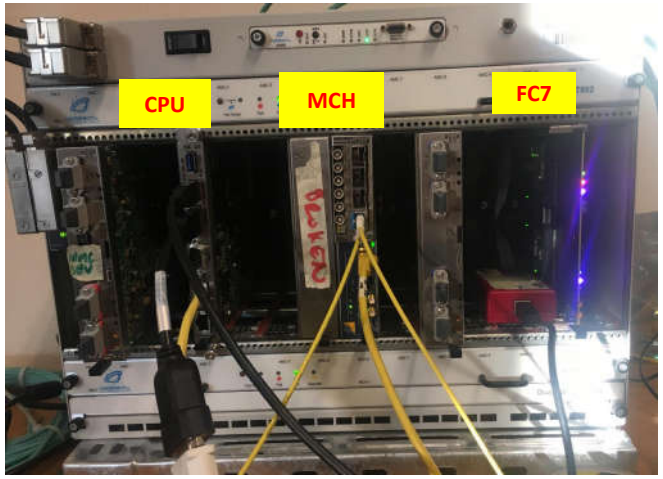
- Firmware ported to Kintex-7 FPGA with Xilinx Dev Boards (PCIe Gen 2 x1).
- Further porting to FC7 (a flexible, uTCA compatible card for generic CMS data acquisition built around Kintex7) to support CMS-uTCA backplane with asynchronous clock

PCIe-IPBus with FC7 and uTCA

Experimental setup at CERN



µTCA



FC7

- The PCIe link has been successfully established (Gen 2 x1) over uTCA backplane using asynchronous clocks (due to absence of FCLKA)
- The PCIe uses our own flavor of DMA engines which are capable of handling staggered TLPs from root complex.
- Linux kernel driver developed completely in-house
- IPbus transactor has been integrated with the DMA engines, which allows fast access to IPbus clients

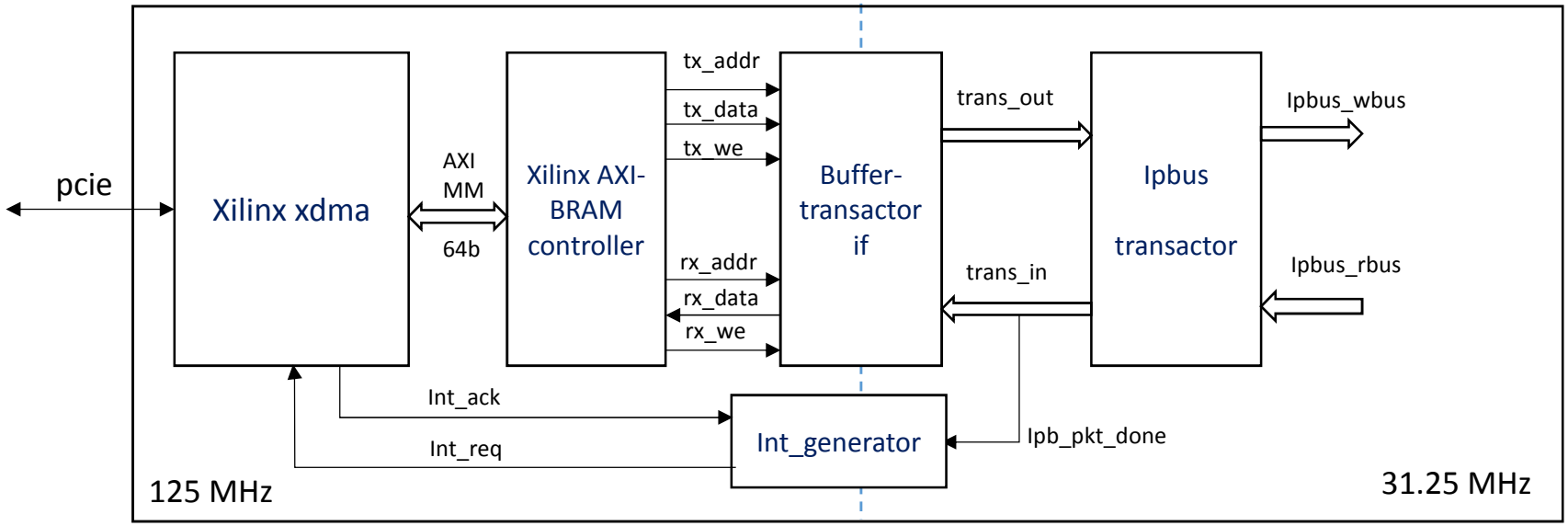
PCIe-IPBus with FC7 and uTCA

- CPU installed in uTCA crate hosts the uHAL
  - uHAL framework has been ported to support PCIe medium natively with necessary user-space bridge program between PCIe kernel driver and uHAL framework
- System tested at CERN

PCIe-IPbus on Xilinx UltraScale

- Xilinx UltraScale and UltraScale+ devices have been chosen as main work horse for demanding upgrades, including HGCAL
- Thus the PCIe-IPbus framework has been extended to support these devices
- The newly introduced Xilinx XDMA SG-DMA syb-system and drivers have been used to provide PCIe interface to FPGA
- uHAL client has been redesigned (currently single threaded) to support new DMA system
- Being used actively for further hardware infrastructure development:
  - Buffers (Algorithm block)
  - Links (DAQ)
  - TTC
  - ..

Overview of the Firmware

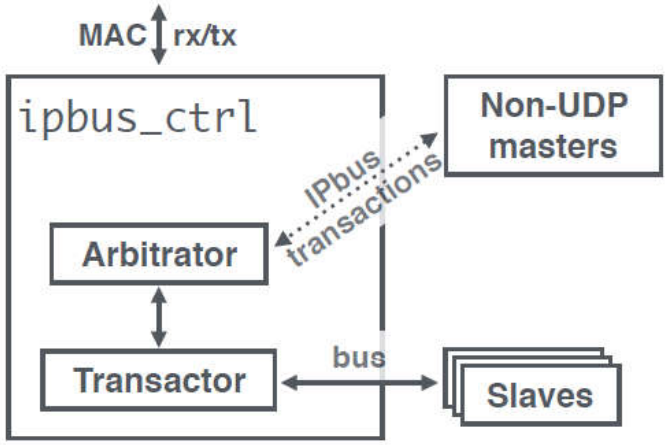
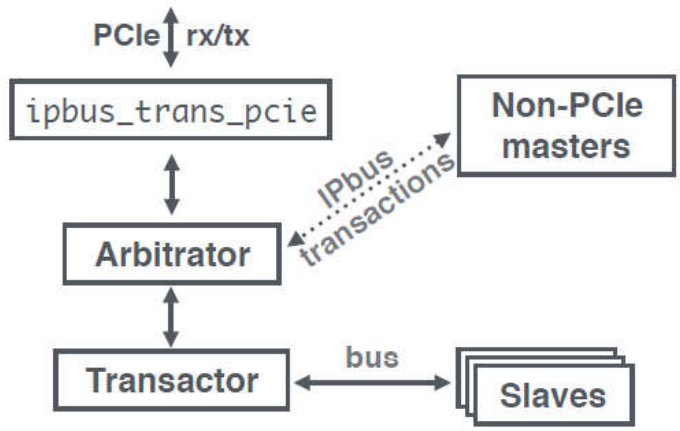


- Earlier implementation used MSI interrupts for DMA engine interrupts and polling mechanism for IPBus transaction completion indication. However lots of stability problem
  - Inefficient ack mechanism and driver implementation
  - Driver reads back system registers (PIO) to determine interrupt source
  - Many times interrupt got lost !
- Newer version implements efficient MSI-X interrupts for DMA engins as well as to indicate IPBus transaction completion (performance improvement > 100 %). Backward compatibility to MSI (may be even Legacy) is under consideration.

Ethernet vs PCIe: User point of view

	Ethernet	PCIe

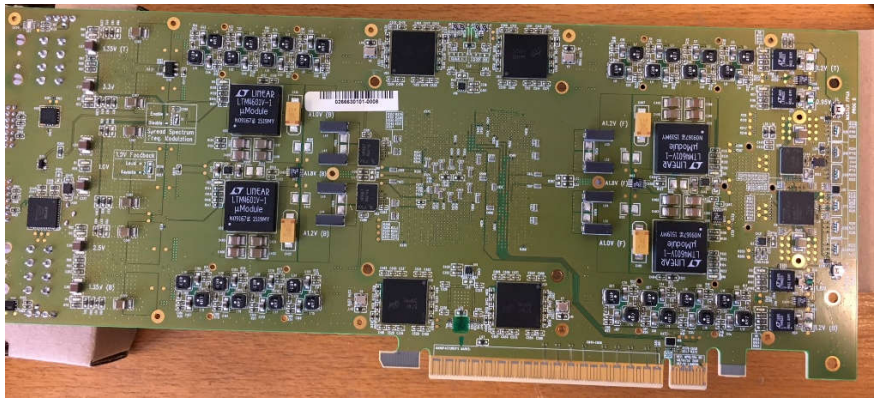
SW	`using namespace uhal;`
	`HwInterface hw("chtcp-2.0://…");`
	
	`auto x = hw.getNode("reg1").read();`
	`auto y = hw.getNode("reg2").read();`
	`hw.dispatch();`
	
	`cout << "reg1 = " << x << endl;`
	`cout << "reg2 = " << y << endl;`
		`using namespace uhal;`
		`HwInterface hw("ipbuspcie-2.0://…");`
		
		`auto x = hw.getNode("reg1").read();`
		`auto y = hw.getNode("reg2").read();`
		`hw.dispatch();`
		
		`cout << "reg1 = " << x << endl;`
		`cout << "reg2 = " << y << endl;`

FW		

Slide courtesy: T. Williams

Testing

- Currently two systems are used for developments and are under test (both KU115 FPGA):
  - MPUltra (CERN): Developed at IC
  - K800 (RAL): Commercial



- Both cards are being tested using standard computer (installed on the motherboard)
- The hardware+ software has underwent standard IPBus soak tests of **more than 500 M transaction !**
- Currently focus is on robustness to release initial version of stable system

Performance

- Performance testing was done using standard IPBus RAM slave and standard testing utility
- IPbus clock ~ 31 MHz =>
  - Fundamental BRAM throughput limit ~ 0.5 Gbps
  - Currently obtained (end-user) throughput ~ 0.4 Gbps !
- Further testing is on-going

Availability

- The uHAL (IPBus-Software) is available at:

  https://github.com/ipbus/ipbus-software
- PCIe-IPbus Firmware available at:

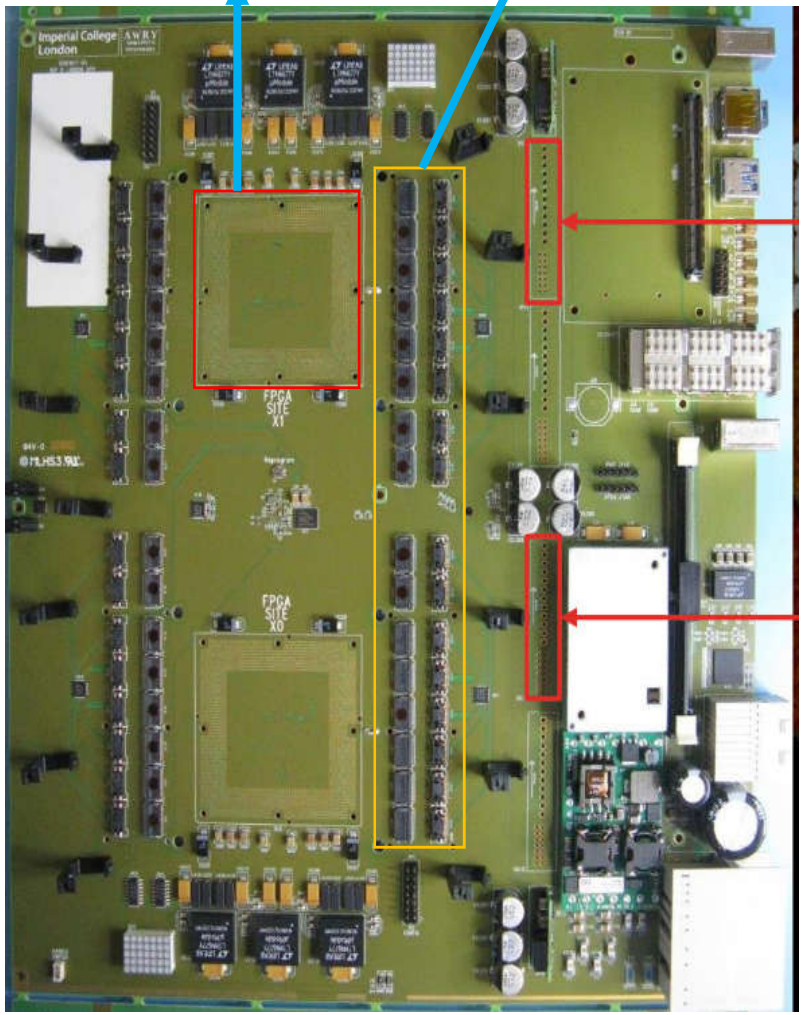
  https://gitlab.cern.ch/p2-xware/phase2-infra-fw
- Documentation:

  https://twiki.cern.ch/twiki/bin/view/CMS/Phase2InfraFw

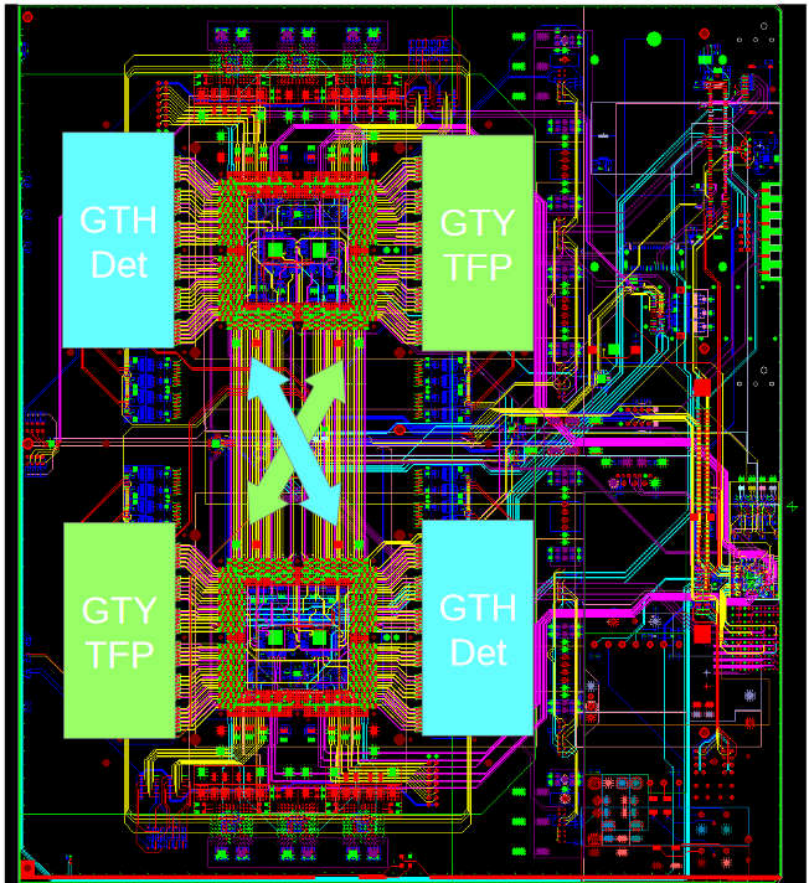
Serenity Motherboard

Interposer site for Daughter Board

Fireflies



Actual PCB

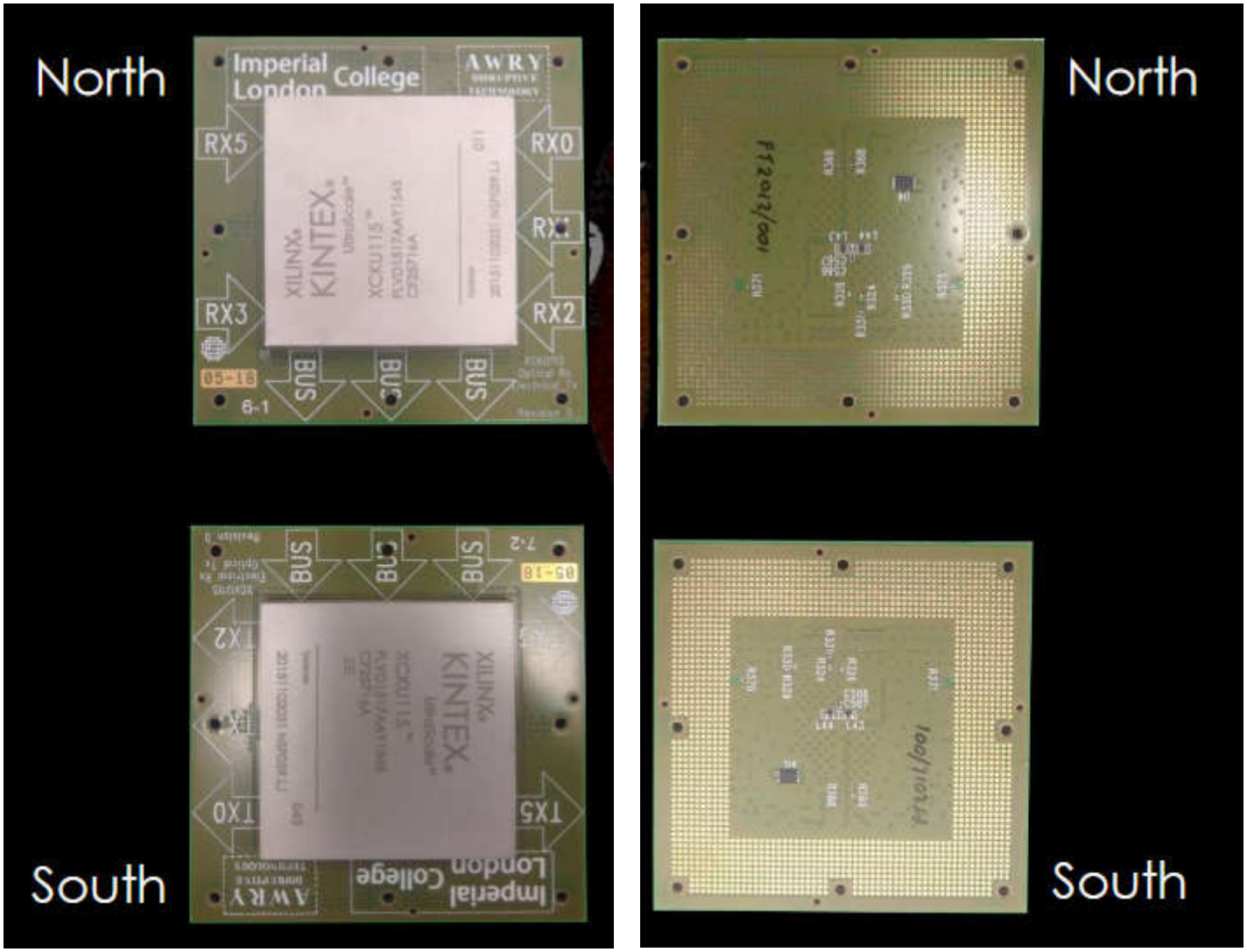


PCB Design

XCVU9P Daughter card design

- Initial study of the PCB (studied KU115 board – next slide):
  - Samtech customized Interposer
  - 16 layer PCB, symmetric layer stack
  - Material: Megatron-6
- Plan is to translate exiting VU9P design to Altium
  - Required Schematic library generated for VU9P board in Altium e.g. FPGA, Interposer etc.
- Further need to study Serenity Daughter Card (DC) sites
  - Detailed study carried out with respect to Serenity Mother board configuration e.g. fire flies placement for various MGT IOs and **Power (and decoupling)**
- Schematics of for VU9P daughter board ready and imported to PCB layout
  - Power (distribution and coupling)
  - Configuration
  - LVDS
  - MGTs

DAUGHTER CARDS: ASSEMBLED, IN-HAND at IC



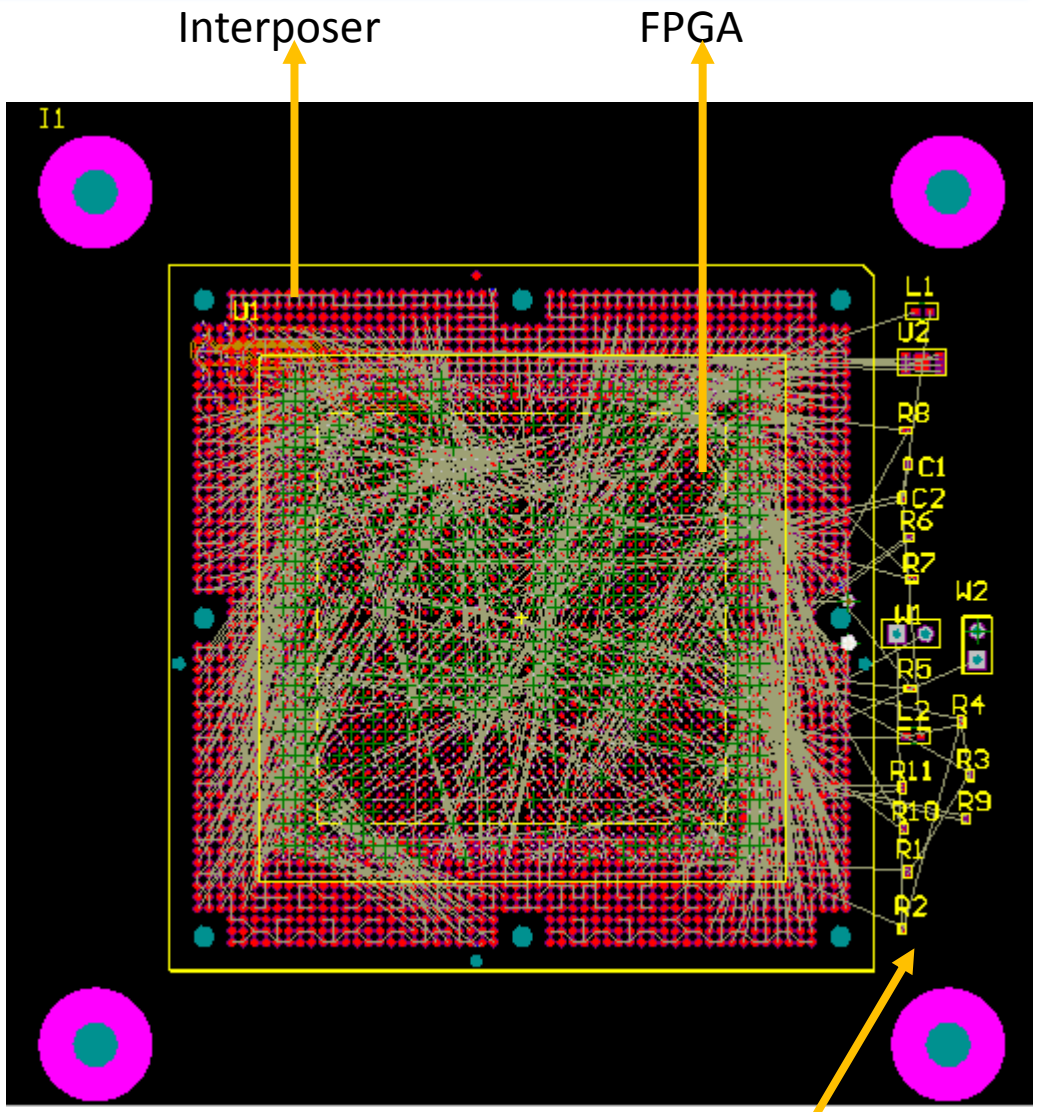
Andy Rose, IC

Top Side

Bottom Side

XCVU9P Daughter card PCB

- Talked to the one of the vendors to check:
  - Material availability and capability
  - Min track width/clearance, via geometry
  - Back drilling
  - Blind via/ micro via/ buried via possibility
  - Cross check controlled impedance calculations



Summery

- Successful implementation of PCIe-IPbus framework carried out various FPGA development boards starting from Spartan 6 to Kintex7 and further testing going on for other high end boards.
- XCVU9P daughter board design study, schematics design carried out and the PCB layout implementation is in progress.

THANK YOU

Backup

Future Roadmap: Multiple Packets in flight

Performance Improvement:

- The IPBus transactor parses the IPBus packets received from uHAL client and performs IPBus transactions, **one packet at a time**
- If multiple user request packets are accommodated in FPGA buffer, transactor can be utilized to full capacity (transfer overhead mitigation)
- Will lead to significant performance improvement

Single vs Multiple Packets in flight