RFNoC™ Deep Dive: Host Side

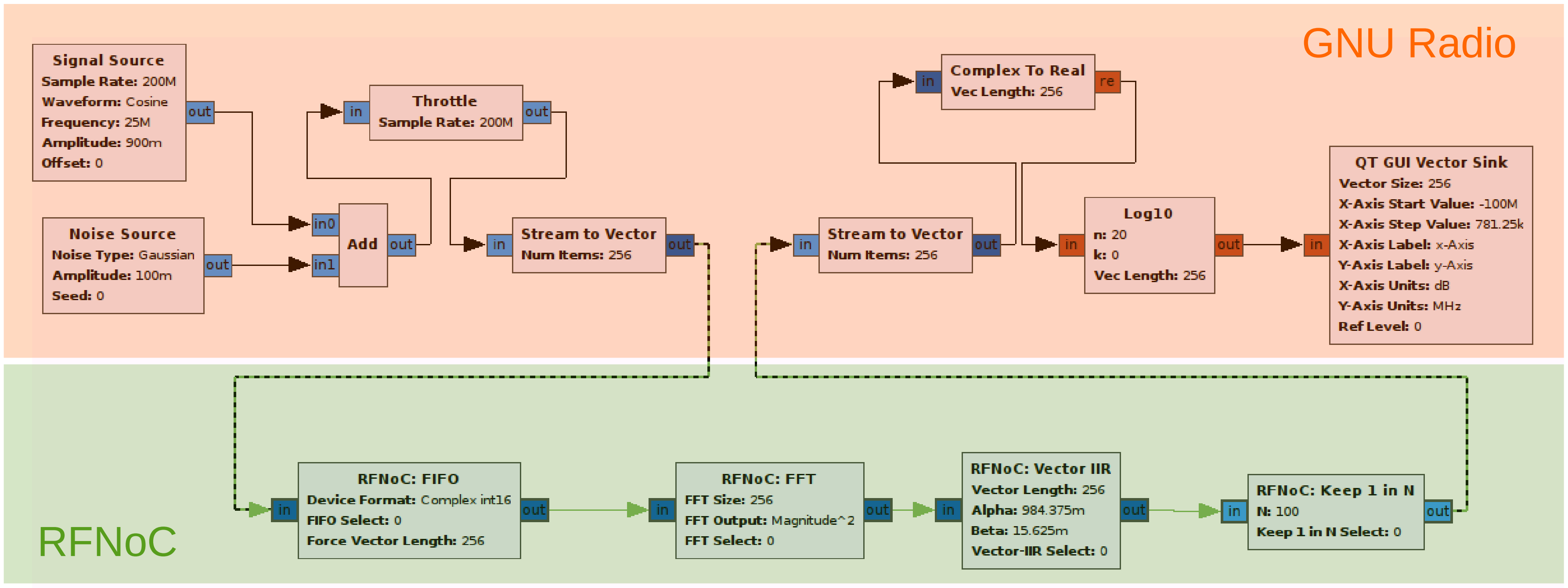
Martin Braun

5/28/2015

Why Host Development?

- Typical application is run and controlled from host-based process (e.g. GNU Radio)
- Setting up heterogeneous processing in a simple fashion requires a lot of software-controlled configuration

Example: Mixing Platforms



Signal Source
Sample Rate: 200M
Waveform: Cosine
Frequency: 25M
Amplitude: 900m
Offset: 0

Noise Source
Noise Type: Gaussian
Amplitude: 100m
Seed: 0

Add

Throttle
Sample Rate: 200M

Stream to Vector
Num Items: 256

Stream to Vector
Num Items: 256

Complex To Real
Vec Length: 256

Log10
n: 20
k: 0
Vec Length: 256

QT GUI Vector Sink
Vector Size: 256
X-Axis Start Value: -100M
X-Axis Step Value: 781.25k
X-Axis Label: x-Axis
Y-Axis Label: y-Axis
X-Axis Units: dB
Y-Axis Units: MHz
Ref Level: 0

RFNoC: FIFO
Device Format: Complex int16
FIFO Select: 0
Force Vector Length: 256

RFNoC: FFT
FFT Size: 256
FFT Output: Magnitude^2
FFT Select: 0

RFNoC: Vector IIR
Vector Length: 256
Alpha: 984.375m
Beta: 15.625m
Vector-IIR Select: 0

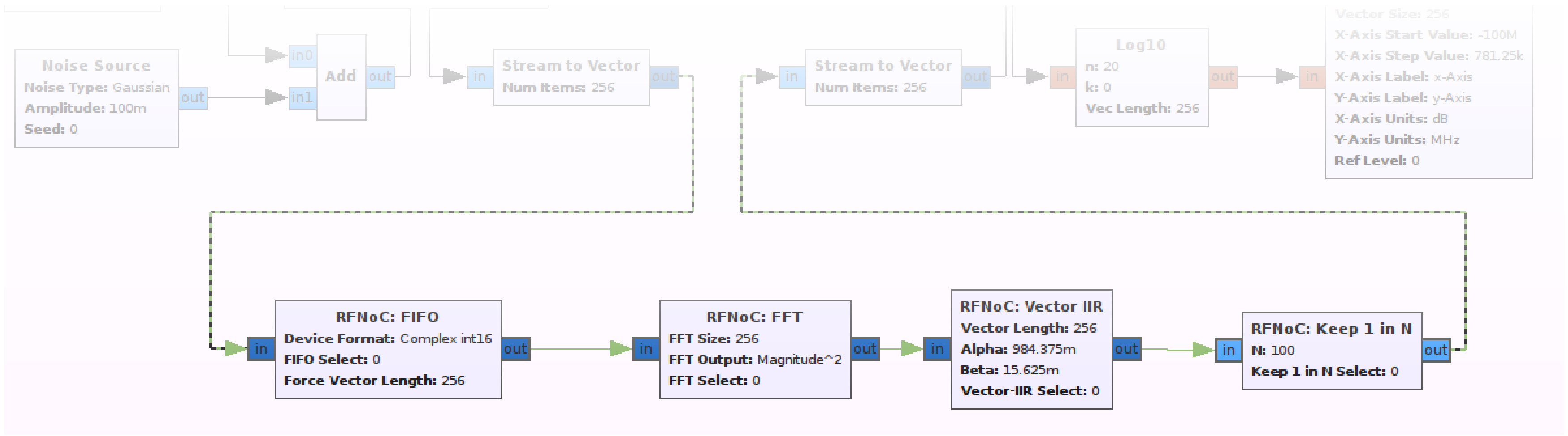
RFNoC: Keep 1 in N
N: 100
Keep 1 in N Select: 0

- Maintaining transparent modularity requires simple access to block settings, regardless of platform
- All controls must be available within the GNU Radio process
- GNU Radio does not control out-of-domain blocks

What does the Host do?

- Configure connections between blocks
  - Set up flow control, configure stream IDs
- Configure block-specific settings (e.g. FFT size, FIR taps, PLL loop bandwidths...)
  - Map settings bus addresses to human-readable settings
- Initiate streaming for domain boundary crossings
  - Abstract transport type (Ethernet, PCIe, AXI)
- Provide API calls for block-specific operations
  - Direct access to FPGA registers is available, but might not be the nicest way to configure blocks

What does the Host do?



Noise Source
Noise Type: Gaussian
Amplitude: 100m
Seed: 0

Add

Stream to Vector
Num Items: 256

Stream to Vector
Num Items: 256

Log10
n: 20
k: 0
Vec Length: 256

Vector Size: 256
X-Axis Start Value: -100M
X-Axis Step Value: 781.25k
X-Axis Label: x-Axis
Y-Axis Label: y-Axis
X-Axis Units: dB
Y-Axis Units: MHz
Ref Level: 0

RFNoC: FIFO
Device Format: Complex int16
FIFO Select: 0
Force Vector Length: 256

RFNoC: FFT
FFT Size: 256
FFT Output: Magnitude^2
FFT Select: 0

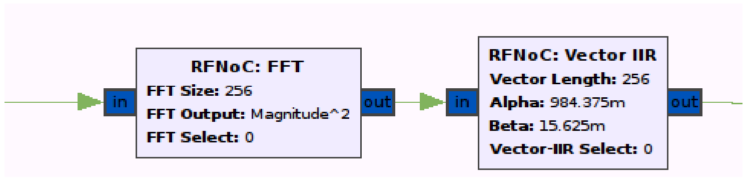
RFNoC: Vector IIR
Vector Length: 256
Alpha: 984.375m
Beta: 15.625m
Vector-IIR Select: 0

RFNoC: Keep 1 in N
N: 100
Keep 1 in N Select: 0

- Maintains graph representation of active RFNoC blocks within UHD context
- Host-side checking of data type matching
- Easy configuration of block connections
- Commands can be passed on to other RFNoC blocks (e.g. streaming commands)

Example: Connection Setup

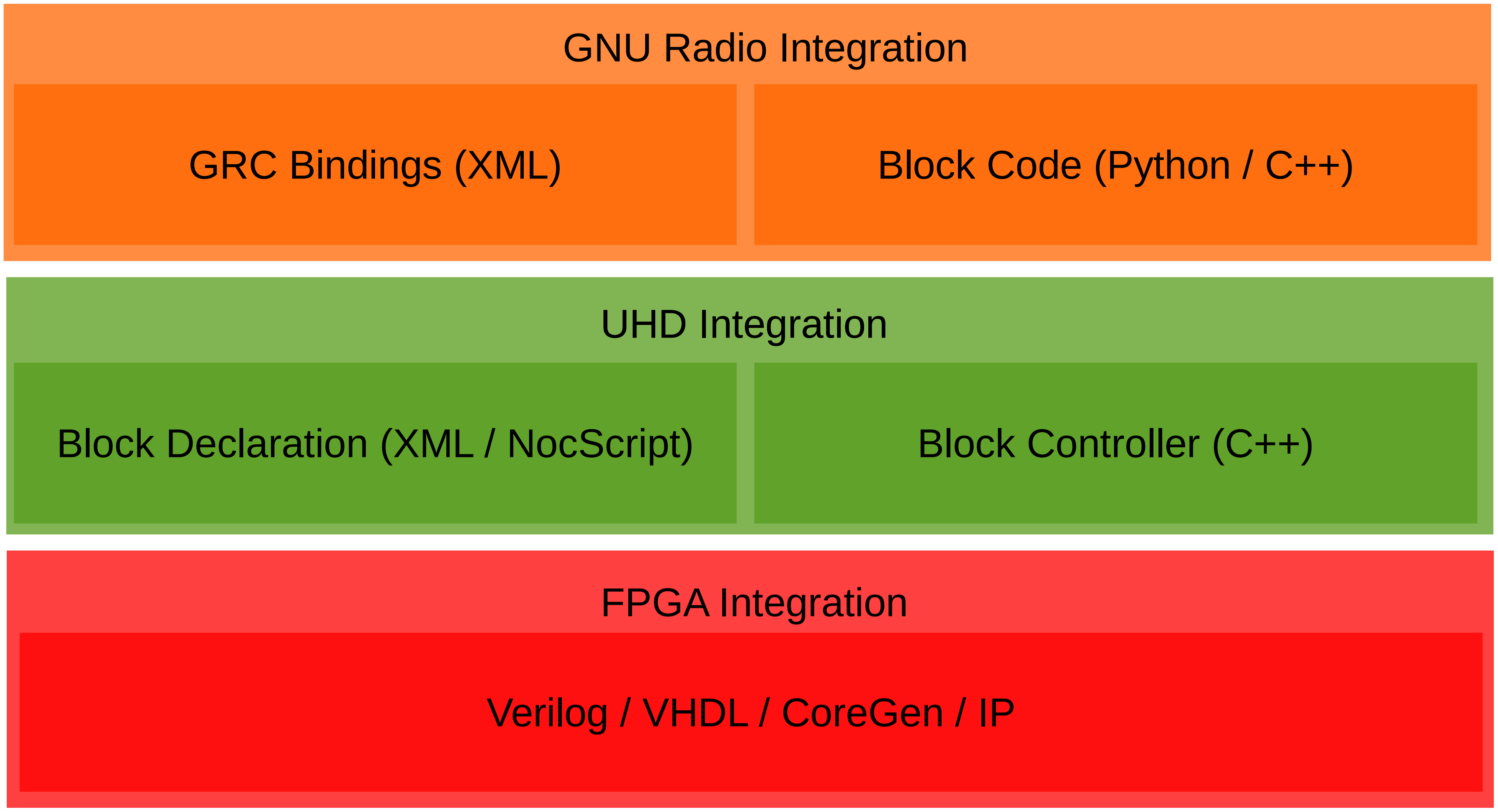
- What happens when we call `connect()`?



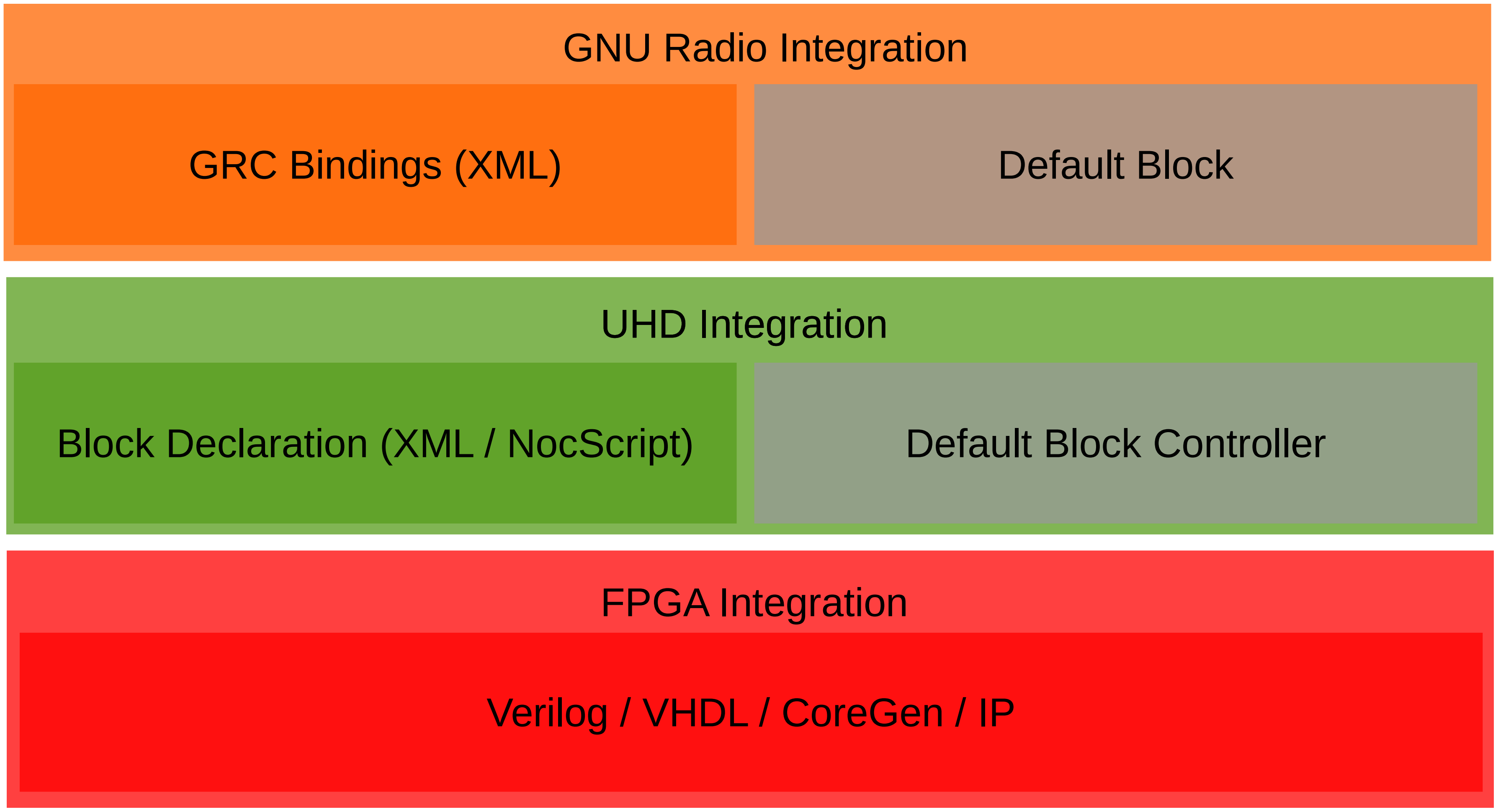
1. Identify port numbers on source and destination
2. Check stream signatures match (type, vector length etc.)
3. Read destination port address, generate SID, write to source block
4. Read packet size from source block, read input buffer size from destination
5. Set flow control registers on source and destination (depending on transport type between blocks)

RFNoC Stack

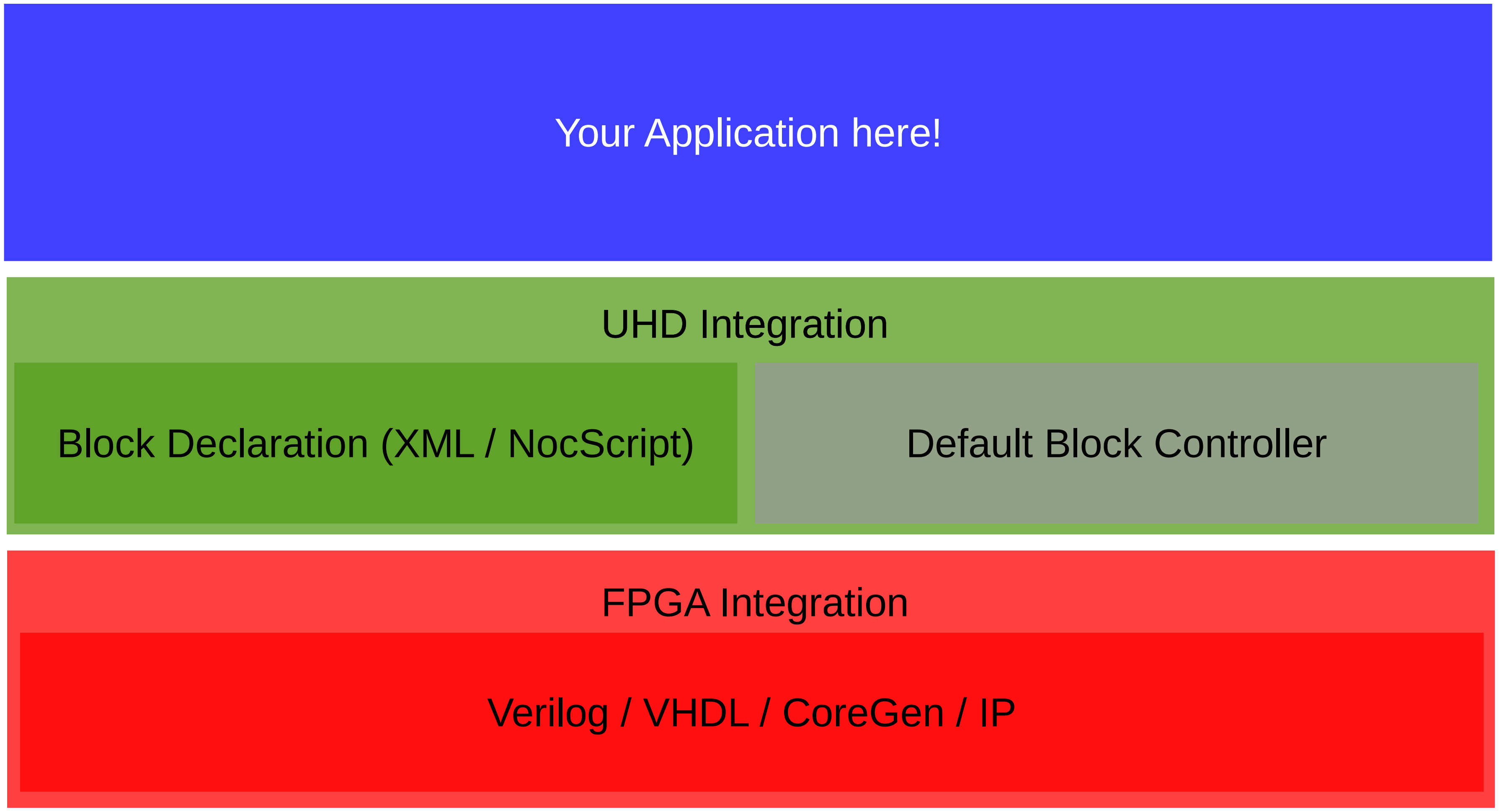
Ettus Research™
A National Instruments Company



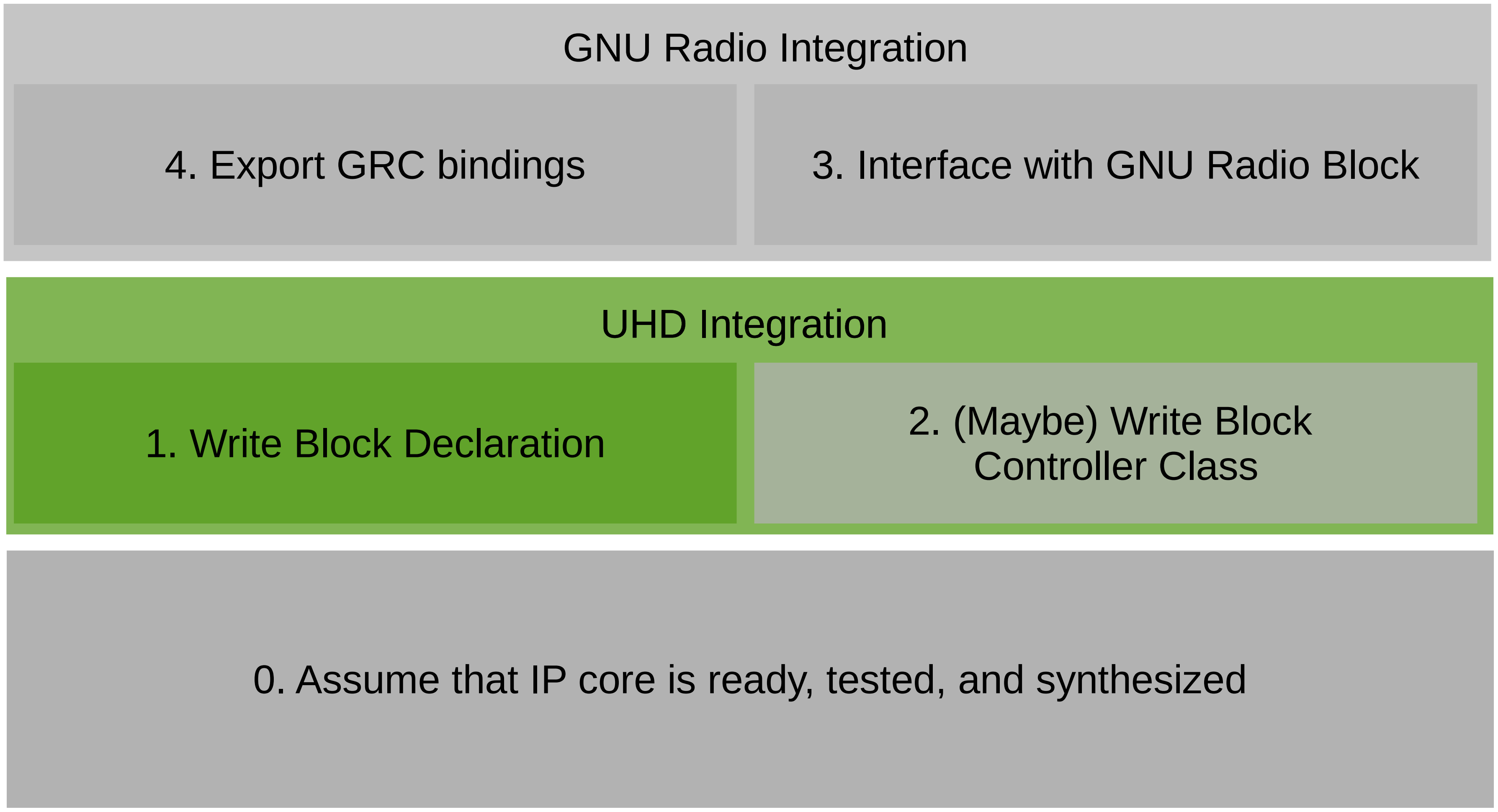
RFNoC Stack (Simple)



RFNoC Stack (Even Simpler)



Let's walk the Stack: UHD


GNU Radio Integration
4. Export GRC bindings
3. Interface with GNU Radio Block
UHD Integration
1. Write Block Declaration
2. (Maybe) Write Block Controller Class
0. Assume that IP core is ready, tested, and synthesized

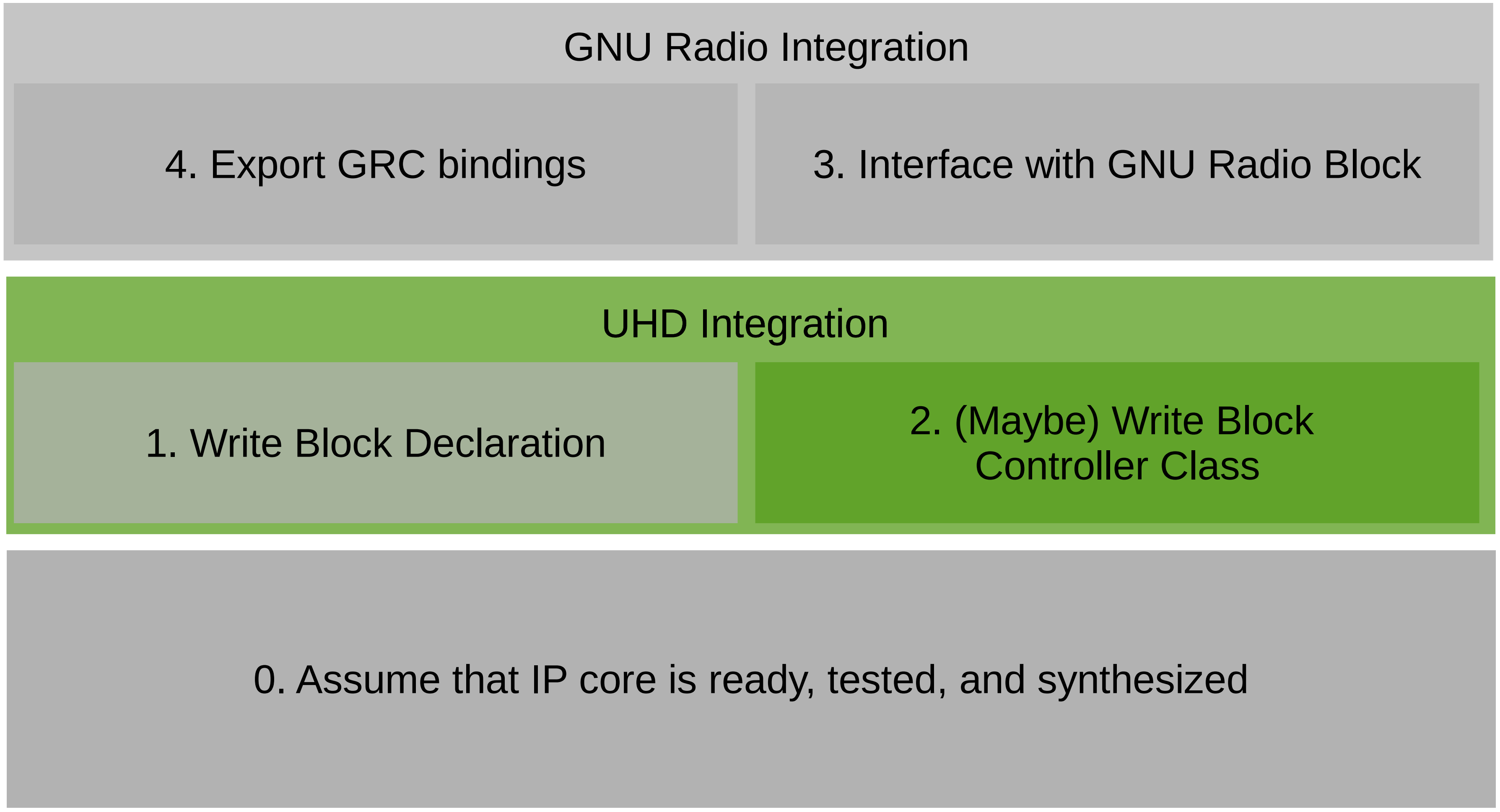
Block Declaration

- XML File
- Blocks are identified by their NoC-ID
- Description of block for UHD
  - Argument List (e.g. FFT size)
  - Input- and output ports (data types, vector length, packet size)
  - Settings- and readback registers
- NocScript: Add control code
- Example: FFT Block

NocScript

- Very simple DSL specific to block configuration
- Statically typed, quasi-functional
- Few basic types: Integers, Strings, Doubles, Vectors
- Lots of uppercase and parentheses
- Allows basic access to block arguments and settings registers
- Basic arithmetic and logic operations available

Let's walk the Stack: UHD


GNU Radio Integration
4. Export GRC bindings
3. Interface with GNU Radio Block
UHD Integration
1. Write Block Declaration
2. (Maybe) Write Block Controller Class
0. Assume that IP core is ready, tested, and synthesized

Block Control Classes

- C++ Code
- Provides FPGA access
- Represents block in RFNoC-Graph

- Default Block Control will do vast majority of required tasks
- Own implementation may not be required

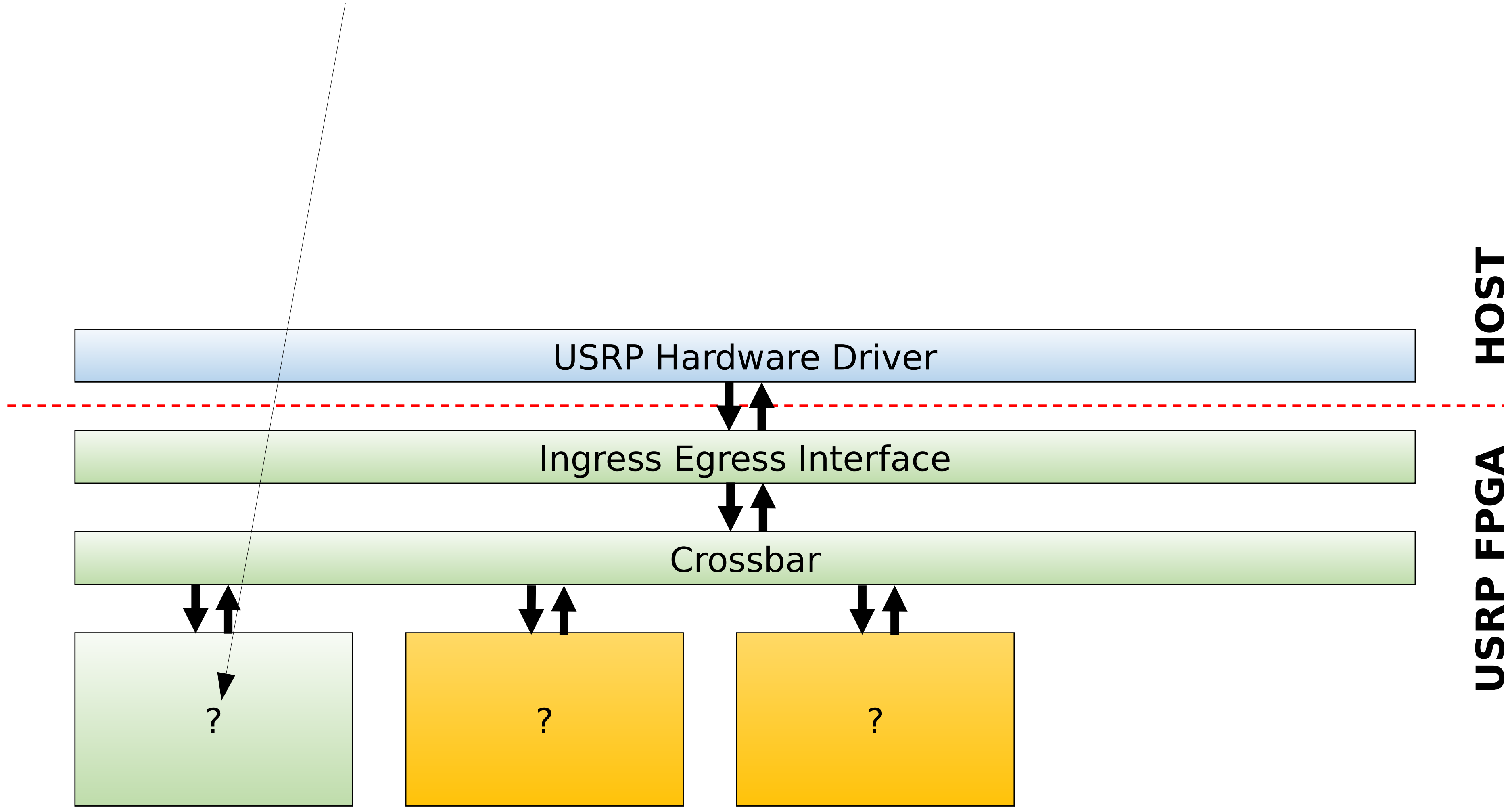
- Example: FFT Block

When to write own class?

- Whenever XML + NocScript are not sufficient!
- Complex operations that are easier expressed in C++-Code than XML + NocScript
  - Example: Radio Controls
- Note: Writing custom block controllers requires re-compilation of UHD + your own library, whereas XML + NocScript is interpreted at runtime
- Example: FFT Block
  - Public header
  - Implementation file

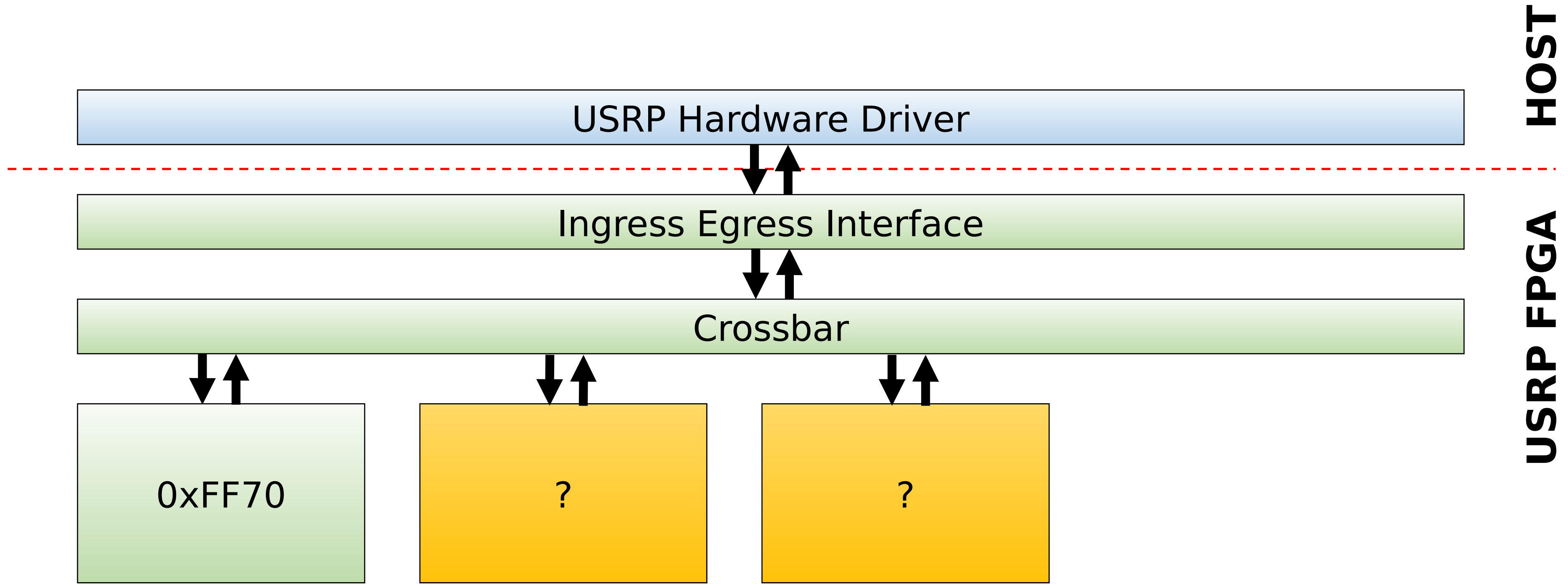
Interlude: Lookup Process

- UHD queries NoC-ID Registers on Block



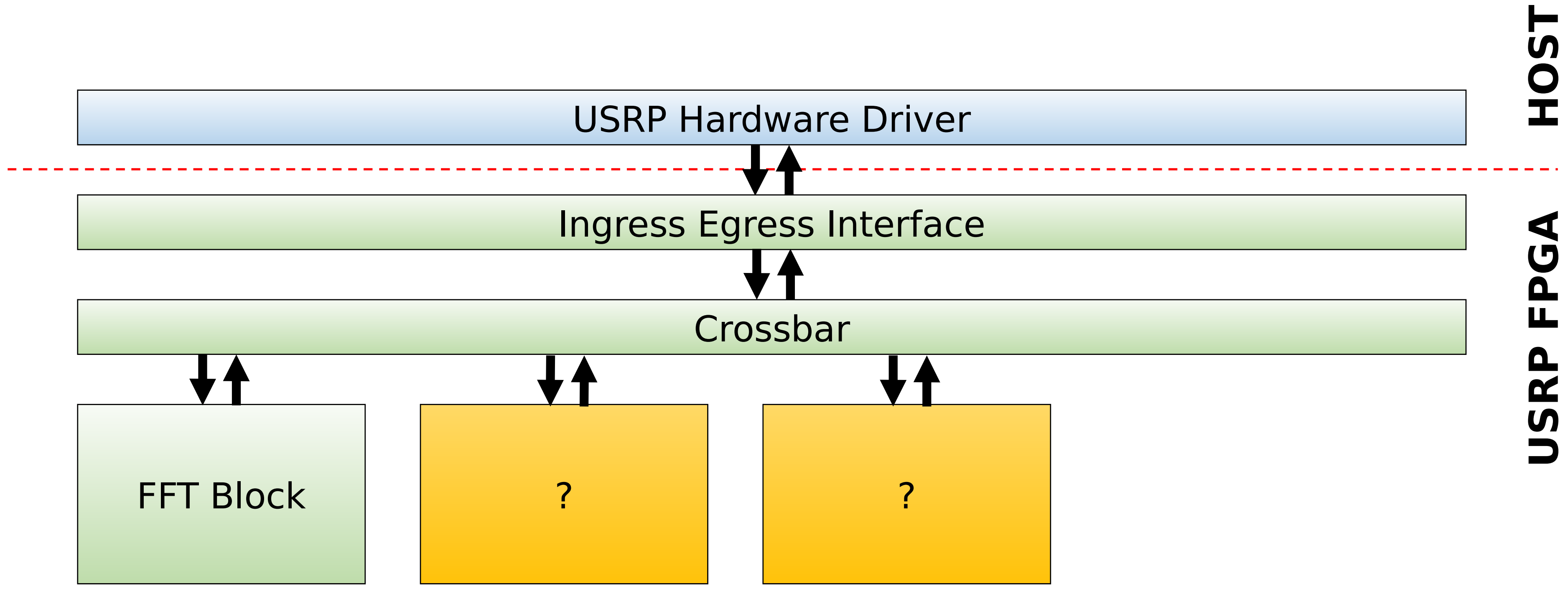
Interlude: Lookup Process

- Query NoC-ID Register on Block
- Look up NoC-ID in XML files



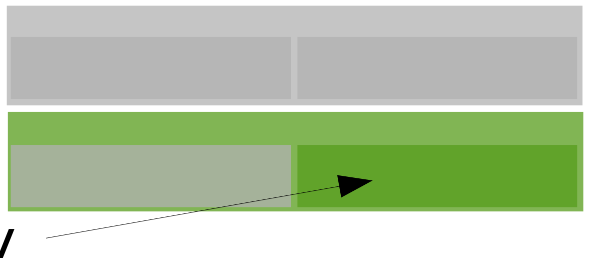
Interlude: Lookup Process

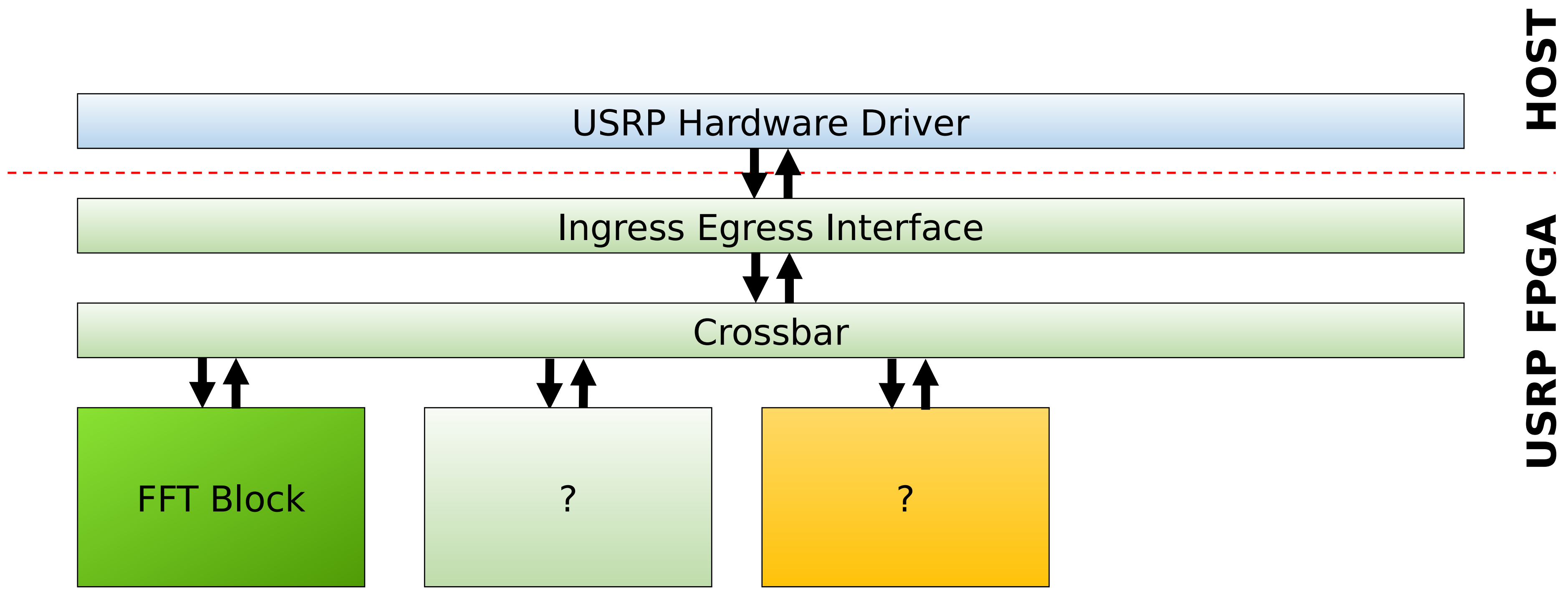
- Query NoC-ID Register on Block
- Look up NoC-ID in XML files



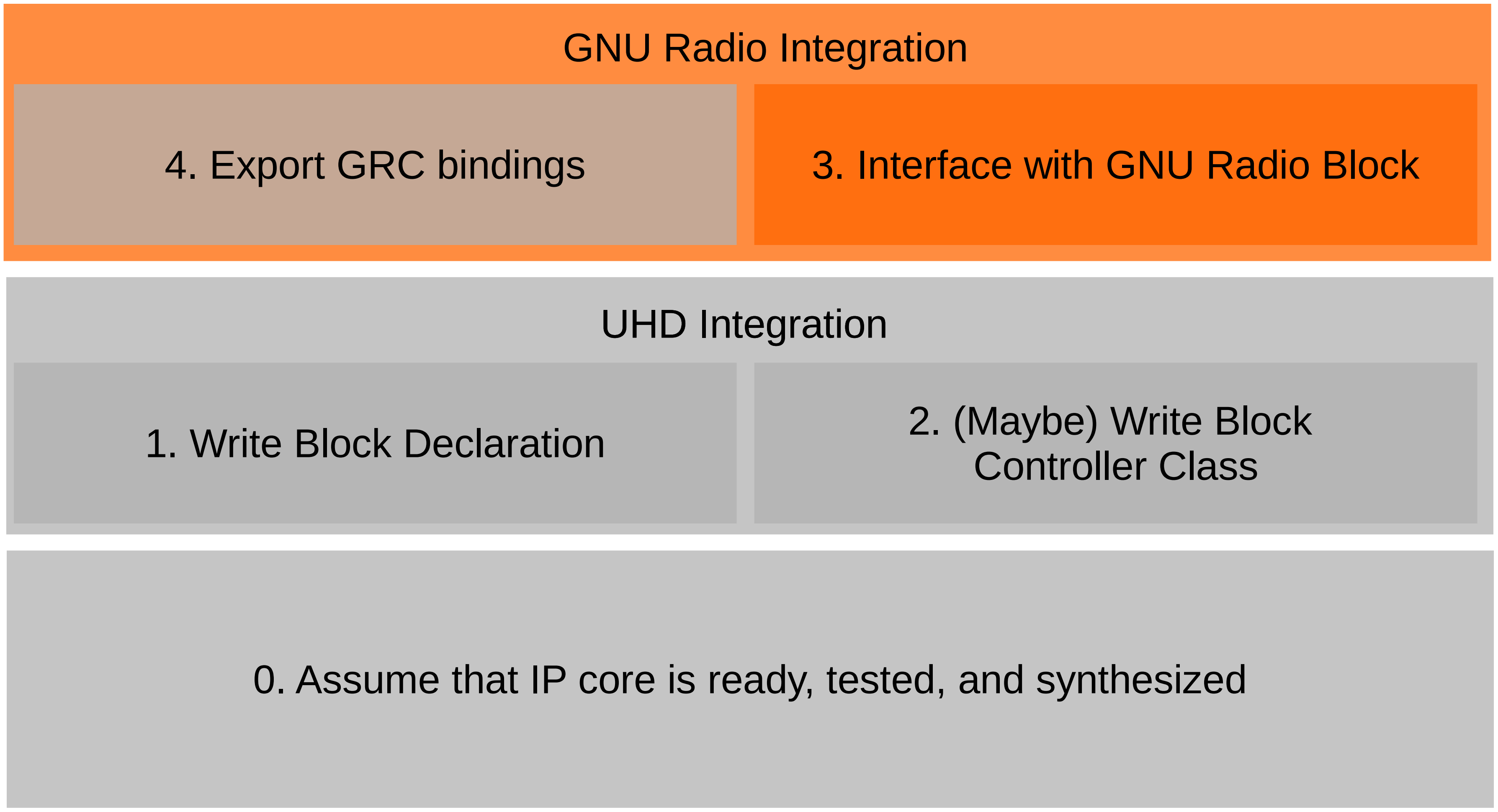
Interlude: Lookup Process

- Query NoC-ID Register on Block
- Look up NoC-ID in XML files
- Find block controller class in registry





GNU Radio Integration


GNU Radio Integration
4. Export GRC bindings
3. Interface with GNU Radio Block
UHD Integration
1. Write Block Declaration
2. (Maybe) Write Block Controller Class
0. Assume that IP core is ready, tested, and synthesized

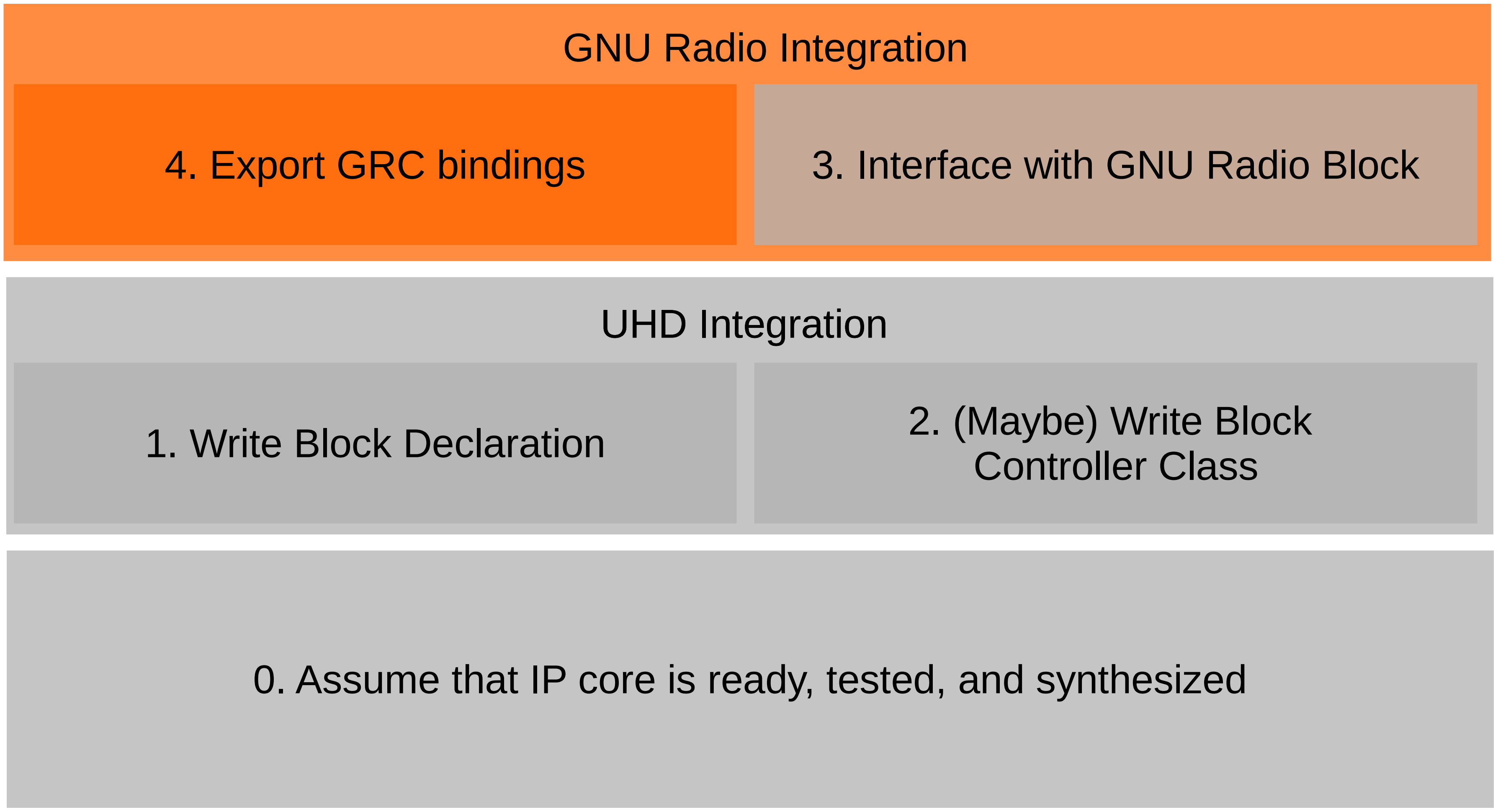
gr-uhd & gr-ettus

- gr-uhd: Stable GNU Radio bindings for all UHD products
- gr-ettus: Out-of-tree module for experimental code, subject to change
- As RFNoC matures, code will migrate from gr-ettus to gr-uhd
- gr-ettus will serve as example for OOT projects including RFNoC
- Now, gr-ettus provides examples for GNU Radio / RFNoC blocks
- Available online: http://github.com/EttusResearch/gr-ettus.git

gr::ettus::rfnoc_*

- gr-ettus provides a generic block that handles most cases of RFNoC blocks (`gr::ettus::rfnoc_generic`)
- In all other cases, derive block from `gr::ettus::rfnoc_block` to make life easier
  - Example: `rfnoc_window_cci` has restrictions that RFNoC has not
- Note: RFNoC/GNU Radio blocks hold reference to USRP object, but do not create it (must be generated externally and passed in)
- gr_modtool still works!

GNU Radio Integration


GNU Radio Integration
4. Export GRC bindings
3. Interface with GNU Radio Block
UHD Integration
1. Write Block Declaration
2. (Maybe) Write Block Controller Class
0. Assume that IP core is ready, tested, and synthesized

GRC Bindings

- XML File
- Describes GNU Radio Block to GNU Radio Companion
- Using `gr::ettus::rfnoc_generic` does not prohibit writing custom GRC bindings
  - Example: FFT Block
- High similarity to block declaration file: In the future, we might provide a tool to convert one to the other

Conclusion

- RFNoC™ requires some host-side modifications
- Best case, this entails writing 1 or 2 XML files
- Worst case, this also means adding up to 2 C++ files with well-documented and simple APIs
- Design goal of RFNoC is to simplify host-side workload as much as possible
- Tools exist to make it as easy as possible (e.g. gr_modtool)