Reconfigurable Virtual Instrumentation based on SoC FPGA

With Emphasis on Scientific Instrumentation and Distributed Systems

Andres Cicuttin
ICTP Multidisciplinary Laboratory

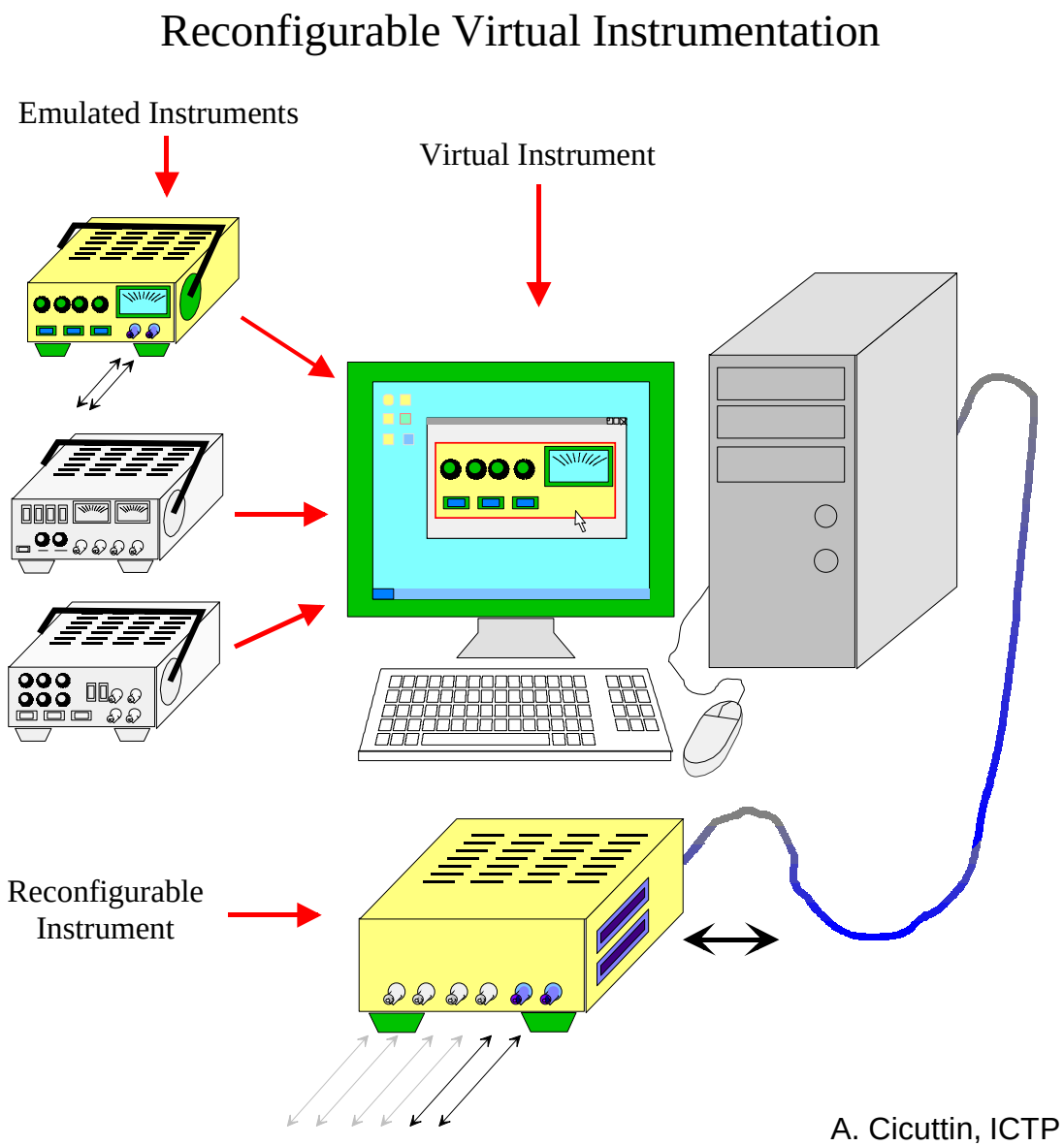
Outline

- Instrumentation for science and other applications
- RVI Concept
- Instrumentation based on SoC-FPGA devices
- Global architecture
- Design for Portability
- RVI Abstract Model: Configuration and Programming
- The Wishbone bus interface standard

Desirable main features of advanced instrumentation

	Scientific	Industrial	Commercial	Academic	Military
Performance	max				max
Accuracy, Precision	max	high			max
Reconfigurability	high		sometimes		
Massively parallel	sometimes				
Physically Distributed	sometimes				
Cost			low	low	
Design time			low	low	
Reliability		high			max

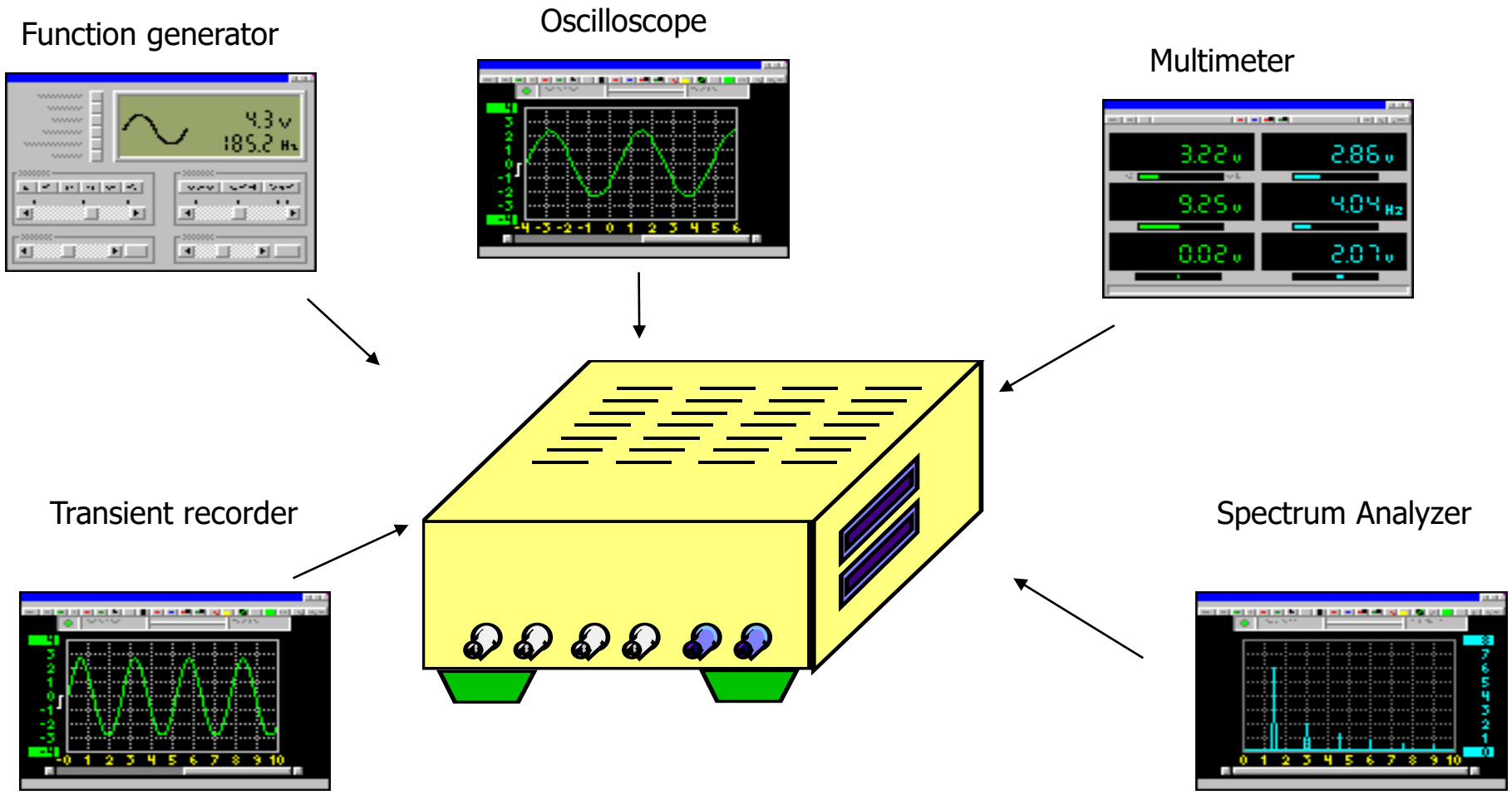
An artistic view of an RVI System



- The RVI system can be seen as a “magic-box” connected to a PC through a standard port

- High-level software application
  - Select a virtual instrument from a library of instruments
  - Configure the RVI system to convert it into the selected instrument
  - Associate a virtual console to operate the instrument.

Reconfigurable Virtual Instrumentation



RVI SYSTEM ARCHITECTURE

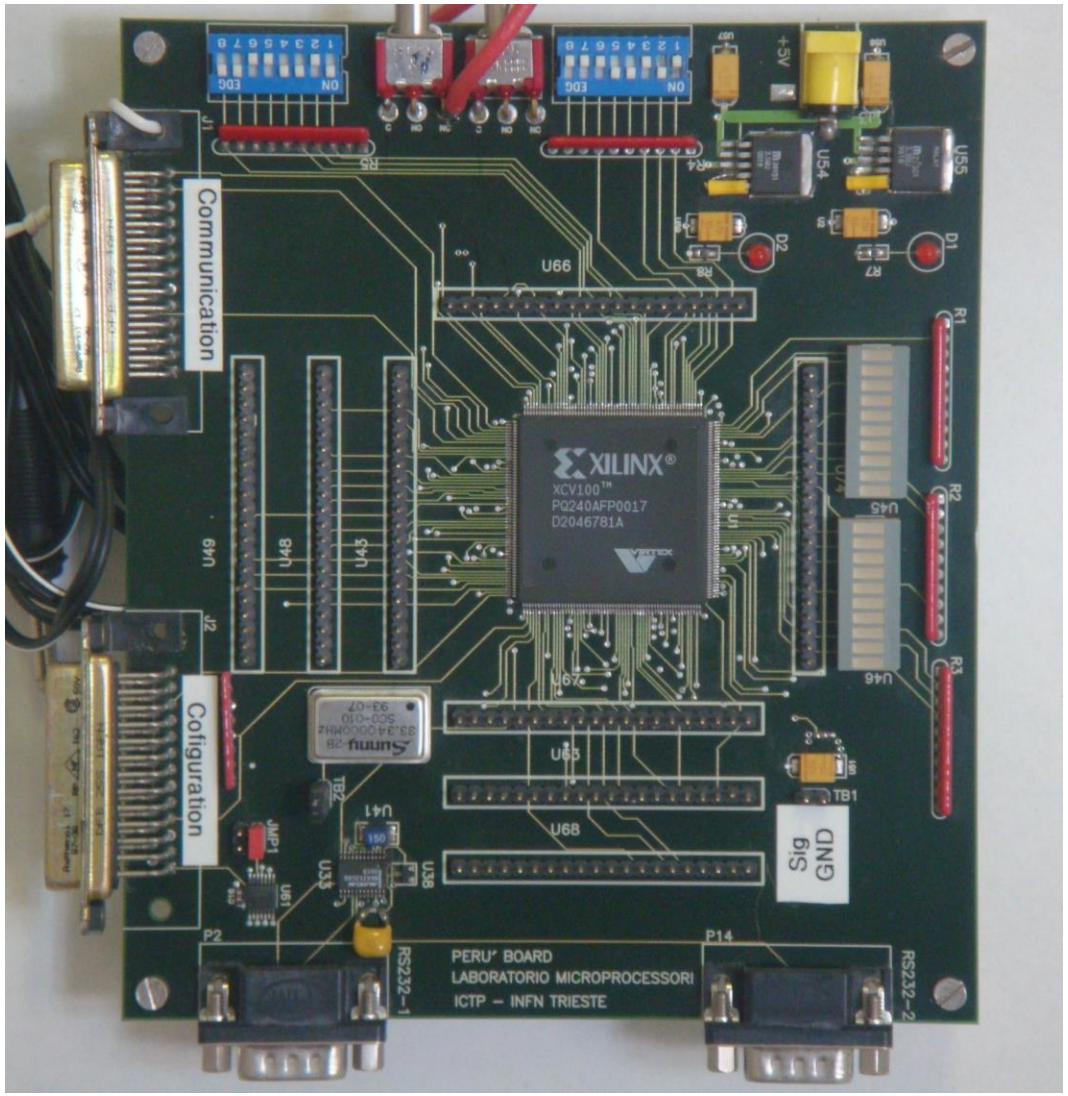
- ***Reconfigurable Instrument***
  - a versatile hardware that can be reconfigured to implement different electronic instruments using a software tool

- ***Virtual Instrumentation***
  - a hardware and software combination that allows the emulation of an instrument through a custom virtual console (graphical user interface)

RVI Hardware: The ICTP Peru Board (2001)

Parallel Ports for Communication and Configuration

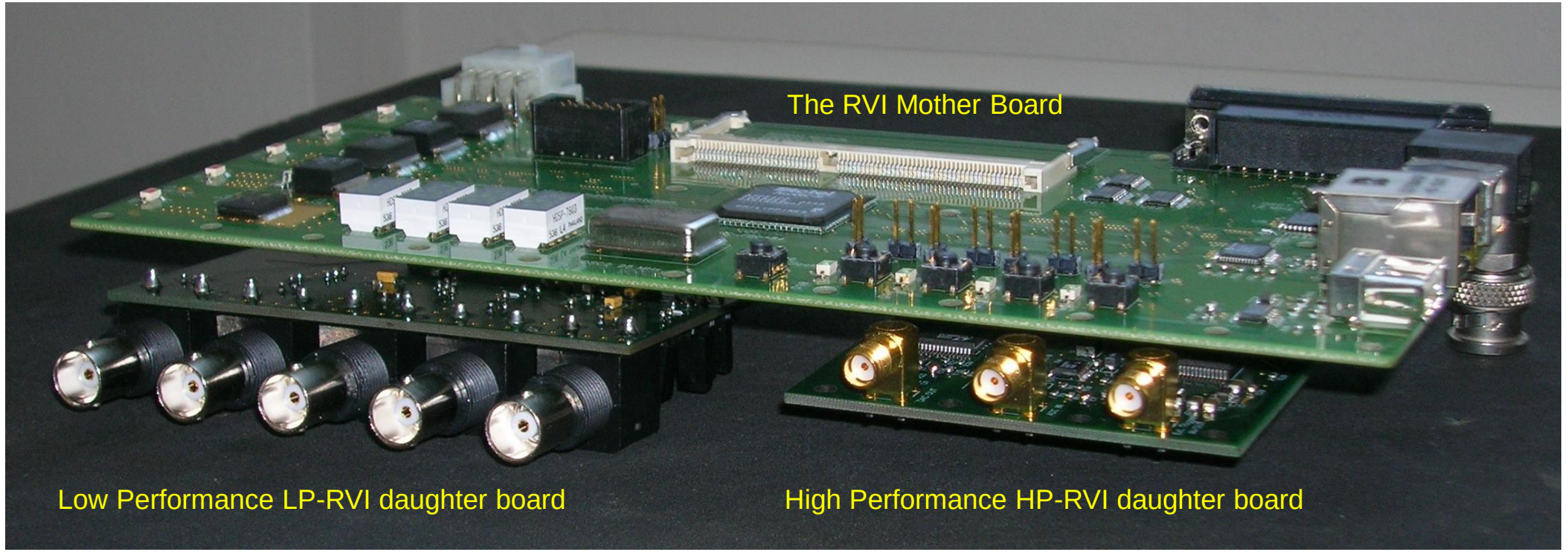
General Purpose Serial Ports



Xilinx Virtex 100
First FPGAs with embedded RAM and DLL

On-board Digital to Analog Converter

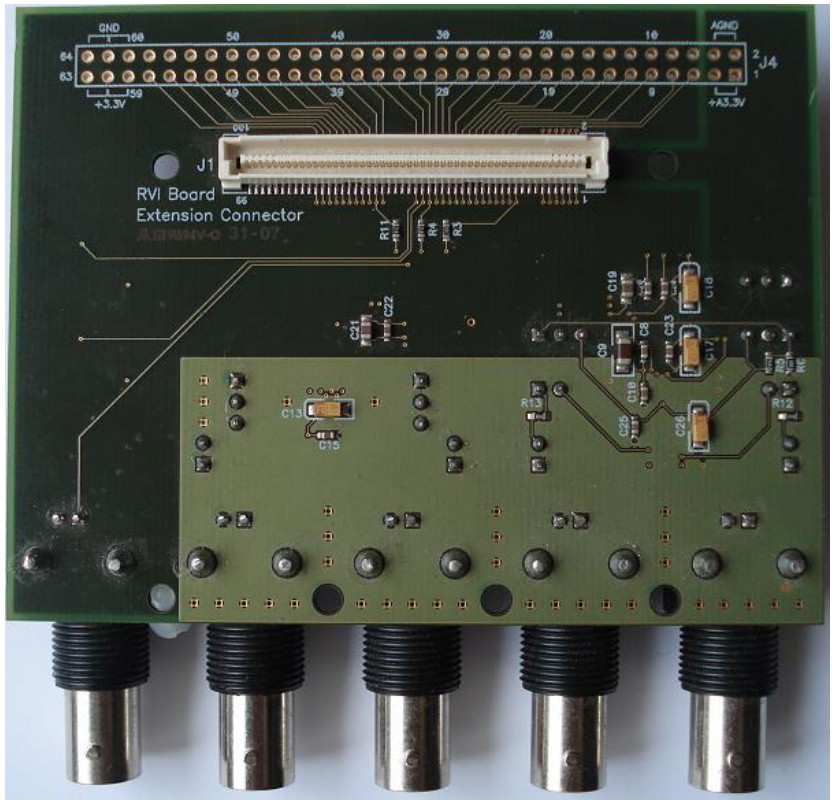
The ICTP RVI Platform (2006)



ADC dual channel 10-bits 20 MSPS
DAC dual channel 14-bit 1 MSPS

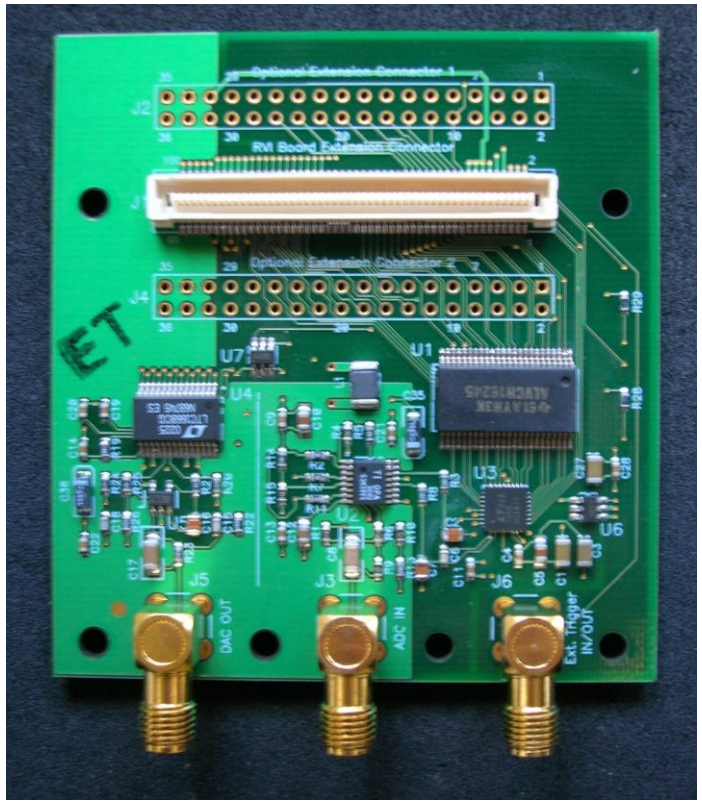
ADC single channel 14-bits 125 MSPS
DAC single channel 16-bit 50 MSPS

Detailed view of ICTP-RVI Daughter Boards



Low Performance Daughter Board

- ADC dual channel 10-bits 20 MSPS
- DAC dual channel 14-bit 1 MSPS

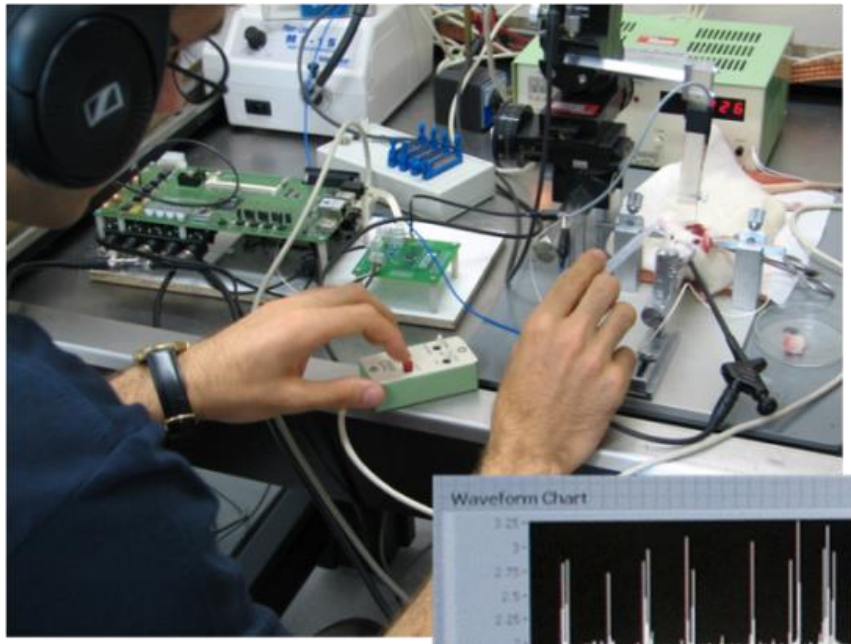


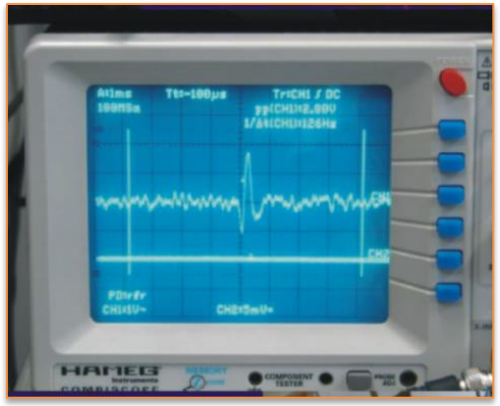
High Performance Daughter Board

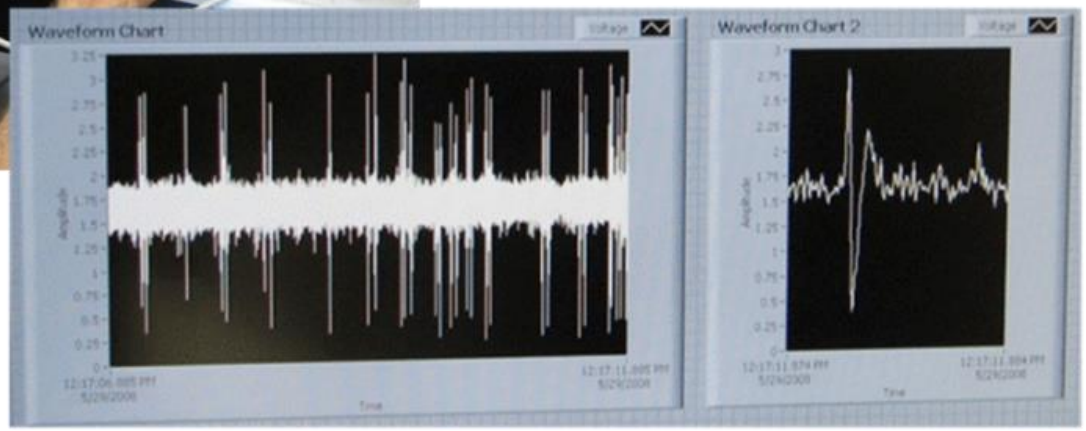
- ADC single channel 14-bits 125 MSPS (LTC2255)
- DAC single channel 16-bit 50 MSPS (LTC1668)

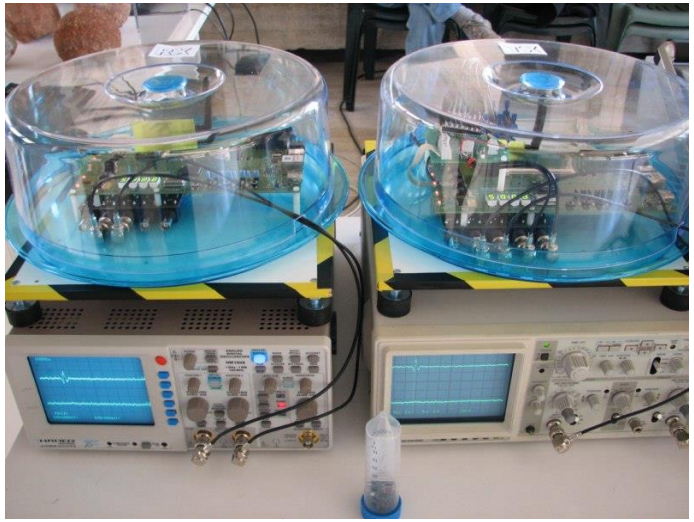
Advanced scientific instrumentation based on FPGA

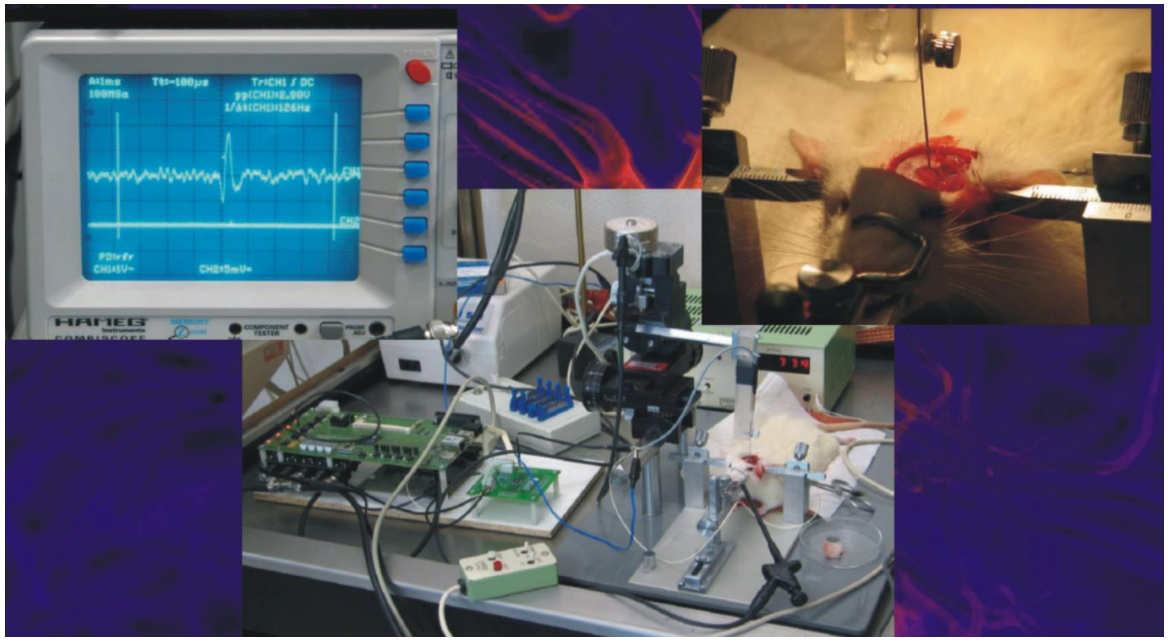
Which characteristics? and which capabilities?

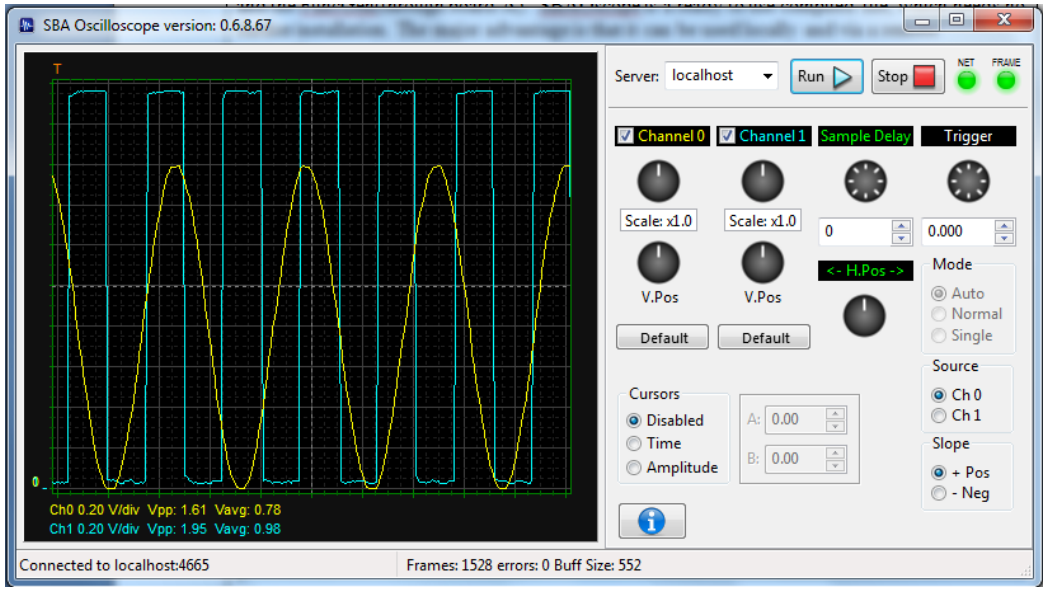


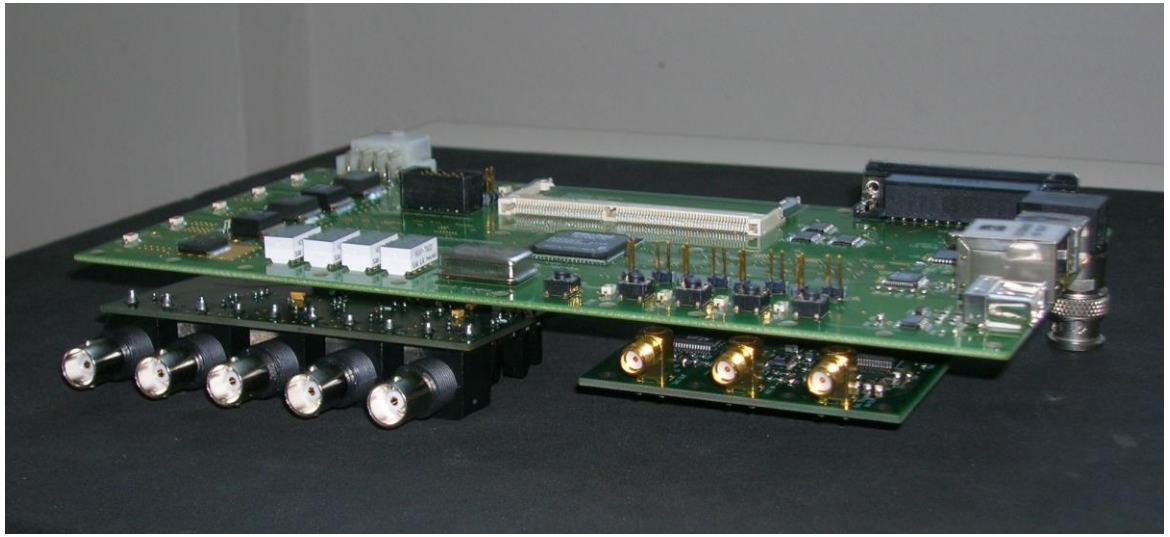


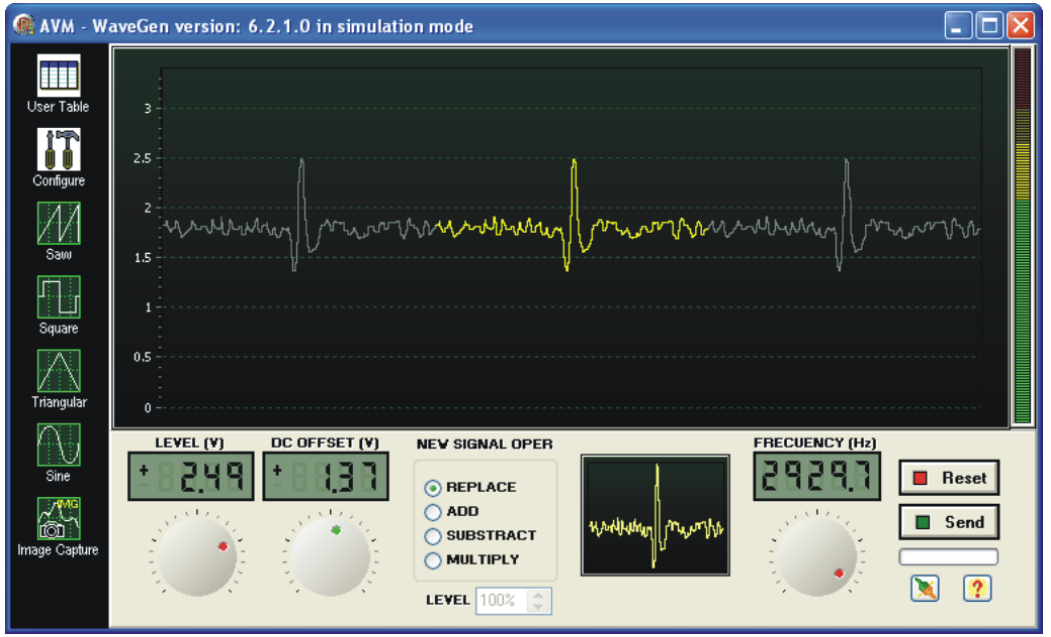




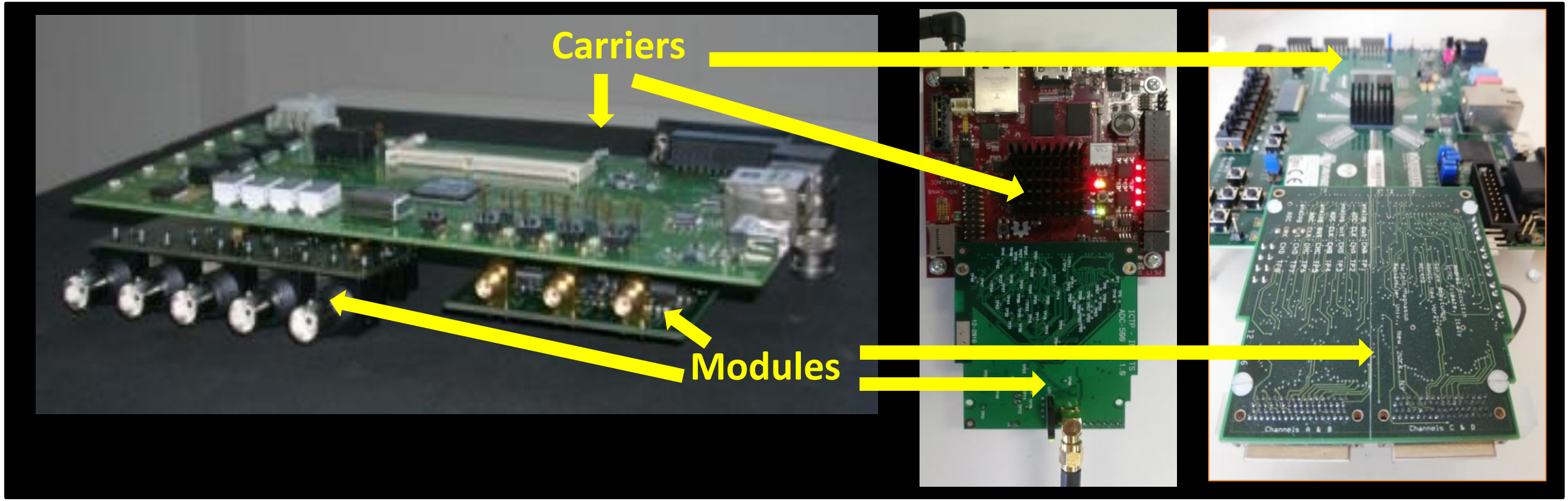



SBA Oscilloscope version: 0.6.8.67
Server: localhost
Run
Stop
Channel 0
Channel 1
Sample Delay
Trigger
Scale: x1.0
Scale: x1.0
V.Pos
V.Pos
Default
Default
<- H.Pos ->
Mode
Auto
Normal
Single
Source
Ch 0
Ch 1
Slope
+ Pos
- Neg
Cursors
Disabled
Time
Amplitude
Ch0 0.20 V/div Vpp: 1.61 Vavg: 0.78
Ch1 0.20 V/div Vpp: 1.95 Vavg: 0.98
Connected to localhost:4665
Frames: 1528 errors: 0 Buff Size: 552




AVM - WaveGen version: 6.2.1.0 in simulation mode
User Table
Configure
Saw
Square
Triangular
Sine
Image Capture
LEVEL (V)
DC OFFSET (V)
NEW SIGNAL OPER
REPLACE
ADD
SUBSTRACT
MULTIPLY
LEVEL
FRECUENCY (Hz)
Reset
Send

Advanced scientific instrumentation based on FPGA



ADC dual channel 10-bits 20 MSPS
(24 MB/s)

DAC dual channel 14-bit 1 MSPS
(24 MB/s)

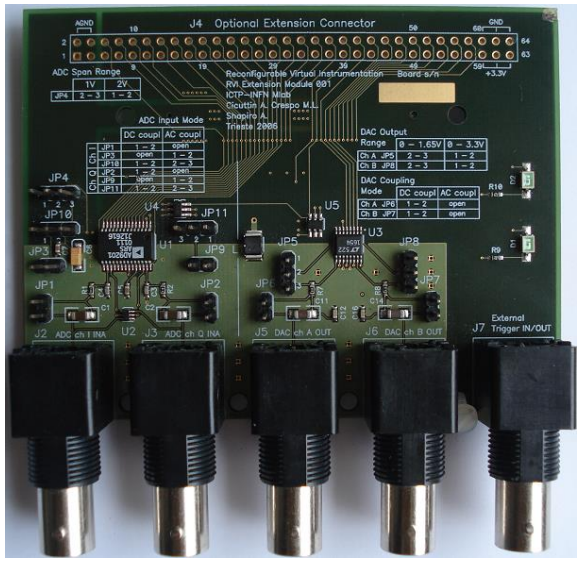
ADC single channel 14-bits 125 MSPS
(218 MB/s)

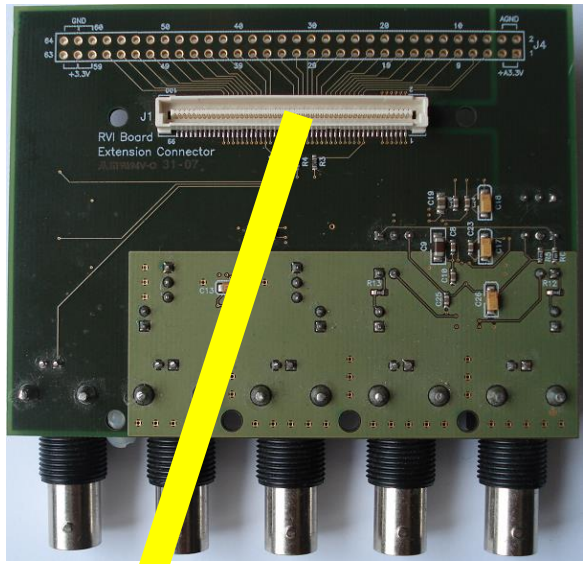
DAC single channel 16-bit 50 MSPS
(100 MB/s)

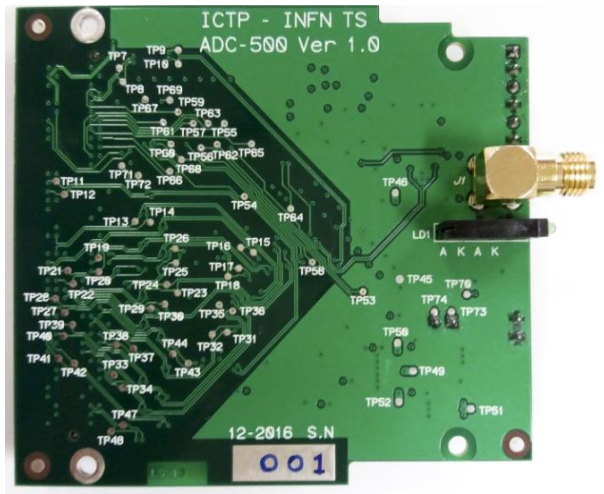
ADC single channel
8-bits @ 500 MSPS
(500 MB/s)

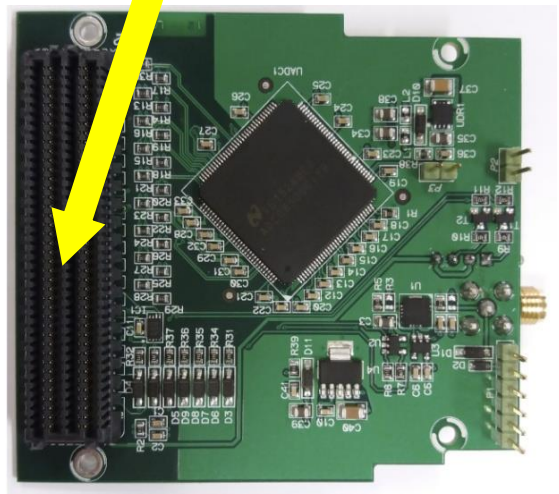
ADC 128 channels
16-bits @ 32 KSPS
(8 MB/s)

Modularization of FPGA-based Hardware







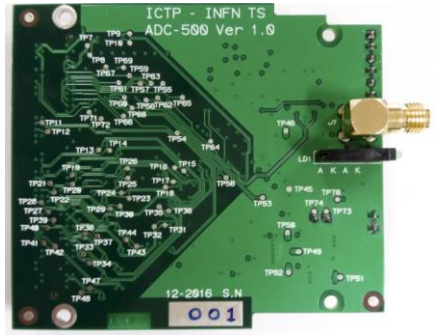


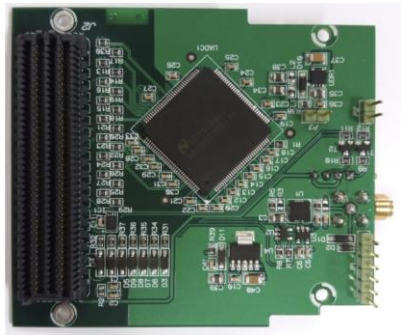
FMC Connector
(FPGA **M**ezzanine **C**ard)

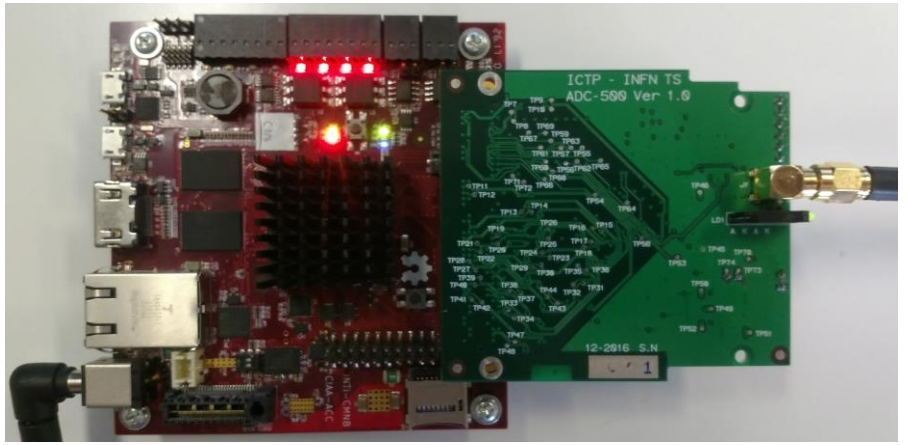
FPGA Based Carrier
+
Mezzanine modules

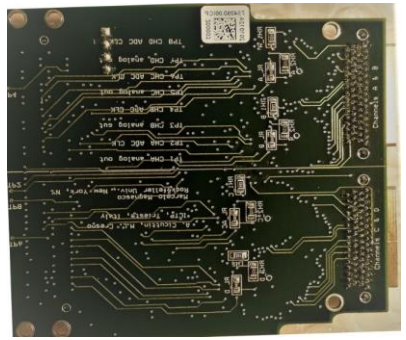
- VITA 57 FPGA Mezzanine Card (FMC) ANSI standard.
- FMC and FMC+ are Standards for electro-mechanical expansion interface between **Mezzanine Boards** and **FPGA-based Carriers** with re-configurable Inputs/Outputs capability.

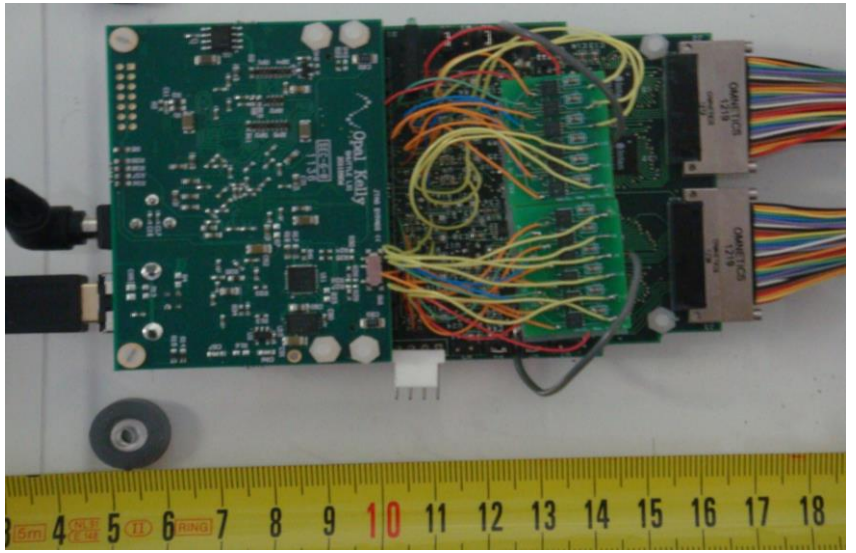
Interoperability between FMC Carriers and Modules



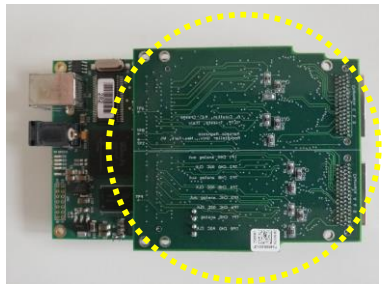




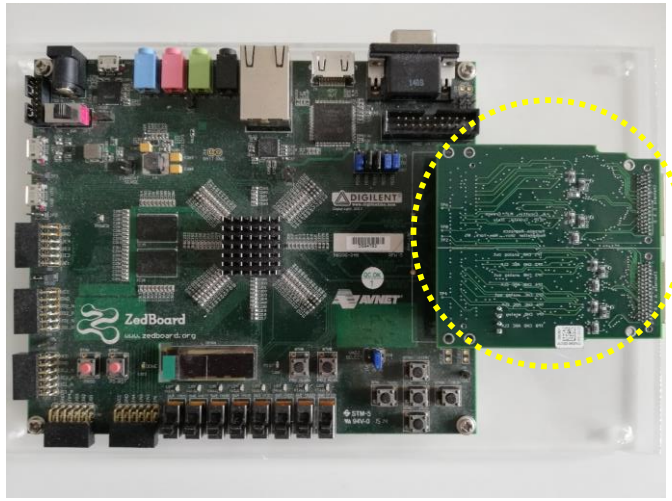




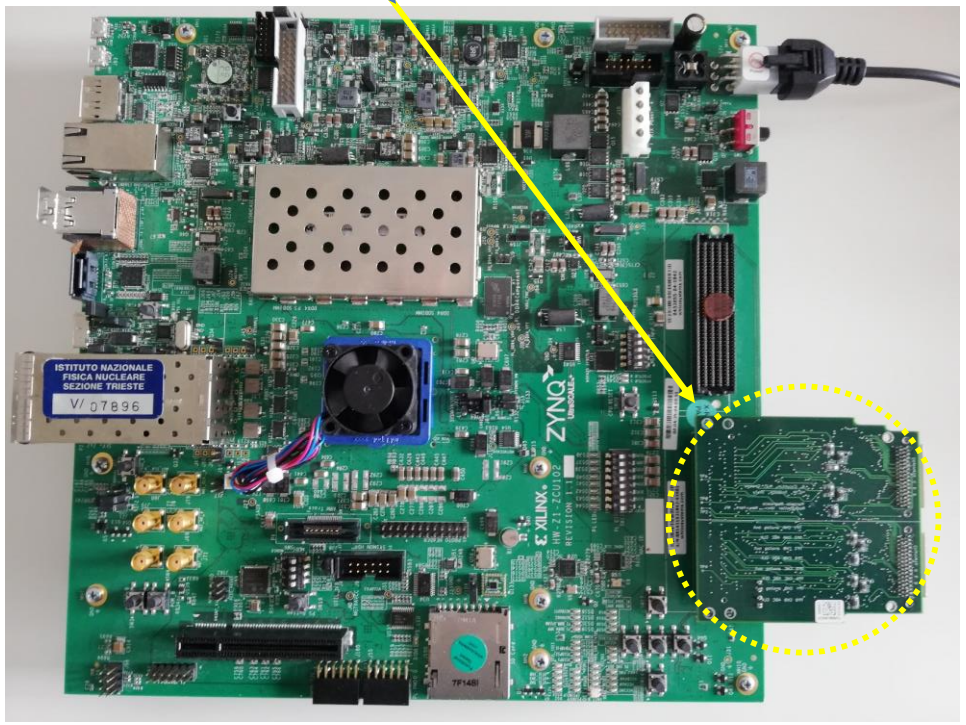
Interoperability between FMC Carriers and Modules



Spartan



Zynq



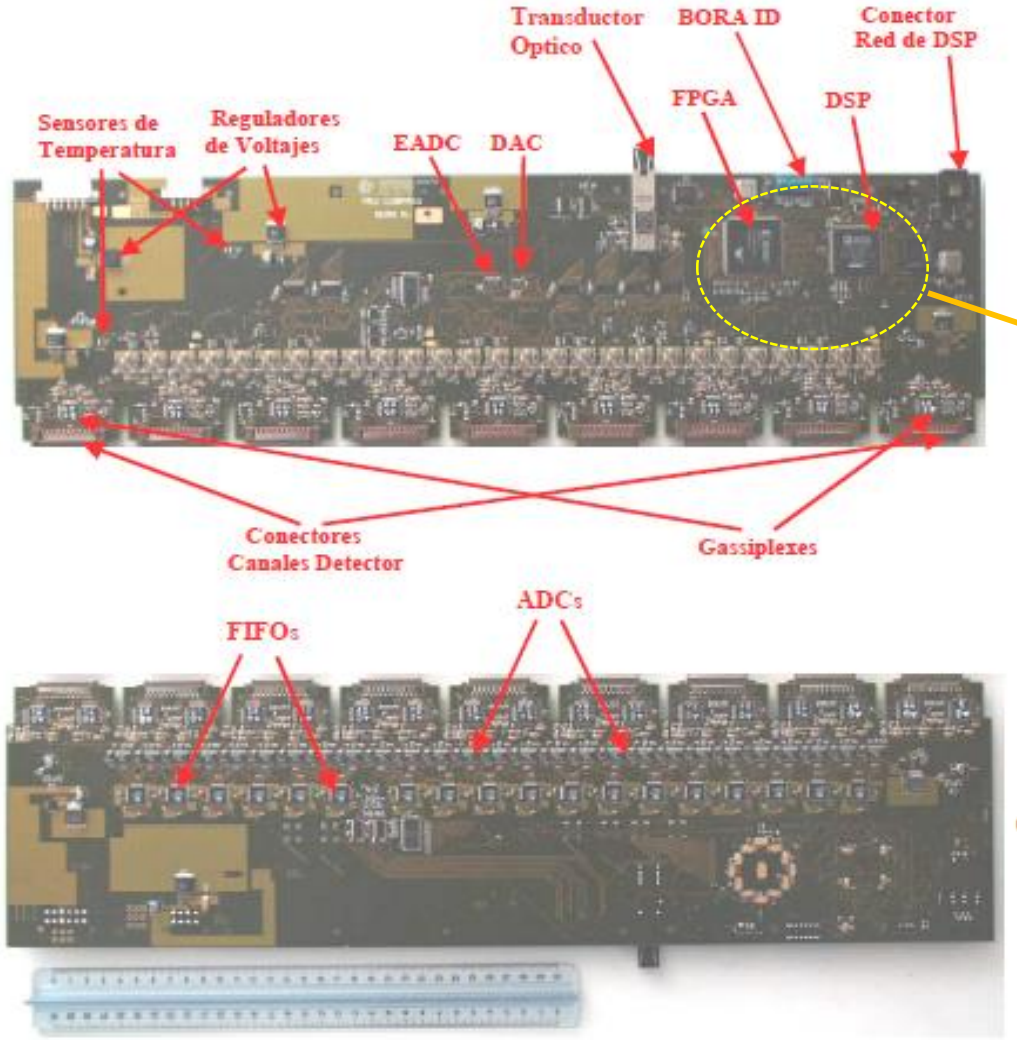
UltraScale

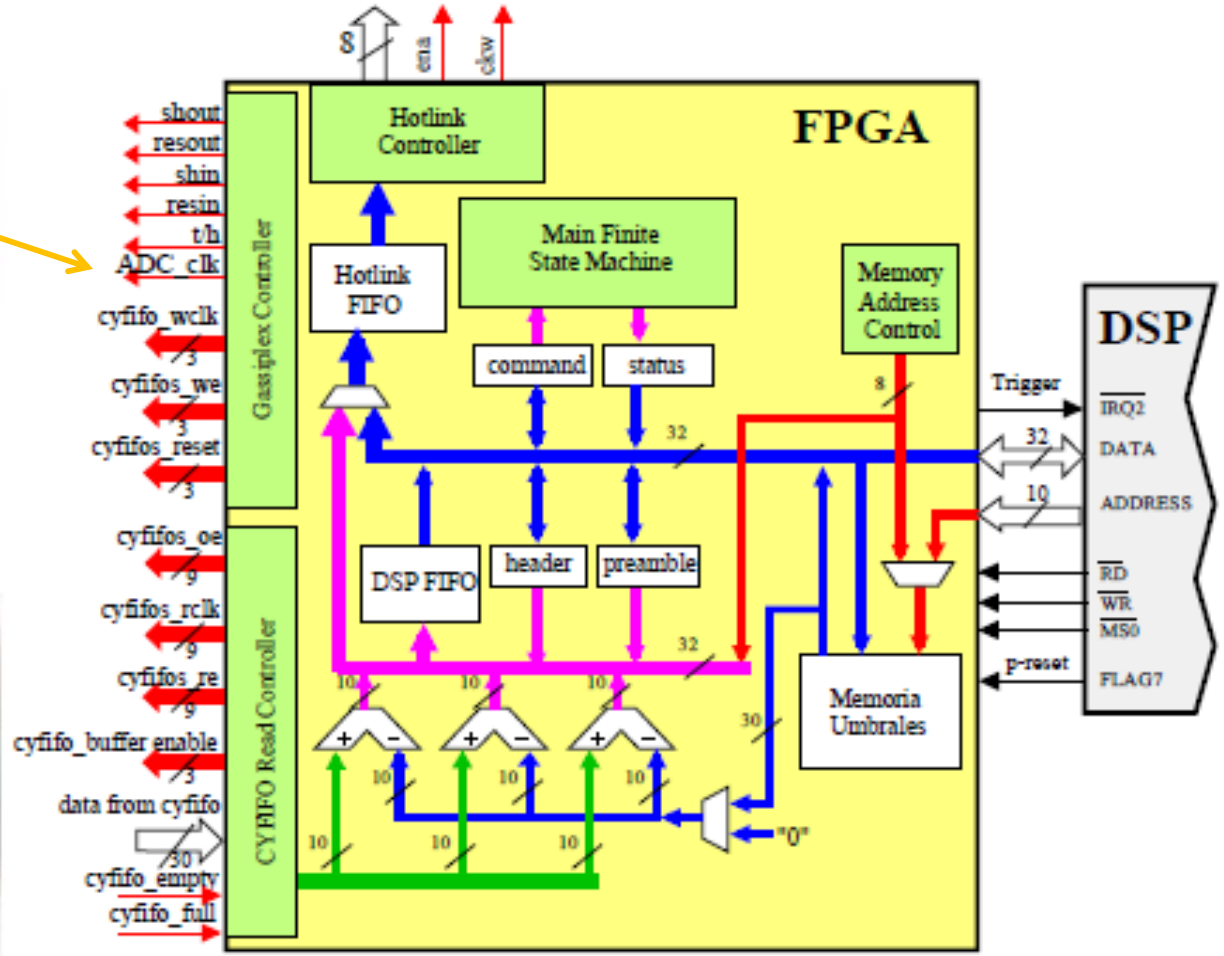
HiCCE-128 FMC Module

FMC Carriers

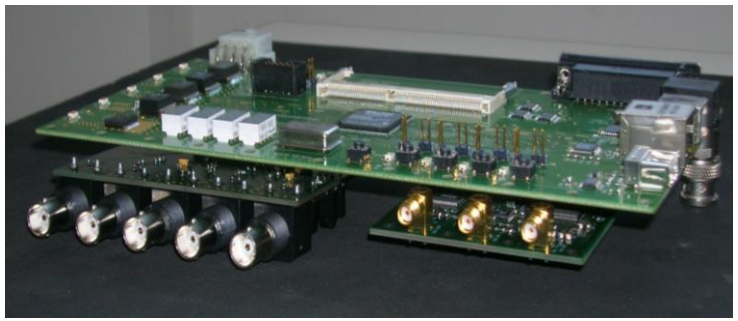
RVI based on DSP–FPGA combination (2000)

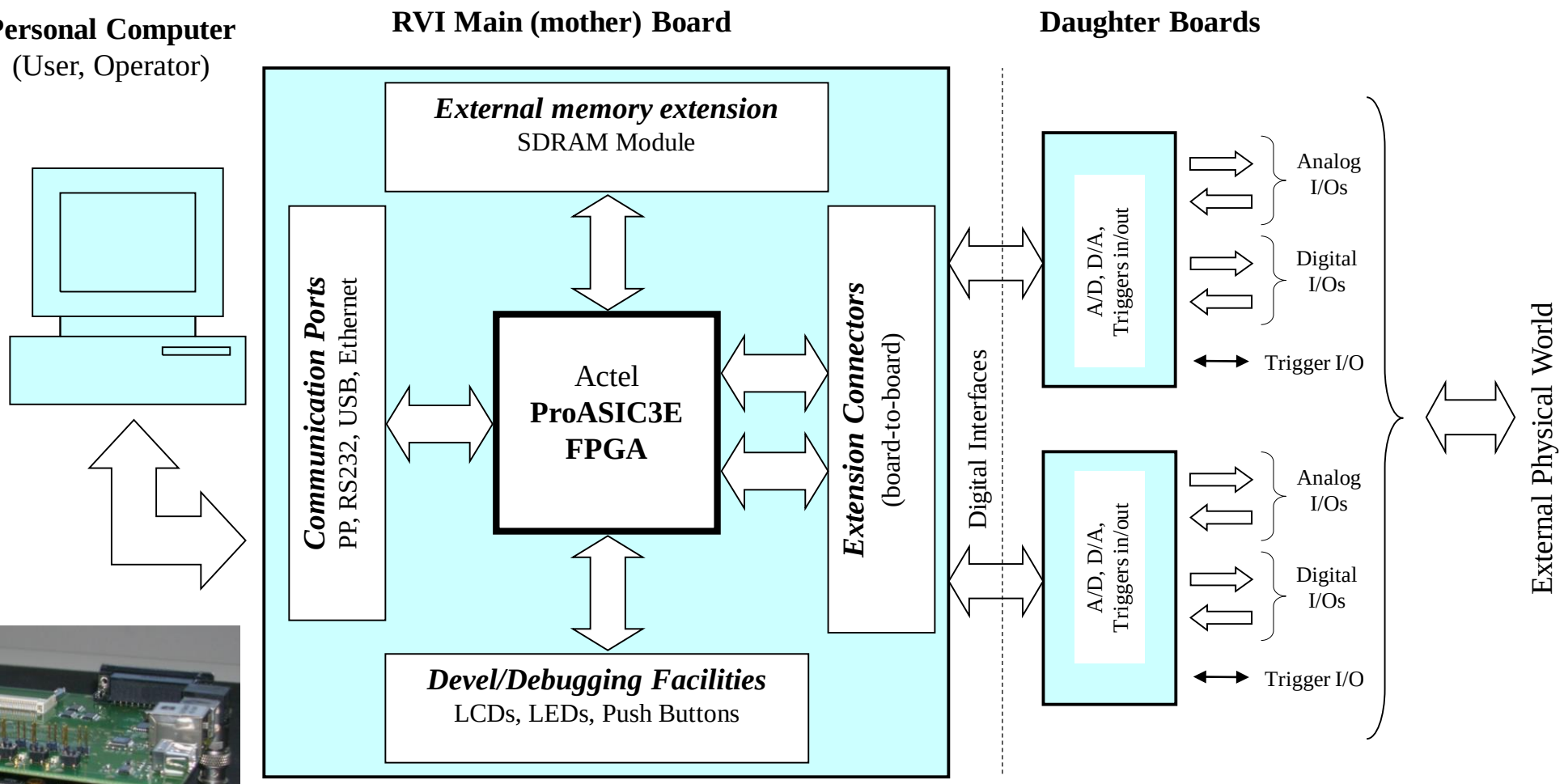
Which advantages of increasing integration?





RVI FPGA-Based Hardware System





A possible FPGA Global Architecture for RVI

1) PC <-> FPGA communication

2) External Hardware

3) Instruments cores

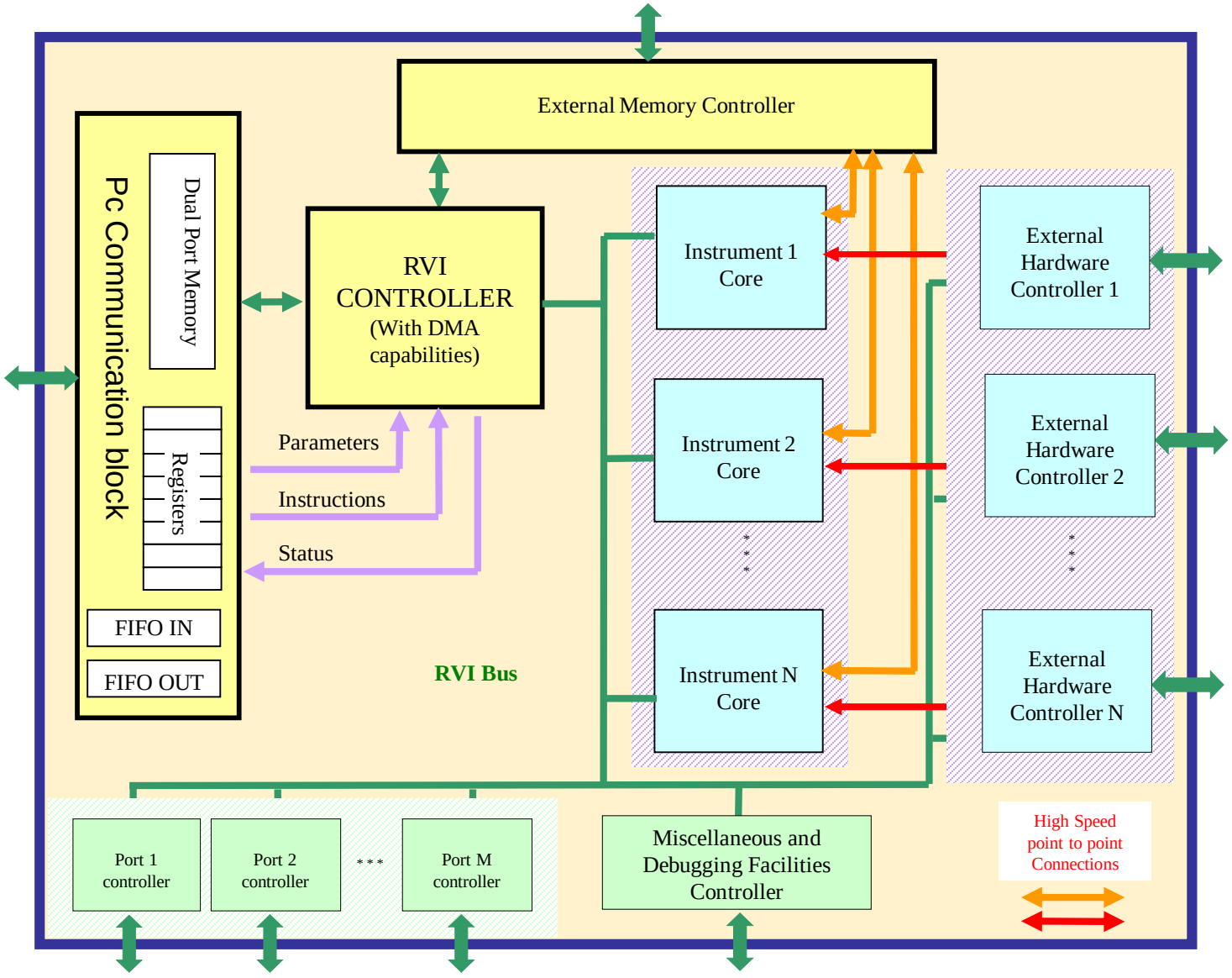
4) External memory

5) Standard Ports

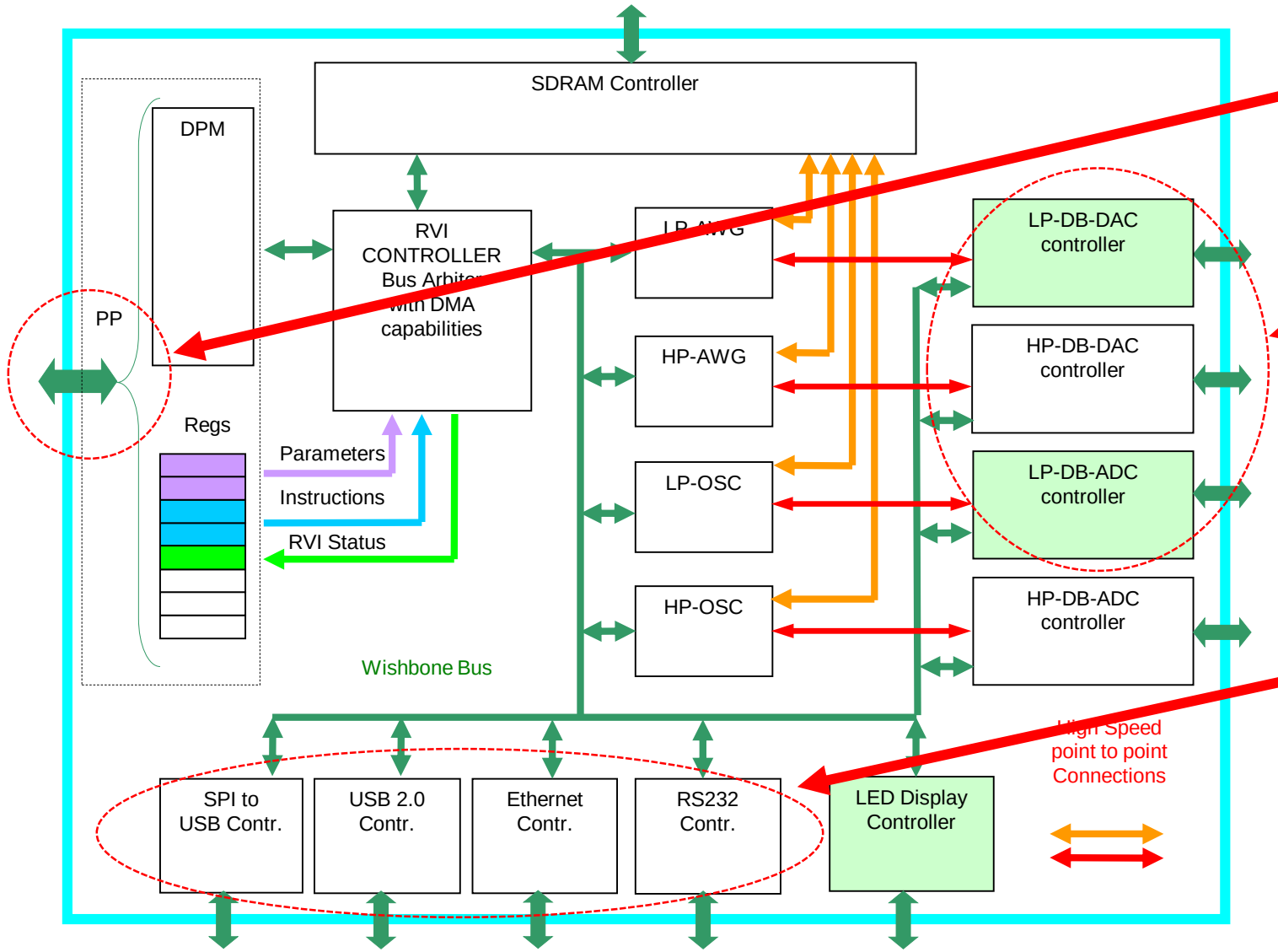
6) Debugging Facilities

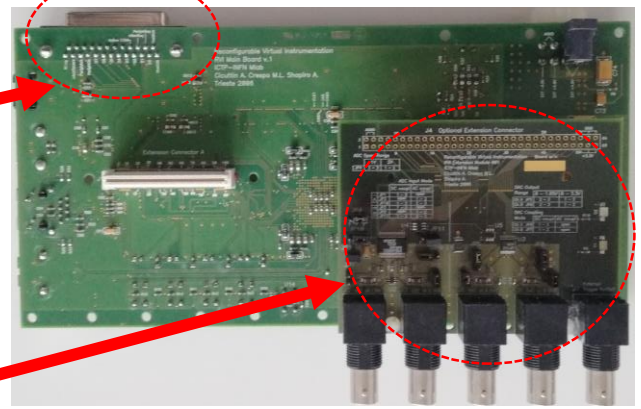
7) Global bus and interconnections

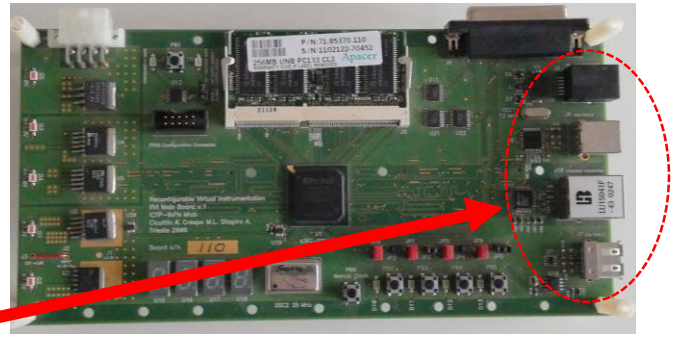
8) RVI Control

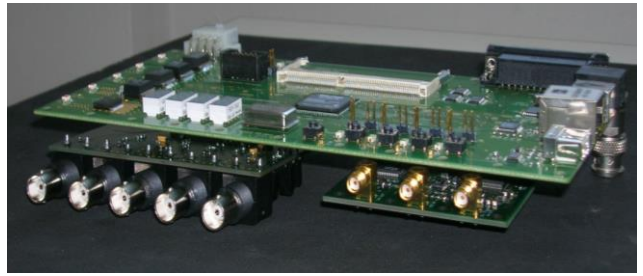


RVI FPGA Internal Architecture





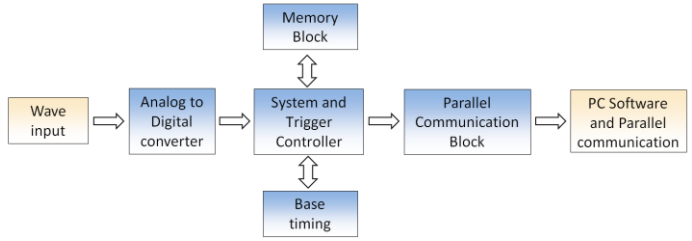




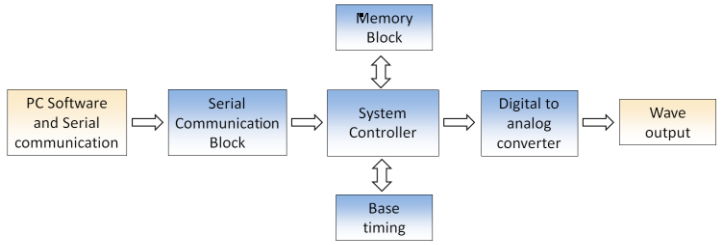
RVI Examples: Block diagrams and GUIs

Block Diagrams

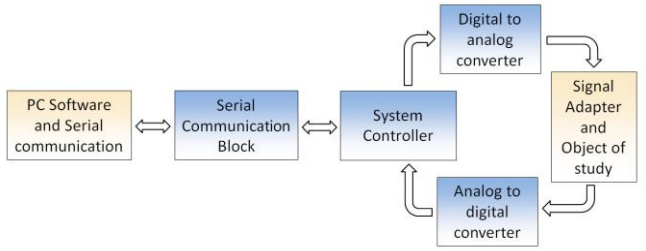
Digital Oscilloscope

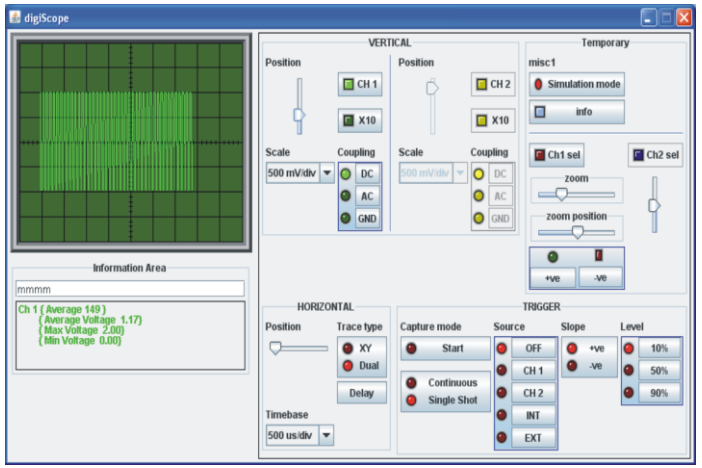


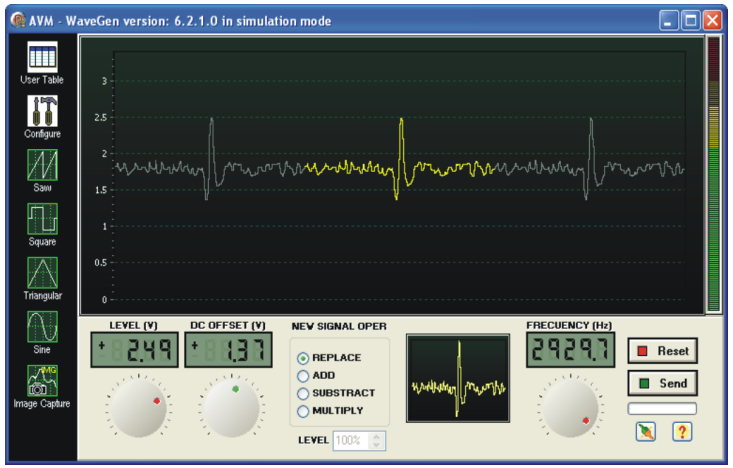
Arbitrary Waveform Generator

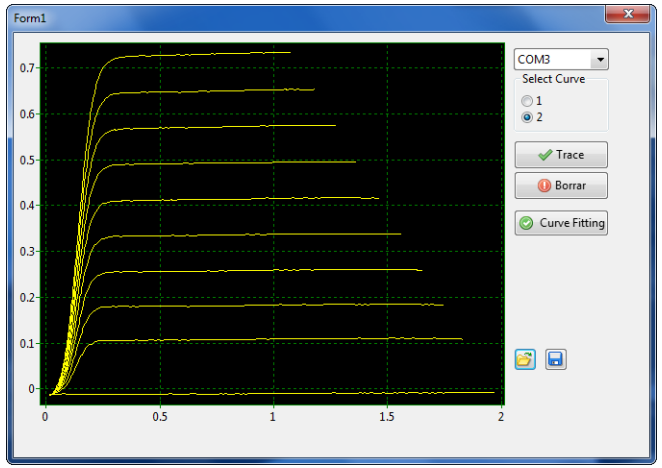


Semiconductor device curve tracer





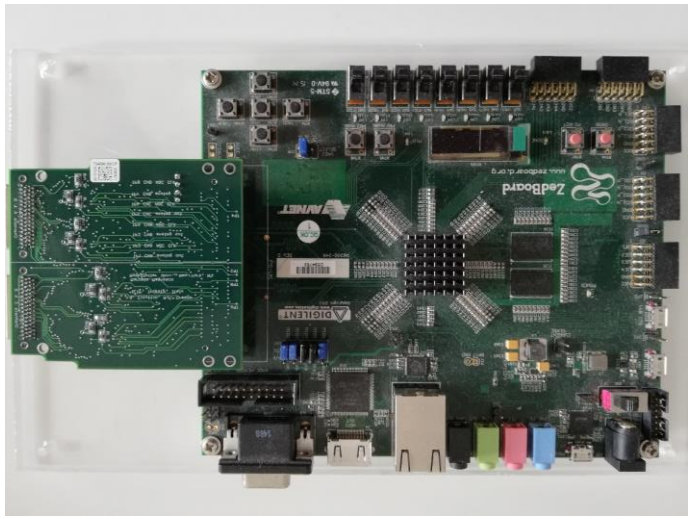




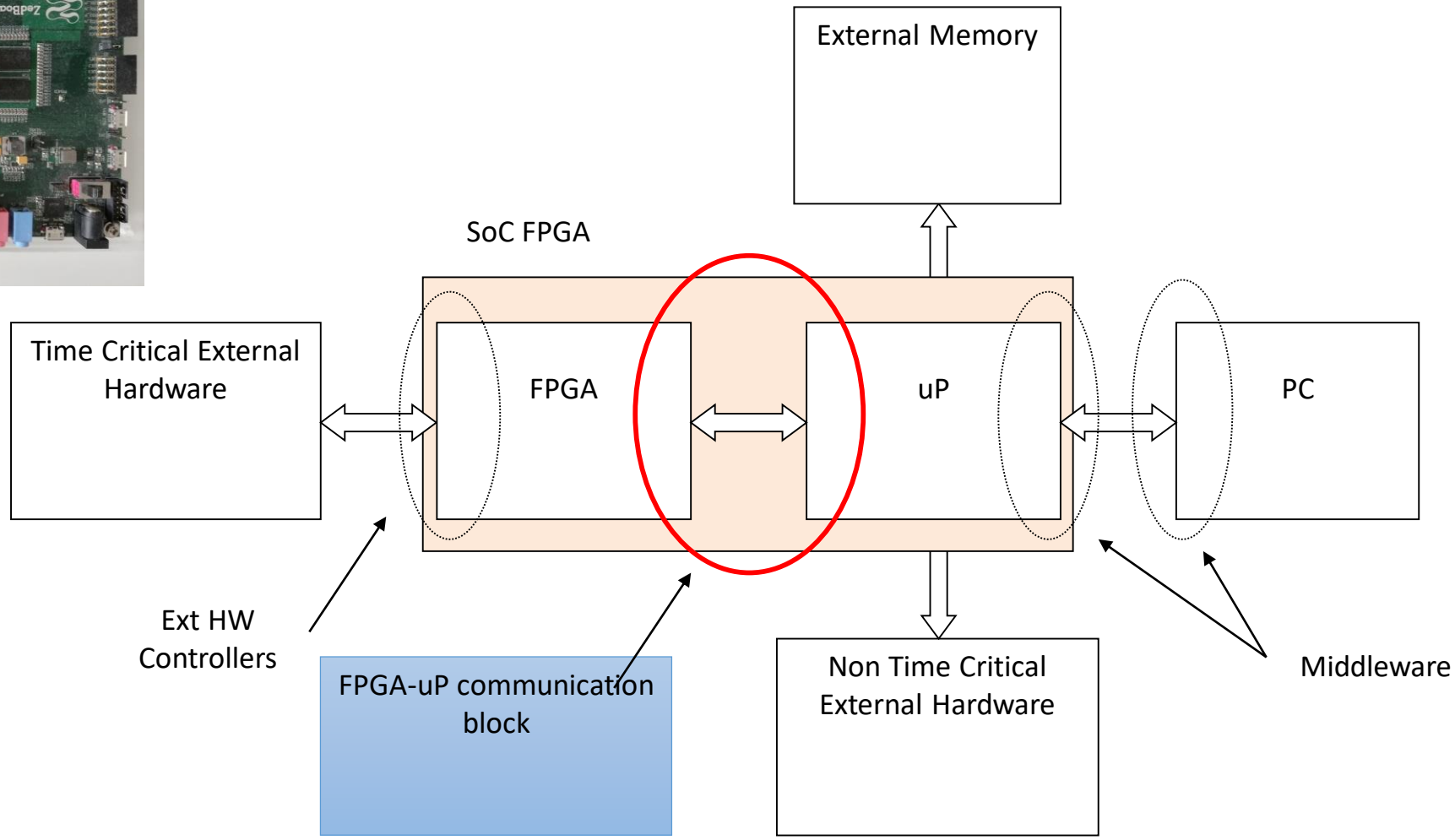
Screen Shots of PC GUI

SoC-FPGA Design for Portability

RVI Based on SoC-FPGA Global Architecture I

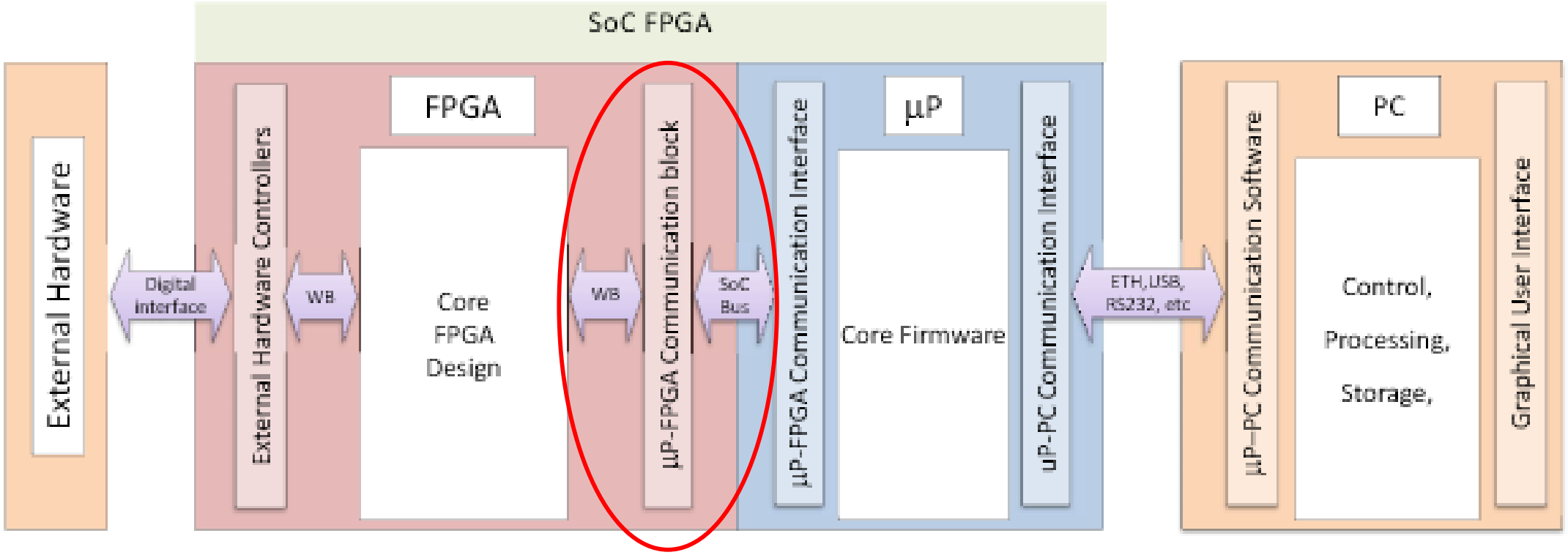


Main parts



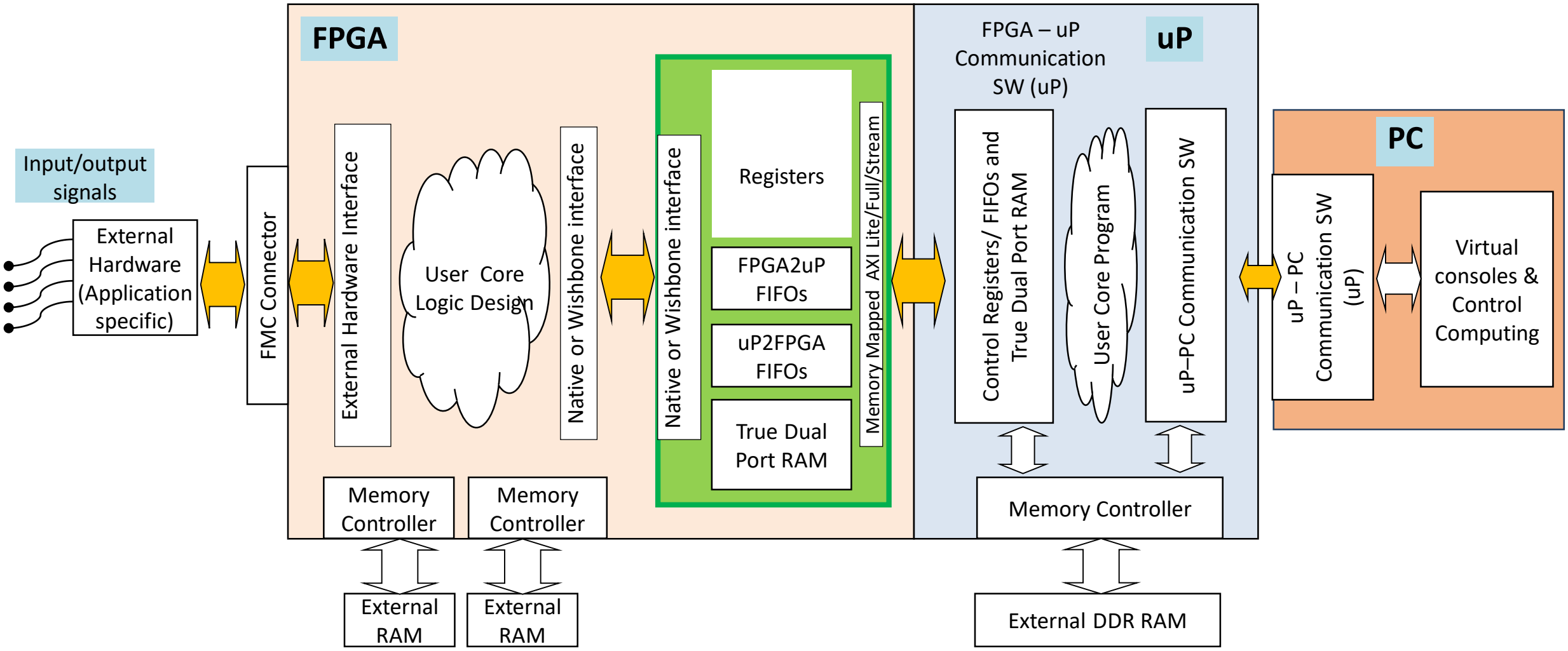
RVI Based on SoC-FPGA Global Architecture II

Global hardware software partition



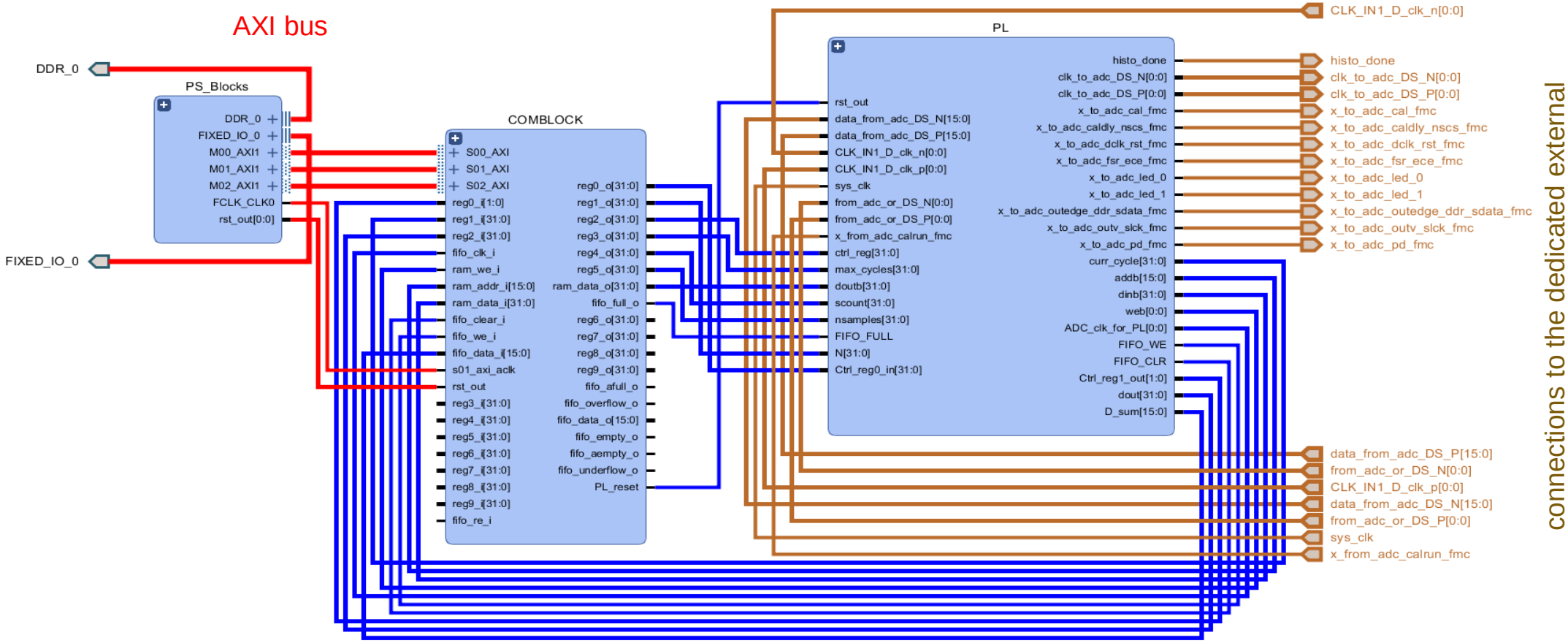
RVI Based on SoC-FPGA Global Architecture III

Three main Subsystems



A *Non-Fully* AXI-based SoC FPGA design

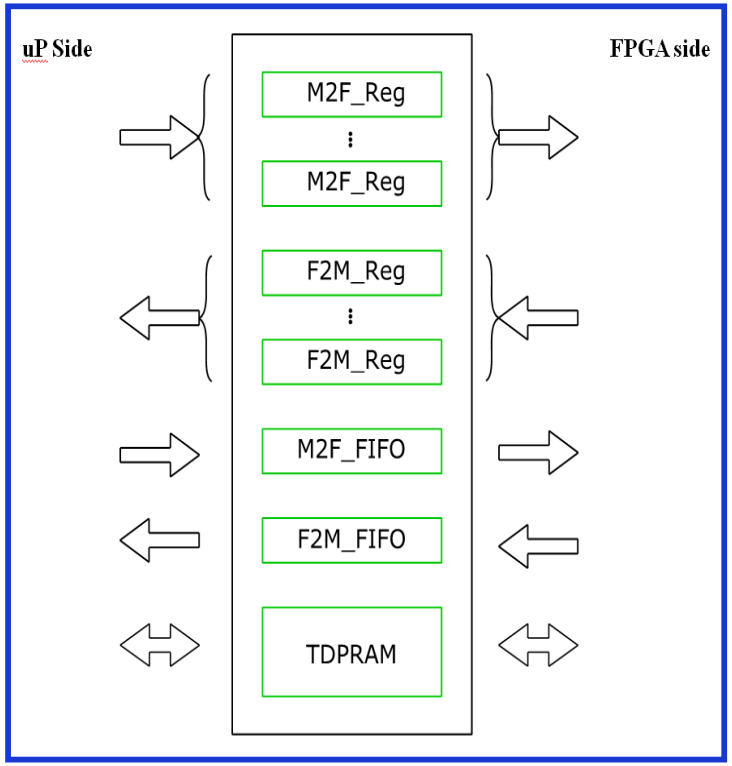
Microprocessor-centric designs Vs FPGA-centric designs or hybrid distributed systems



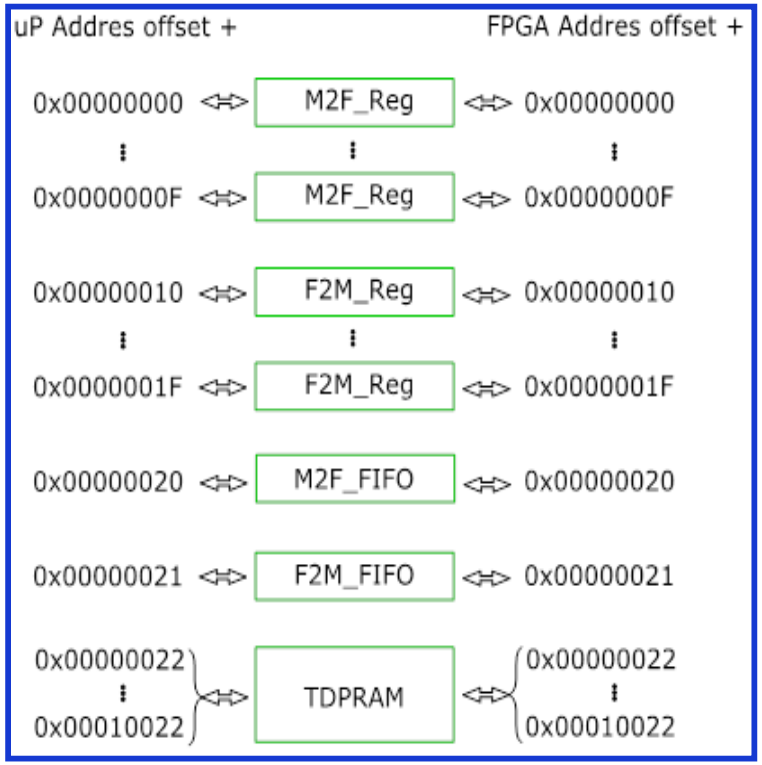
Connections to the FPGA-subsystem through the native interfaces of the CB

The communication block: Comblock

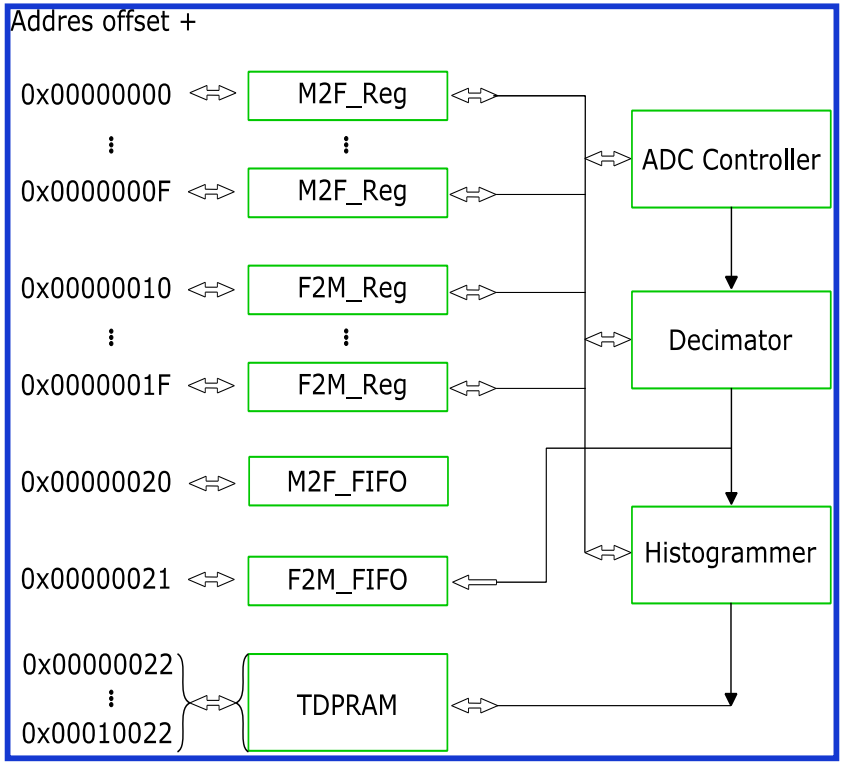
Three type of memory elements: registers, FIFO, and TDPRAM



M2F: uP to FPGA
F2M: FPGA to uP



Access from FPGA with a bus



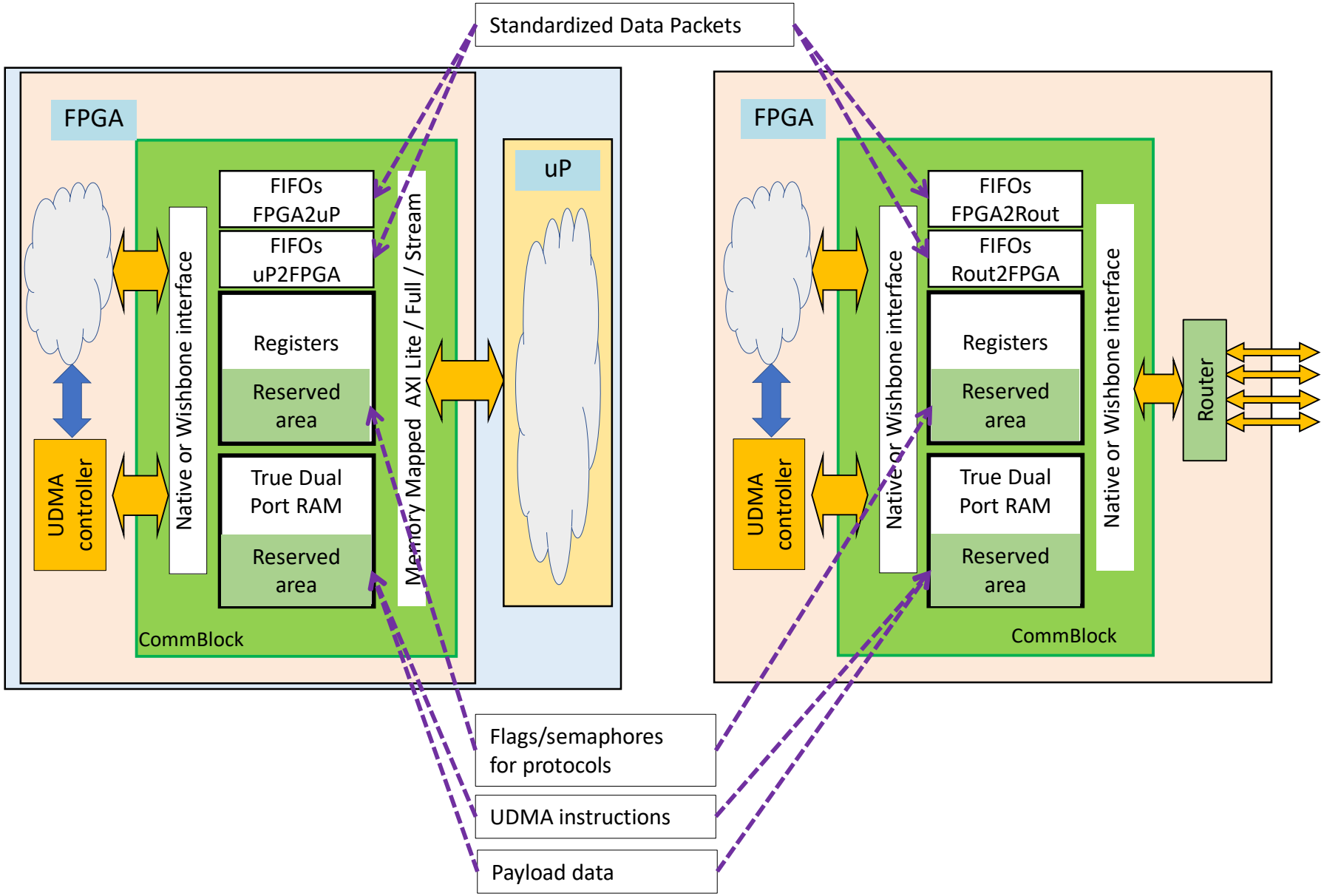
Direct connections

Communication through FPGAs in clusters of reconfigurable computational units

Data packet transmission over

- *On demand point-to-point connections*
- *Buses*
- *Time-Division Multiplexing* on *common signal paths*

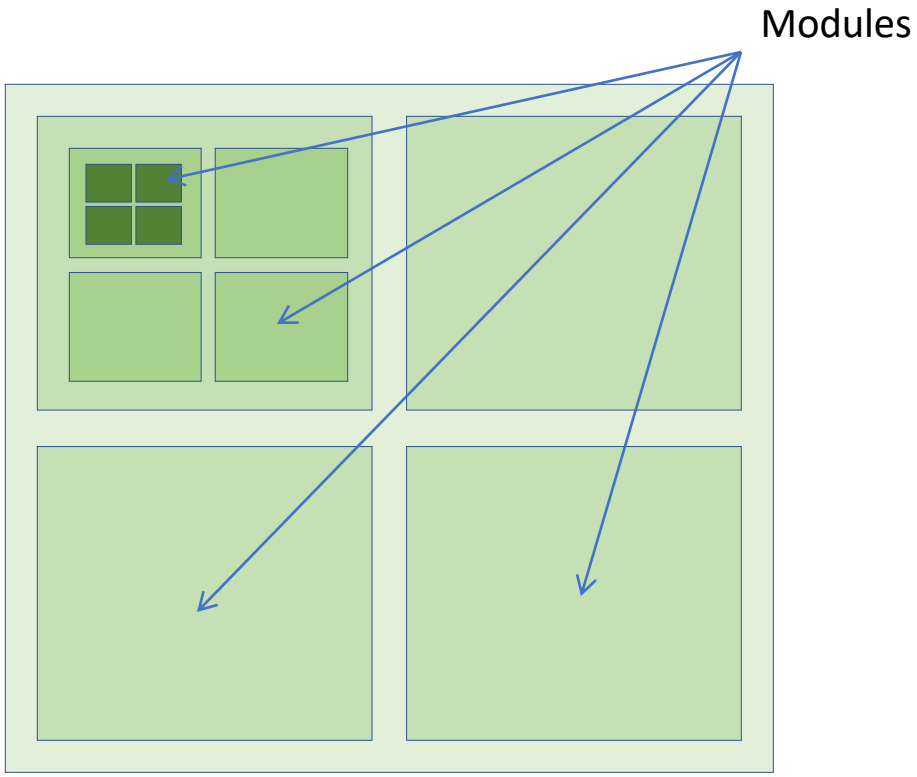
Slave Communication Blocks (Abstraction Layers)



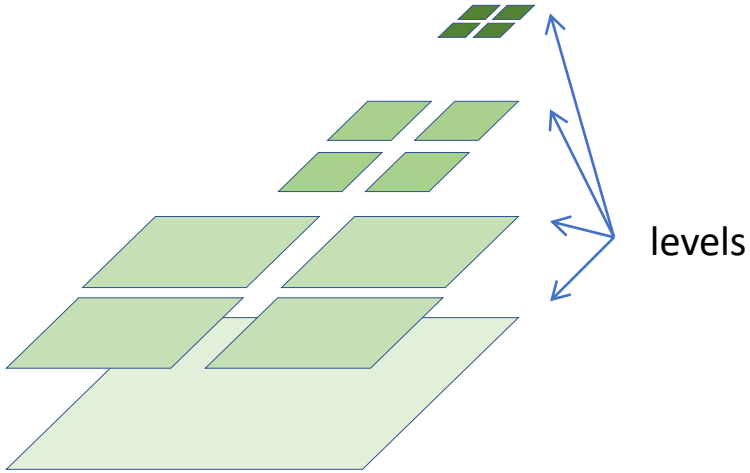
RVI Abstract Model: Elemental Concepts I

How to deal with large complexity? *Divide et impera*

Modularity



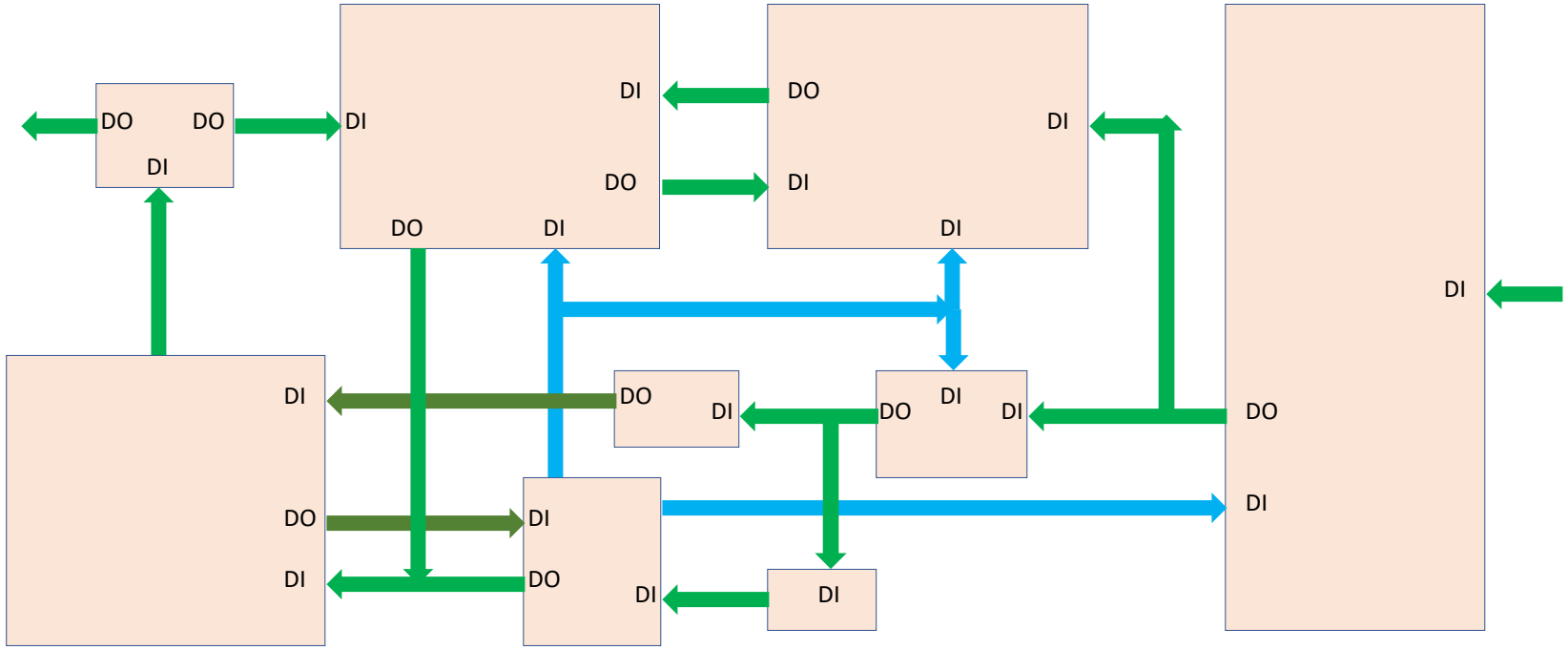
Hierarchy



RVI Abstract Model: Elemental Concepts II

What activity at given hierarchical level? *Direct Memory Access (DMA), Concurrent data transfer*

Functional blocks Ports Data exchange



RVI Abstract Model: Elemental Concepts III

How to effectively describe the hardware and its activity ?

Hardware Configuration

Instantiation of functional blocks

Memory mapping of registered ports

All HW Resources

Address	Ports
0x00000001 0x0000FFFF	Ext_RAM
0x000A0000 0x000AEEEE	Ext_ROM
0x000AEEEE	FIFO_a_in
0x000AEEF0	FIFO_b_out
0x001A0000 0x001AEEEE	RAM_block_p
0x002A0000	Register_h
0x002A000A	Operand _i
0x002A000B	Operand _j
0x003A0001	Operator_m_Out_k
0x003A0001	Ext_HW_in_port_x
0x003A0001	Ext_HW_out_port_y
	Register_k

Software Programming

Description of the HW actvity

Concurrent execution of ***U**niversal **D**irect **M**emory **A**ccess (**UDMA**)* instructions

Universal Direct Memory Access Instruction

UDMA SA DA SAinc DAinc N <BC> <activate, suspend, abort>

Source Address

Destination Address

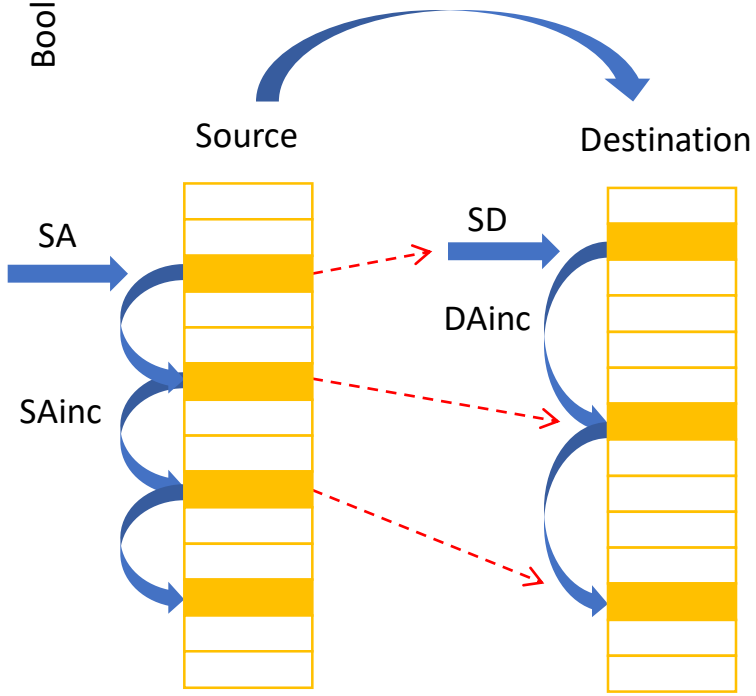
Increment of Source Address

Increment of Destination Address

Number of Words

Boolean condition

Reaction



Universal Direct Memory Access Instructions

Some examples

```
UDMA  0x0000F001  0x0000F00A  1  1  256
```

RAM to RAM

```
UDMA  0x0000F002  0x0002F00B  1  0  1024
```

RAM to FIFO

```
UDMA  0x0000F003  0x0004F00C  0  1  1024
```

FIFO to RAM

```
UDMA  0xAAAAF003  0x008FAA80  4  1  2000
```

RAM to RAM

```
UDMA  0xAAAA4004  0x000FAA40  0   0  0
```

Permanent link

```
UDMA  0xFFFF4004  0x000FAA00 4  1  1024  “timer > countmax“ Abort
```

Conditional data transfer

```
UDMA  0xFFFF4004  0x000FAA00 4  1  1024  “counter1 ==  31“ Suspend
```

Conditional data transfer

Universal Direct Memory Access Instruction for **Distribution**

UDMA_BC SA $\{DA_1,\ldots,DA_k\}$ N <NC> <BC> <R>

Source Address

Destination Addresses

Number of Words per cycle

Number of cycles

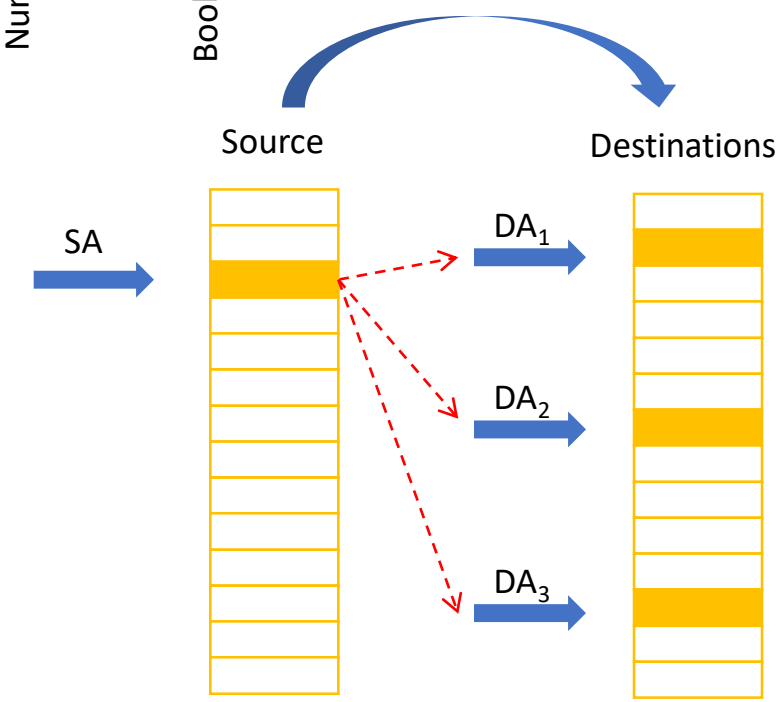
Boolean condition

Reaction

UDMA_BC (BroadCast)

UDMA_DRR (Distribute Round Robin)

UDMA_DTFF (Distribute Till FIFO Full)



Universal Direct Memory Access Instruction for **Collection**

UDMA_CRR $\{SA_1,\ldots,SA_k\}$ DA N NC <BC> <R>

Source Addresses

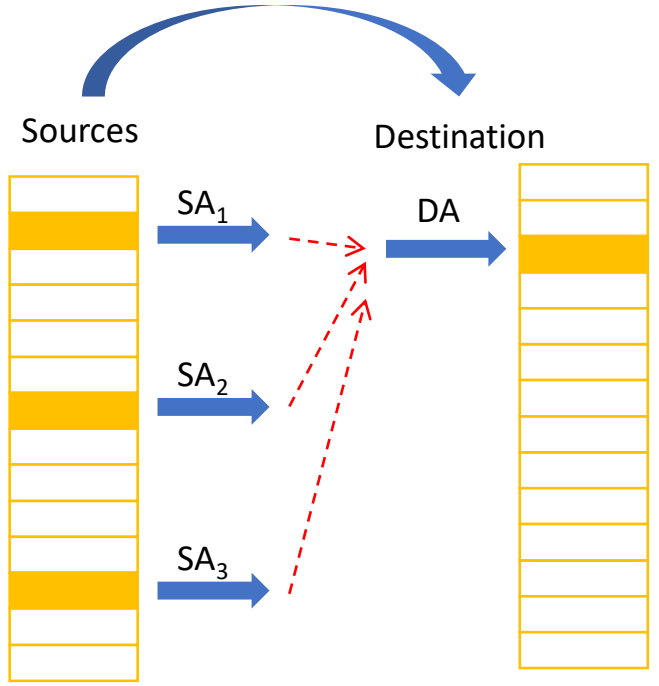
Destination Address

Number of Words per cycle

Number of cycles

Boolean condition

Reaction

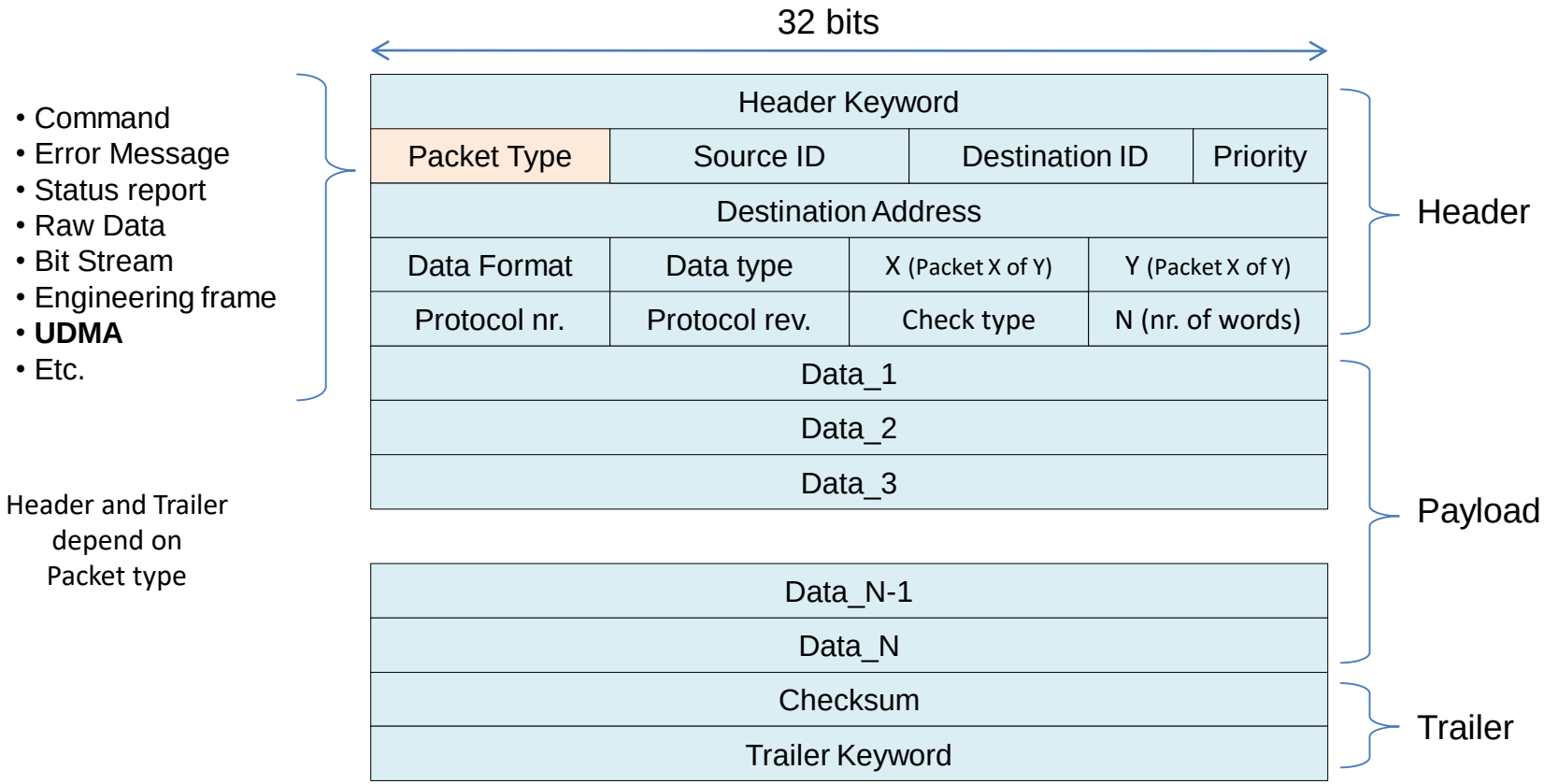


UDMA_CRR (Collection Round Robin)

UDMA_CTFE (Collection Till FIFO Empty)

UDMA instructions for hybrid systems and distributed hardware

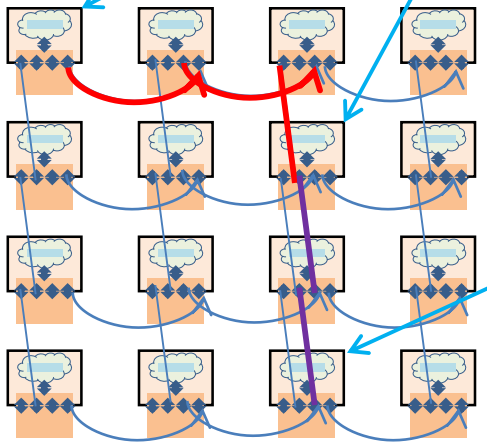
Standardized Data Packets



Standardized *data packets* and corresponding *handling mechanisms* for moving data across entire hybrid systems

UDMA SA DA SAinc DAinc N

(1) **UDMA-Packet** is sent from "i" to "j" to move data from data source "j" to destination "k"



Corresponding *Acknowledge-Packets* can optionally be sent back to conclude transactions

(2) A **Data-Packet** is prepared and sent from data source "j" to destination "k"

At this level of abstraction we don't care about underlying networks and low level communication layers.

Data also include *instructions*, *commands*, *error messages*, etc.

Intro to the Wishbone standard bus-interface

System on Chip: The Wishbone Bus-Interface Standard Definitions

The four main components of the Wishbone system:

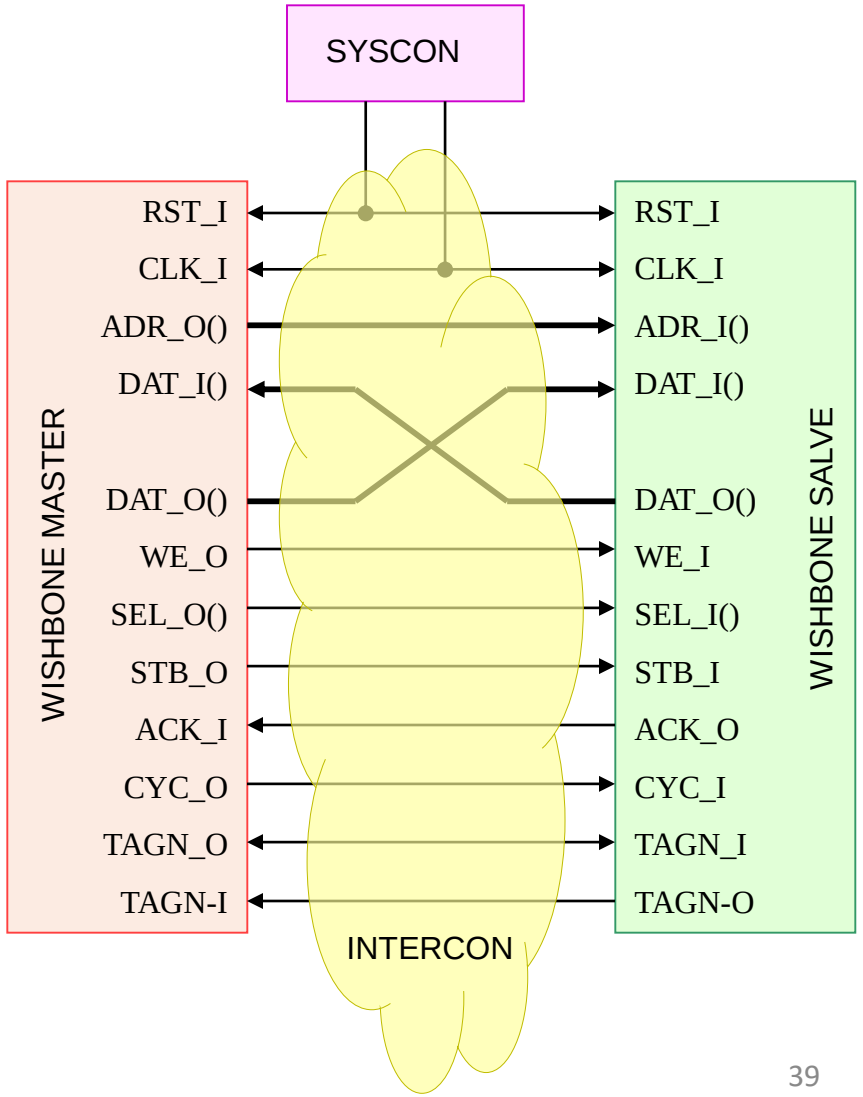
Master and *Slave interfaces*, *Syscon* and *Intercon*.

SYSCON: drives the system clock and reset signals.

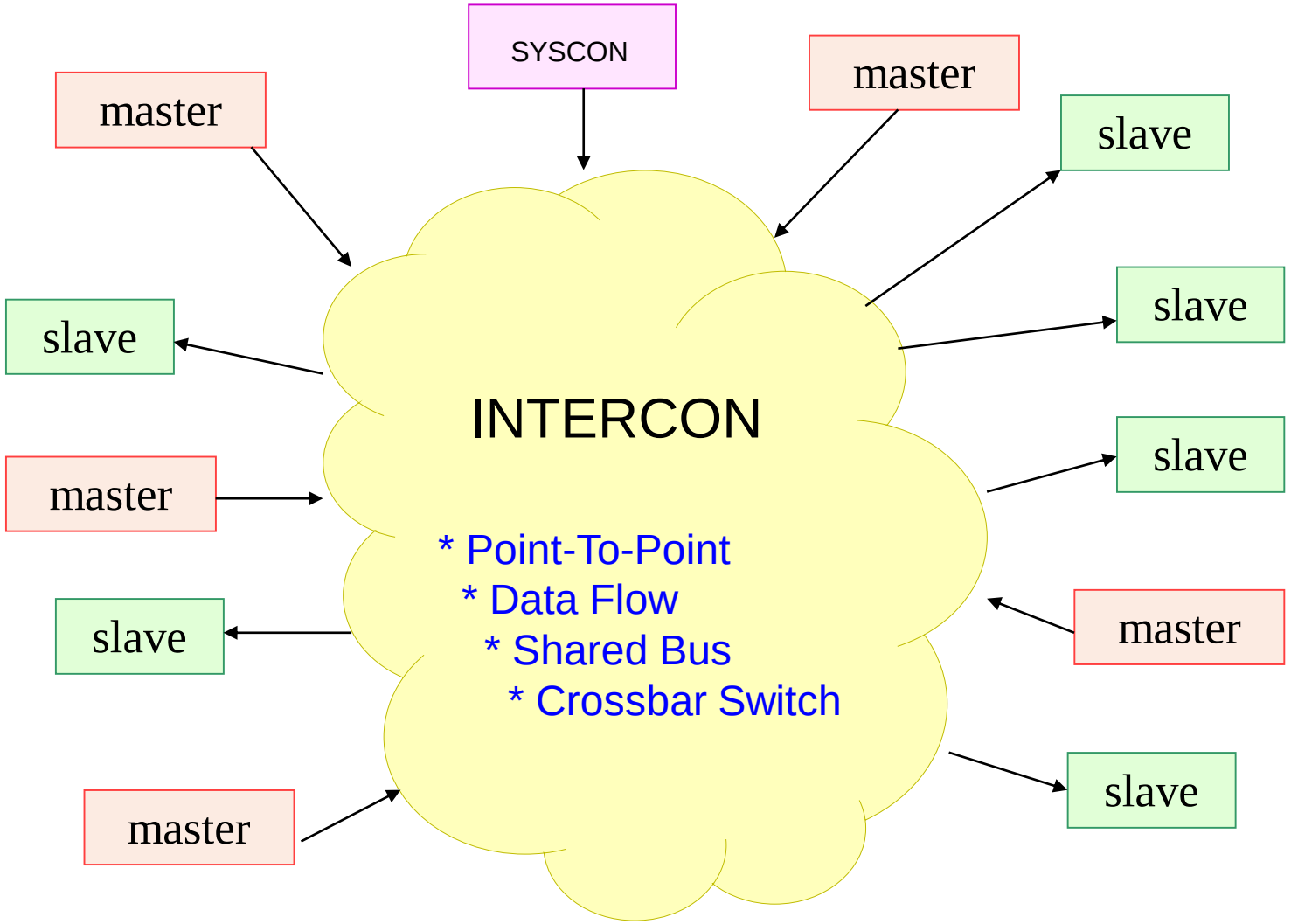
MASTER: IP Core interface that generates bus cycles.

SLAVE: IP Core interface that receives bus cycles.

INTERCON: an IP Core that connects all of the MASTER and SLAVE interfaces together.

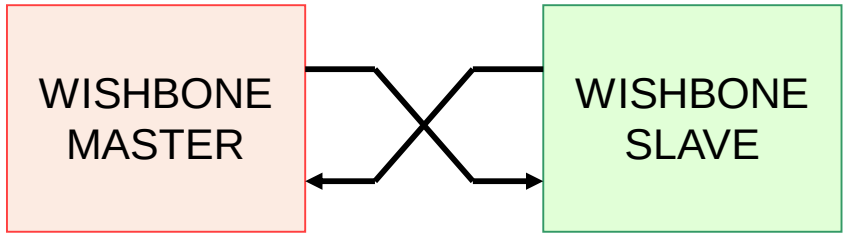


WB Interconnections I

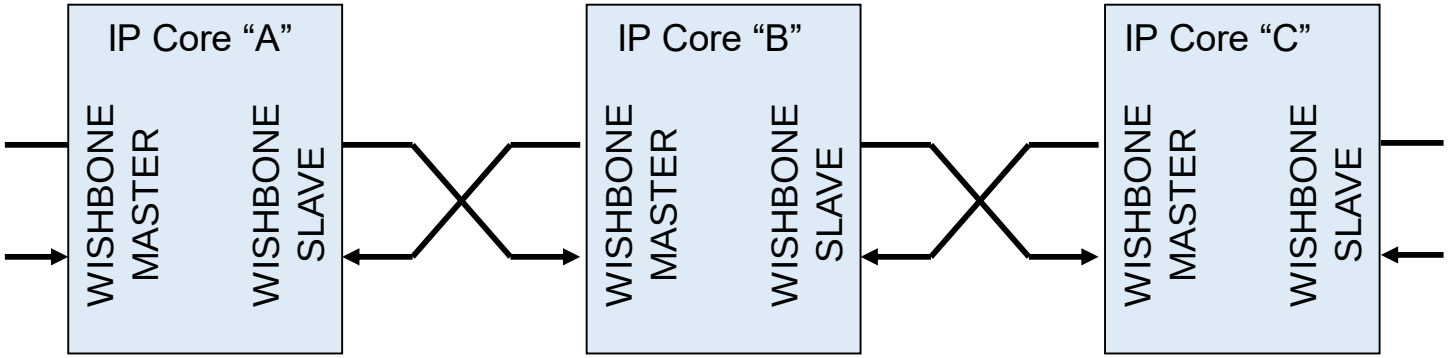


The Wishbone Interconnection is created by the SYSTEM INTEGRATOR, who has total control of its design

WB Interconnections II

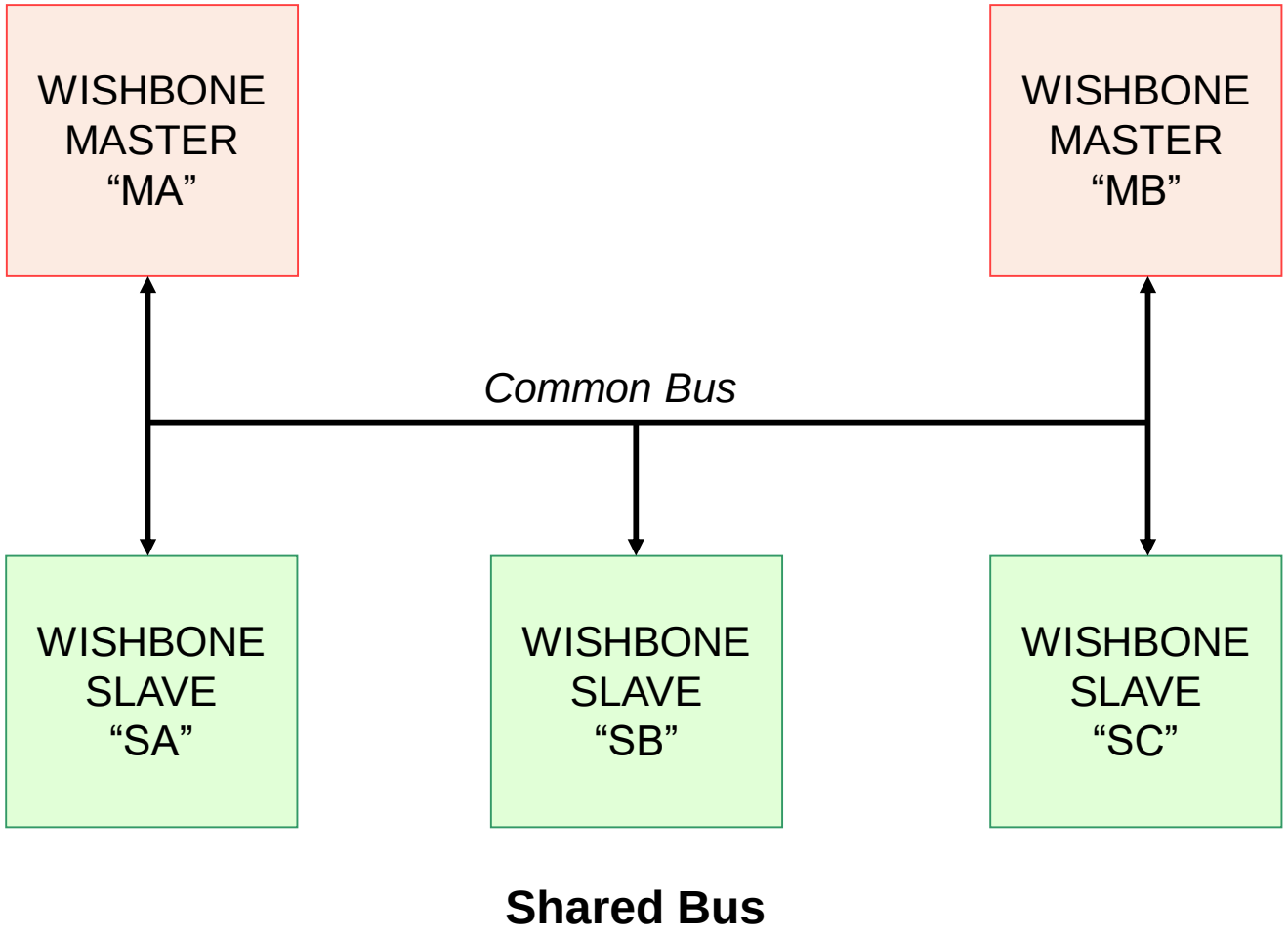


Point-To-Point

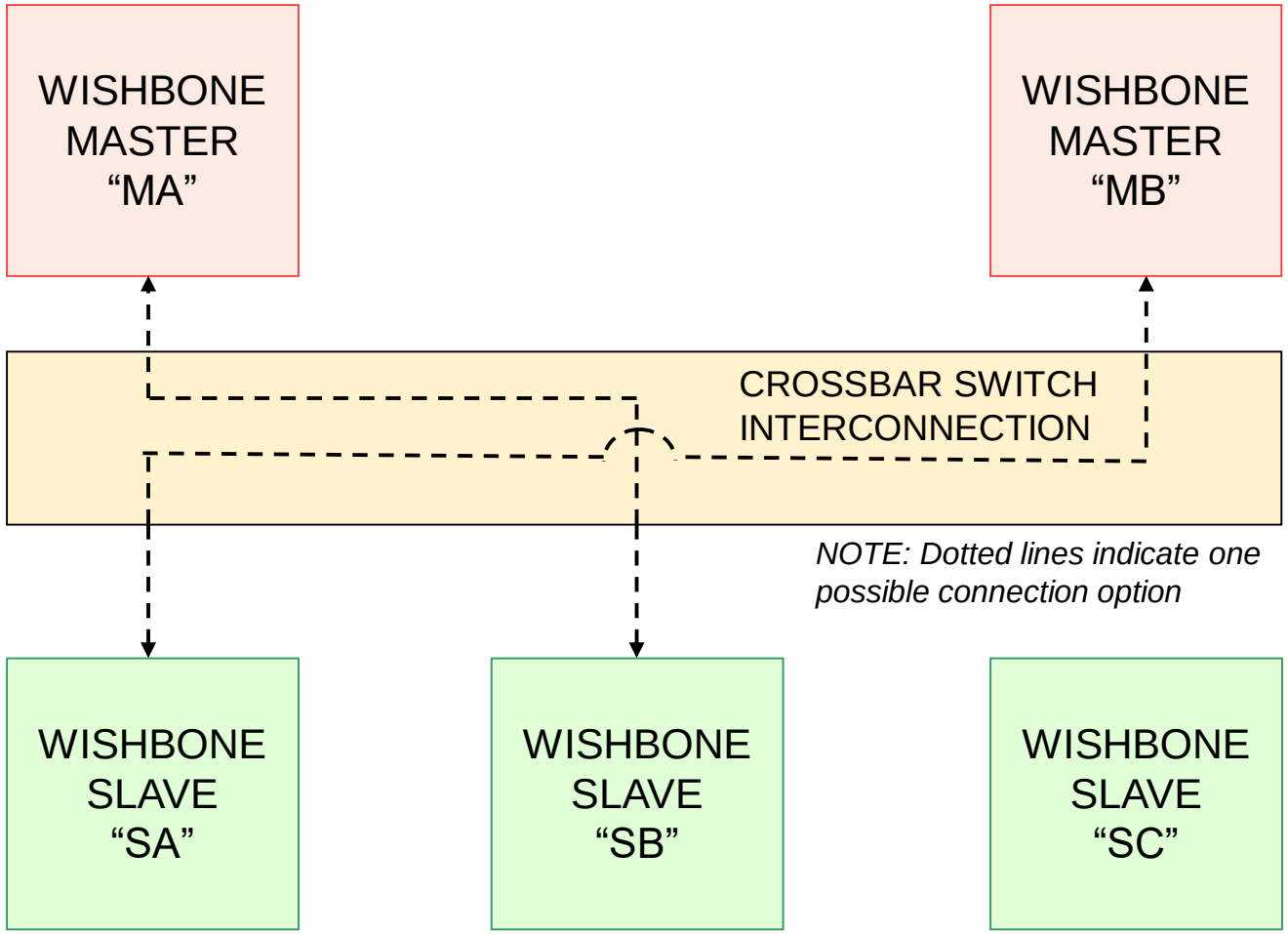


Data Flow

WB Interconnections III

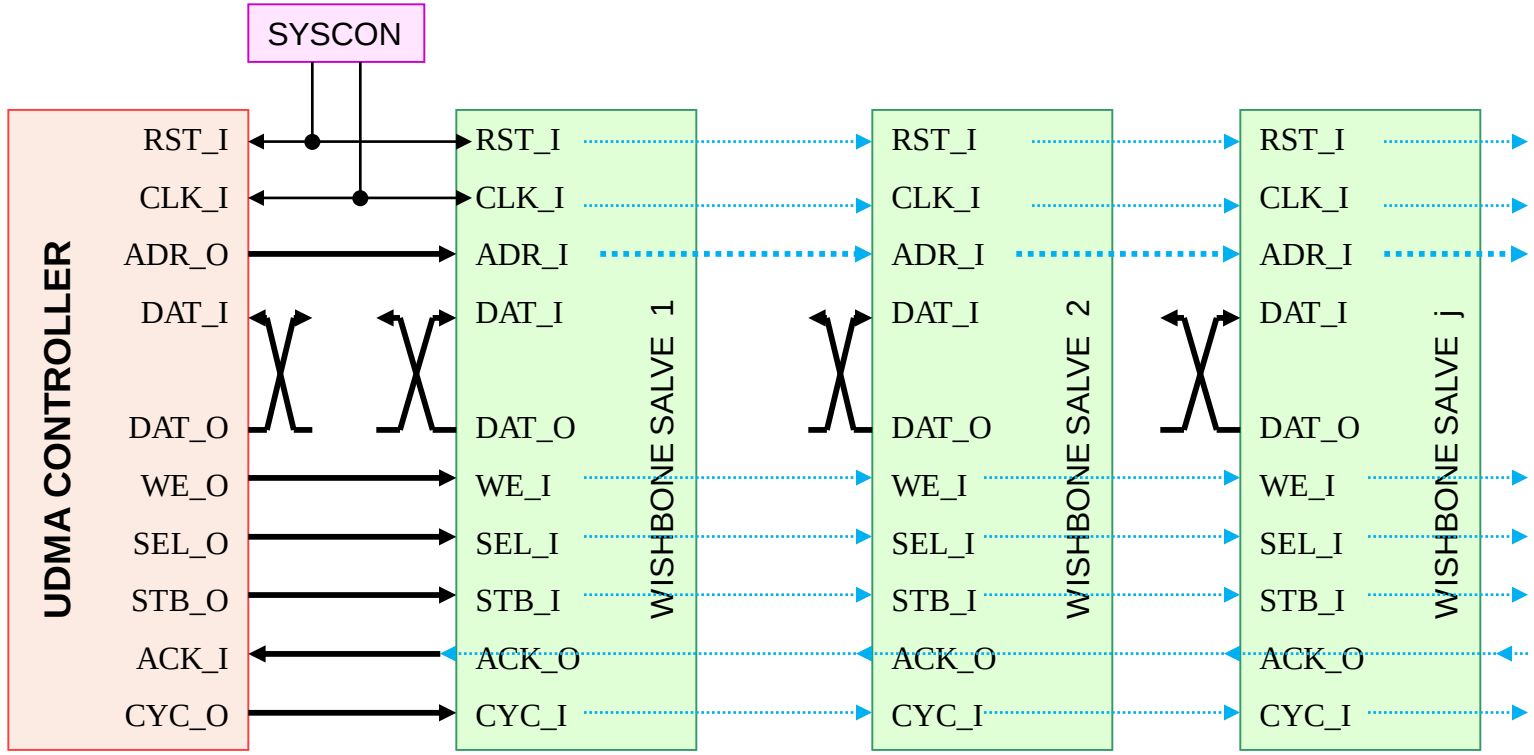


WB Interconnections IV



Crossbar Switch

UDMA controller for a system based on Wishbone compliant modules



- UDMA instructions could be stored in a WB module
- One WB module must be a ***communication block*** which could also store UDMA Instructions in a reserved area.