FPGA Low Level Interface

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

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

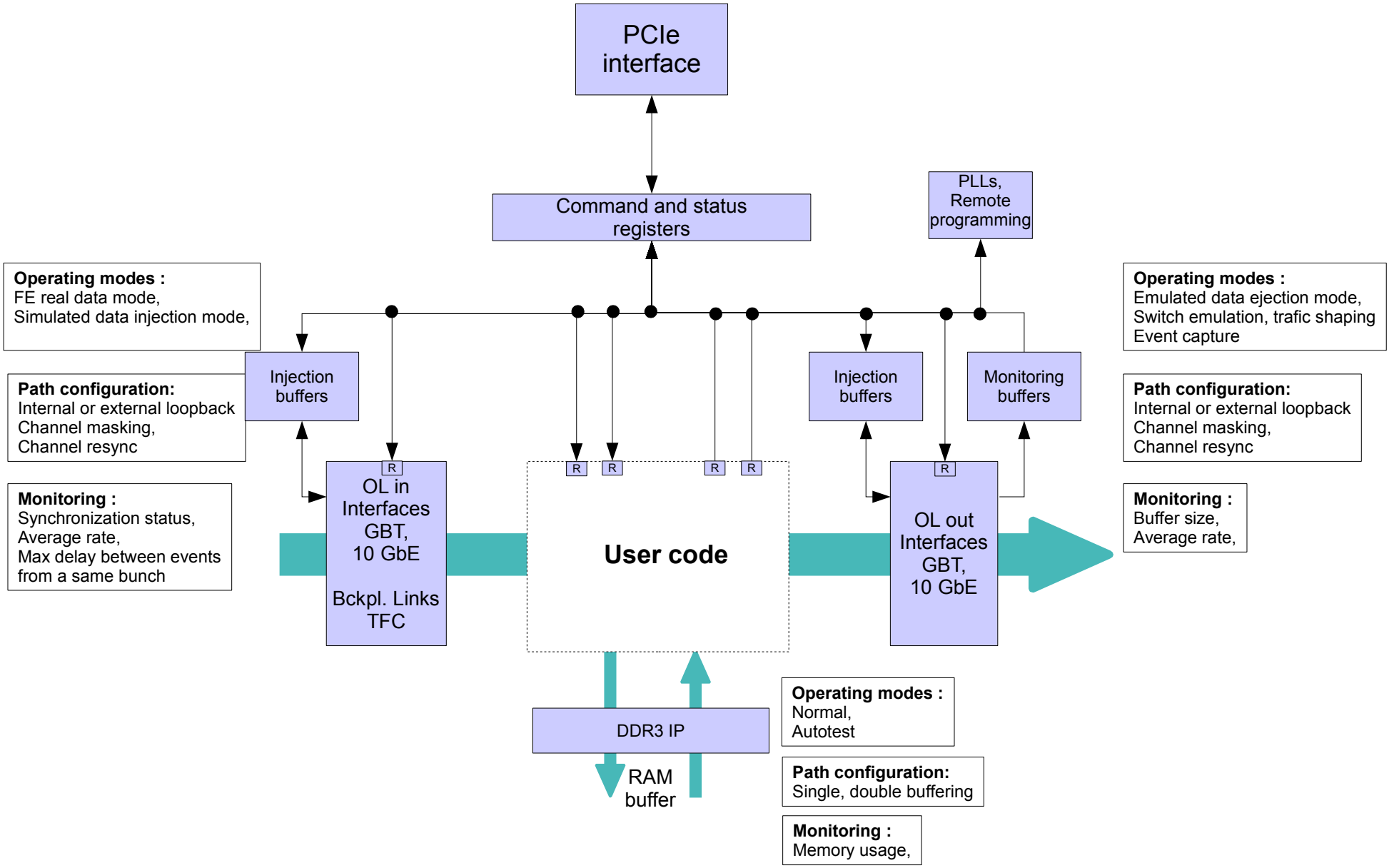
Many firmwares but common data flows and requirements

- 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

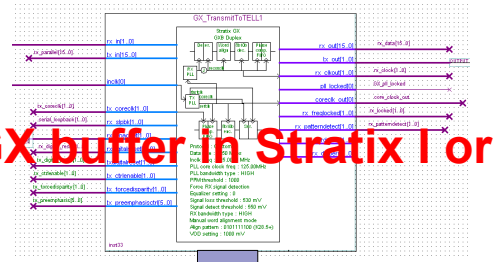
Need for a flexible low level interface in which user code can be “plugged”

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

Low level interface features



Additional requirements (1)



GX buffer in Stratix I or II

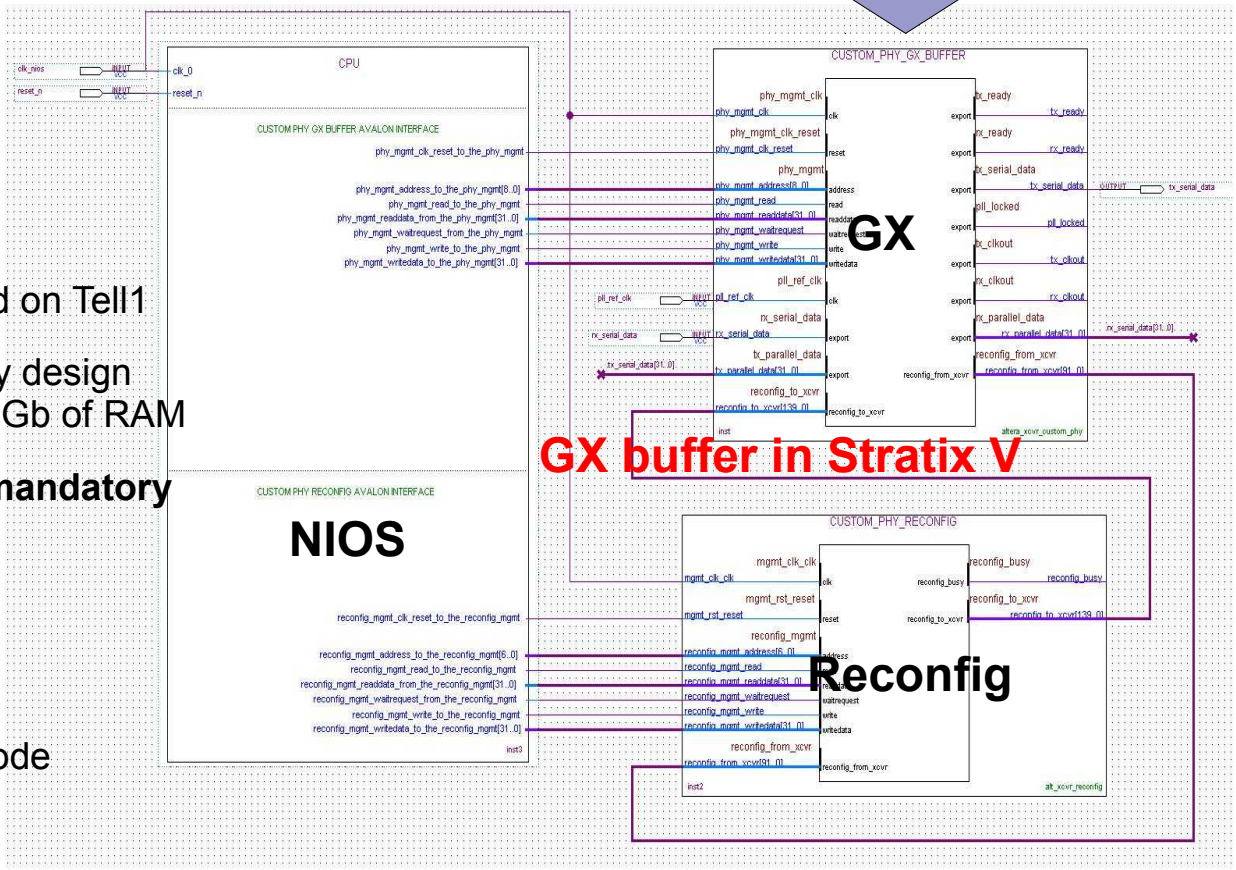
Stratix V complexity increases

- Large differences with previous versions
- Very complex GX interfaces
- GBT layer

  → **need to encapsulate**

- 25 times larger than Stratix I used on Tell1
- 7 hours to compile a nearly empty design with a 4 cores PC embedding 16 Gb of RAM

  → **Incremental compilation mandatory to shorten design cycles**



GX buffer in Stratix V

Many firmwares

- Clear separation between user code and common code

  → **Object oriented approach with well defined interface specification**

Additional requirements (2)

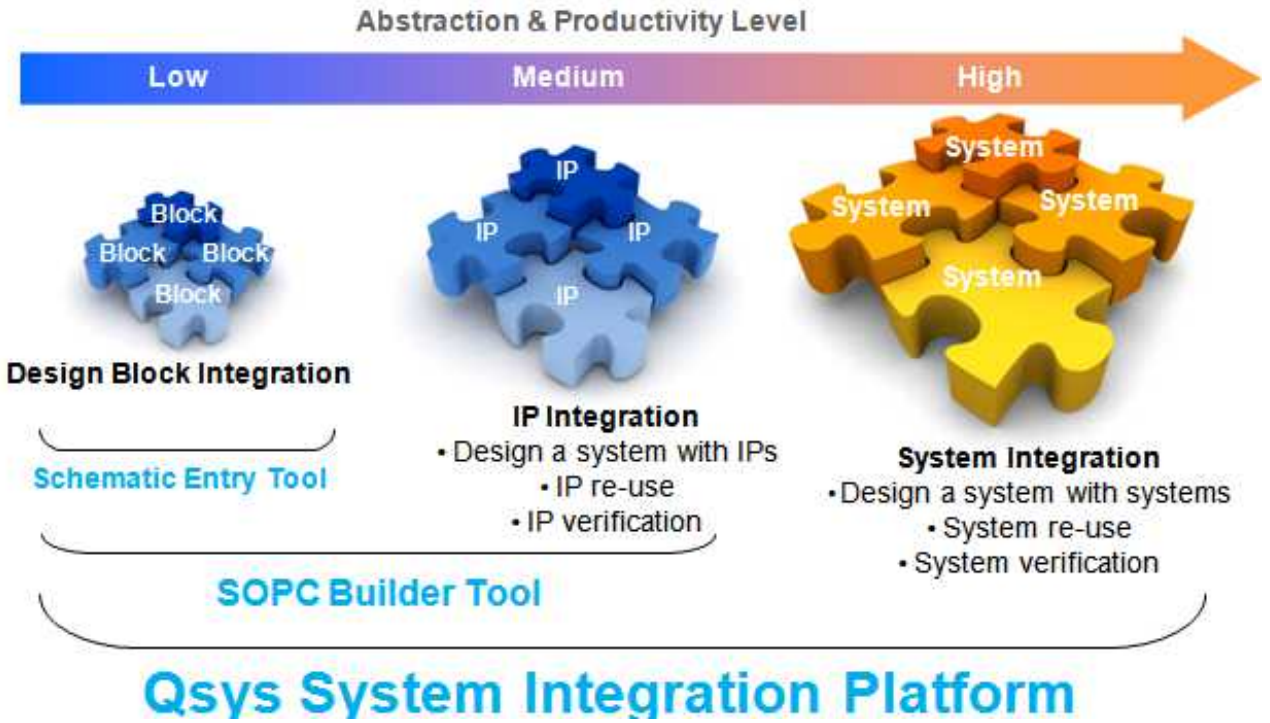
Development tools

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

Marseille proposal

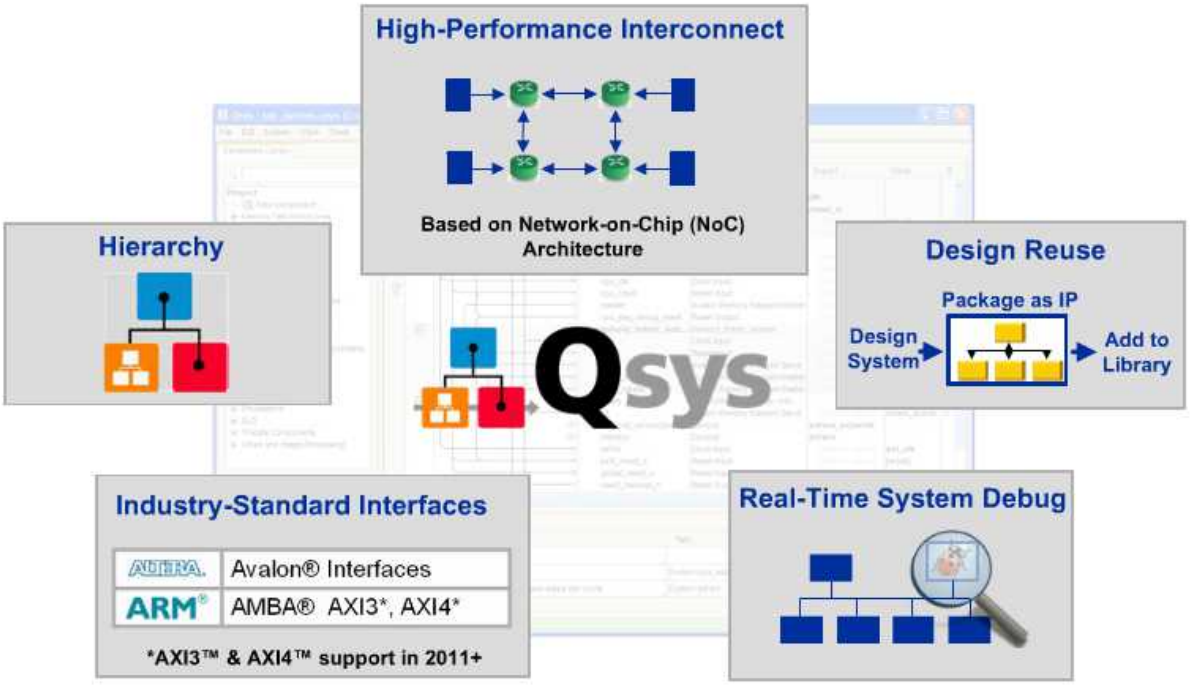
- **Quartus** + **QSYS**

What is QSYS ?



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

Key features



Powerful system integration tool

- High level of abstraction for design capture.
- Facilitates design reuse

Save time by avoiding writing HDL code for interconnection

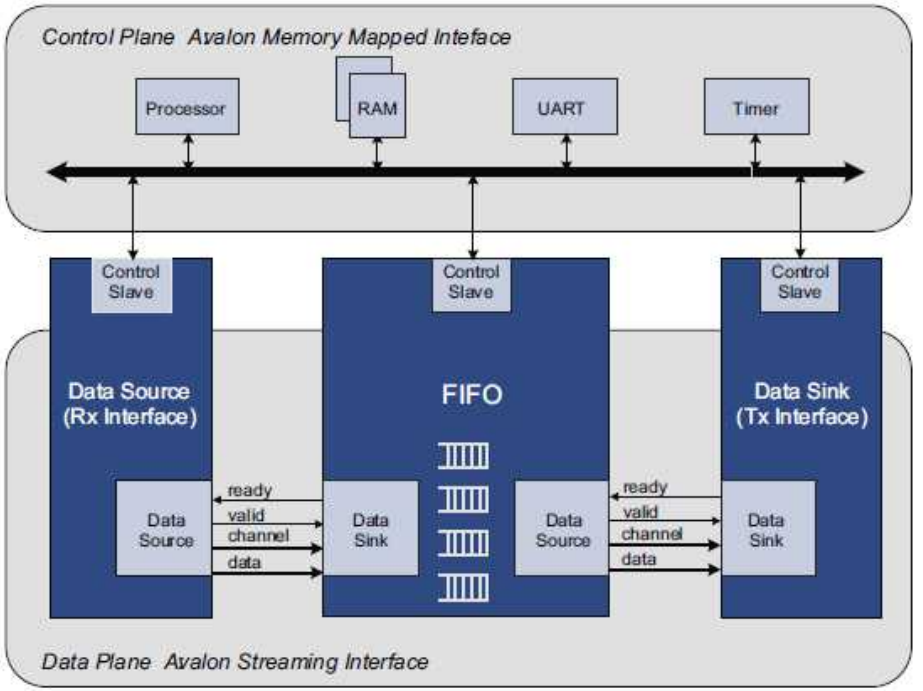
- Automatically creates high-performance interconnect logic.

Easy way to normalize interfaces in the system

- Standard interfaces
- Documentation maintained and already available

Automatic test bench tools

Standard interfaces

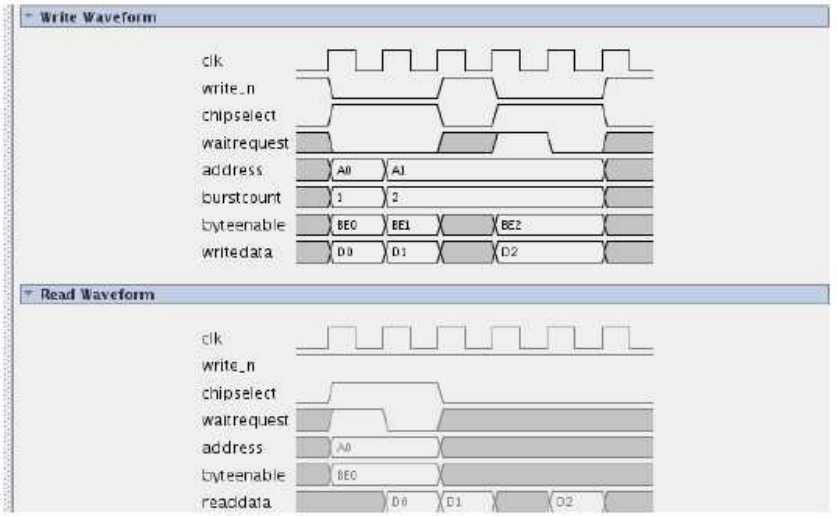


Mainly two kinds:

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

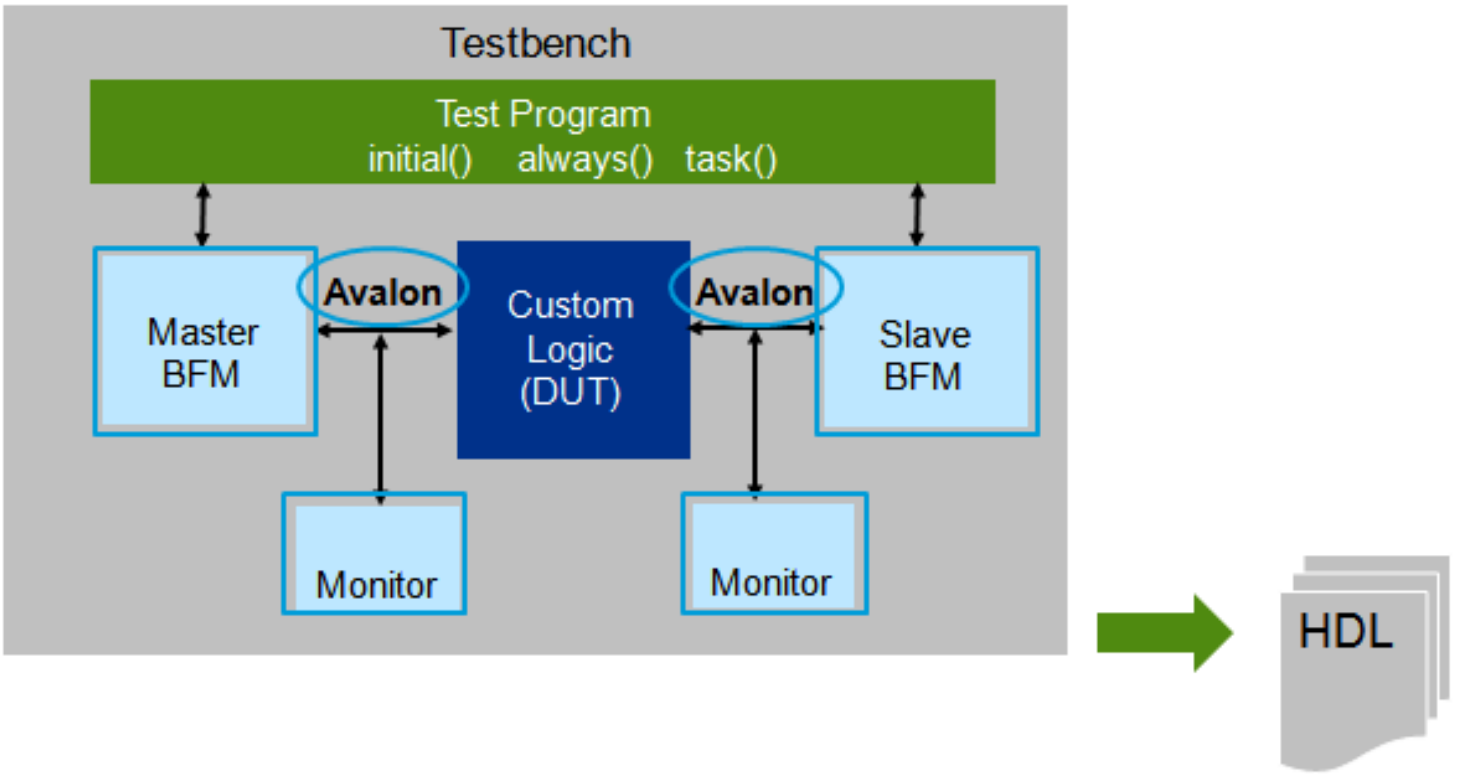
Simplify design entry and team-based design

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

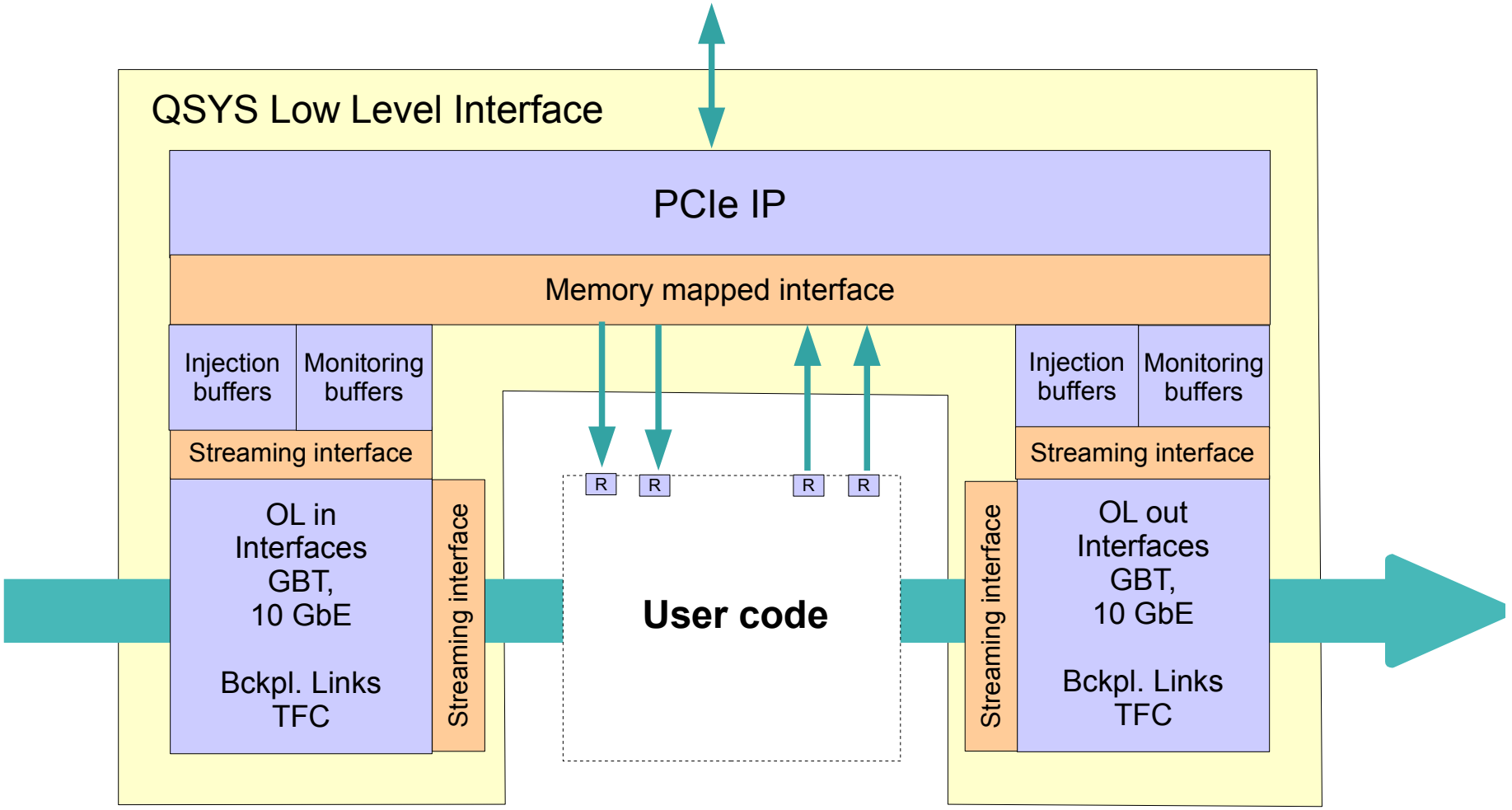


Test benches

- 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 !
  - → You need at least **20 Gb of RAM and 64 bits processor** (32 bits does not compile !)
  - → 4 cores or more strongly advised.