FPGA Low Level Interface



J.P. Cachemiche, P.Y. Duval, F. Hachon, R. Le Gac, F. Rethore

FPGA Low Level Interface

CPPM 1

Outline

- Motivations for Low level interface
- Fonctional requirements
- Development methodology requirements
- QSYS features
- Conclusion

Outline

- Motivations for Low level interface
- Fonctional requirements
- Development methodology requirements
- QSYS features
- Conclusion

Motivations for a LL Interface

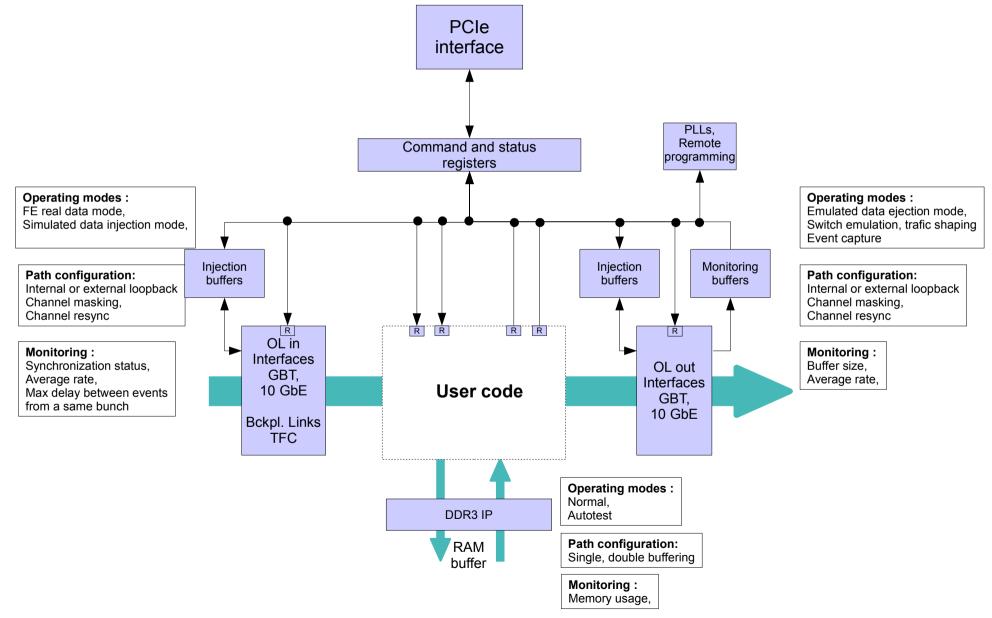
Many firmwares but common data flows and requirements

- Same data path for control
- Same input and output data path for processing
- Need for embedded stand-alone simulators (FE, TFC, LLT, Farm, ...)
- Monitoring buffers

Need for a flexible low level interface in which user code can be "plugged"

- Hide the underlying complexity (GX buffers, GBT, PCIe, 10 GbE, ...)

Low level interface features



Additional requirements (1) GX buffer in Stratix Stratix V complexity increases Large differences with previous versions Very complex GX interfaces CUSTOM PHY GX BUFFE CPU BIPUT eset r phy_mgmt_c **GBT** layer CUSTOM PHY GX BUFFER AVALON INTERFACE phy mamt clk re read w_mgmt_cik_reset phy mgmt clk reset to the phy mg \rightarrow need to encapsulate serial data nhy m dress[8_0 phy mont address to the phy months 01 nhy mant read. phy mant read to the phy mant readdata from the phy mgmt[31.0 GX phy mant waitrequest y_mgmt_waitrequest_from_the_phy_mgm clkout phy_mgmt_write phy mant write to the phy mant w mamt writedata to the phy mant/31.0 pll_ref_ clkout 25 times larger than Stratix I used on Tell1 pl ref cl nx serial da narallel data rx serial data[31,0 rx parallel data[3] i data tx_parallel_da onfig from xcvr 7 hours to compile a nearly empty design el data[31_0] inandly from reconfig to xe with a 4 cores PC embedding 16 Gb of RAM GX buffer in Stratix \rightarrow Incremental compilation mandatory CLISTOM PHY RECONFIG AVAI ON INTERFACE to shorten design cycles NIOS mamt clk mgmt_rst_re nfig_to_xcvr mt rst re reconfig mamt clk reset to the reconfig mam reconfia ma reconfig mamt address to the reconfig mamt/6.01 **Many firmwares** config mant read Reconfig recontig mamt read to the recontig mamt data from the reconfig mamt[31..0] notic mant readdata(31 econfia mant waitrequest ig_mgmt_waitrequest_from_the_reconfig_mgmt econfig_mgmt_write econfig_mgmt_write_to_the_reconfig_mgmt config mant writedata to the reconfig mant[31..0] reconfig_from_xc Clear separation between user code at xowr recond and common code \rightarrow Object oriented approach

with well defined interface specification

Additional requirements (2)

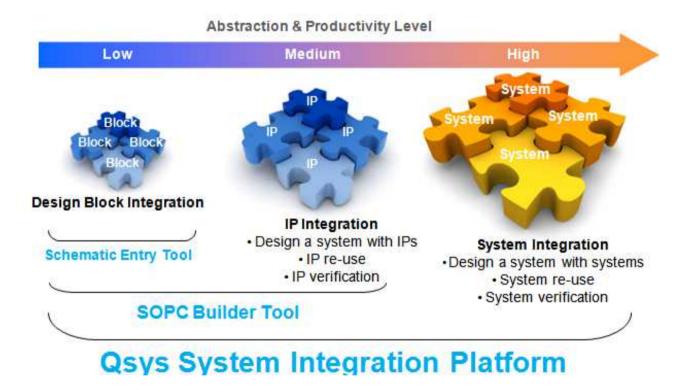
Development tools

- Simple and if possible unique tool
- Hierarchical approach
- Quick redesign
- Test bench for every developed function

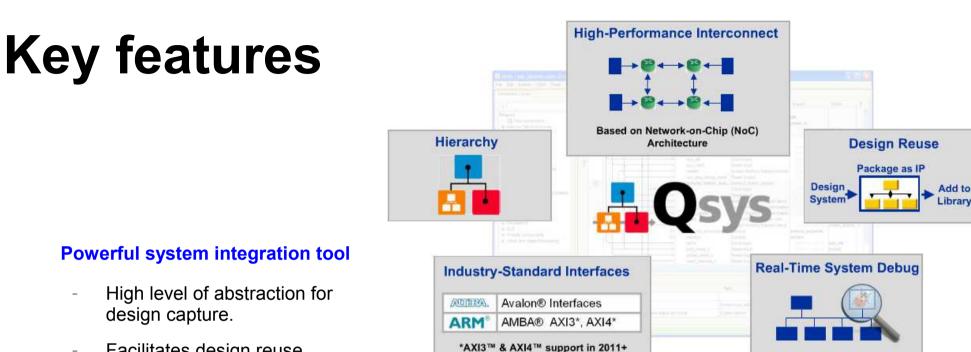
Marseille proposal

- Quartus + QSYS

What is QSYS ?



QSYS: Generalisation of SOPC builder for **designing at system level**



Facilitates design reuse

Save time by avoiding writing HDL code for interconnection

- Automatically creates high-performance interconnect logic.

Easy way to normalize interfaces in the system

- Standard interfaces
- Documentation maintained and already available

Automatic test bench tools

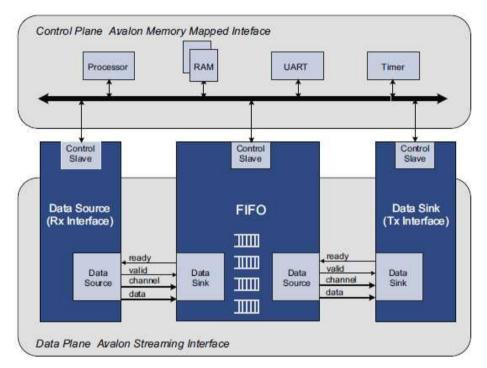
Standard interfaces

Mainly two kinds:

- Memory mapped interfaces:
 - \rightarrow control plane
 - \rightarrow Reading and writing of control registers
- Streaming interface:
 - \rightarrow data plane
 - → Data switching (muxing and demuxing), aggregation, bridges

Simplify design entry and team-based design

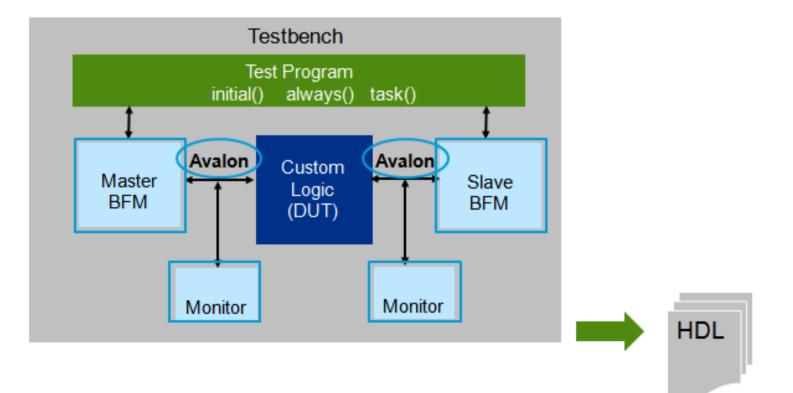
- Signal behavior defined by interface
- Simplified documentation
- No manual wiring or mapping of control, data, and status signals
- Easy system changes



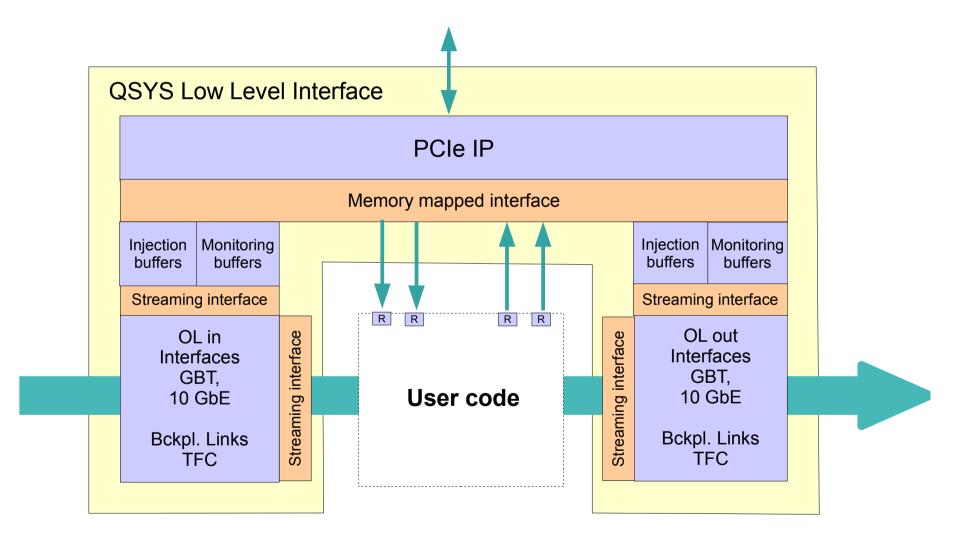
	cik	1			
		국산			
	write_n	-	\		
	chipselect =				
	waitrequest	1			
	address) A0	AI	X	
	burstcount 📋) 1) 2	X	
	byteenable	BEO) EE1 X (BE2	X	
	writedata	You	01 X XD2	X	
	Decomposition -		·······		
Read Waveform					
	cik	100			
	write_n =			يصا اسا ا	
	chipselect				
			1		
	waitrequest	<u> </u>	1		
	waitrequest) (A0	<u> </u>		
		A0 8E0	<u> </u>		

Test benches

- High level function calls through Bus Functional Modules (BFM)
- QSYS automatically generates a test bench



Low level interface implementation



Conclusion

Interesting features in QSYS

- Quick and high level design
- Independent objects linked by a standard interface
- Allows team design with components sharing
- We will test all the concepts during debug
- **First conclusions** and first encapsulation of low level interface **by end of year**

Concurrent team design

- Specification of interfaces is a priority requirement
 - → **Draft specification** for low level interface circulated **in coming months**

Compilation is tremendously long

- Use partitioning and incremental design
- Steal gamers PCs to your children or order very powerful PCs !
 - \rightarrow You need at least **20 Gb of RAM and 64 bits processor** (32 bits does not compile !)
 - \rightarrow 4 cores or more strongly advised.