RUN 4 upgrade: FoCal E pixel readout



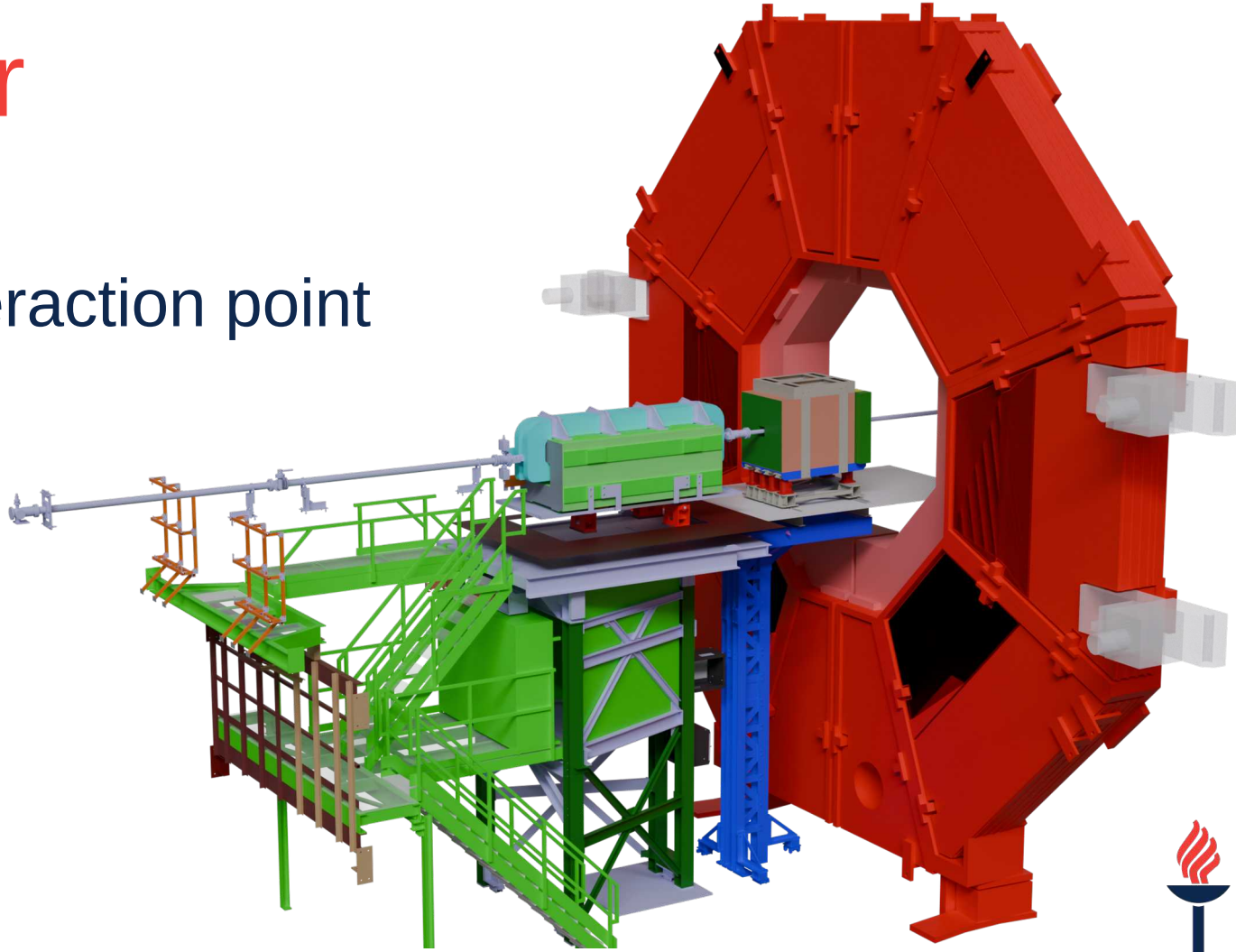
Goal of FoCal

- Direct ɣ (photon) measurements
  - π^0, ω, η identification
- Correlations between
  - π^0, γ_{dir}, Jet
- Ultra peripheral (UPC) J/ψ production



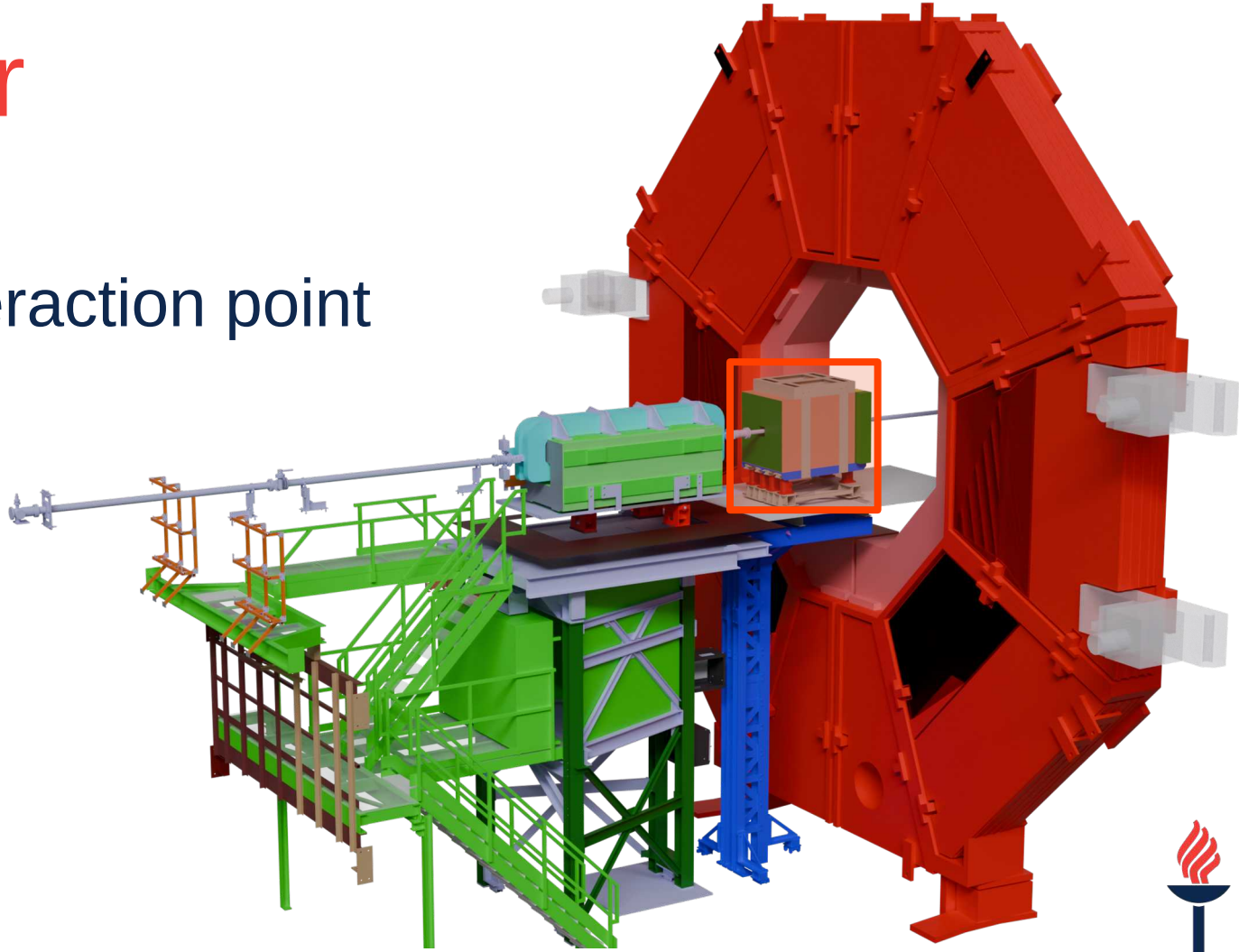
The detector

- Forward region
  - 7m from the interaction point
  - $3.4 < \eta < 5.5$



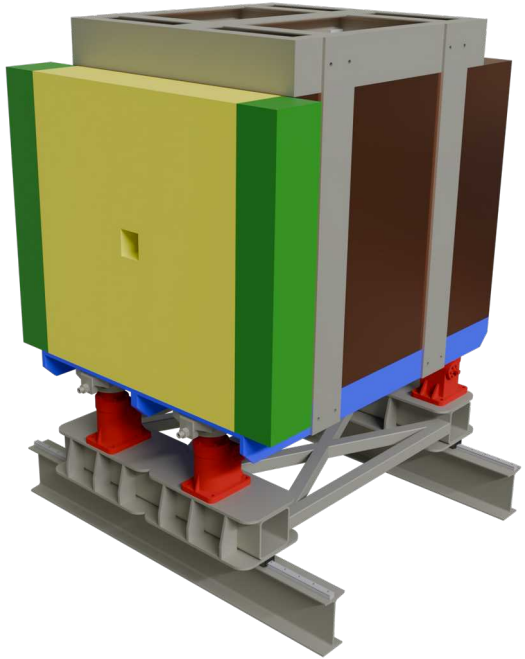
The detector

- Forward region
  - 7m from the interaction point
  - $3.4 < \eta < 5.5$



FoCal

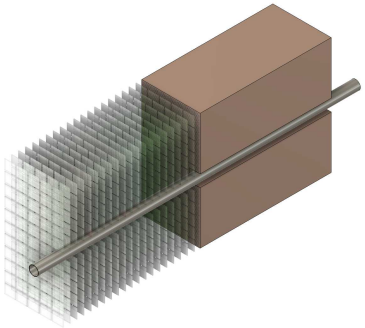
Two subdetectors:

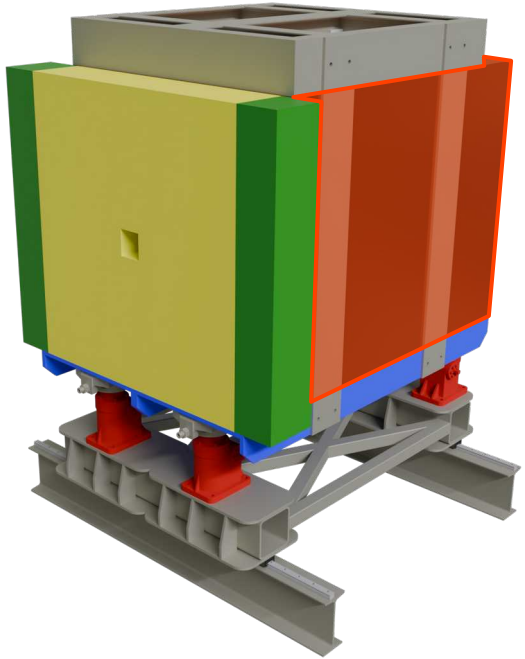


FoCal

Two subdetectors

- FoCal-H
  - Hadronic showers

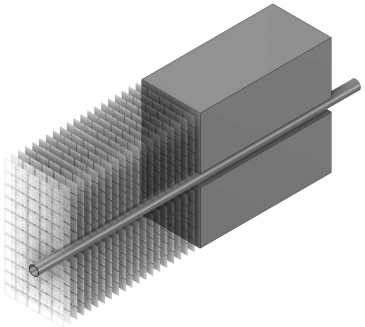


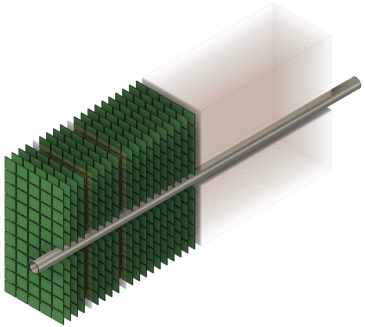


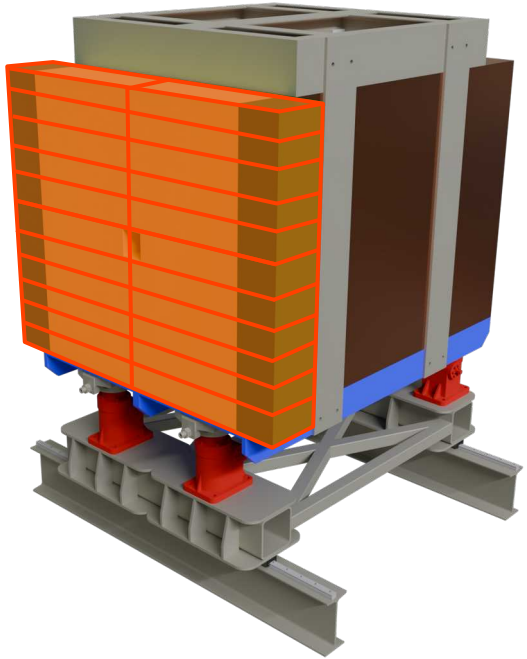
FoCal

Two subdetectors

- FoCal-H
  - Hadronic showers
- FoCal-E
  - EM showers
  - 22 modules
  - FoCal-E pads



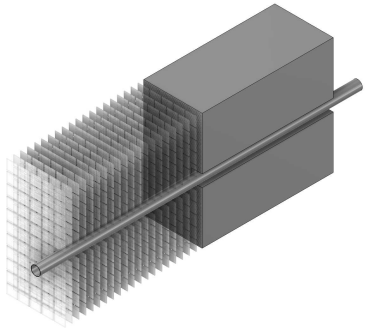


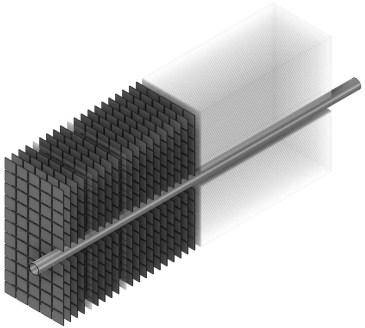


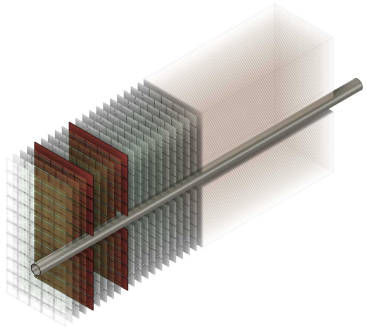
FoCal

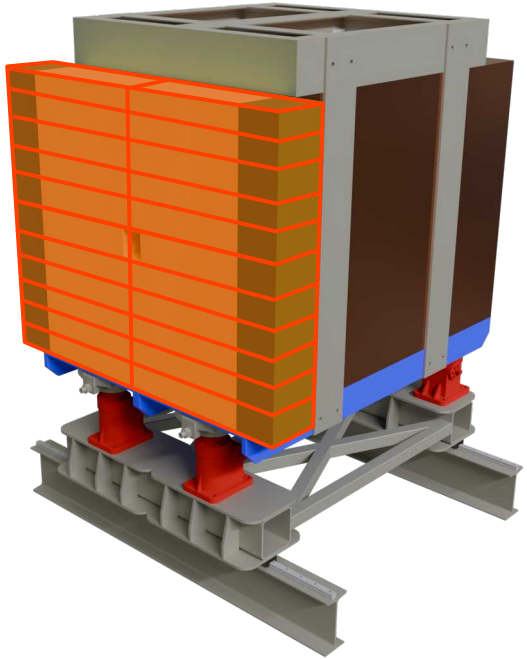
Two subdetectors

- FoCal-H
  - Hadronic showers
- FoCal-E
  - EM showers
  - 22 modules
  - FoCal-E pads
  - FoCal-E pixels



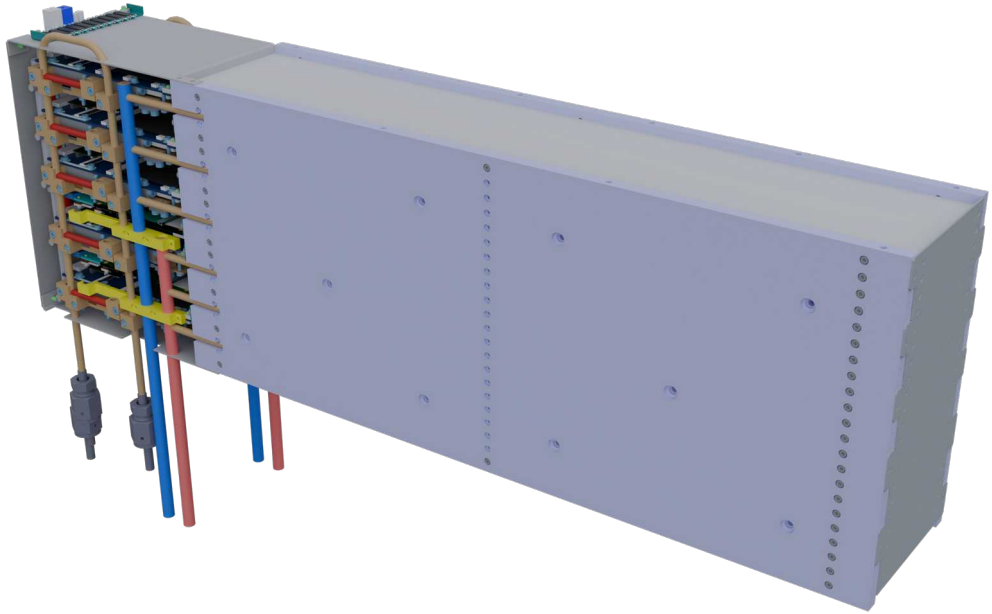






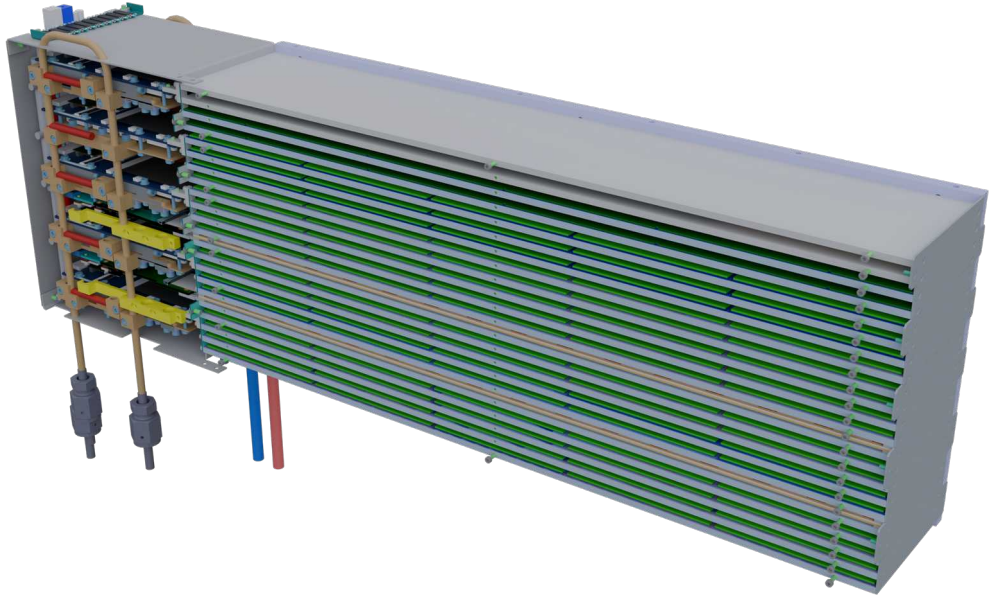
FoCal-E modules

- Readout and cooling



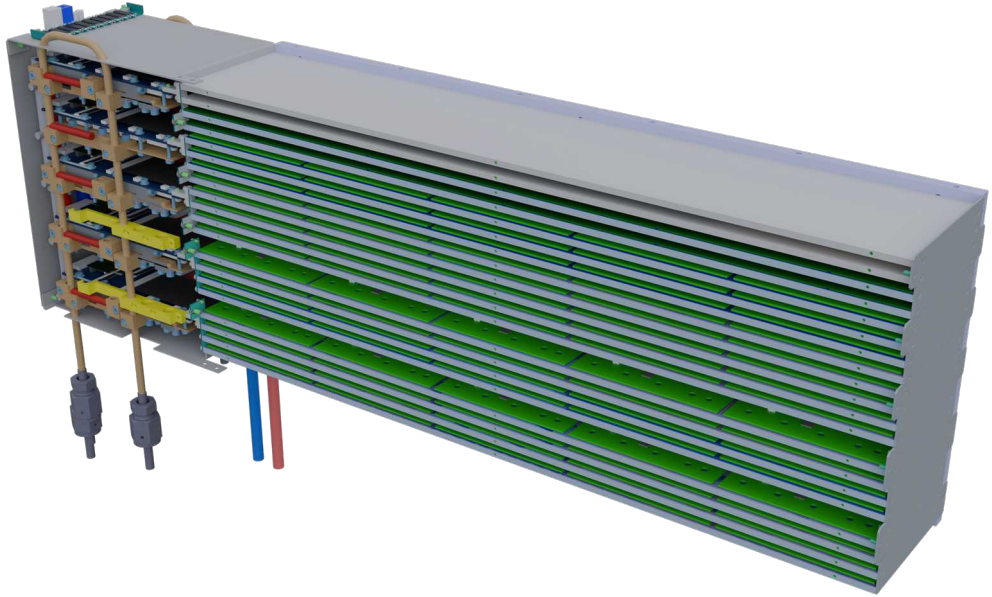
FoCal-E modules

- Readout and cooling
- 20 layers
  - Absorber: Tungsten
  - Active material



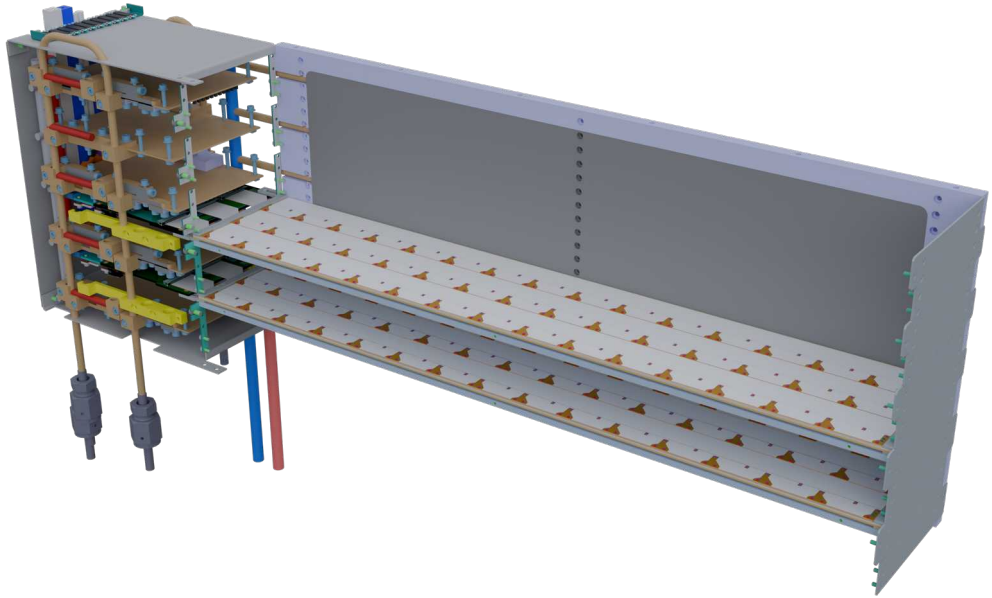
FoCal-E modules

- Readout and cooling
- 20 layers
  - Absorber: Tungsten
  - Active material
    - 18 layers of Pads
      - Energy deposition



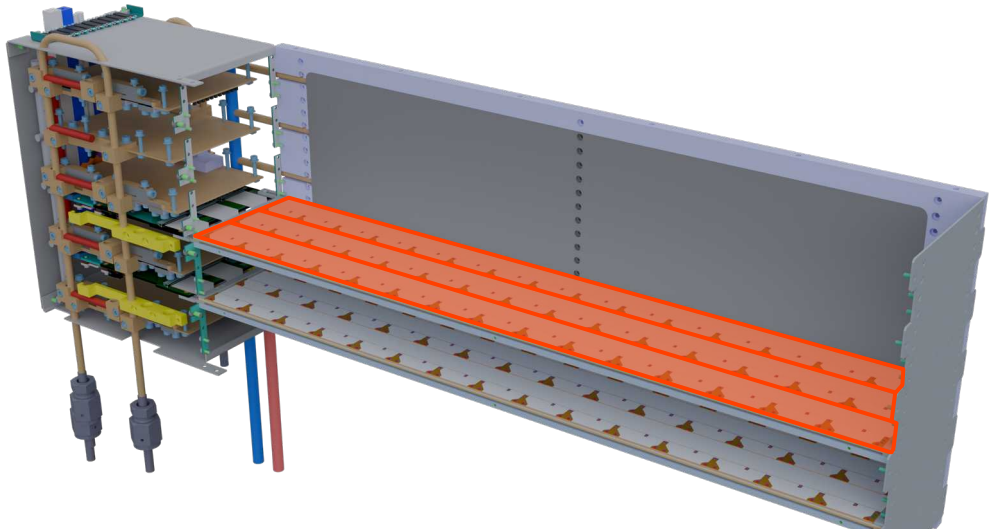
FoCal-E modules

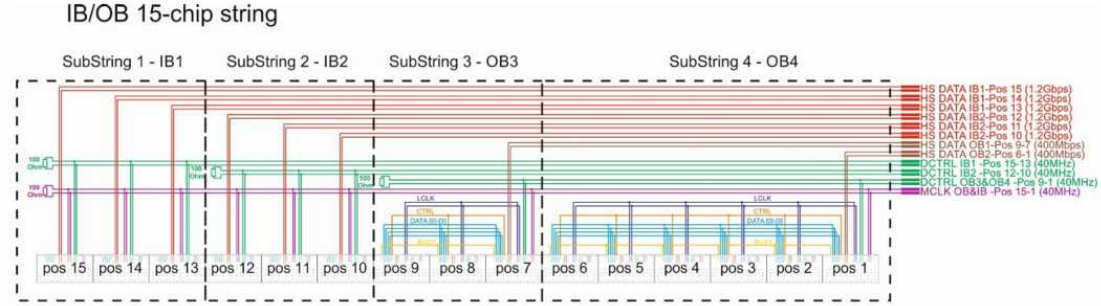
- Readout and cooling
- 20 layers
  - Absorber: Tungsten
  - Active material
    - 18 layers of Pads
      - Energy deposition
    - 2 layers of Pixels
      - positioning



FoCal-E pixels

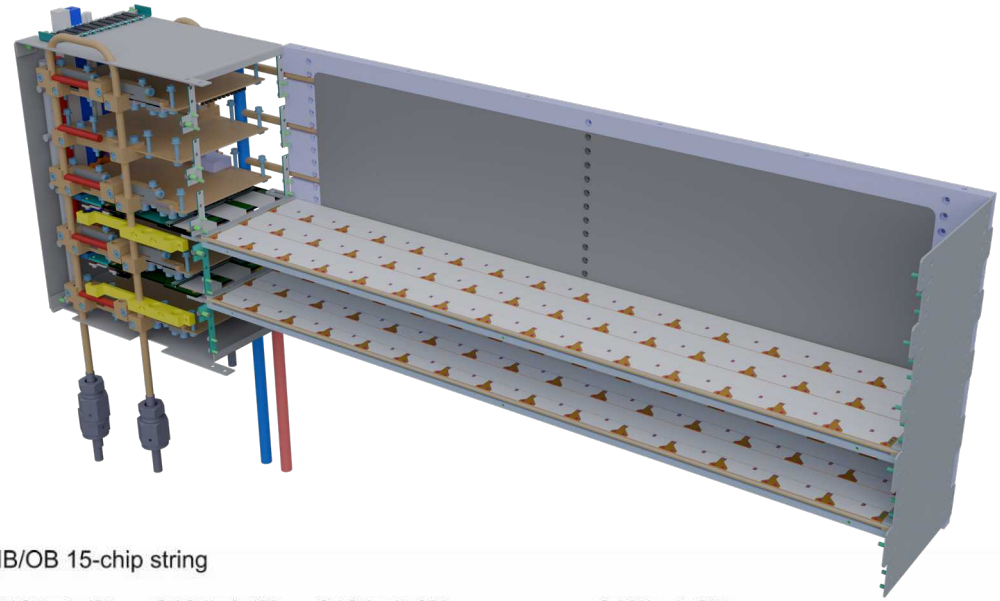
- 6 strings

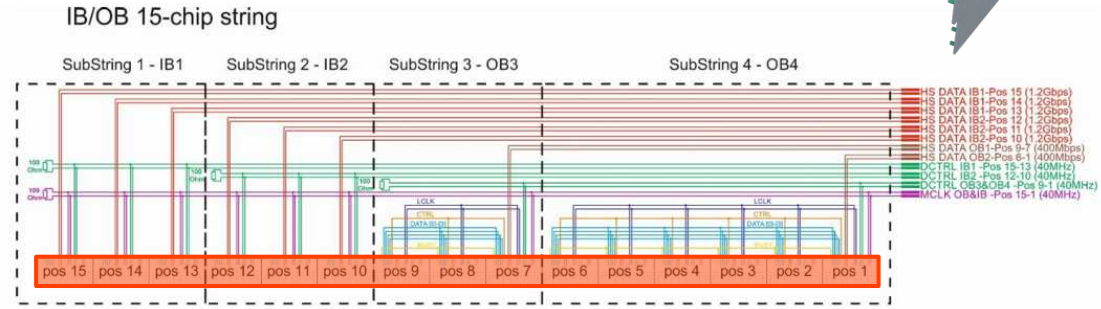




FoCal-E pixels

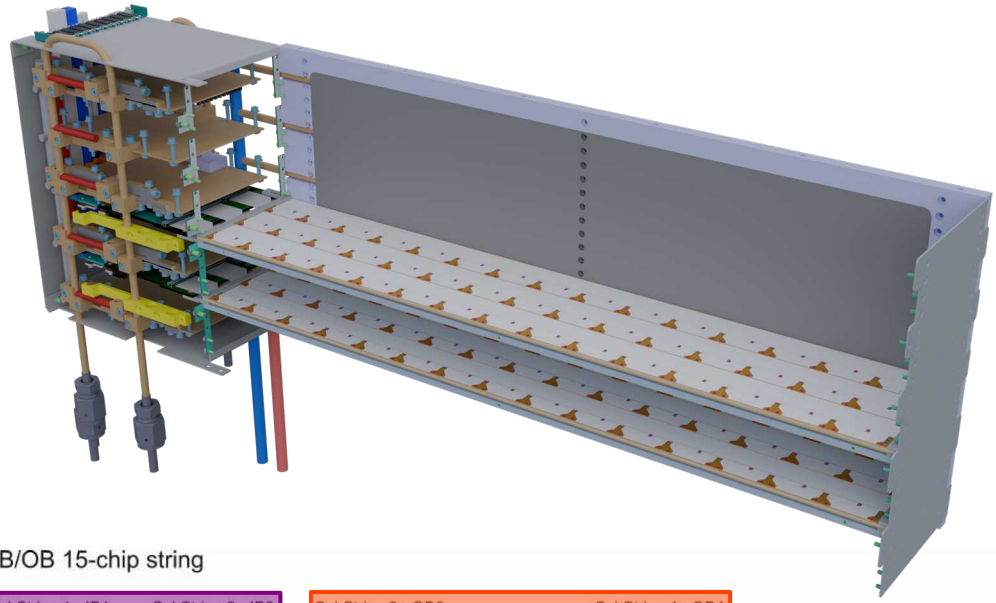
- 6 strings
  - 15 ALPIDE chips





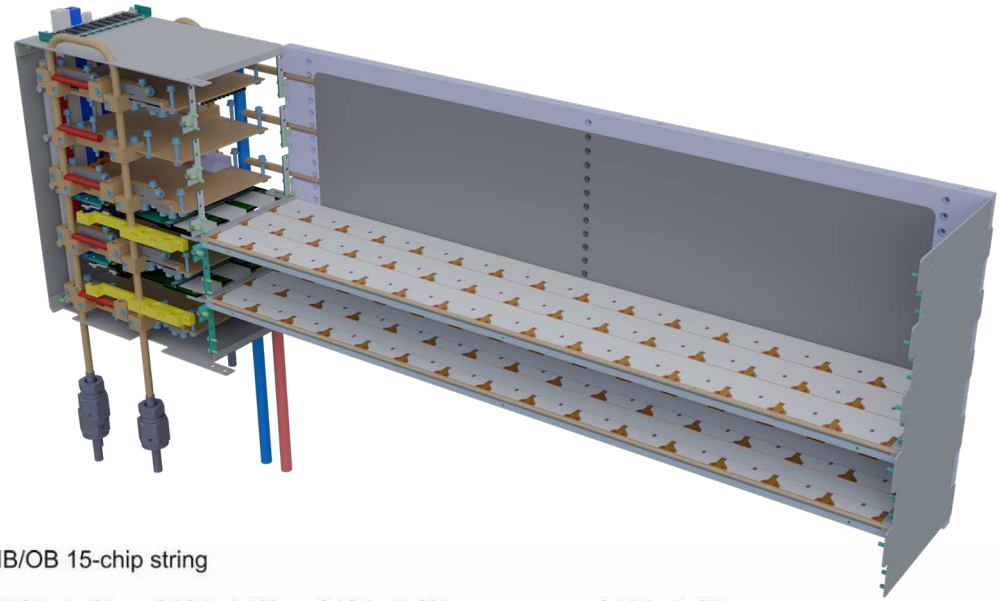
FoCal-E pixels

- 6 strings
  - 15 ALPIDE chips
  - Outer barrel
  - Inner barrel



FoCal-E pixels

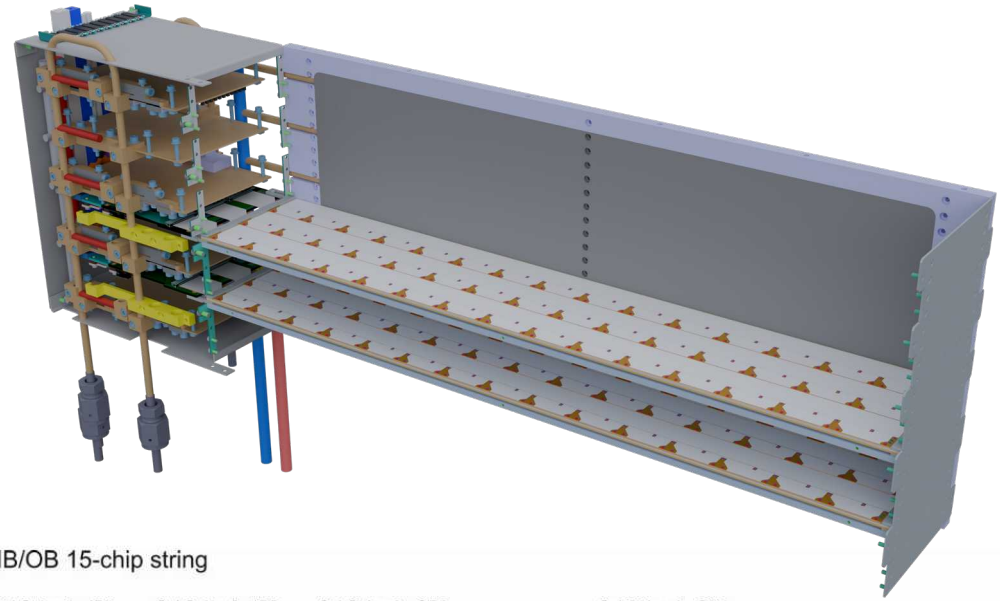
- 6 strings
  - 15 ALPIDE chips
  - Outer barrel
    - 1 master chip
  - Inner barrel
    - Only master chips

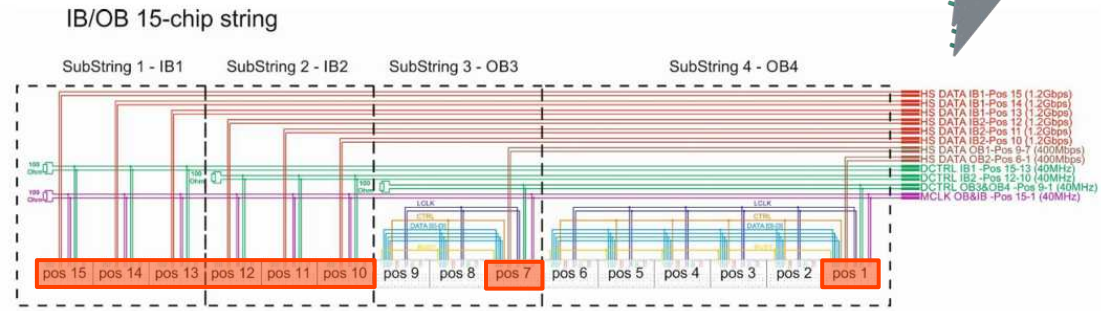




FoCal-E pixels

- 6 strings
  - 15 ALPIDE chips
  - Outer barrel
    - 1 master chip
  - Inner barrel
    - Only master chips

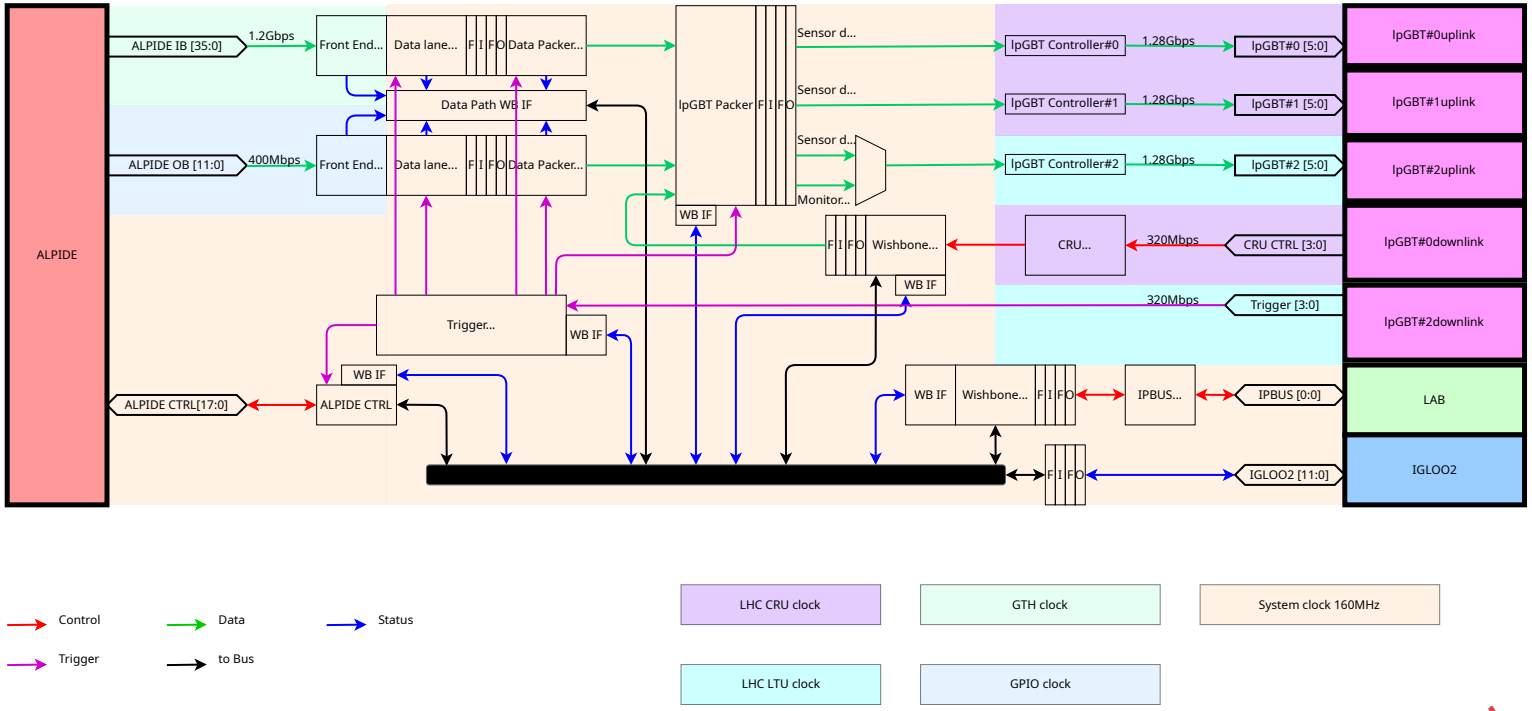




- Master chip Connected directly to FPGA readout

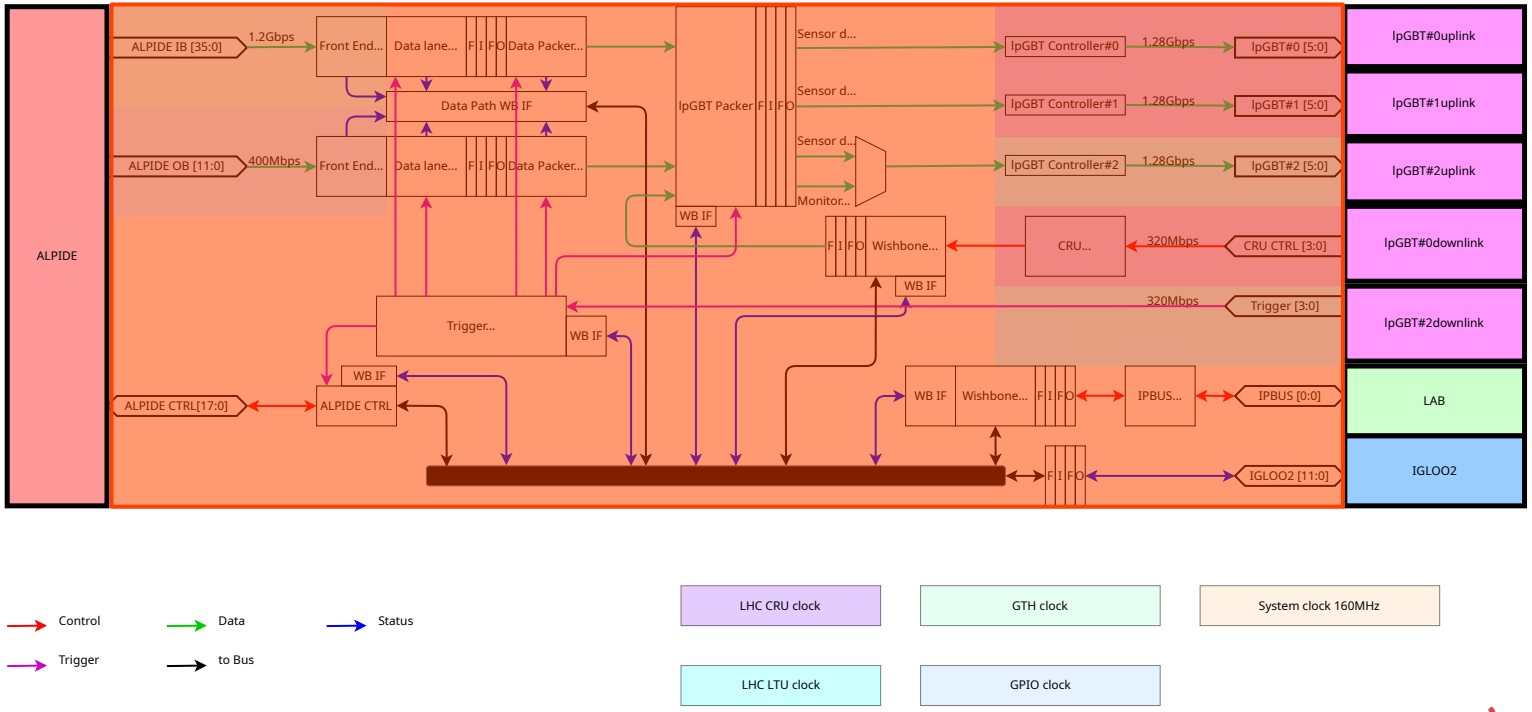
FoCal-E pixels readout

- Complex



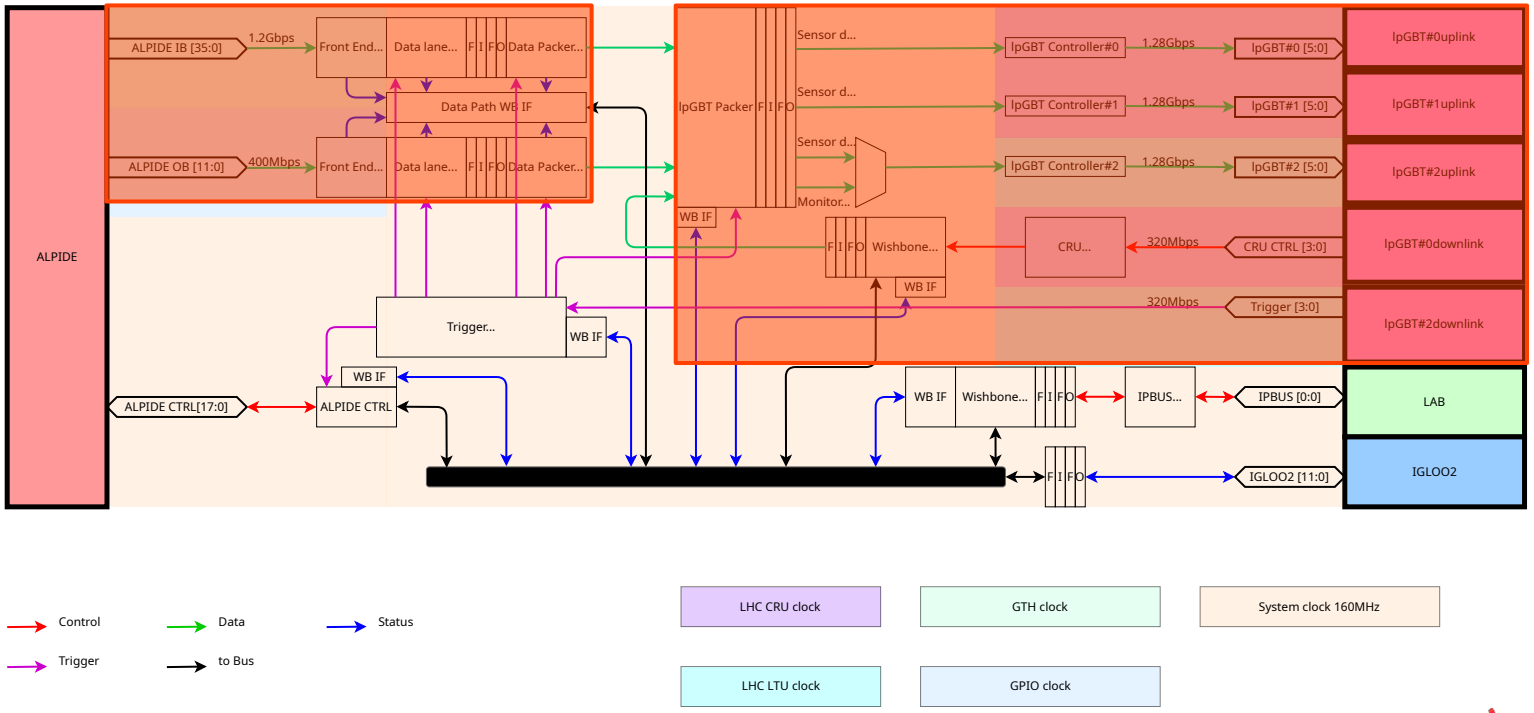
FoCal-E pixels readout

- Complex
- FPGA



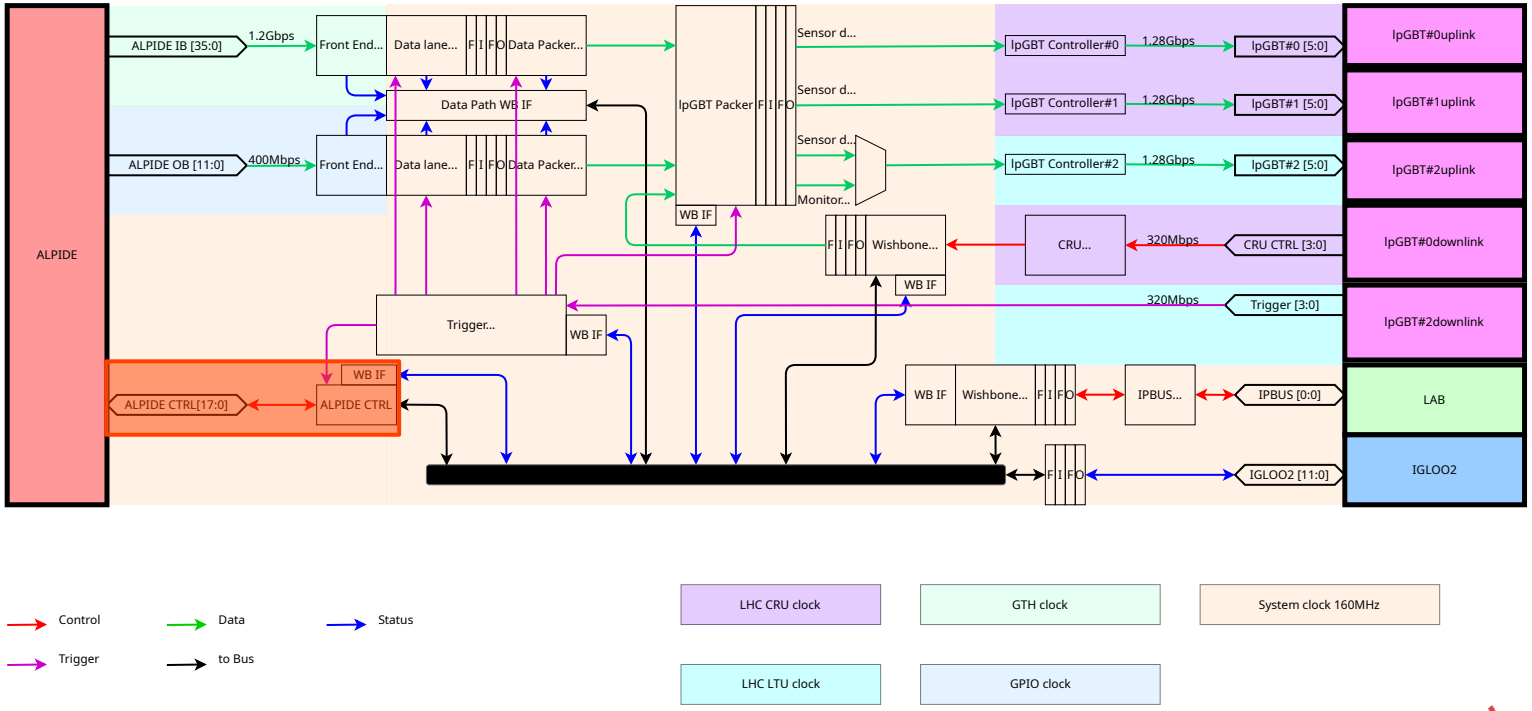
FoCal-E pixels readout

- Complex
- FPGA
- Data lines
- CRU lines



FoCal-E pixels readout

- Complex
- FPGA
- Data lines
- CRU lines
- Control!



FoCal-E control module

- Handles ALL communications with the Pixel layers
  - Operational codes (opcodes)
  - Triggers
  - Resets
  - starting/stopping readout
  - System setup
- Receives external instructions
  - Transforms them such that the ALPIDE understands it



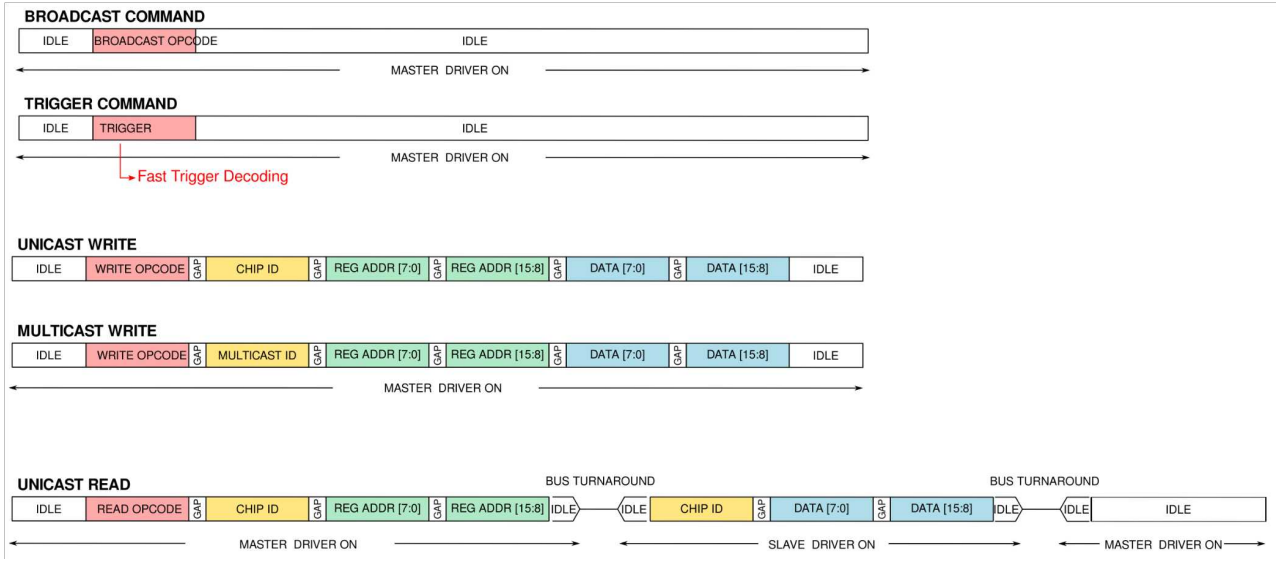
FoCal-E control module: Example

- Received instructions are:
  - OPCODE: what to do
  - CHIP ID: to which ALPIDE(s)
  - ADDRESS: where in the ALPIDE(s)
  - DATA: what



FoCal-E control module: Example

- Received instructions are:
  - OPCODE
  - CHIP ID
  - ADDRESS
  - DATA
- Transforms into:



HDL programming workflow

- Code inspiration
  - Proton CT code
  - ITS2 code



HDL programming workflow

- Code inspiration
  - Proton CT code
    - Not error free
  - ITS2 code
    - Not easily applicable



HDL programming workflow

- Code inspiration
  - Proton CT code
    - Not error free
  - ITS2 code
    - Not easily applicable
- Program it in the HDL, VHDL
  - This requires a completely different flow than Cpp
  - This different workflow requires different tools



HDL programming workflow

- Many tools!

purpose	Software
Operating system	Windows/Linux
Synthesis	Vivado
Place & Route	Vivado
Simulator Backend	Rieviera-PRO
Testbench framework	pyuvm
Unit test framework	CocoTB
Regression testing	VUnit
Build system	Hog
TMR generator	tmrg
Register gen.	PeakRDL
Linter	Verible/VSG



HDL programming workflow

- Many tools!
  - Used so far

purpose	Software
Operating system	Windows/Linux
Synthesis	Vivado
Place & Route	Vivado
Simulator Backend	Rieviera-PRO
Testbench framework	pyuvm
Unit test framework	CocoTB
Regression testing	VUnit
Build system	Hog
TMR generator	tmrg
Register gen.	PeakRDL
Linter	Verible/VSG



HDL programming workflow

- Many tools!
  - Used so far
- Flow
  - Create VHDL code
  - Verify with a test bench
  - Triple modulation redundancy
  - Verify test bench
  - Synthesize
  - Place and route
  - Static timing analysis
  - Test the final code

purpose	Software
Operating system	Windows/Linux
Synthesis	Vivado
Place & Route	Vivado
Simulator Backend	Rieviera-PRO
Testbench framework	pyuvm
Unit test framework	CocoTB
Regression testing	VUnit
Build system	Hog
TMR generator	tmrg
Register gen.	PeakRDL
Linter	Verible/VSG



HDL programming workflow

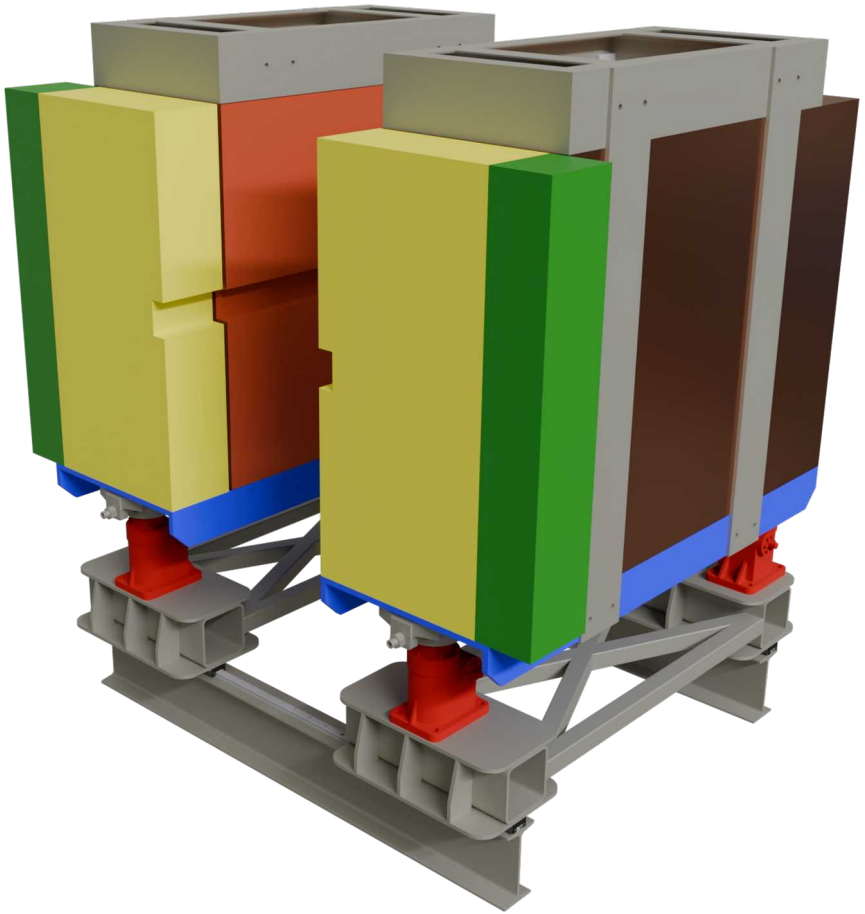
- Many tools!
  - Used so far
- Flow
  - Create VHDL code
  - Verify with a test bench
  - Triple modulation redundancy
  - Verify test bench
  - Synthesize
  - Place and route
  - Static timing analysis
  - Test the final code

purpose	Software
Operating system	Windows/Linux
Synthesis	Vivado
Place & Route	Vivado
Simulator Backend	Rieviera-PRO
Testbench framework	pyuvm
Unit test framework	CocoTB
Regression testing	VUnit
Build system	Hog
TMR generator	tmrg
Register gen.	PeakRDL
Linter	Verible/VSG



Outlook

- Busy with
  - Test bench
- Others are busy with
  - lpGBT interface
  - Wishbone
  - IGLOO (auxiliary FPGA)
- Lots of work to do
  - Help is more than welcome

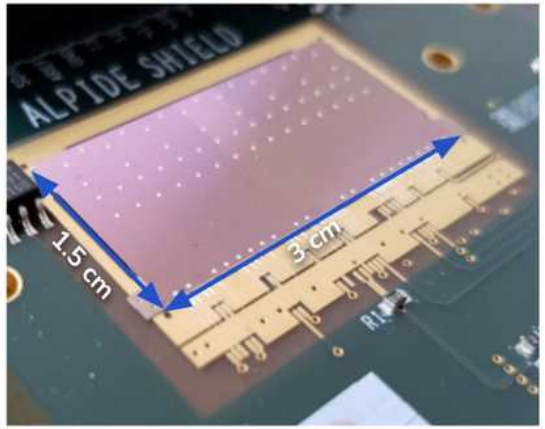


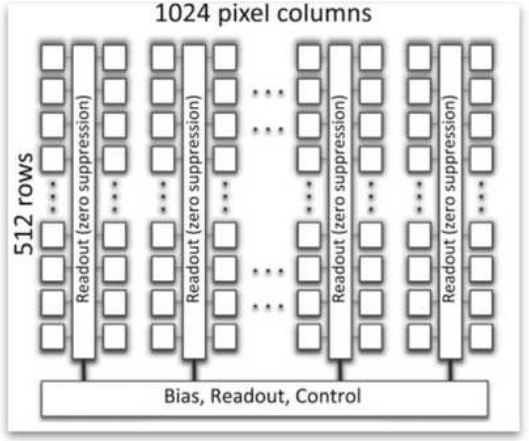
Backup-slides



Interlude: ALPIDE chips

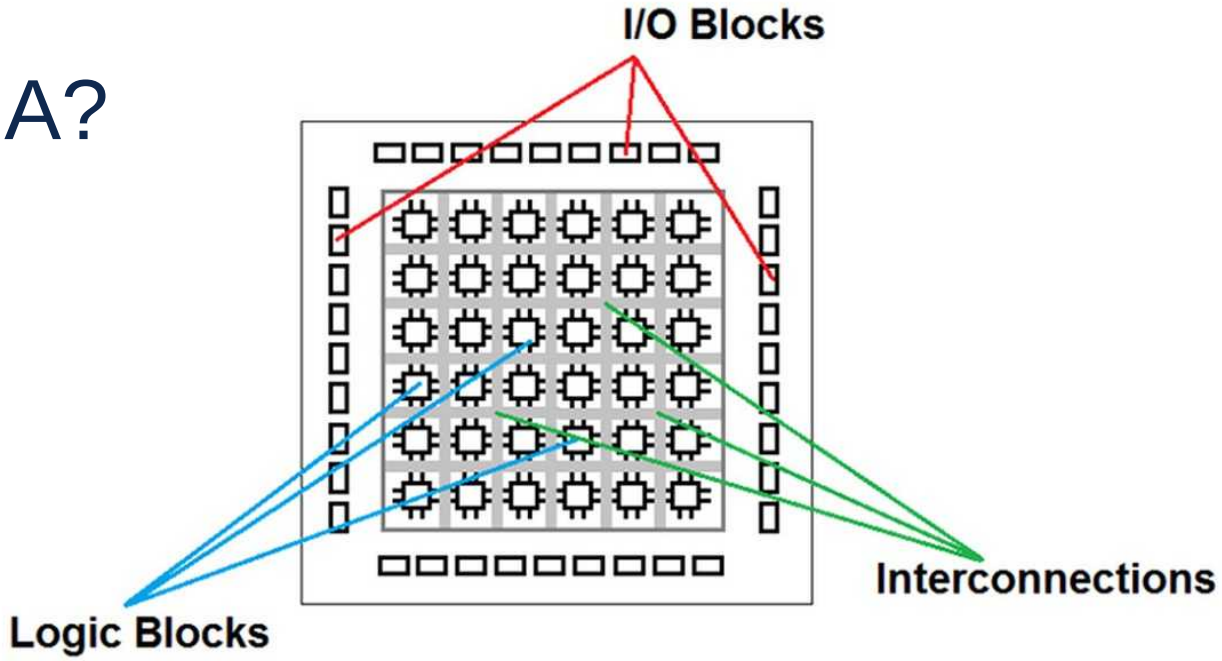
- Monolithic Active Pixel Sensors (MAPS)
  - Integrated circuitry
    - Signal processing
    - Temporary storage (3 bits)
    - more
- 512 x 1024 pixels
- Binary Pixels
  - On or off





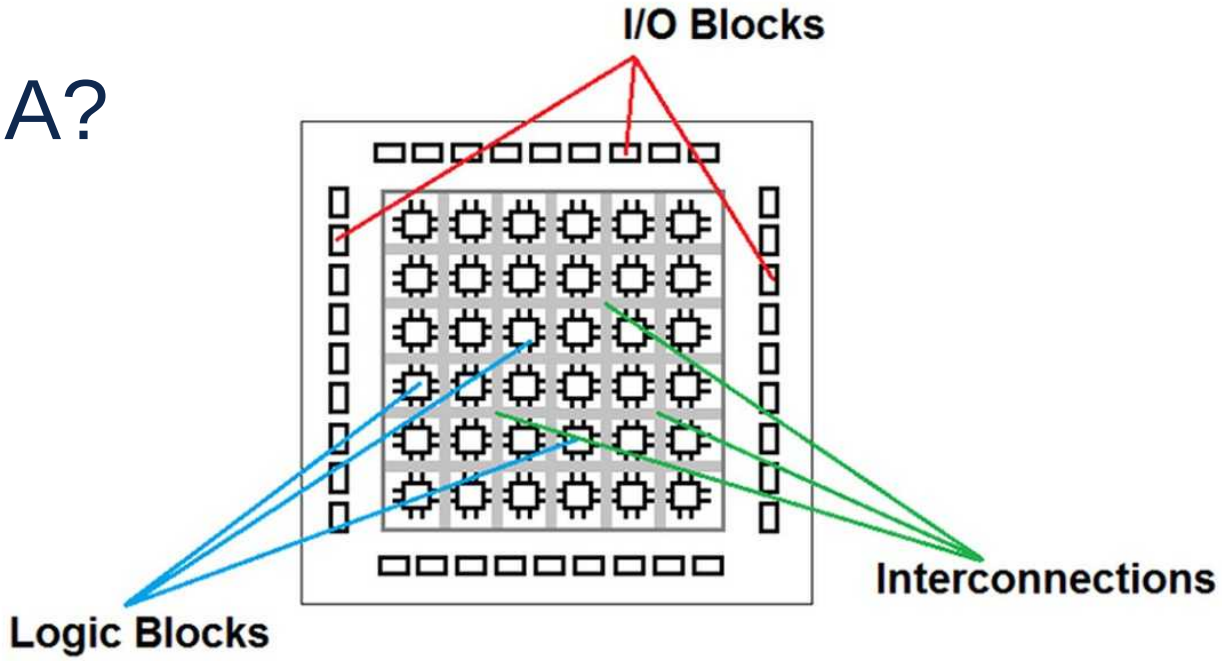
Interlude: FPGA

- What is an FPGA?



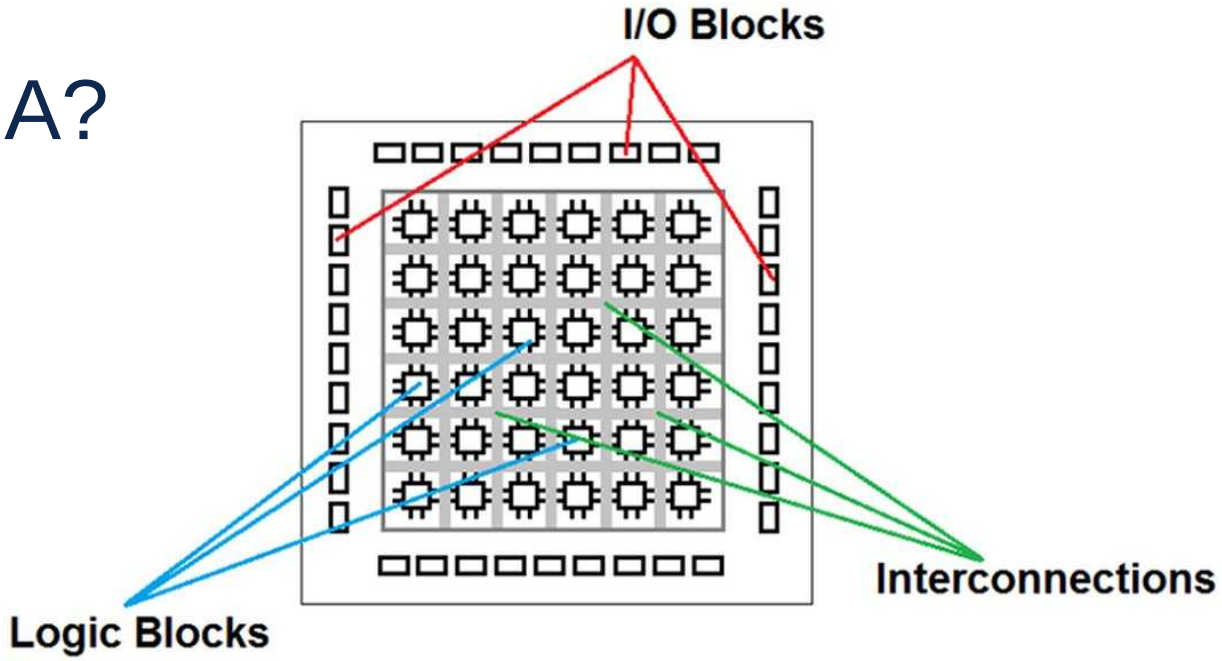
Interlude: FPGA

- What is an FPGA?
- Core
  - Logic blocks
  - Interconnections



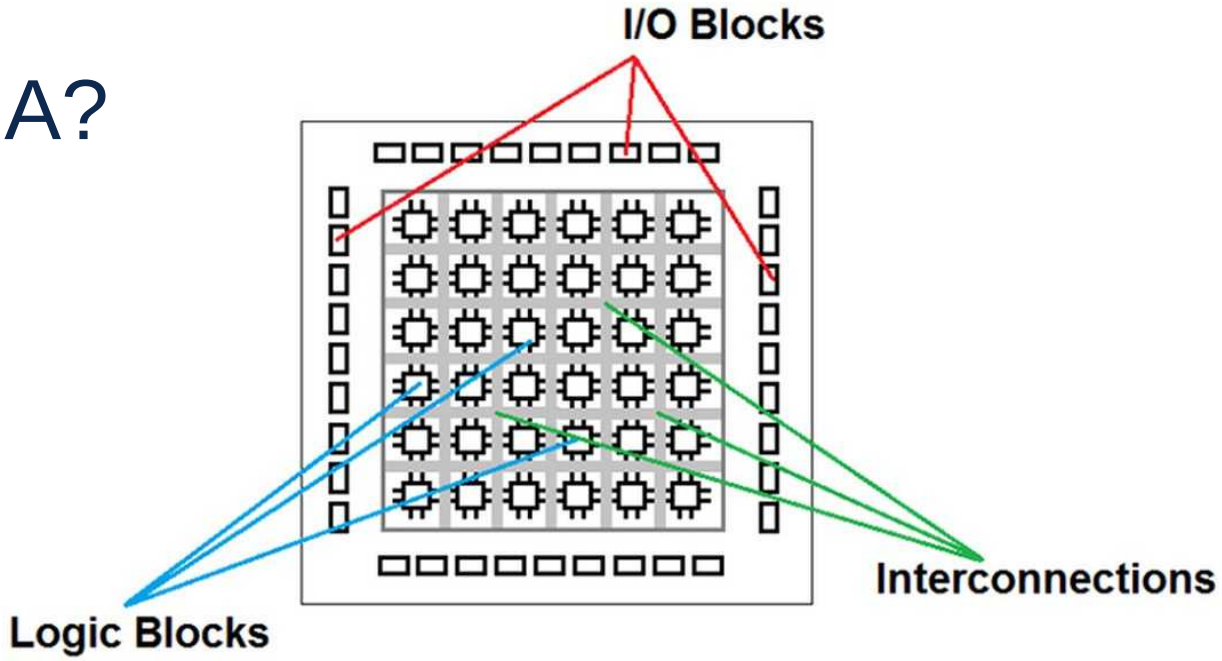
Interlude: FPGA

- What is an FPGA?
- Core
  - Logic blocks
  - Interconnections
- Periphery
  - IO blocks



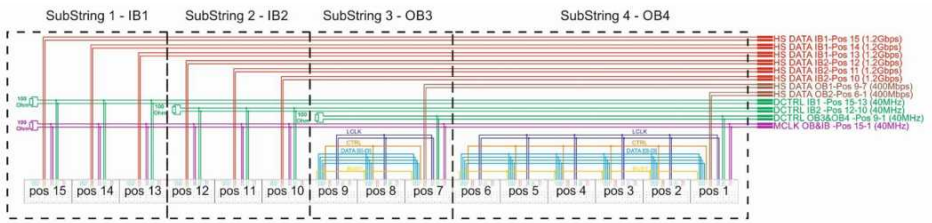
Interlude: FPGA

- What is an FPGA?
- Core
  - Logic blocks
  - Interconnections
- Periphery
  - IO blocks
- Programmable: Hardware Description Language



RU's: link numbers

- 1 IB/OB RU contains (1 * 6 * 3) 18 links
  - 1 IB/OB RU: 1 IB/OB module
  - 1 IB/OB module: 6 strings
  - 1 IB/OB string: 3 DCTRL links



- 1 OB RU contains (3 * 6 * 1) 18 links
  - 1 OB RU: 3 OB modules
  - 1 OB module: 6 strings
  - 1 OB string: 1 DCTRL link

