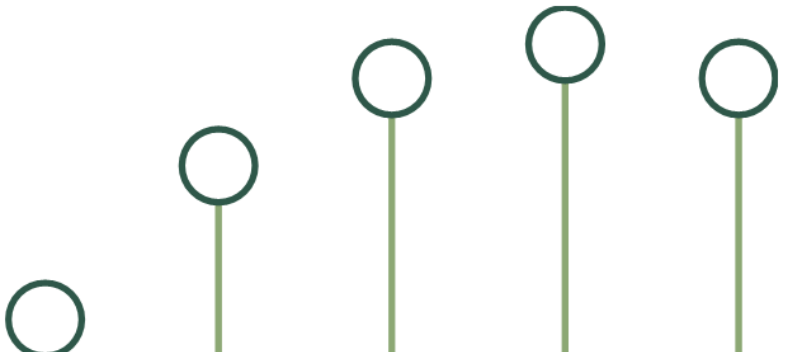
Ettus Research™
A National Instruments Company

RFNoC 3 Workshop

Part 2

Jonathon Pendlum – Ettus Research
Neel Pandeya – Ettus Research

August 2020



Schedule

Ettus Research™
A National Instruments Company

- Part 1
  - RFNoC Framework Overview
  - Hands on Demos
- Part 2
  - RFNoC FPGA & Software Architecture
  - Hands on RFNoC Block Development
  - Advanced RFNoC Topics

Getting RFNoC

Ettus Research™ A National Instruments Company

- Two main ways to get RFNoC
- Install with PyBOMBs
  - PyBOMBs install already done in our VM!
  - ~/rfnoc-workshop
  - To reproduce see RFNoC Getting Started Guide
  - https://kb.ettus.com/Getting_Started_with_RFNoC_Development
- Manual install
  - Install dependencies
  - Build UHD, GNU Radio, and gr-ettus

RFNoC modtool

Ettus Research™
A National Instruments Company

- Installed with gr-ettus (via PyBOMBs)
- Generate OOT RFNoC modules
  - Similar concept to gr_modtool
  - Creates skeleton code for a pass through RFNoC block
    - Verilog for RFNoC Block
    - Test bench
    - UHD C++ block controller / NoC Script XML
    - GNU Radio Companion XML
- Everything in one place!

Using RFNoC modtool

Ettus Research™
A National Instruments Company

- cd ~/rfnoc-workshop
- source setup_env.sh
- rfnocmodtool help

```
rfnoc@rfnoc-vm ~/rfnoc-workshop $ source ~/rfnoc-workshop/setup_env.sh
rfnoc@rfnoc-vm ~/rfnoc-workshop $ rfnocmodtool help
Usage:
rfnocmodtool <command> [options] -- Run <command> with the given options.
rfnocmodtool help -- Show a list of commands.
rfnocmodtool help <command> -- Shows the help for a given command.

List of possible commands:

Name      Aliases          Description
==========================================================================
disable   dis              Disable block (comments out CMake entries for files)
info      getinfo,inf      Return information about a given module
remove    rm,del           Remove block (delete files and remove Makefile entries)
makexml   mx               Make XML file for GRC block bindings
add       insert           Add block to the out-of-tree module.
newmod    nm,create        Create a new out-of-tree module
rename    mv               Rename a block in the RFNoC out-of-tree module.
```

Using RFNoC modtool

Ettus Research™ A National Instruments Company

- Steps to create our block:
- **source ~/rfnoc-workshop/setup_env.sh**
- **cd ~/rfnoc-workshop/src**
- **rfnocmodtool newmod tutorial**
- **cd rfnoc-tutorial**
- **rfnocmodtool add gain**
  - Enter valid argument list, including default arguments: ***(leave blank)***
  - Add Python QA Code? [y/N] **N**
  - Add C++ QA Code? [y/N] **N**
  - Block NoC ID (Hexadecimal): **E7757E5700000001**
  - Skip Block Controllers Generation? [UHD block ctrl files] [y/N] **N**
  - Skip Block interface files Generation? [GRC block ctrl files] [y/N] **N**

RFNoC modtool output

Ettus Research™
A National Instruments Company

```
rfnoc@rfnoc-vm ~/rfnoc-workshop $ cd src
rfnoc@rfnoc-vm ~/rfnoc-workshop/src $ rfnocmodtool newmod tutorial
Creating out-of-tree module in ./rfnoc-tutorial... Done.
Use 'rfnocmodtool add' to add a new block to this currently empty module.
rfnoc@rfnoc-vm ~/rfnoc-workshop/src $ cd rfnoc-tutorial
rfnoc@rfnoc-vm ~/rfnoc-workshop/src/rfnoc-tutorial $ rfnocmodtool add gain
RFNoC module name identified: tutorial
Block/code identifier: gain
Enter valid argument list, including default arguments:
Add Python QA code? [y/N] N
Add C++ QA code? [y/N] N
Block NoC ID (Hexadecimal): E7757E5700000001
Skip Block Controllers Generation? [UHD block ctrl files] [y/N] N
Skip Block interface files Generation? [GRC block ctrl files] [y/N] N
Adding file 'lib/gain_impl.h'...
Adding file 'lib/gain_impl.cc'...
Adding file 'include/tutorial/gain.h'...
Adding file 'include/tutorial/gain_block_ctrl.hpp'...
Adding file 'lib/gain_block_ctrl_impl.cpp'...
Editing swig/tutorial_swig.i...
Adding file 'grc/tutorial_gain.xml'...
Adding file 'rfnoc/blocks/gain.xml'...
Adding file 'rfnoc/fpga-src/noc_block_gain.v'...
rfnoc/testbenches/noc_block_gain_tb folder created
Adding file 'rfnoc/testbenches/noc_block_gain_tb/noc_block_gain_tb.sv'...
Adding file 'rfnoc/testbenches/noc_block_gain_tb/Makefile'...
Adding file 'rfnoc/testbenches/noc_block_gain_tb/CMakeLists.txt'...
```

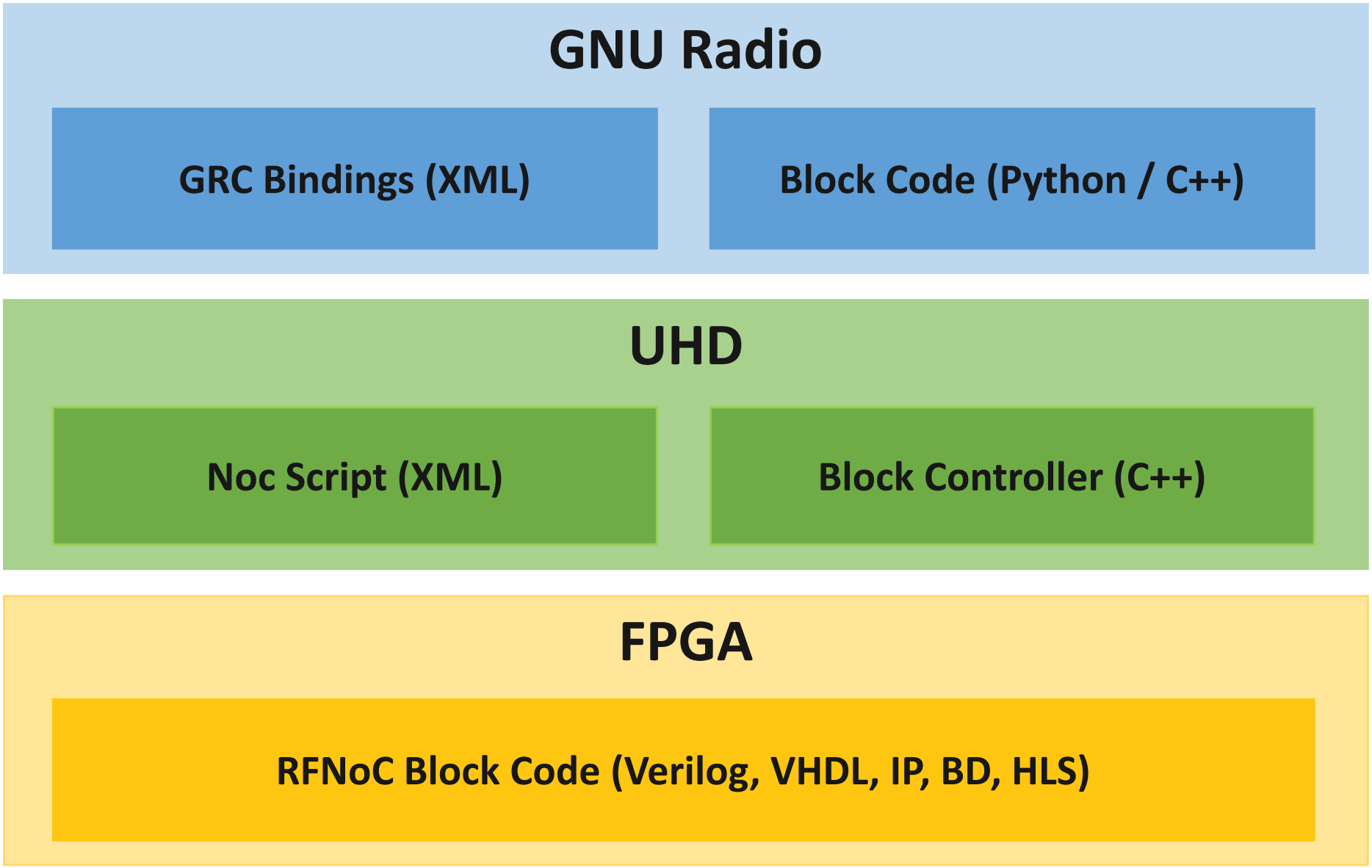
Directory Structure

Ettus Research™ A National Instruments Company

- grc
  - tutorial_gain.xml – GRC XML Block Description
- include/tutorial
  - GNU Radio & UHD Block Controller C++ headers
- lib
  - GNU Radio & UHD Block Controller C++ impls
- rfnoc/blocks
  - gain.xml – Noc Script XML
- rfnoc/fpga-src
  - noc_block_gain.v – RFNoC Block HDL
- rfnoc/testbench/noc_block_gain_tb
  - noc_block_gain_tb.sv – RFNoC Block HDL test bench

RFNoC Framework

Ettus Research™
A National Instruments Company



RFNoC Framework

Ettus Research™
A National Instruments Company

GNU Radio

GRC Bindings (XML)

Block Code (Python / C++)

UHD

Noc Script (XML)

Block Controller (C++)

FPGA

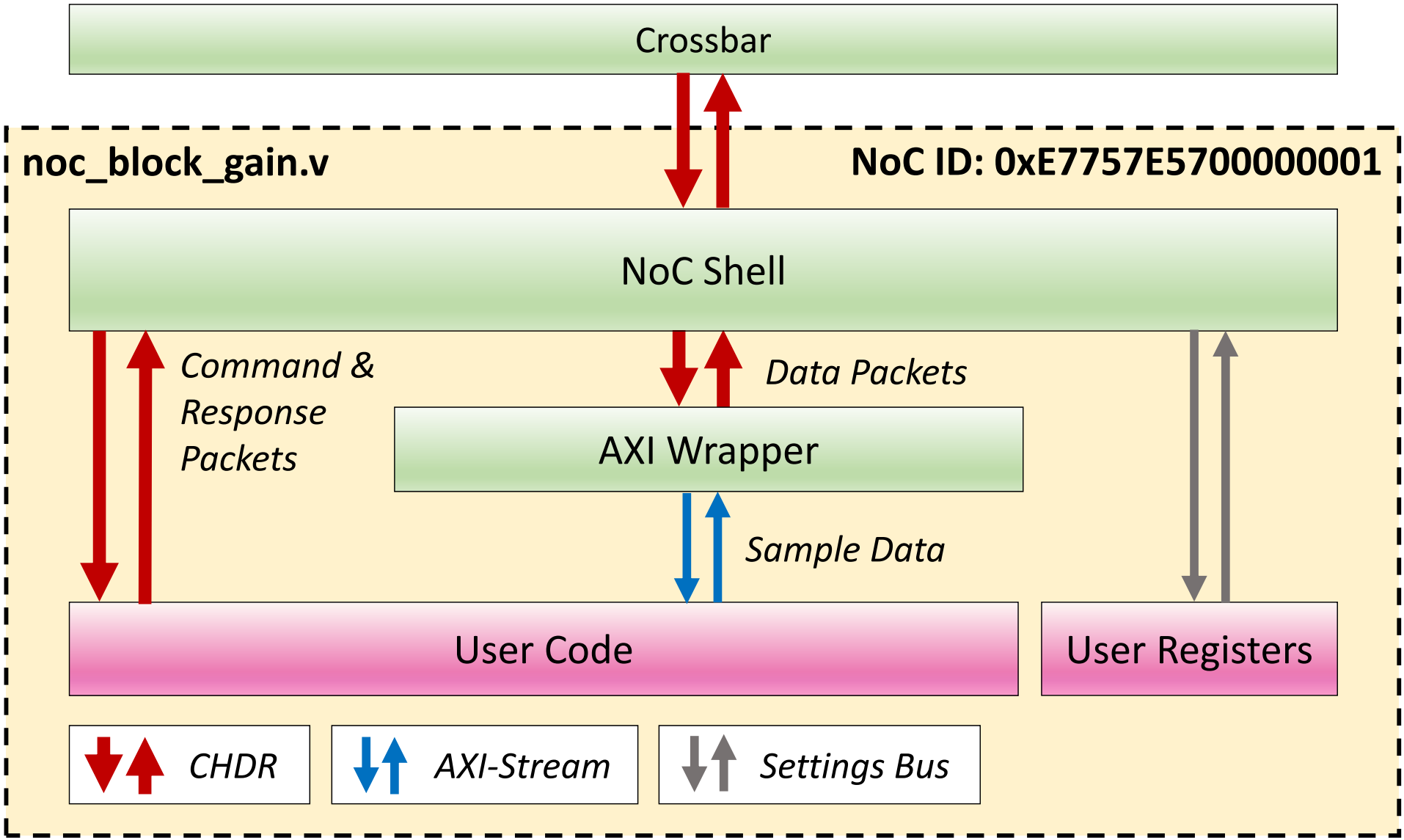
RFNoC Block Code (Verilog, VHDL, IP, BD, HLS)

Gain RFNoC Block

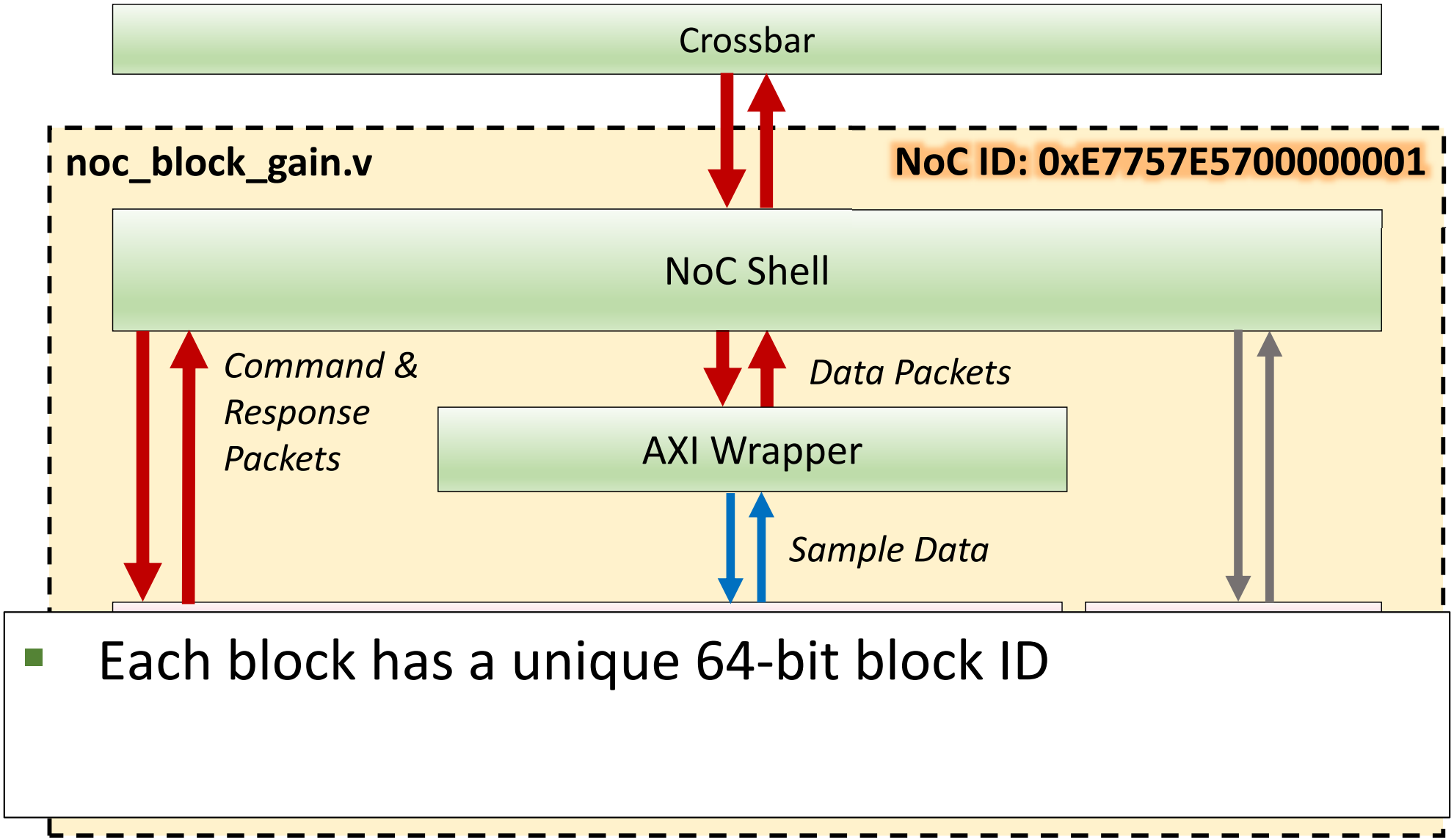
Ettus Research™
A National Instruments Company

- Open ~/rfnoc-workshop/src/rfnoc-tutorial/rfnoc/fpga-src/noc_block_gain.v
- Editor choices in VM:
  - (g)vim
  - emacs
  - nano
  - gedit
  - geany
  - notepadqq
  - mousepad

Gain RFNoC Block



Gain RFNoC Block



- Each block has a unique 64-bit block ID

Gain RFNoC Block Parameters

Ettus Research™
A National Instruments Company

```
21
22 //
23 module noc_block_gain #(
24   parameter NOC_ID = 64'hE7757E5700000001,
25   parameter STR_SINK_FIFOSIZE = 11)
26 (
27   input bus_clk, input bus_rst,
28   input ce_clk, input ce_rst,
29   input  [63:0] i_tdata, input  i_tlast, input  i_tvalid, output i_tready,
30   output [63:0] o_tdata, output o_tlast, output o_tvalid, input  o_tready,
31   output [63:0] debug
32 );
33
```

- Each block has a unique 64-bit block ID

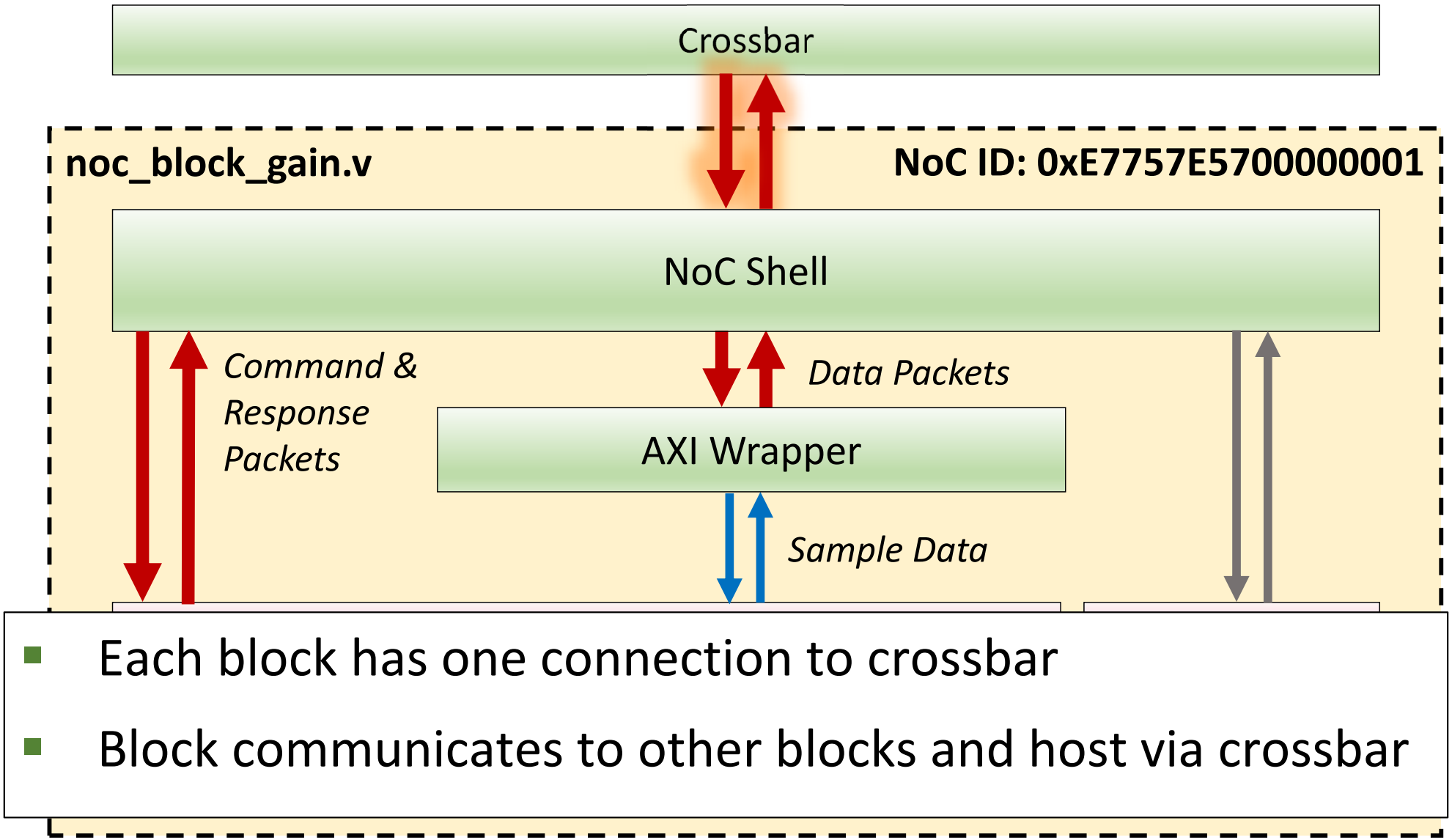
Gain RFNoC Block Parameters

Ettus Research™
A National Instruments Company

```
21
22 //
23 module noc_block_gain #(
24   parameter NOC_ID = 64'hE7757E5700000001,
25   parameter STR_SINK_FIFOSIZE = 11)
26 (
27   input bus_clk, input bus_rst,
28   input ce_clk, input ce_rst,
29   input  [63:0] i_tdata, input  i_tlast, input  i_tvalid, output i_tready,
30   output [63:0] o_tdata, output o_tlast, output o_tvalid, input  o_tready,
31   output [63:0] debug
32 );
33
```

- STR_SINK_FIFOSIZE (power of 2) used for flow control
- Sets flow control receive FIFO depth, generally leave at 11

Gain RFNoC Block



- Each block has one connection to crossbar
- Block communicates to other blocks and host via crossbar

Gain RFNoC Block I/O

Ettus Research™
A National Instruments Company

```
21
22 //
23 module noc_block_gain #(
24   parameter NOC_ID = 64'hE7757E5700000001,
25   parameter STR_SINK_FIFOSIZE = 11)
26 (
27   input bus_clk, input bus_rst,
28   input ce_clk, input ce_rst,
29   input  [63:0] i_tdata, input  i_tlast, input  i_tvalid, output i_tready,
30   output [63:0] o_tdata, output o_tlast, output o_tvalid, input  o_tready,
31   output [63:0] debug
32 );
33
```

- Each block has one connection to crossbar
- Input and output AXI stream
  - Carries CHDR packets (discussed later)
  -

Gain RFNoC Block I/O

Ettus Research™
A National Instruments Company

```
21
22 //
23 module noc_block_gain #(
24   parameter NOC_ID = 64'hE7757E5700000001,
25   parameter STR_SINK_FIFOSIZE = 11)
26 (
27   input bus_clk, input bus_rst,
28   input ce_clk, input ce_rst,
29   input  [63:0] i_tdata, input  i_tlast, input  i_tvalid, output i_tready,
30   output [63:0] o_tdata, output o_tlast, output o_tvalid, input  o_tready,
31   output [63:0] debug
32 );
33
```

- Each block has one connection to crossbar
- Input and output AXI stream
  - Carries CHDR packets (discussed later)
  - AXI stream on bus_clk domain. User logic uses ce_clk domain. Can be the same clock.

Gain RFNoC Block I/O

Ettus Research™
A National Instruments Company

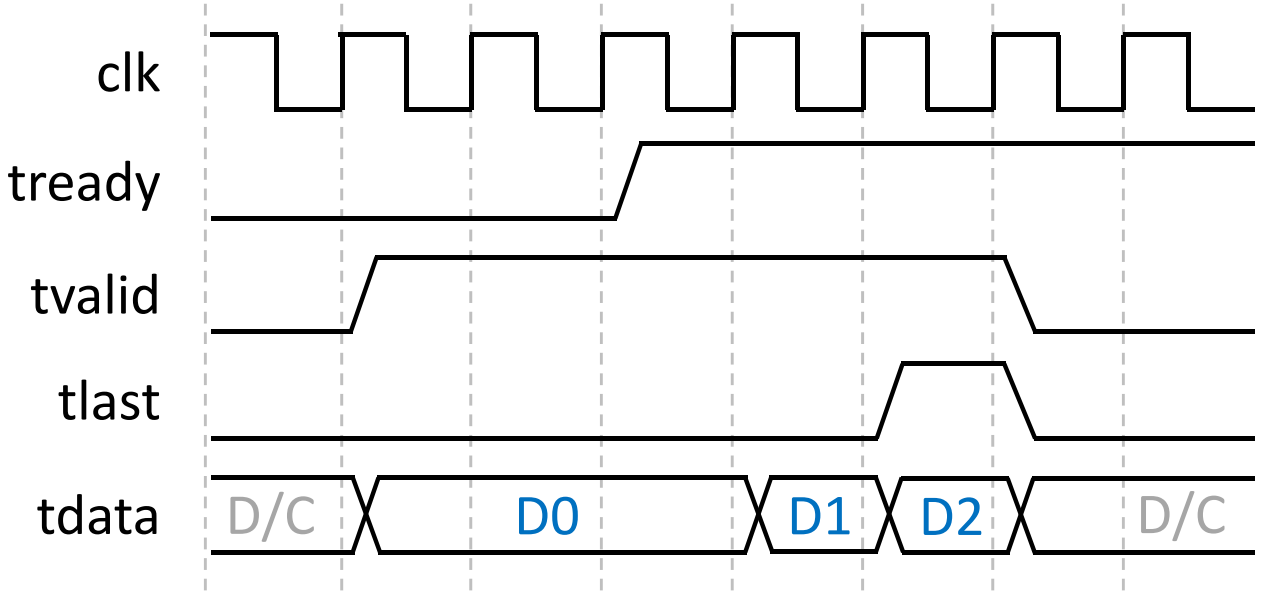
```
21
22 //
23 module noc_block_gain #(
24   parameter NOC_ID = 64'hE7757E5700000001,
25   parameter STR_SINK_FIFOSIZE = 11)
26 (
27   input bus_clk, input bus_rst,
28   input ce_clk, input ce_rst,
29   input  [63:0] i_tdata, input  i_tlast, input  i_tvalid, output i_tready,
30   output [63:0] o_tdata, output o_tlast, output o_tvalid, input  o_tready,
31   output [63:0] debug
32 );
33
```

- Debug not currently used for anything

AXI-Stream

Ettus Research™ A National Instruments Company

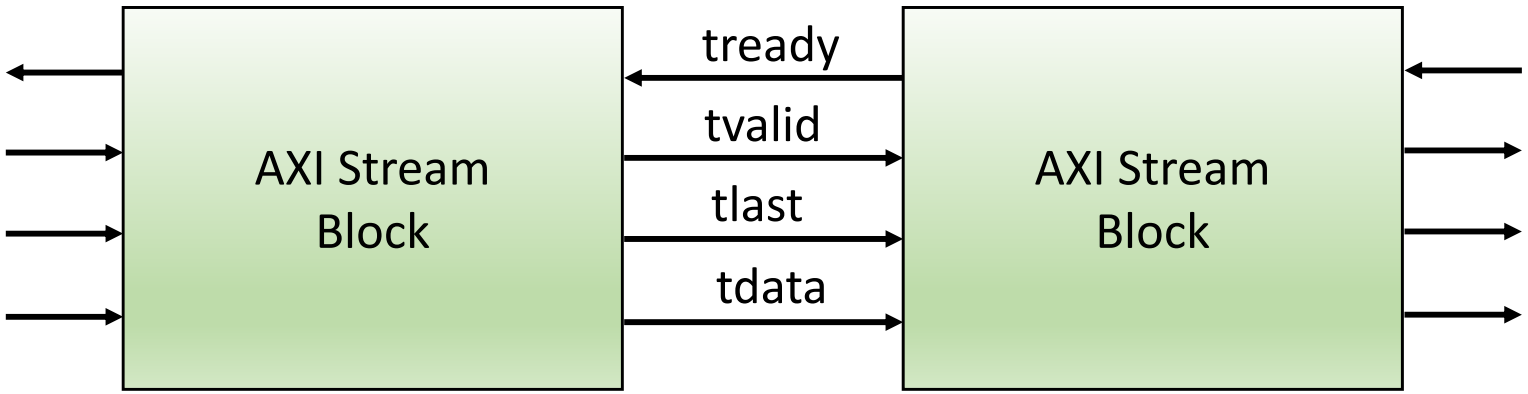
- Part of ARM AMBA standard
- Simple handshake / flow control protocol:
  - Upstream block asserts **tvalid**
  - Downstream block asserts **tready**
  - Data is consumed when tvalid & tready == 1
  - **tlast** used to delimit packets



AXI-Stream

Ettus Research™ A National Instruments Company

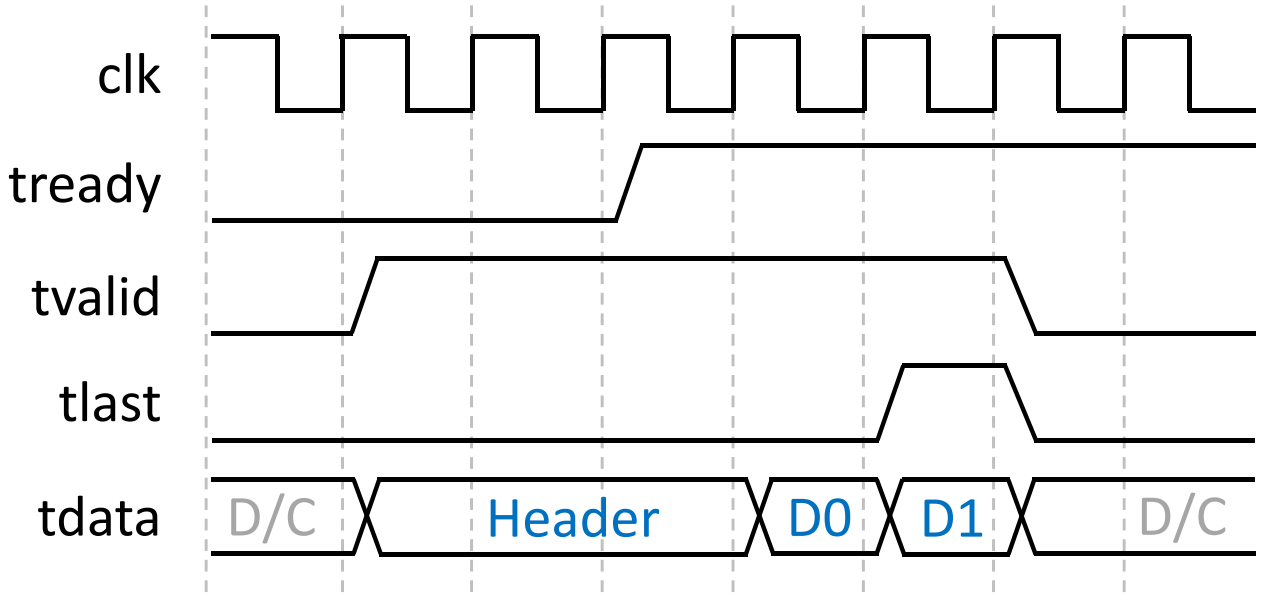
- tdata & tlast have no meaning when tvalid == 0
- **Once tvalid is asserted, it cannot be deasserted without at least one tready cycle**
- Why AXI-Stream?
  - Industry standard => Lots of existing IP
  - No need for complicated strobes – data flows through
  - Enhances reusability & composability



CHDR over AXI-Stream

Ettus Research™ A National Instruments Company

- RFNoC's packet format
- CHDR = Compressed Header



CHDR Packet Protocol

Ettus Research™ A National Instruments Company

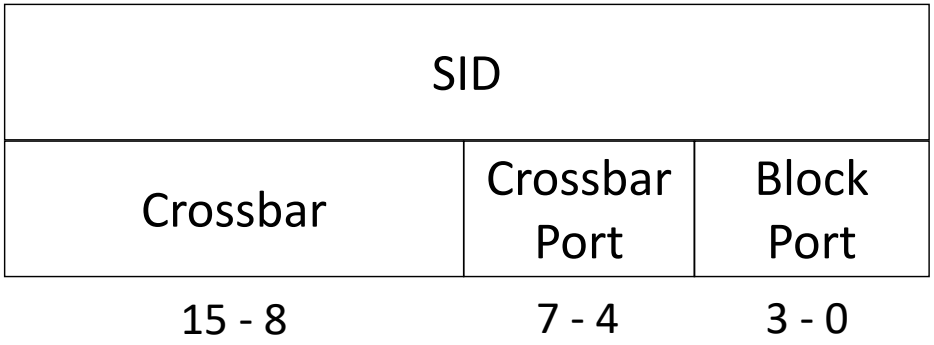
63 - 62	61	60	59 - 48	47 - 32	31 - 16	15 - 0
Pkt Type	Has Time	EOB	Seq #	Length (in bytes)	SRC SID	DST SID
Fractional Time (Optional, Has Time = 1)						
Payload						
…						

- Packet type based on bits 63, 62, & 60

63	62	60	Packet Type
0	0	0	Data
0	0	1	Data (End of Burst)
0	1	0	Flow Control
1	0	0	Command
1	1	0	Response
1	1	1	Response (Error)

Stream IDs

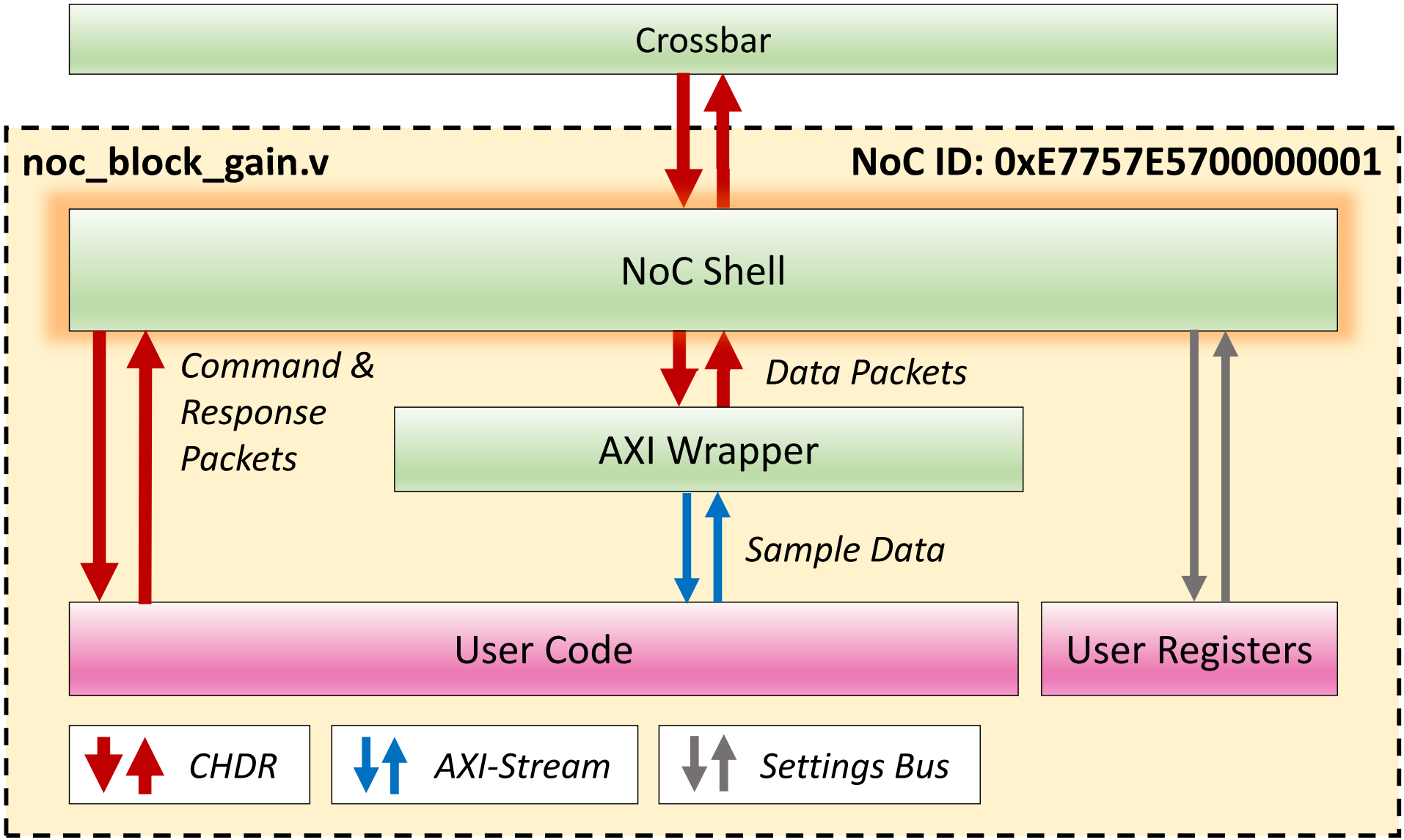
Ettus Research™ A National Instruments Company



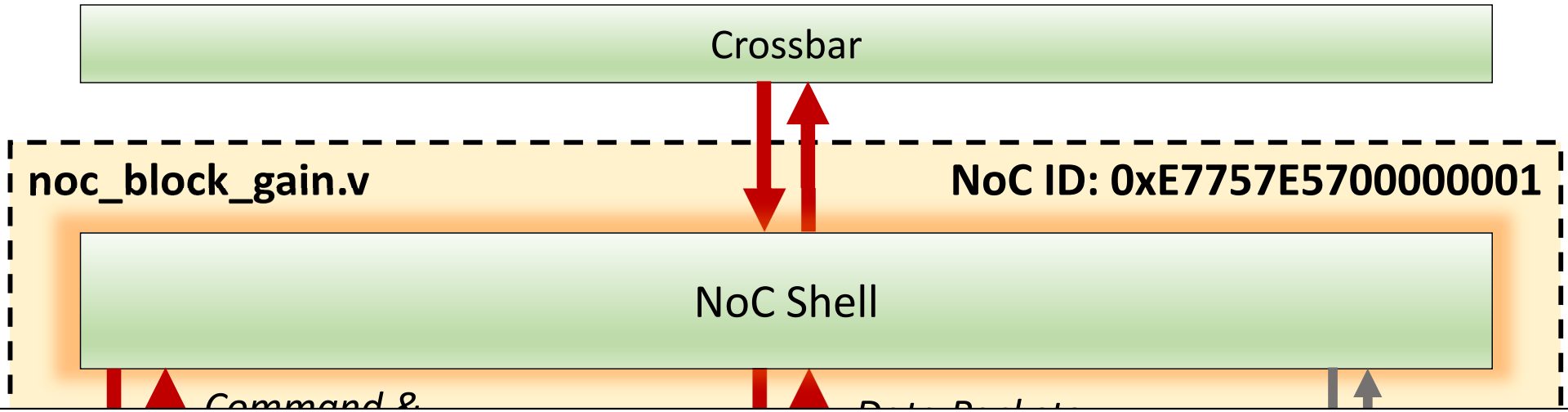
- 16 bit Stream ID
  - 256 unique crossbar (or device) IDs
  - 16 ports per crossbar
  - 16 (logical) ports per block
- Example Crossbar ID: 2, Port: 1, Block Port: 0

  SID: 2.1.0

Gain RFNoC Block



Gain RFNoC Block



- Standard boilerplate RFNoC interfacing code
  - Not strictly required
- Muxes / demuxes CHDR packets
  - Data, command, response
- Handles flow control
- Crosses clock domains

Noc Shell Parameters

Ettus Research™
A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73    .str_sink_tdata(str_sink_tdata), .str_sink_tlast(str_sink_tlast), .str_sink_tvalid(str_sink_tvalid), .str_sink_t
74    // Stream Source
75    .str_src_tdata(str_src_tdata), .str_src_tlast(str_src_tlast), .str_src_tvalid(str_src_tvalid), .str_src_tready(s
76    // Stream IDs set by host
77    .src_sid(src_sid),                   // SID of this block
78    .next_dst_sid(next_dst_sid),         // Next destination SID
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- NOC_ID and STR_SINK_FIFOSIZE module parameters

Noc Shell I/O

Ettus Research™ A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73    .str_sink_tdata(str_sink_tdata), .str_sink_tlast(str_sink_tlast), .str_sink_tvalid(str_sink_tvalid), .str_sink_t
74    // Stream Source
75    .str_src_tdata(str_src_tdata), .str_src_tlast(str_src_tlast), .str_src_tvalid(str_src_tvalid), .str_src_tready(s
76    // Stream IDs set by host
77    .src_sid(src_sid),                   // SID of this block
78    .next_dst_sid(next_dst_sid),         // Next destination SID
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- Connection to crossbar

Noc Shell I/O

Ettus Research™ A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73    .str_sink_tdata(str_sink_tdata), .str_sink_tlast(str_sink_tlast), .str_sink_tvalid(str_sink_tvalid), .str_sink_t
74    // Stream Source
75    .str_src_tdata(str_src_tdata), .str_src_tlast(str_src_tlast), .str_src_tvalid(str_src_tvalid), .str_src_tready(s
76    // Stream IDs set by host
77    .src_sid(src_sid),                   // SID of this block
78    .next_dst_sid(next_dst_sid),         // Next destination SID
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- clk is the user logic clock domain, usually connected to ce_clk
- Noc Shell internally crosses clock domains

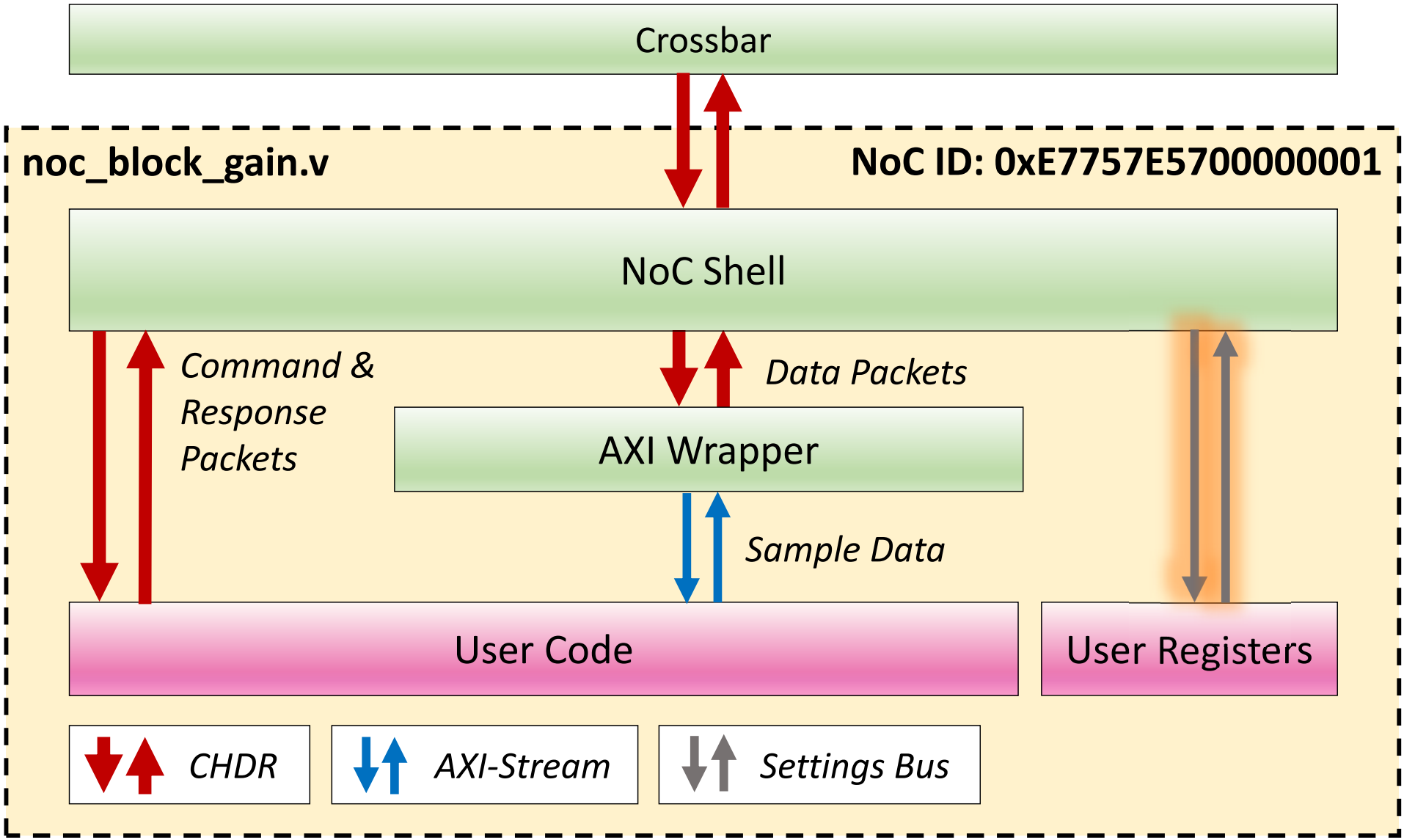
Noc Shell I/O

Ettus Research™ A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73    .str_sink_tdata(str_sink_tdata), .str_sink_tlast(str_sink_tlast), .str_sink_tvalid(str_sink_tvalid), .str_sink_t
74    // Stream Source
75    .str_src_tdata(str_src_tdata), .str_src_tlast(str_src_tlast), .str_src_tvalid(str_src_tvalid), .str_src_tready(s
76    // Stream IDs set by host
77    .src_sid(src_sid),                   // SID of this block
78    .next_dst_sid(next_dst_sid),         // Next destination SID
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- Settings bus: User register write and readback
- set_time and set_has_time for timed commands

Gain RFNoC Block



Register Space

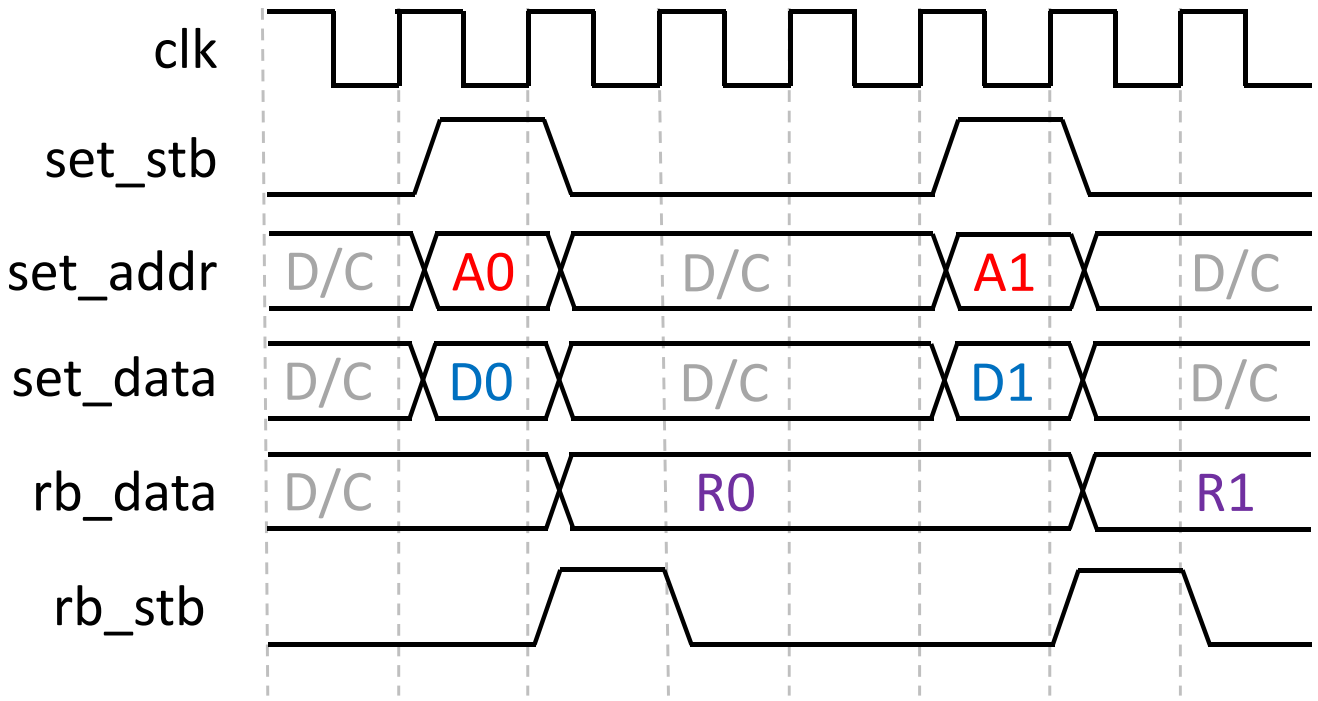
Ettus Research™
A National Instruments Company

- Settings Bus 8-bit address = 256 regs
- NoC Shell Regs: 0 – 127
  - Special registers to setup NoC Shell
  - 0 – 4: Flow Control
  - 5 – 8: SIDs (Src, Next Dst, Resp In, Resp Out)
  - 124: User readback address
  - 127: NoC Shell readback address
- User Regs: 128 – 255

Settings Bus

Ettus Research™ A National Instruments Company

- Control packet payload sets addr & data
  - [63:0] = {24'd0, set_addr[7:0], set_data[31:0]}
- Response packet payload has readback data
- User must assert readback strobe (or leave 1'b1)



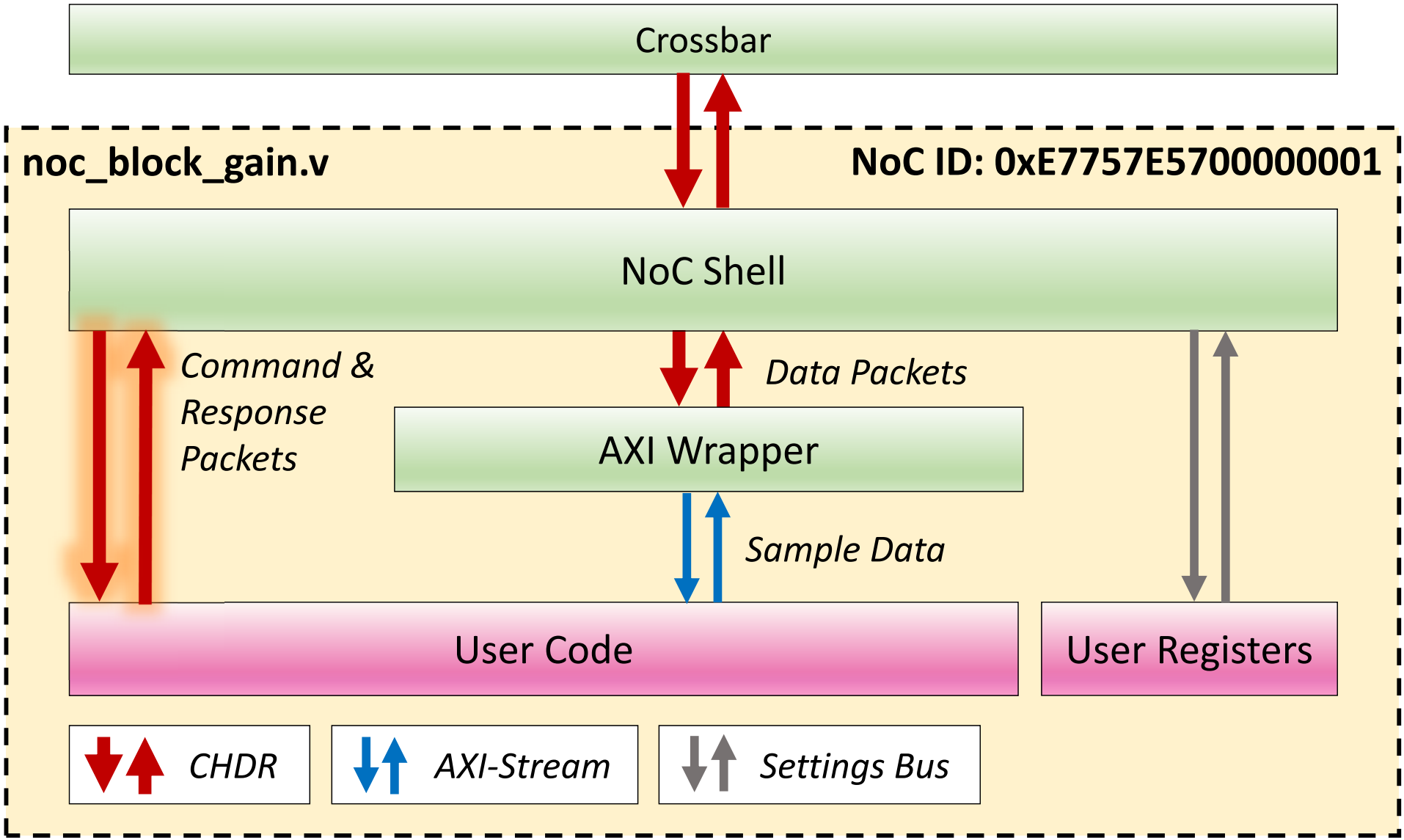
Noc Shell I/O

Ettus Research™ A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
```

- Access registers in other blocks
- cmdout: CHDR command packets out
  - Register write (i.e. settings bus writes) in other blocks
- ackin: CHDR response (ack) packets in
  - Register readback from other blocks
  - Output by other block's noc shell (remember rb_stb?)
- We will go over an example block using this port later

Gain RFNoC Block



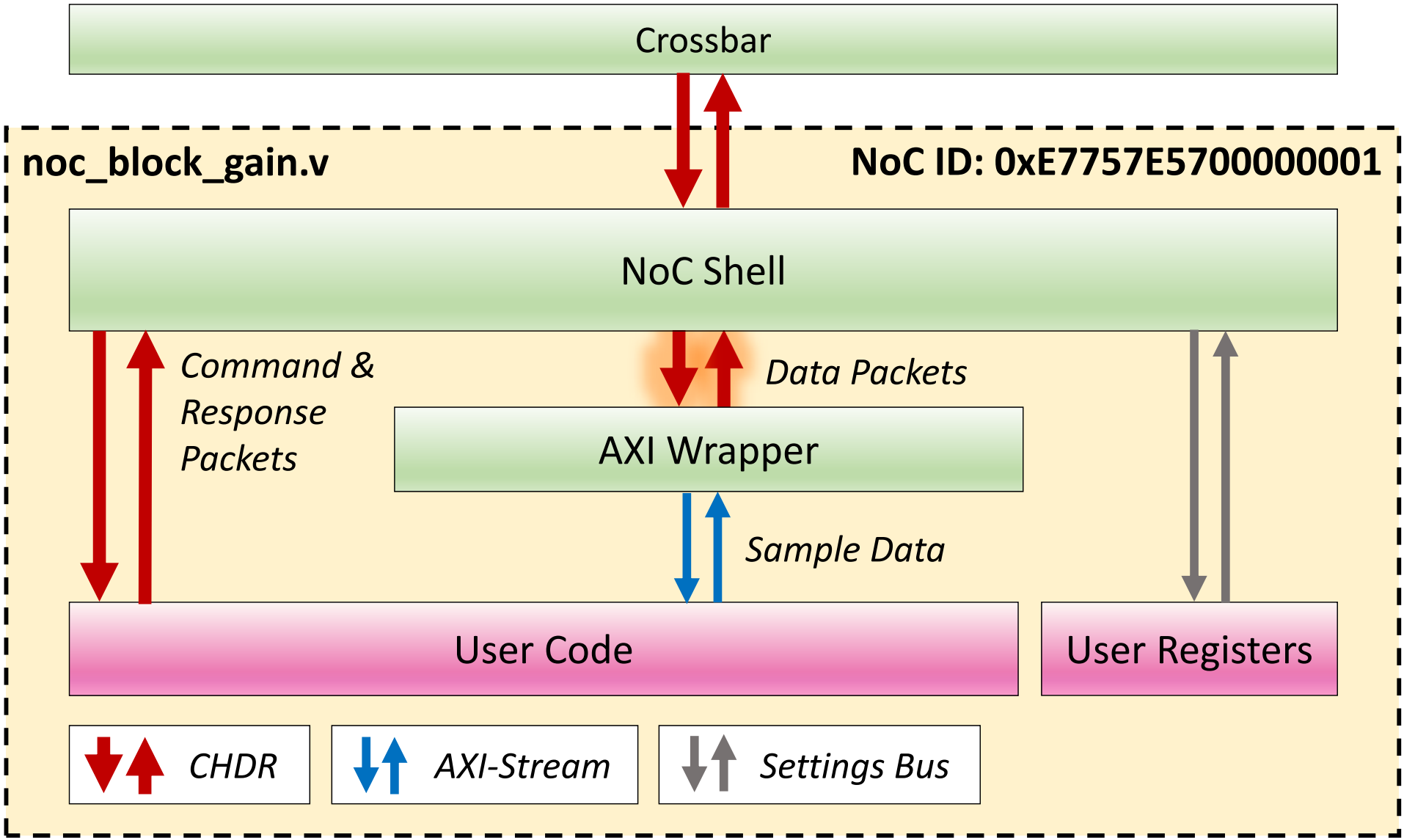
Noc Shell I/O

Ettus Research™ A National Instruments Company

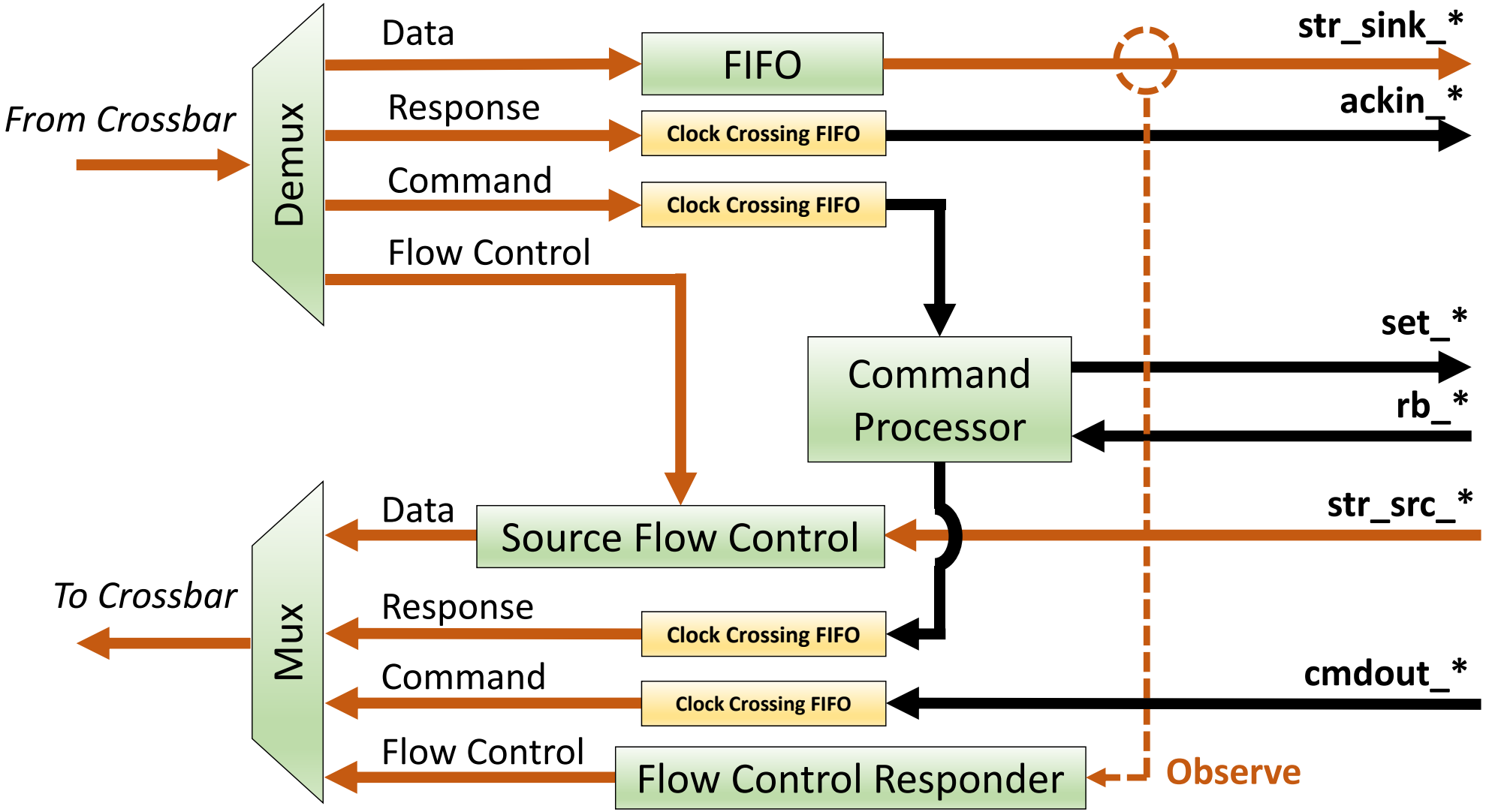
```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73    .str_sink_tdata(str_sink_tdata), .str_sink_tlast(str_sink_tlast), .str_sink_tvalid(str_sink_tvalid), .str_sink_t
74    // Stream Source
75    .str_src_tdata(str_src_tdata), .str_src_tlast(str_src_tlast), .str_src_tvalid(str_src_tvalid), .str_src_tready(s
76    // Stream IDs set by host
77    .src_sid(src_sid),                    // SID of this block
78    .next_dst_sid(next_dst_sid),          // Next destination SID
79    .resp_in_dst_sid(resp_in_dst_sid),    // Response destination SID for input stream responses / errors
80
81
82
83
```

- str_sink: CHDR data packets out
- str_src: CHDR data packets in

Gain RFNoC Block



NoC Shell Internals



Noc Shell I/O

Ettus Research™ A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73    .str_sink_tdata(str_sink_tdata), .str_sink_tlast(str_sink_tlast), .str_sink_tvalid(str_sink_tvalid), .str_sink_t
74    // Stream Source
75    .str_src_tdata(str_src_tdata), .str_src_tlast(str_src_tlast), .str_src_tvalid(str_src_tvalid), .str_src_tready(s
76    // Stream IDs set by host
77    .src_sid(src_sid),                   // SID of this block
78    .next_dst_sid(next_dst_sid),         // Next destination SID
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- Useful stream IDs for forming packets

Noc Shell I/O

Ettus Research™ A National Instruments Company

```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65
66
67
68
69
70                                                                          lout_t
71                                                                          dy),
72
73                                                                          sink_t
74
75                                                                          eady(s
76
77
78
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- VITA time input to control timed settings bus writes
- Avoid using this approach
  - Only useful in special circumstances such as Radio Core
- For timed command examples, see DDC / DUC blocks

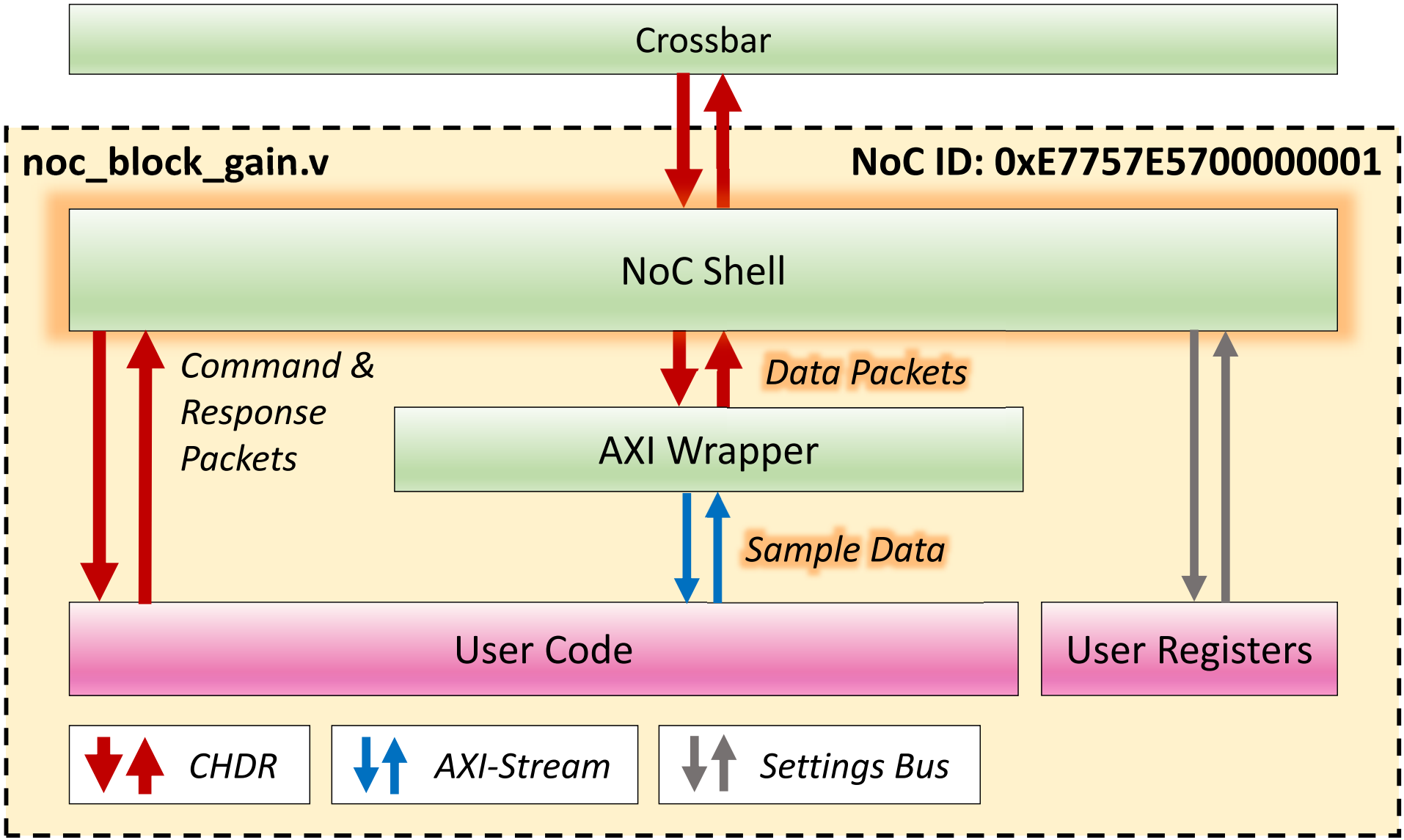
Noc Shell I/O

Ettus Research™ A National Instruments Company

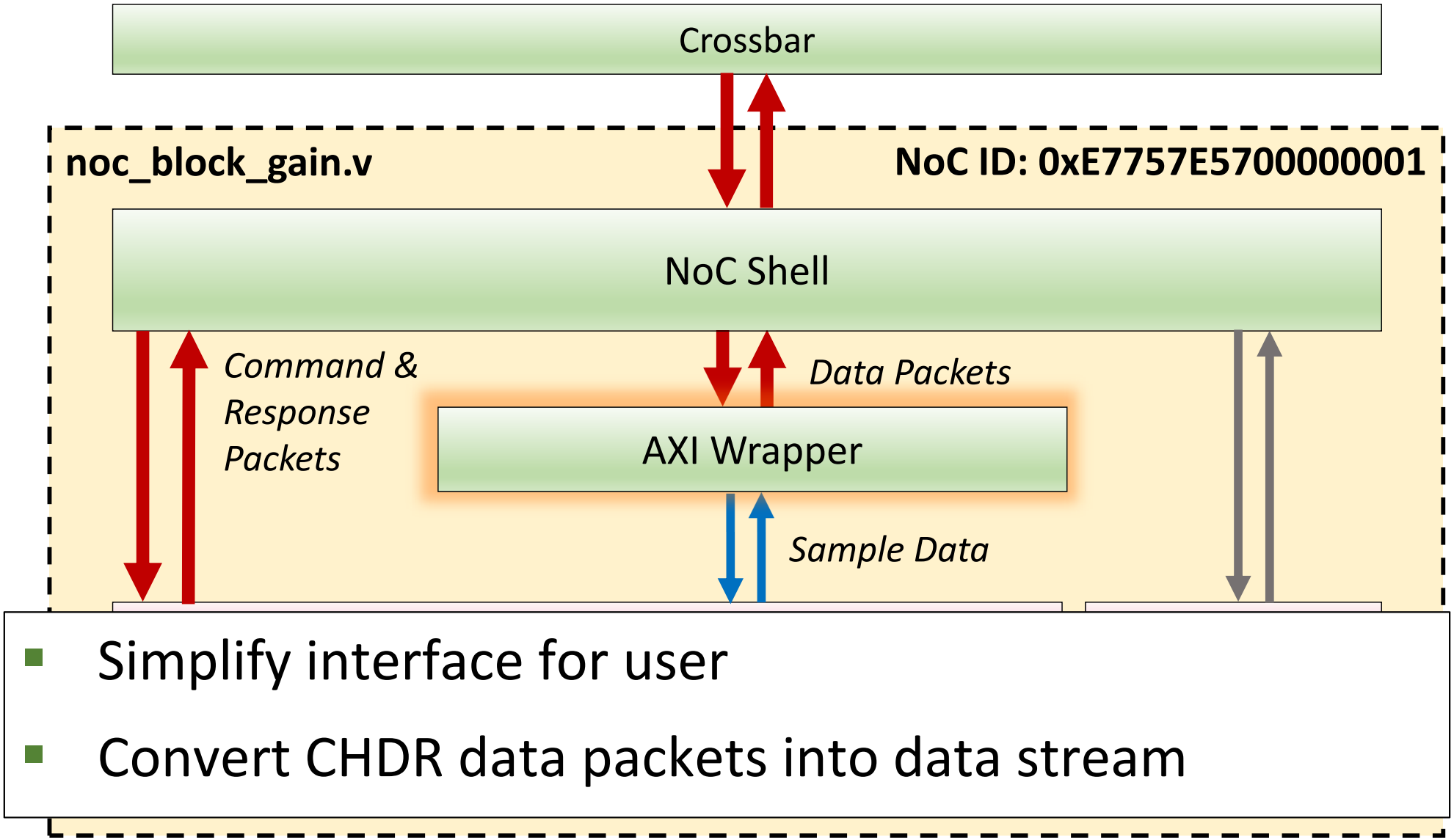
```
57  noc_shell #(
58    .NOC_ID(NOC_ID),
59    .STR_SINK_FIFOSIZE(STR_SINK_FIFOSIZE))
60  noc_shell (
61    .bus_clk(bus_clk), .bus_rst(bus_rst),
62    .i_tdata(i_tdata), .i_tlast(i_tlast), .i_tvalid(i_tvalid), .i_tready(i_tready),
63    .o_tdata(o_tdata), .o_tlast(o_tlast), .o_tvalid(o_tvalid), .o_tready(o_tready),
64    // Computer Engine Clock Domain
65    .clk(ce_clk), .reset(ce_rst),
66    // Control Sink
67    .set_data(set_data), .set_addr(set_addr), .set_stb(set_stb), .set_time(), .set_has_time(),
68    .rb_stb(1'b1), .rb_data(rb_data), .rb_addr(rb_addr),
69    // Control Source
70    .cmdout_tdata(cmdout_tdata), .cmdout_tlast(cmdout_tlast), .cmdout_tvalid(cmdout_tvalid), .cmdout_tready(cmdout_t
71    .ackin_tdata(ackin_tdata), .ackin_tlast(ackin_tlast), .ackin_tvalid(ackin_tvalid), .ackin_tready(ackin_tready),
72    // Stream Sink
73                                                                                          sink_t
74
75                                                                                          eady(s
76
77
78
79    .resp_in_dst_sid(resp_in_dst_sid),   // Response destination SID for input stream responses / errors
80    .resp_out_dst_sid(resp_out_dst_sid), // Response destination SID for output stream responses / errors
81    // Misc
82    .vita_time('d0), .clear_tx_seqnum(clear_tx_seqnum),
83    .debug(debug));
```

- Automatically asserted when UHD initializes the block at start up
- Can be used as a user logic reset

Gain RFNoC Block



Gain RFNoC Block



- Simplify interface for user
- Convert CHDR data packets into data stream

AXI Wrapper Parameters

Ettus Research™ A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
```

- SIMPLE_MODE: auto generate output packet header
  - Useful if input / output rate is 1:1
  - Examples:
    - FIR Filter that does not decimate or interpolate
    - FFT
    - Gain block
  - If not 1:1, header must be handled manually
    - N:M integer ratio blocks can use the module AXI Rate Change
    - See DUC / DDC RFNoC blocks

AXI Wrapper Parameters

Ettus Research™
A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
111    .m_axis_data_tdata(m_axis_data_tdata),
112    .m_axis_data_tlast(m_axis_data_tlast),
113    .m_axis_data_tvalid(m_axis_data_tvalid)
114    .m_axis_data_tready(m_axis_data_tready),
115    .m_axis_config_tdata(),
116    .m_axis_config_tvalid(),
117    .m_axis_config_tready(),
118    .m_axis_pkt_len_tdata(),
119    .m_axis_pkt_len_tvalid(),
120    .m_axis_pkt_len_tready());
```

- CHDR data packets to / from Noc Shell
- Notice ce_clk & bus_clk, there is an clock crossing FIFO inside
- next_dst_sid: Stream ID of next RFNoC block (or host) in flowgraph

AXI Wrapper Parameters

Ettus Research™ A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
111    .m_axis_data_tdata(m_axis_data_tdata),
112    .m_axis_data_tlast(m_axis_data_tlast),
113    .m_axis_data_tvalid(m_axis_data_tvalid),
114    .m_axis_data_tready(m_axis_data_tready),
115    .m_axis_data_tuser(),
116    .s_axis_data_tdata(s_axis_data_tdata),
117    .s_axis_data_tlast(s_axis_data_tlast),
118    .s_axis_data_tvalid(s_axis_data_tvalid),
119    .s_axis_data_tready(s_axis_data_tready),
120    .s_axis_data_tuser(),
121    .m_axis_config_tdata()
```

- Sample stream to / from user
- 32-bit wide, typically sc16 (16-bit I, 16-bit Q)
- tlast indicates packet boundaries

AXI Wrapper Parameters

Ettus Research™
A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
111    .m_axis_data_tdata(m_axis_data_tdata),
112    .m_axis_data_tlast(m_axis_data_tlast),
113    .m_axis_data_tvalid(m_axis_data_tvalid),
114    .m_axis_data_tready(m_axis_data_tready),
115    .m_axis_data_tuser(),
116    .s_axis_data_tdata(s_axis_data_tdata),
117    .s_axis_data_tlast(s_axis_data_tlast),
118    .s_axis_data_tvalid(s_axis_data_tvalid),
119    .s_axis_data_tready(s_axis_data_tready),
120    .s_axis_data_tuser(),
121    .m_axis_config_tdata(),
122    .m_axis_config_tlast(),
```

- Packet headers
- Must set s_axis_data_tuser if SIMPLE_MODE = 0

AXI Wrapper Parameters

Ettus Research™ A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
111    .m_axis_data_tdata(m_axis_data_tdata),
112    .m_axis_data_tlast(m_axis_data_tlast),
113    .m_axis_data_tvalid(m_axis_data_tvalid),
114    .m_axis_data_tready(m_axis_data_tready),
115    .m_axis_data_tuser(),
116    .s_axis_data_tdata(s_axis_data_tdata),
117    .s_axis_data_tlast(s_axis_data_tlast),
118    .s_axis_data_tvalid(s_axis_data_tvalid),
119    .s_axis_data_tready(s_axis_data_tready),
120    .s_axis_data_tuser(),
121    .m_axis_config_tdata(),
122    .m_axis_config_tlast(),
123    .m_axis_config_tvalid(),
124    .m_axis_config_tready(),
```

- Incoming packet header
- Outgoing packet header
- Must set s_axis_data_tuser if SIMPLE_MODE = 0

AXI Wrapper Parameters

Ettus Research™
A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
111    .m_axis_data_tdata(m_axis_data_tdata),
112    .m_axis_data_tlast(m_axis_data_tlast),
113    .m_axis_data_tvalid(m_axis_data_tvalid),
114    .m_axis_data_tready(m_axis_data_tready),
115
116
117
118    .s_axis_data_tvalid(s_axis_data_tvalid),
119    .s_axis_data_tready(s_axis_data_tready),
120    .s_axis_data_tuser(),
121    .m_axis_config_tdata(),
122    .m_axis_config_tlast(),
123    .m_axis_config_tvalid(),
124    .m_axis_config_tready(),
125    .m_axis_pkt_len_tdata(),
126    .m_axis_pkt_len_tvalid(),
127    .m_axis_pkt_len_tready());
```

- Legacy, can be ignored

AXI Wrapper Parameters

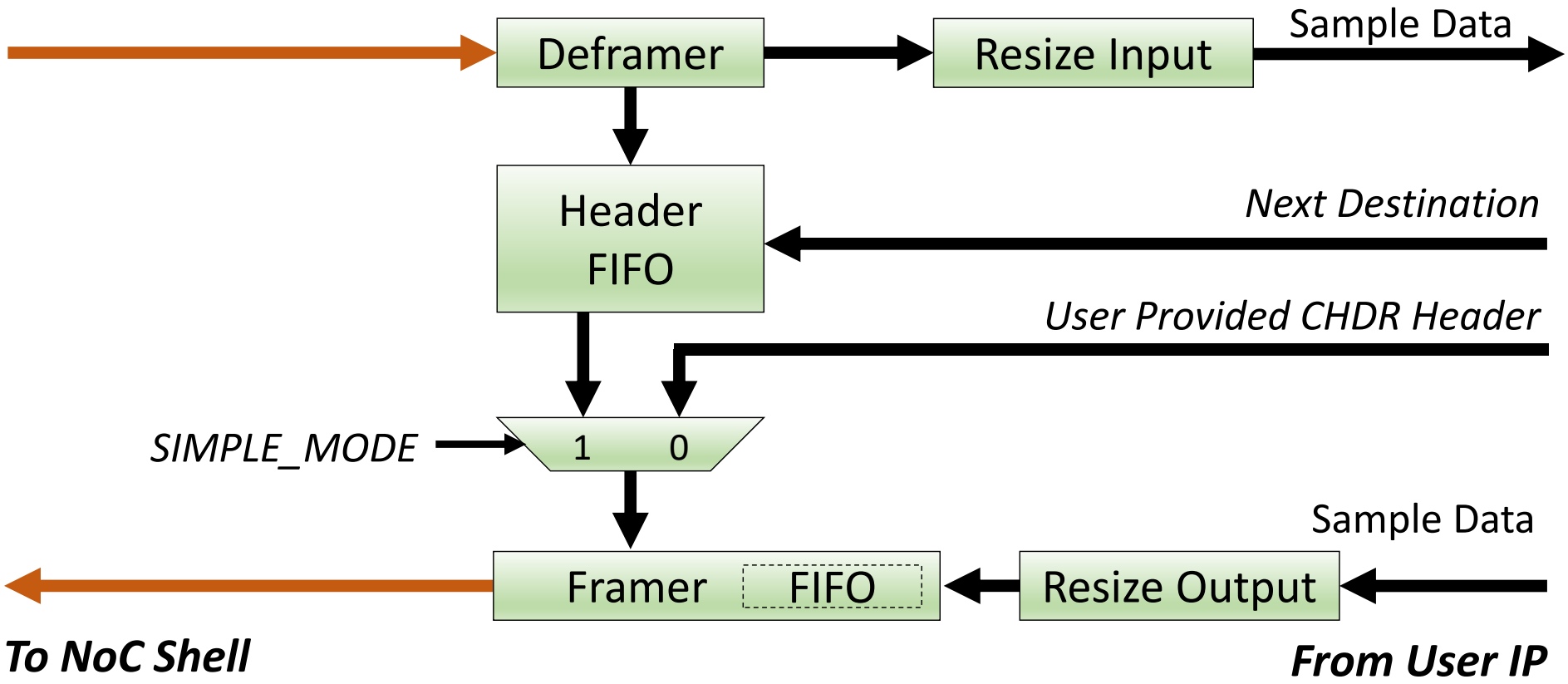
Ettus Research™ A National Instruments Company

```
101  axi_wrapper #(
102    .SIMPLE_MODE(1))
103  axi_wrapper (
104    .clk(ce_clk), .reset(ce_rst),
105    .bus_clk(bus_clk), .bus_rst(bus_rst),
106    .clear_tx_seqnum(clear_tx_seqnum),
107    .next_dst(next_dst_sid),
108    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
109    .i_tdata(str_sink_tdata), .i_tlast(str_sink_tlast), .i_tvalid(str_sink_tvalid), .i_tready(str_sink_tready),
110    .o_tdata(str_src_tdata), .o_tlast(str_src_tlast), .o_tvalid(str_src_tvalid), .o_tready(str_src_tready),
111    .m_axis_data_tdata(m_axis_data_tdata),
112    .m_axis_data_tlast(m_axis_data_tlast),
113
114
115
116
117
118
119
120
121
122    .m_axis_config_tlast(),
123    .m_axis_config_tvalid(),
124    .m_axis_config_tready(),
125    .m_axis_pkt_len_tdata(),
126    .m_axis_pkt_len_tvalid(),
127    .m_axis_pkt_len_tready());
```

- Resize incoming packet length
- Must set parameter RESIZE_INPUT_PACKET = 1 (not shown)

AXI Wrapper

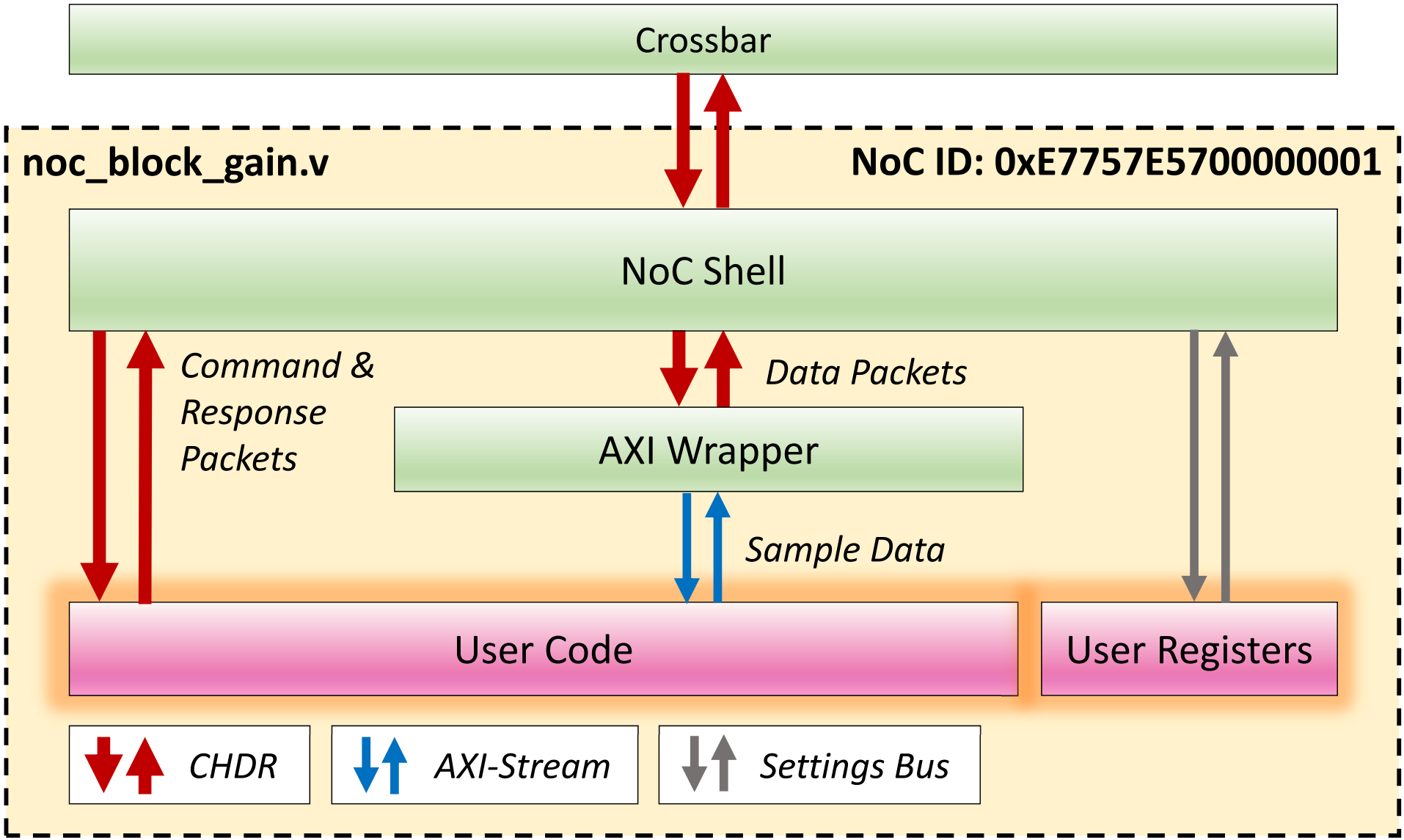
Ettus Research™
A National Instruments Company



bus_clk ce_clk

Gain RFNoC Block

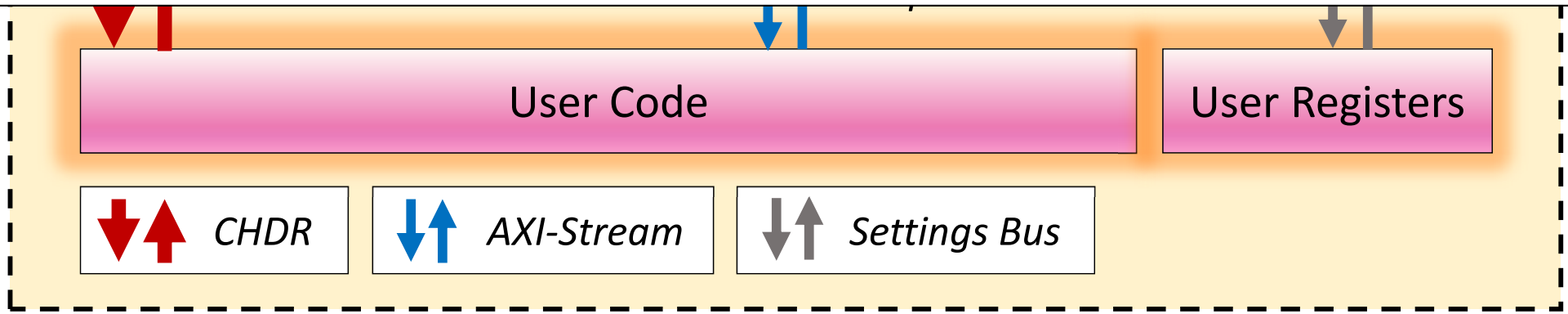
Ettus Research™
A National Instruments Company



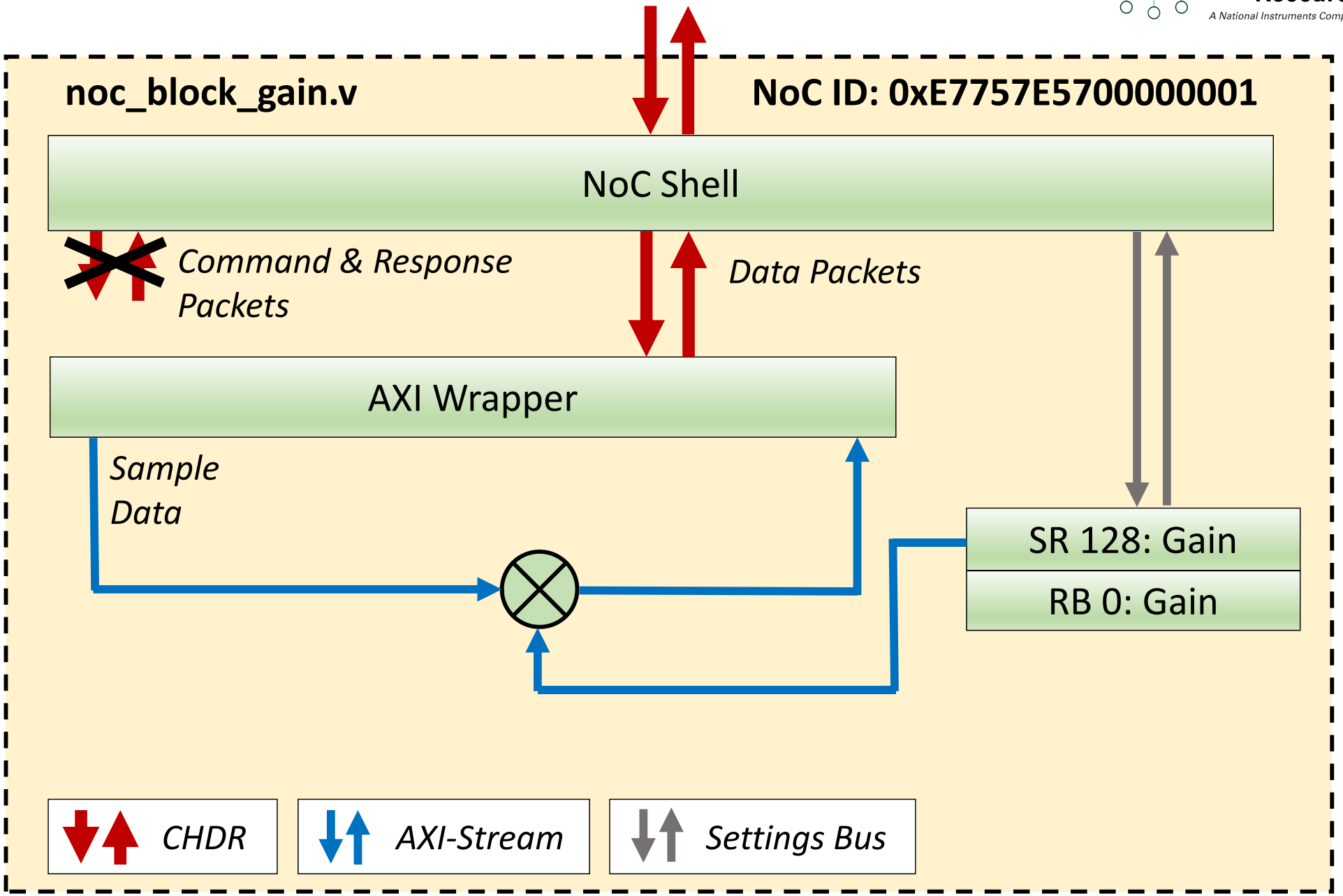
Gain RFNoC Block

Ettus Research™ A National Instruments Company

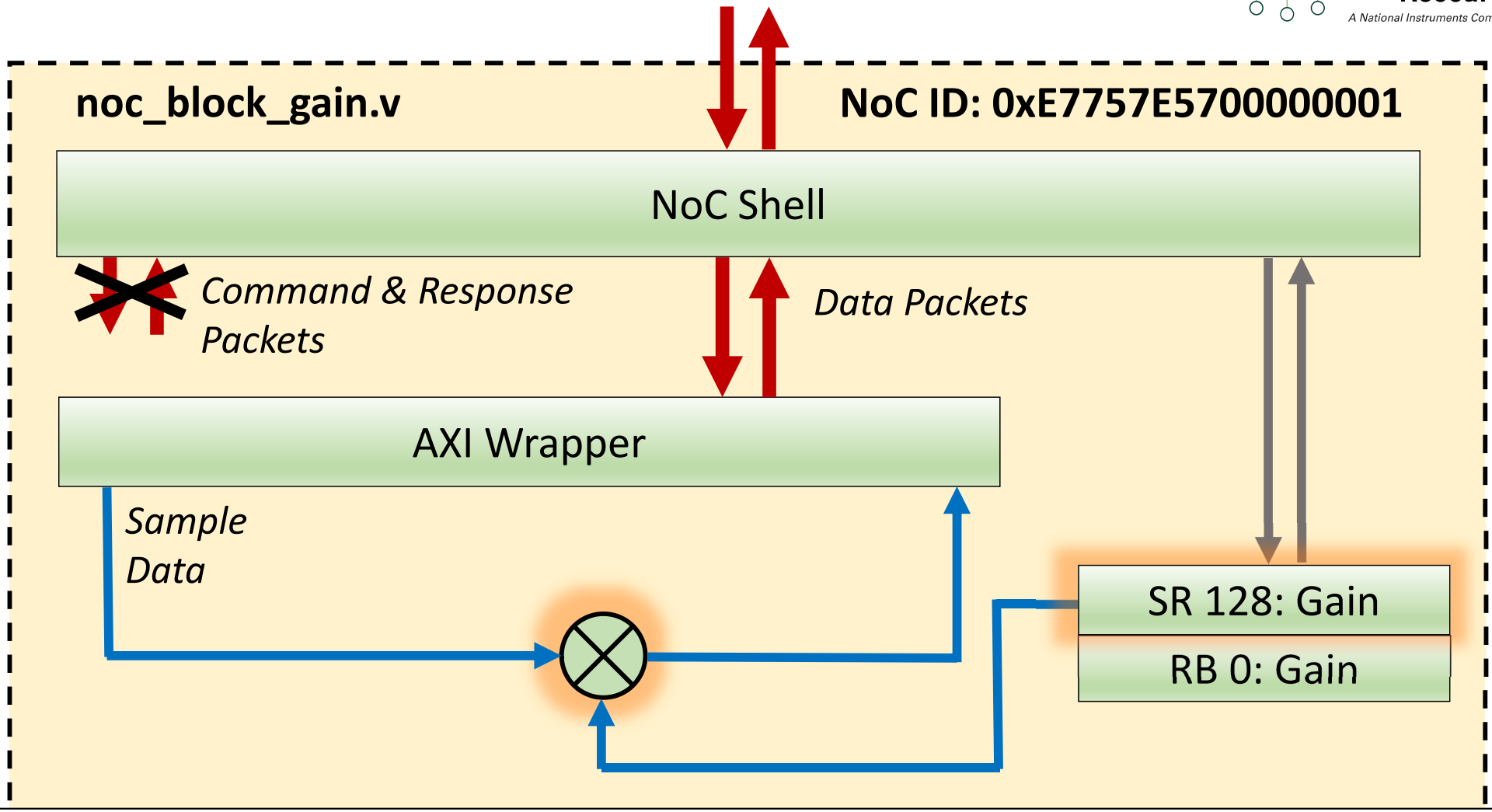
- Many implementation options for user code
  - **HDL**: Verilog, SystemVerilog, VHDL
  - **Vivado IP**: FIR, FFT, DDS, CORDIC, Turbo Decoder, etc
  - **Vivado Block Diagrams (BD)**: Microblaze
  - **Vivado High-Level Synthesis (HLS)**: C and C++
- User read/write registers to support configuration, control, and status readback



Gain RFNoC Block



Gain RFNoC Block



- Multiply data stream with constant value set via user register

Write Custom HDL

Ettus Research™ A National Instruments Company

- Open **rfnoc-tutorial/rfnoc/fpga-src/noc_block_gain.v**
- Change **TEST_REG_0** into a 16-bit register **SR_GAIN**

```
localparam [7:0] SR_GAIN = SR_USER_REG_BASE;
localparam [7:0] SR_TEST_REG_1 = SR_USER_REG_BASE + 8'd1;

wire [15:0] gain;
setting_reg #(
  .my_addr(SR_GAIN), .awidth(8), .width(16))
sr_gain (
  .clk(ce_clk), .rst(ce_rst),
  .strobe(set_stb), .addr(set_addr), .in(set_data), .out(gain), .changed());
```

```
always @(posedge ce_clk) begin
  case(rb_addr)
    8'd0 : rb_data <= {48'd0, gain};
    8'd1 : rb_data <= {32'd0, test_reg_1};
    default : rb_data <= 64'h0BADC0DE0BADC0DE;
  endcase
end
```

Write Custom HDL

Ettus Research™

```
167  localparam [7:0] SR_GAIN = SR_USER_REG_BASE;
168  localparam [7:0] SR_TEST_REG_1 = SR_USER_REG_BASE + 8'd1;
169
170  wire [15:0] gain;
171  setting_reg #(
172    .my_addr(SR_GAIN), .awidth(8), .width(16))
173  sr_gain (
174    .clk(ce_clk), .rst(ce_rst),
175    .strobe(set_stb), .addr(set_addr), .in(set_data), .out(gain), .changed());
176
177  wire [31:0] test_reg_1;
178  setting_reg #(
179    .my_addr(SR_TEST_REG_1), .awidth(8), .width(32))
180  sr_test_reg_1 (
181    .clk(ce_clk), .rst(ce_rst),
182    .strobe(set_stb), .addr(set_addr), .in(set_data), .out(test_reg_1), .changed());
183
184  // Readback registers
185  // rb_stb set to 1'b1 on NoC Shell
186  always @(posedge ce_clk) begin
187    case(rb_addr)
188      8'd0 : rb_data <= {48'd0, gain};
189      8'd1 : rb_data <= {32'd0, test_reg_1};
190      default : rb_data <= 64'h0BADC0DE0BADC0DE;
191    endcase
192  end
```

Write Custom HDL

Ettus Research™ A National Instruments Company

- Write constant multiplier
- Vivado will infer a multiplier, likely using a DSP48

```
wire [15:0] i, q;
wire [31:0] i_mult_gain, q_mult_gain;

assign i = m_axis_data_tdata[31:16];
assign q = m_axis_data_tdata[15:0];

assign i_mult_gain = i*gain;
assign q_mult_gain = q*gain;

assign m_axis_data_tready = s_axis_data_tready;
assign s_axis_data_tvalid = m_axis_data_tvalid;
assign s_axis_data_tlast  = m_axis_data_tlast;
assign s_axis_data_tdata  = {i_mult_gain[15:0], q_mult_gain[15:0]};
```

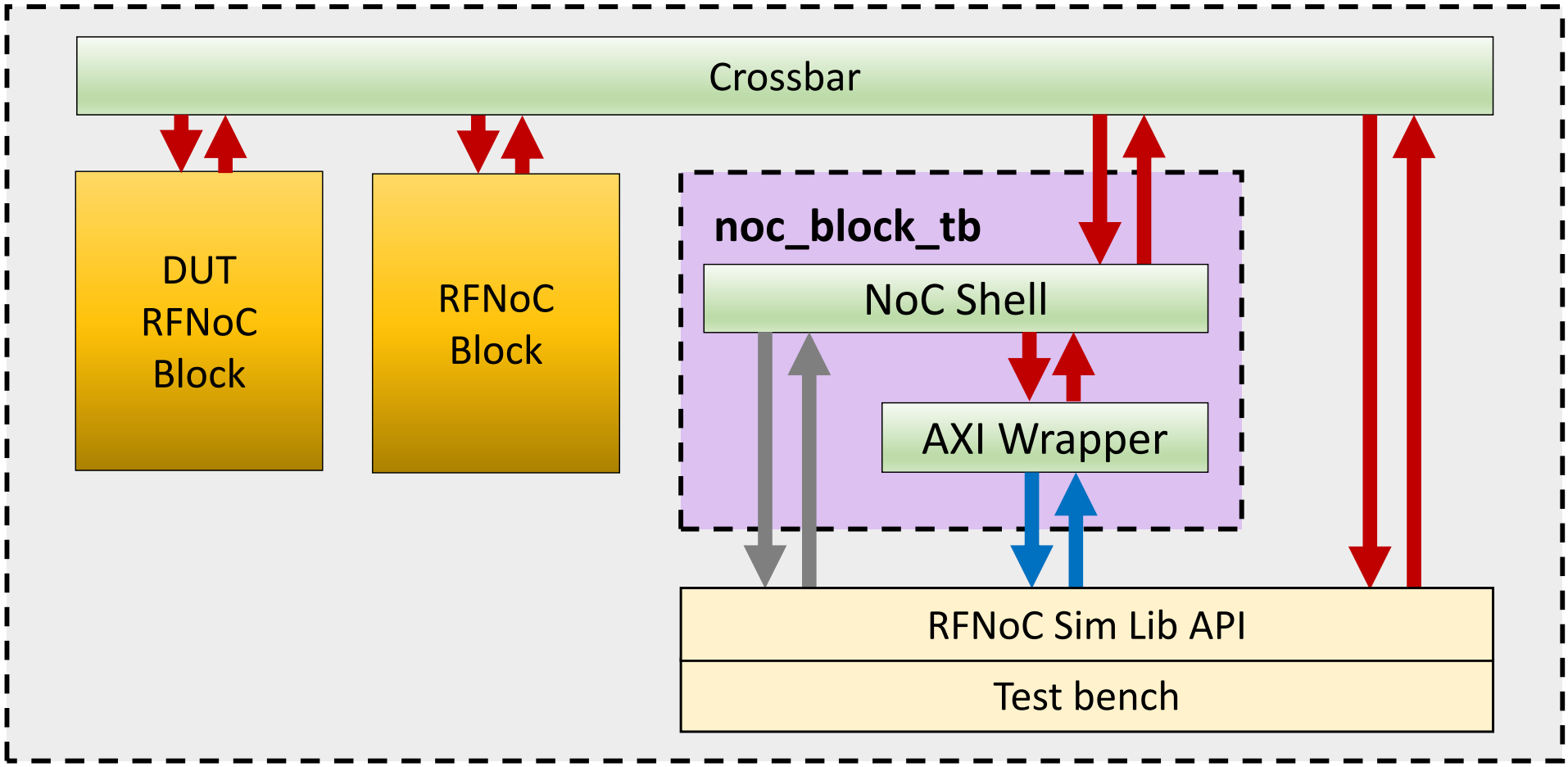
Write Custom HDL

Ettus Research™

```
185   // Readback registers
186   // rb_stb set to 1'b1 on NoC Shell
187   always @(posedge ce_clk) begin
188     case(rb_addr)
189       8'd0 : rb_data <= {32'd0, test_reg_0};
190       8'd1 : rb_data <= {32'd0, test_reg_1};
191       default : rb_data <= 64'h0BADC0DE0BADC0DE;
192     endcase
193   end
194
195   wire [15:0] i, q;
196   wire [31:0] i_mult_gain, q_mult_gain;
197
198   assign i = m_axis_data_tdata[31:16];
199   assign q = m_axis_data_tdata[15:0];
200
201   assign i_mult_gain = i*gain;
202   assign q_mult_gain = q*gain;
203
204   assign s_axis_data_tdata  = {i_mult_gain[15:0], q_mult_gain[15:0]};
205   assign m_axis_data_tready = s_axis_data_tready;
206   assign s_axis_data_tvalid = m_axis_data_tvalid;
207   assign s_axis_data_tlast  = m_axis_data_tlast;
208
209 endmodule
210
```

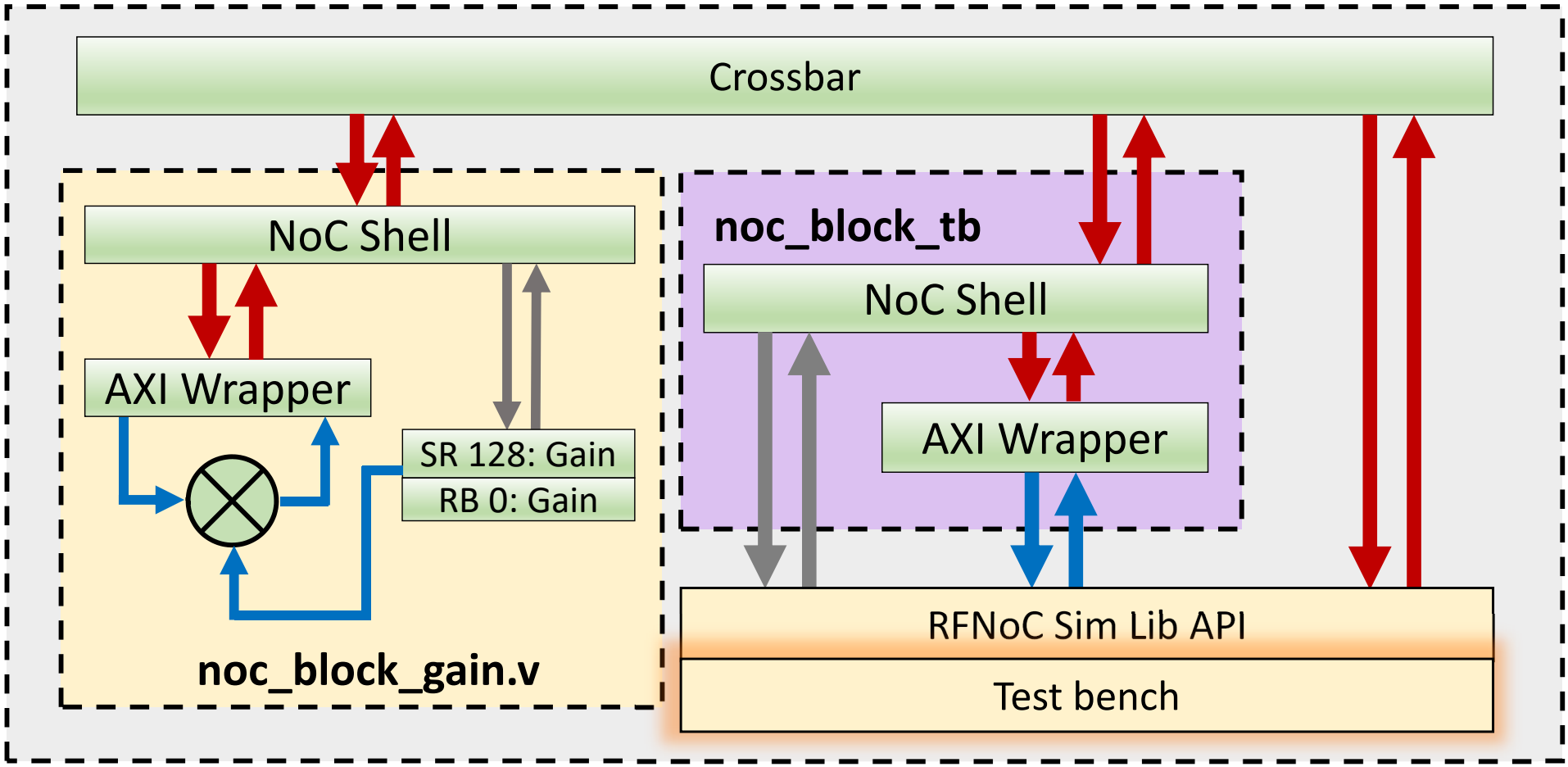
RFNoC Test Bench Infrastructure

Ettus Research™ A National Instruments Company



- Sets up a realistic, simulated RFNoC infrastructure
- Simple API for interacting with blocks

RFNoC Test Bench Infrastructure



- Need to fill out testbench

Noc Block Gain Test Bench

- Initial setup:
- **source ~/rfnoc-workshop/setup_env.sh**
- **cd ~/rfnoc-workshop/src/rfnoc-tutorial**
- **mkdir build && cd build/**
- **cmake ../**
- **make test_tb**
- **make noc_block_gain_tb**
- Test bench failed… need to update it for our block!

Write Test Bench

Ettus Research™ A National Instruments Company

- Edit rfnoc-tutorial/rfnoc/testbenches/noc_block_gain_tb/noc_block_gain_tb.sv

```
initial begin : tb_main
  string s;
  logic [31:0] random_word;
  logic [63:0] readback;
  shortint gain;
```

```
`TEST_CASE_START("Write / readback user registers");
random_word = $random();
tb_streamer.write_user_reg(sid_noc_block_gain, noc_block_gain.SR_GAIN,
                           random_word[15:0]);
tb_streamer.read_user_reg(sid_noc_block_gain, 0, readback);
$sformat(s, "User register 0 incorrect readback! Expected: %0d, Actual %0d",
         readback[15:0], random_word[15:0]);
`ASSERT_ERROR(readback[15:0] == random_word[15:0], s);
```

Write Test Bench

Ettus Research™
A National Instruments Company

```
41  /*********************************************************
42  ** Verification
43  *********************************************************/
44  initial begin : tb_main
45    string s;
46    logic [31:0] random_word;
47    logic [63:0] readback;
48    shortint gain;
49
```

```
76    /*********************************************************
77    ** Test 4 -- Write / readback user registers
78    *********************************************************/
79    `TEST_CASE_START("Write / readback user registers");
80    random_word = $random();
81    tb_streamer.write_user_reg(sid_noc_block_gain, noc_block_gain.SR_GAIN, random_word[15:0]);
82    tb_streamer.read_user_reg(sid_noc_block_gain, 0, readback);
83    $sformat(s, "User register 0 incorrect readback! Expected: %0d, Actual %0d", readback[15:0], random_word[15:0]);
84    `ASSERT_ERROR(readback[15:0] == random_word[15:0], s);
85    random_word = $random();
86    tb_streamer.write_user_reg(sid_noc_block_gain, noc_block_gain.SR_TEST_REG_1, random_word);
87    tb_streamer.read_user_reg(sid_noc_block_gain, 1, readback);
88    $sformat(s, "User register 1 incorrect readback! Expected: %0d, Actual %0d", readback[31:0], random_word);
89    `ASSERT_ERROR(readback[31:0] == random_word, s);
90    `TEST_CASE_DONE(1);
```

Write Test Bench

Ettus Research™ A National Instruments Company

- Edit rfnoc-tutorial/rfnoc/testbenches/noc_block_gain_tb/noc_block_gain_tb.sv

```
`TEST_CASE_START("Test sequence");
gain = 100;
tb_streamer.write_user_reg(sid_noc_block_gain, noc_block_gain.SR_GAIN, gain);
  fork
    begin
      cvita_payload_t send_payload;
      for (int i = 0; i < SPP/2; i++) begin
        send_payload.push_back({16'(i),16'(i),16'(i),16'(i)});
      end
      tb_streamer.send(send_payload);
    end
```

```
tb_streamer.recv(recv_payload,md);
  for (int i = 0; i < SPP/2; i++) begin
    expected_value = {16'(i*gain),16'(i*gain),16'(i*gain),16'(i*gain)};
```

Write Test Bench

Ettus Research™
A National Instruments Company

```
92    /********************************************************
93    ** Test 5 -- Test sequence
94    ********************************************************/
95    // gain's user code is a loopback, so we should receive
96    // back exactly what we send
97    `TEST_CASE_START("Test sequence");
98    gain = 100;
99    tb_streamer.write_user_reg(sid_noc_block_gain, noc_block_gain.SR_GAIN, gain);
100   fork
101     begin
102       cvita_payload_t send_payload;
103       for (int i = 0; i < SPP/2; i++) begin
104         send_payload.push_back({16'(i),16'(i),16'(i),16'(i)});
105       end
106       tb_streamer.send(send_payload);
107     end
108     begin
109       cvita_payload_t recv_payload;
110       cvita_metadata_t md;
111       logic [63:0] expected_value;
112       tb_streamer.recv(recv_payload,md);
113       for (int i = 0; i < SPP/2; i++) begin
114         expected_value = {16'(i*gain),16'(i*gain),16'(i*gain),16'(i*gain)};
115         $sformat(s, "Incorrect value received! Expected: %0d, Received: %0d", expected_value, recv_payload[i]);
116         `ASSERT_ERROR(recv_payload[i] == expected_value, s);
117       end
118     end
119   join
120   `TEST_CASE_DONE(1);
121   `TEST_BENCH_DONE;
```

Check Test Bench Again

- source ~/rfnoc-workshop/setup_env.sh
- cd ~/rfnoc-workshop/src/rfnoc-tutorial/build
- make noc_block_gain_tb

```
============================================================
TESTBENCH STARTED: noc_block_gain
============================================================
[TEST CASE   1] (t=000000000) BEGIN: Wait for Reset...
[TEST CASE   1] (t=000001002) DONE... Passed
[TEST CASE   2] (t=000001002) BEGIN: Check NoC ID...
Read gain NOC ID: e7757e5700000001
[TEST CASE   2] (t=000001238) DONE... Passed
[TEST CASE   3] (t=000001238) BEGIN: Connect RFNoC blocks...
Connecting noc_block_tb (SID: 1:0) to noc_block_gain (SID: 0:0)
Connecting noc_block_gain (SID: 0:0) to noc_block_tb (SID: 1:0)
[TEST CASE   3] (t=000005277) DONE... Passed
[TEST CASE   4] (t=000005277) BEGIN: Write / readback user registers...
[TEST CASE   4] (t=000006623) DONE... Passed
[TEST CASE   5] (t=000006623) BEGIN: Test sequence...
[TEST CASE   5] (t=000007363) DONE... Passed
============================================================
TESTBENCH FINISHED: noc_block_gain
 - Time elapsed:   7400 ns
 - Tests Expected: 5
 - Tests Run:      5
 - Tests Passed:   5
Result: PASSED
============================================================
```

Break?

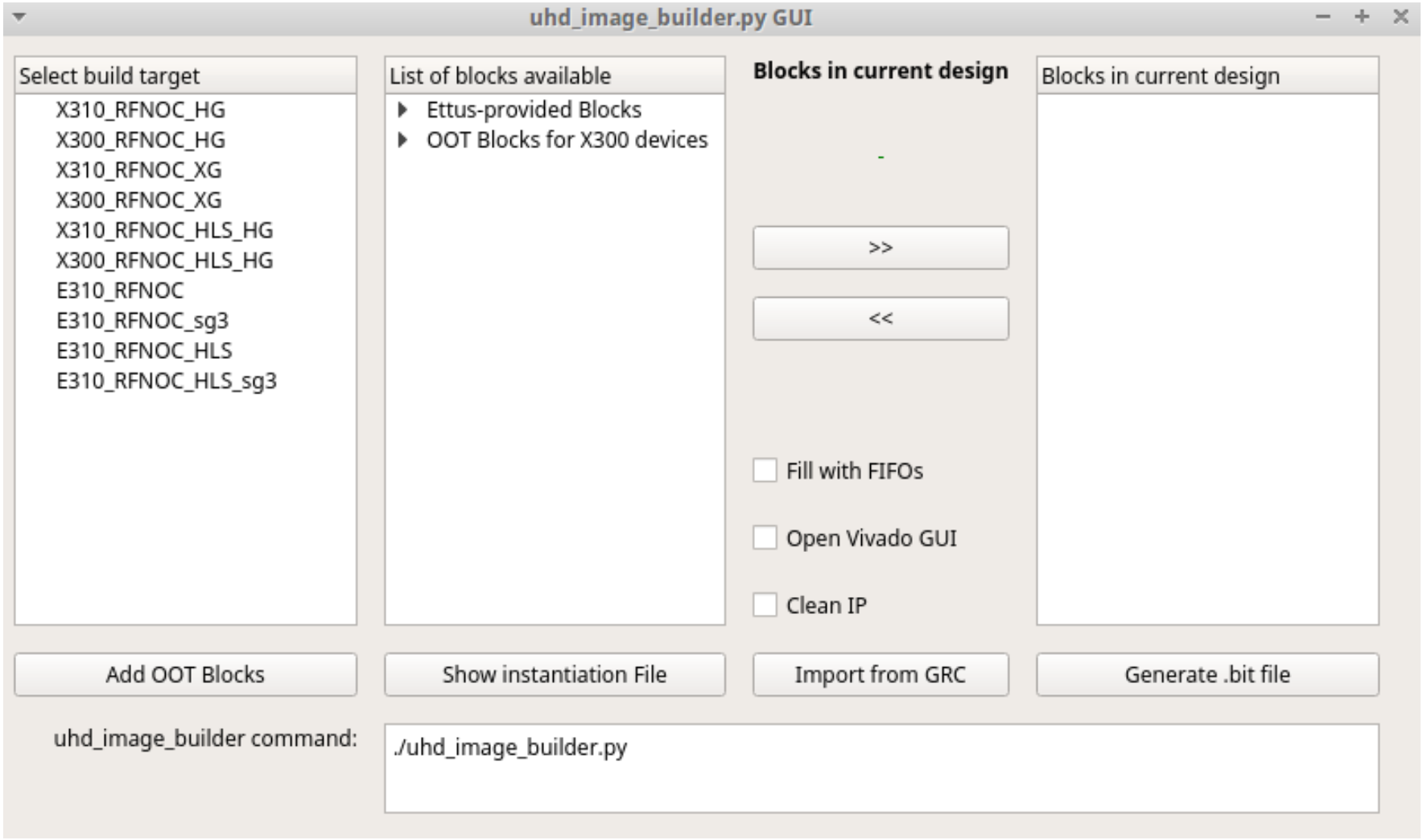
Generate Bitstream

Ettus Research™ A National Instruments Company

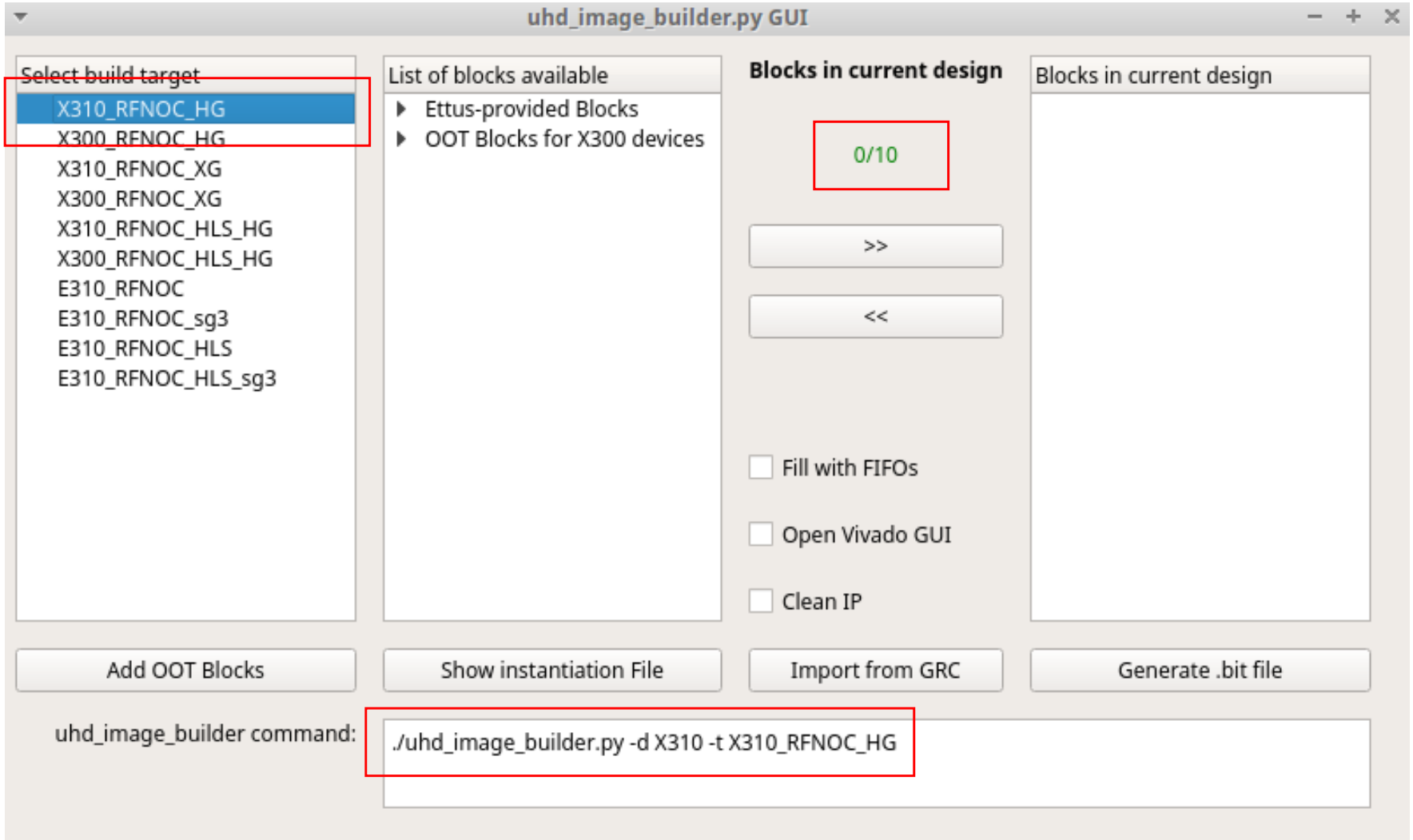
- uhd_image_builder
  - Command line tool for building bitstreams
  - Allows specifying RFNoC blocks to include
  - Note: make sure to include DDC and DUC!
  - For full details on usage, see getting started
- GUI version: uhd_image_builder_gui.py
  - **source ~/rfnoc-workshop/setup_env.sh**
  - **cd ~/rfnoc-workshop/src/uhd-fpga/usrp3/tools/scripts**
  - **./uhd_image_builder_gui.py**

uhd_image_builder_gui.py

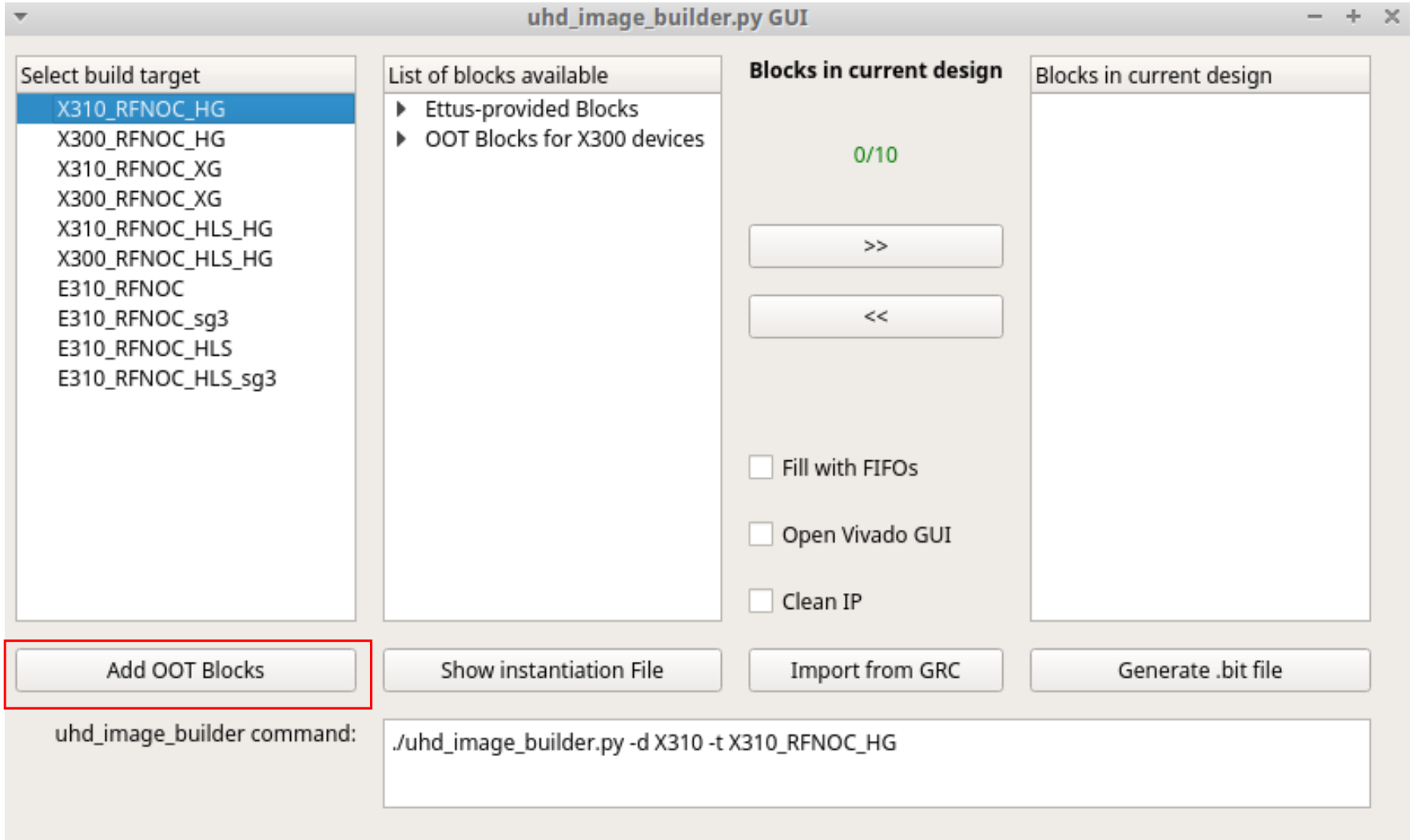
Ettus Research™
A National Instruments Company



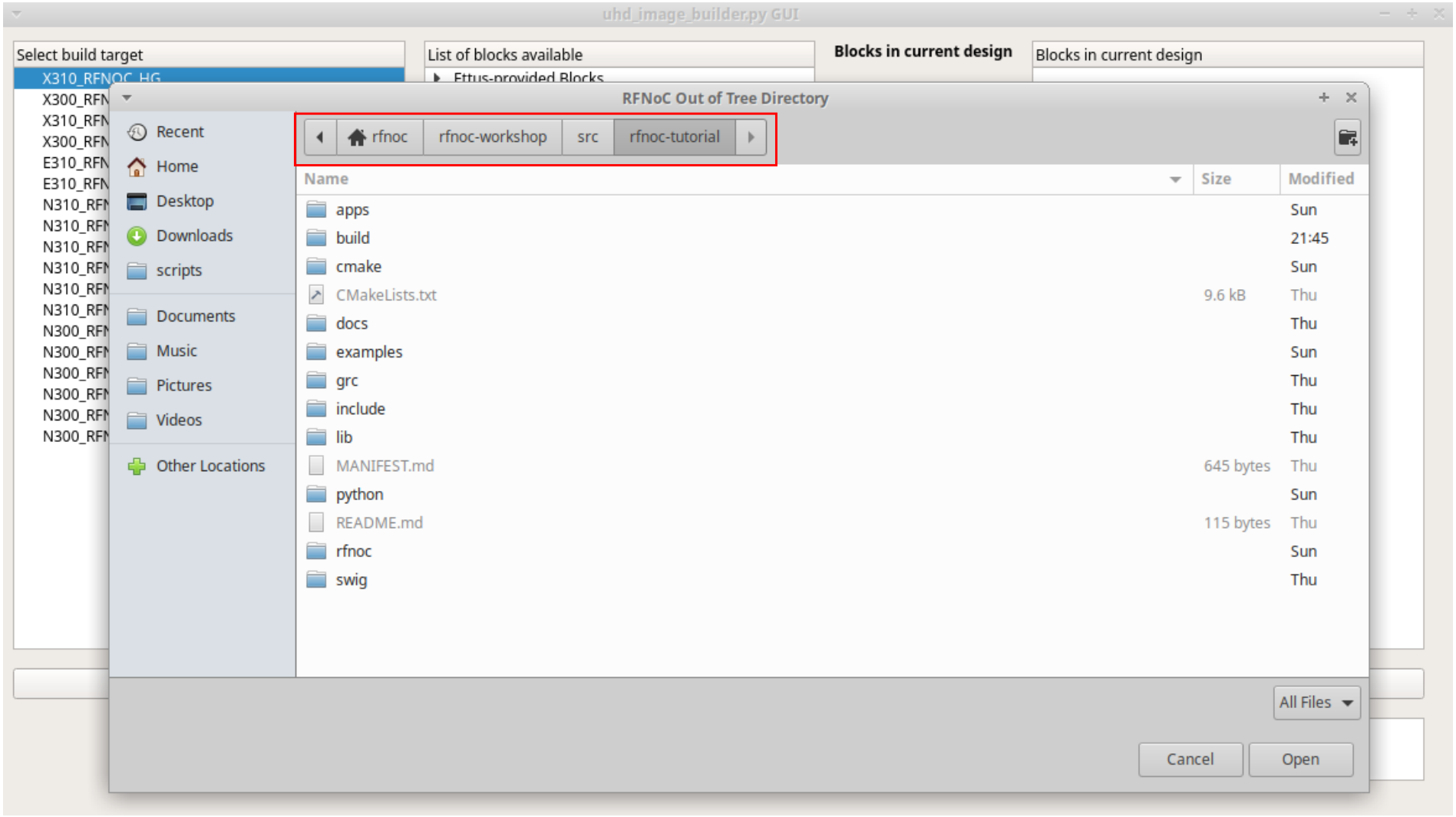
uhd_image_builder_gui.py



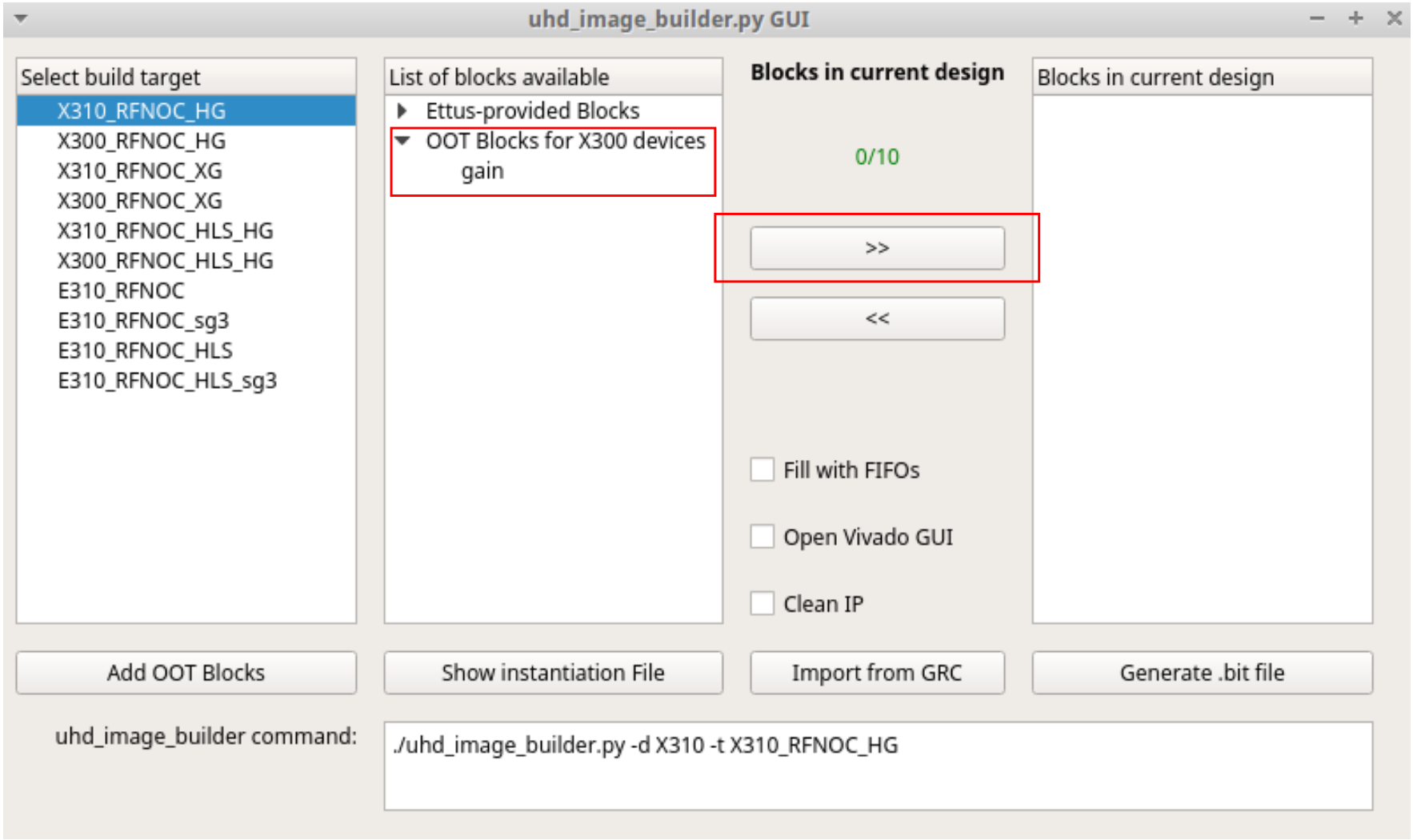
uhd_image_builder_gui.py



uhd_image_builder_gui.py

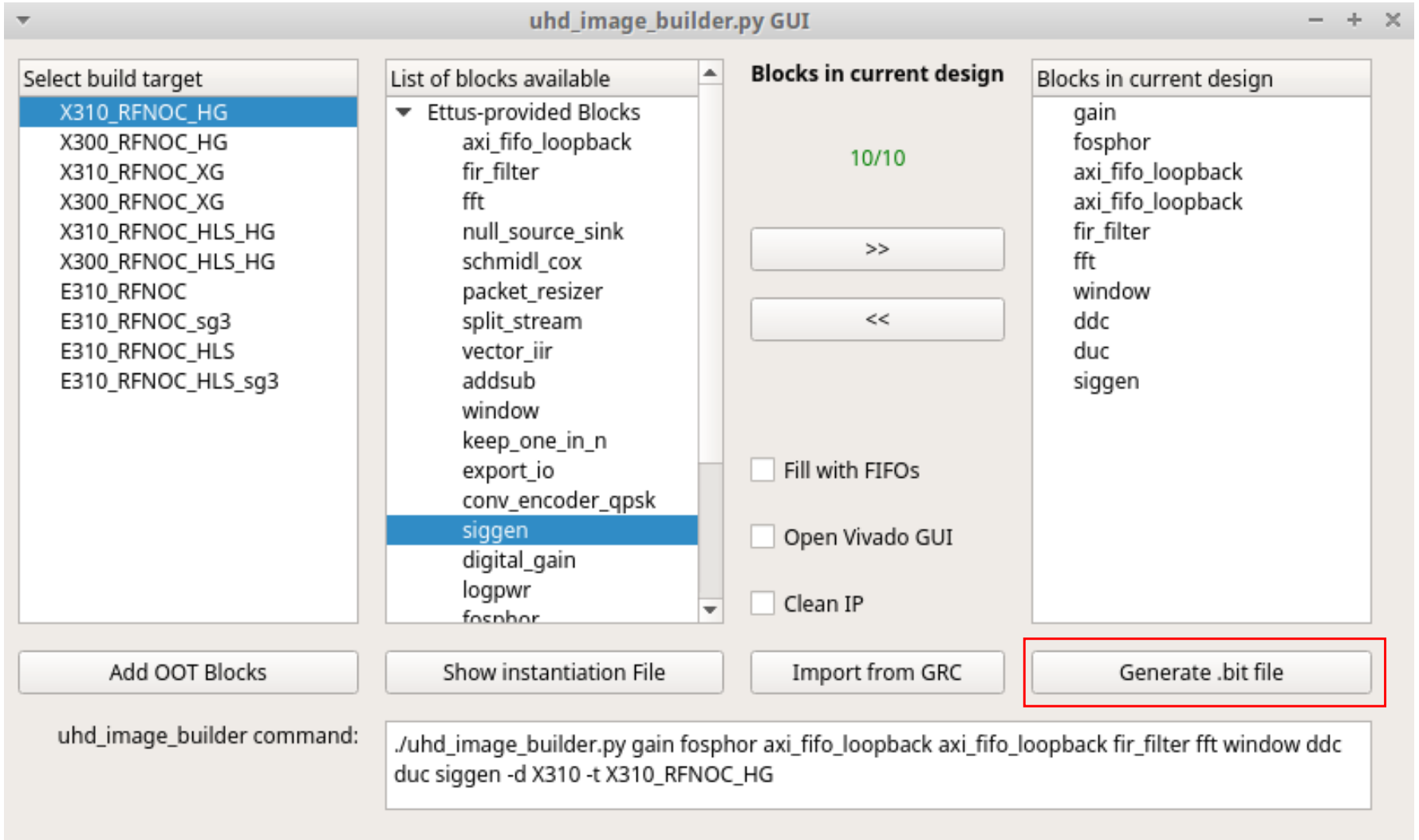


uhd_image_builder_gui.py



uhd_image_builder_gui.py

Ettus Research™ A National Instruments Company



Steps to using GUI

Ettus Research™
A National Instruments Company

- Select device (X310_RFNOC_HG, etc…)
- '**Add OOT Blocks**' to import RFNoC OOT Blocks
- Select RFNoC blocks and add them to image with '**>>**' button
- Use '**Generate .bit file**' to call Vivado to build bitstream
- Bitstream will be in **uhd-fpga/usrp3/top/<device>/build**
- Our premade bitstream is located at **rfnoc-workshop/src/uhd_x310_fpga_RFNOC_HG_1.bit**

Fix Timing Error

- Vivado occasionally fails to make timing with our simple design
- Multipliers in FPGAs can have long critical paths
- Add a register stage to break up critical path
- Our premade bitstream has this register

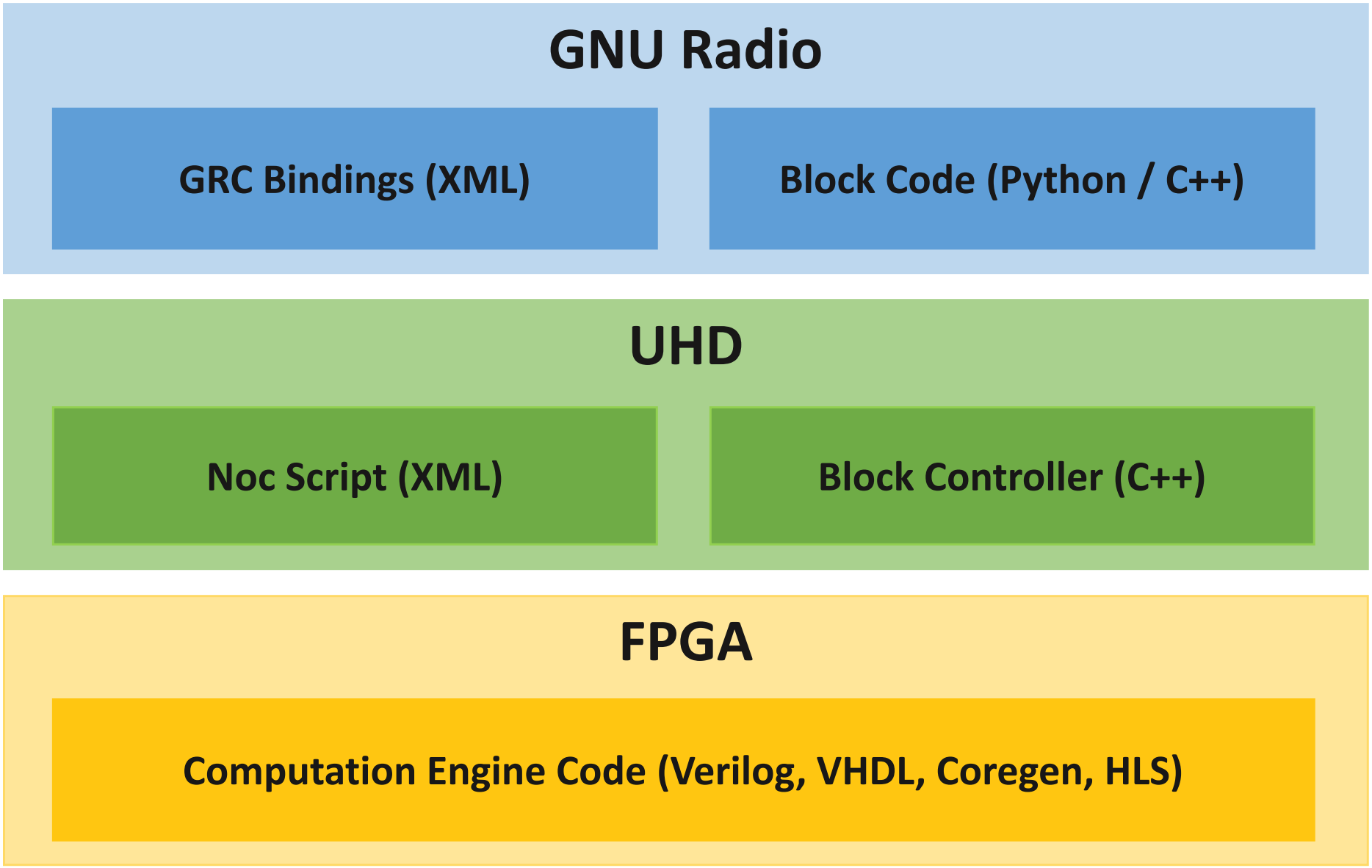
Fix Timing Error

Ettus Research™
A National Instruments Company

```
194  // Apply gain
195  wire [15:0] i, q;
196  wire [31:0] i_mult_gain, q_mult_gain;
197
198  assign i = m_axis_data_tdata[31:16];
199  assign q = m_axis_data_tdata[15:0];
200
201  assign i_mult_gain = i*gain;
202  assign q_mult_gain = q*gain;
203
204  generate
205    if (USE_REG) begin
206      axi_fifo_flop #(.WIDTH(1+16+16)) axi_fifo_flop_gain (
207        .clk(ce_clk),.reset(ce_rst), .clear(clear_tx_seqnum),
208        .i_tdata({m_axis_data_tlast, i_mult_gain[15:0], q_mult_gain[15:0]}),
209        .i_tvalid(m_axis_data_tvalid),
210        .i_tready(m_axis_data_tready),
211        .o_tdata({s_axis_data_tlast, s_axis_data_tdata}),
212        .o_tvalid(s_axis_data_tvalid),
213        .o_tready(s_axis_data_tready),
214        .space(), .occupied());
215    end else begin
216      // Fails timing occasionally
217      assign s_axis_data_tdata  = {i_mult_gain[15:0], q_mult_gain[15:0]};
218      assign m_axis_data_tready = s_axis_data_tready;
219      assign s_axis_data_tvalid = m_axis_data_tvalid;
220      assign s_axis_data_tlast  = m_axis_data_tlast;
221    end
222  endgenerate
223
```

RFNoC Framework

Ettus Research™
A National Instruments Company



RFNoC Framework

Ettus Research™
A National Instruments Company

GNU Radio

GRC Bindings (XML)

Block Code (Python / C++)

UHD

Noc Script (XML)

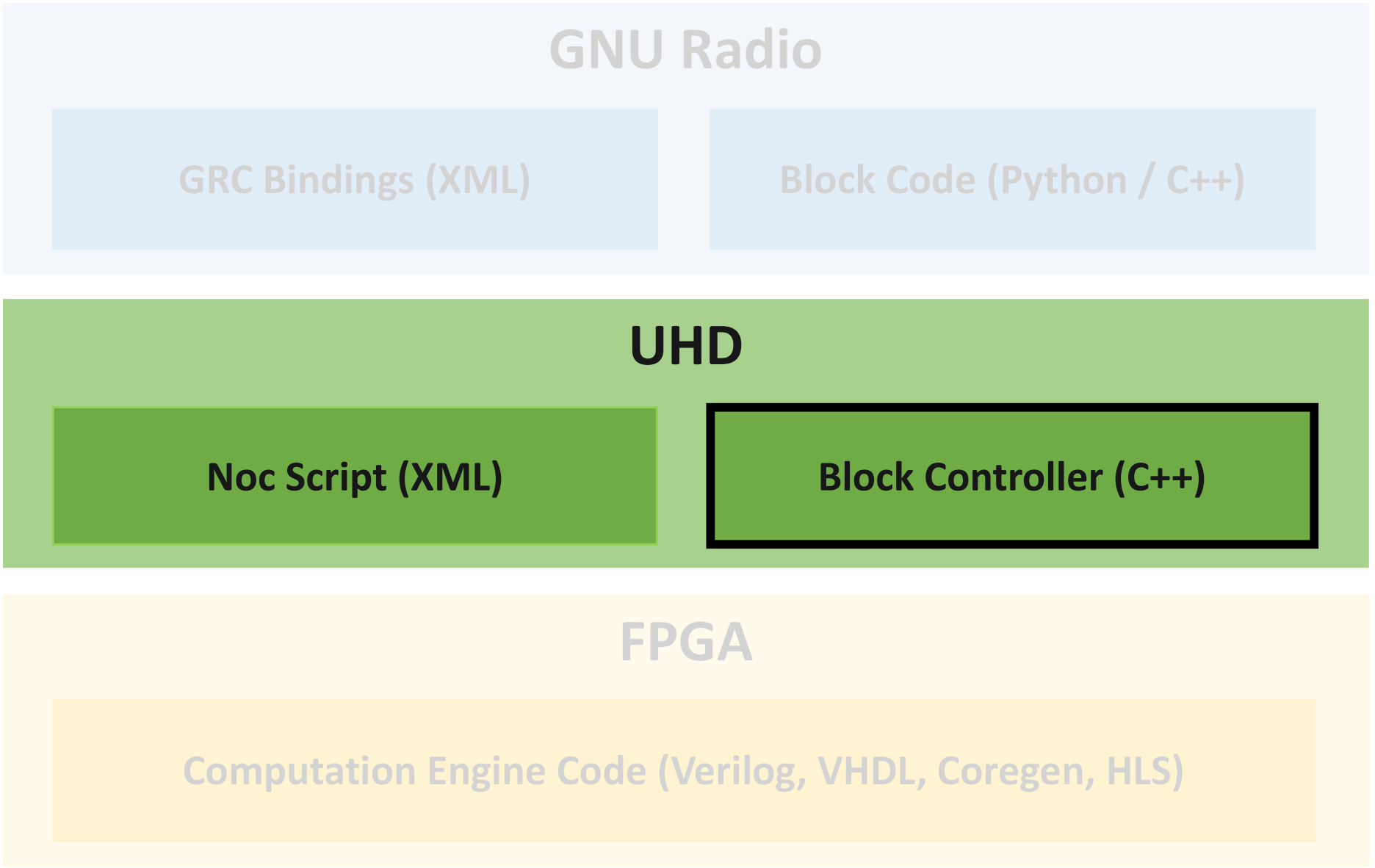
Block Controller (C++)

FPGA

Computation Engine Code (Verilog, VHDL, Coregen, HLS)

RFNoC Framework

Ettus Research™
A National Instruments Company



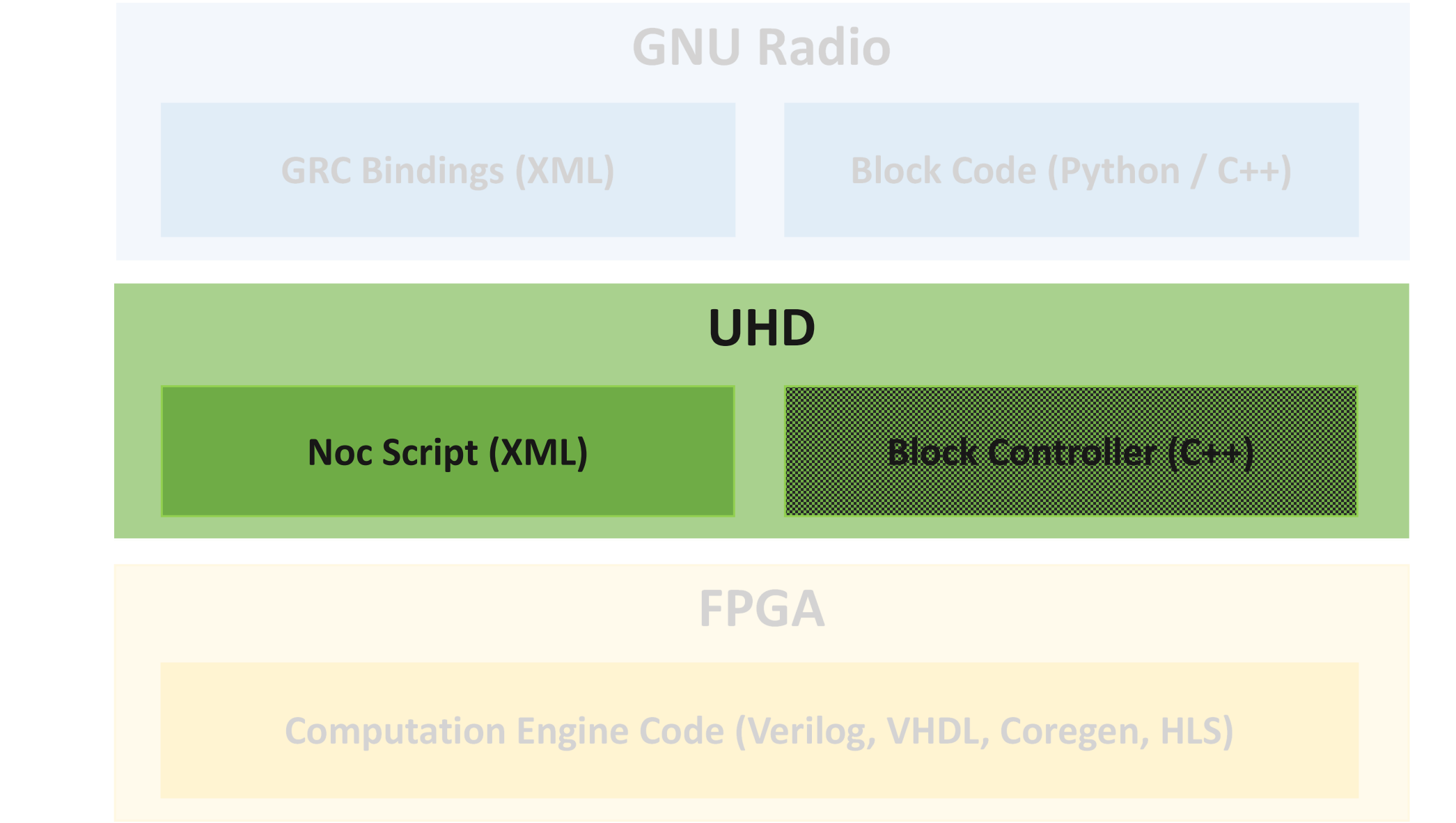
Block Controller Class

Ettus Research™ A National Instruments Company

- C++ Code
- Tells UHD about block's capabilities, configuration
- Exposes methods to access and control block
  - Example: set_taps() for FIR filter RFNoC block
- Default block controller class covers most cases
- Custom implementation often not required!
  - Read / write registers
  - Independently perform operation(s) on a data stream
  - Examples: FIFO, FFT, Signal Generator, Fosphor
- rfnoc-tutorial/lib/gain_block_ctrl_impl.cpp
  - Our simple block does not need to do anything!

RFNoC Framework

Ettus Research™



RFNoC Framework

GNU Radio

GRC Bindings (XML)

Block Code (Python / C++)

UHD

Noc Script (XML)

Block Controller (C++)

FPGA

Computation Engine Code (Verilog, VHDL, Coregen, HLS)

Noc Script

Ettus Research™
A National Instruments Company

- XML based DSL for block configuration
- No recompilation!
- Describe NOC ID, user registers, input & output ports
- Create args to customize writing to registers
  - Statically typed, quasi-functional
  - Basic types: Integers, Strings, Doubles
  - Basic arithmetic and logic operations available
  - Validate input
- Skeleton file generated by RFNoC modtool!

Noc Script

Ettus Research™ A National Instruments Company

- Add Gain settings register and arg to our block
- Open **rfnoc-tutorial/rfnoc/blocks/gain.xml**:

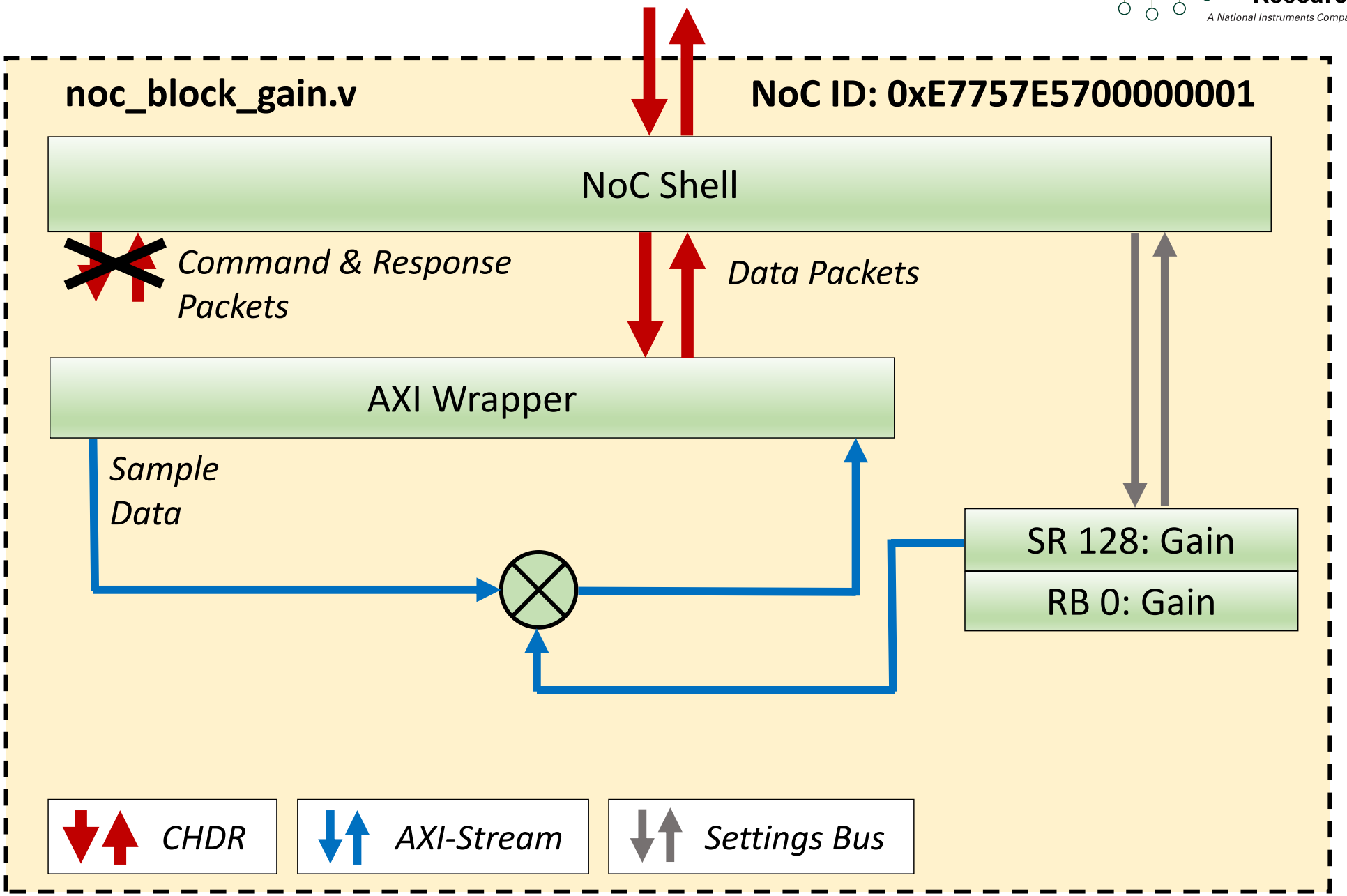
```
<registers>
  <setreg>
    <name>GAIN</name>
    <address>128</address>
  </setreg>
</registers>
<args>
  <arg>
    <name>gain</name>
    <type>double</type>
    <value>1.0</value>
    <check>GE($gain, 0.0) OR LE($gain, 32767.0)</check>
    <check_message>Gain must be in range [0, 32767].</check_message>
    <action>SR_WRITE("GAIN", IROUND($gain))</action>
  </arg>
</args>
```

Gain RFNoC Block Noc Script

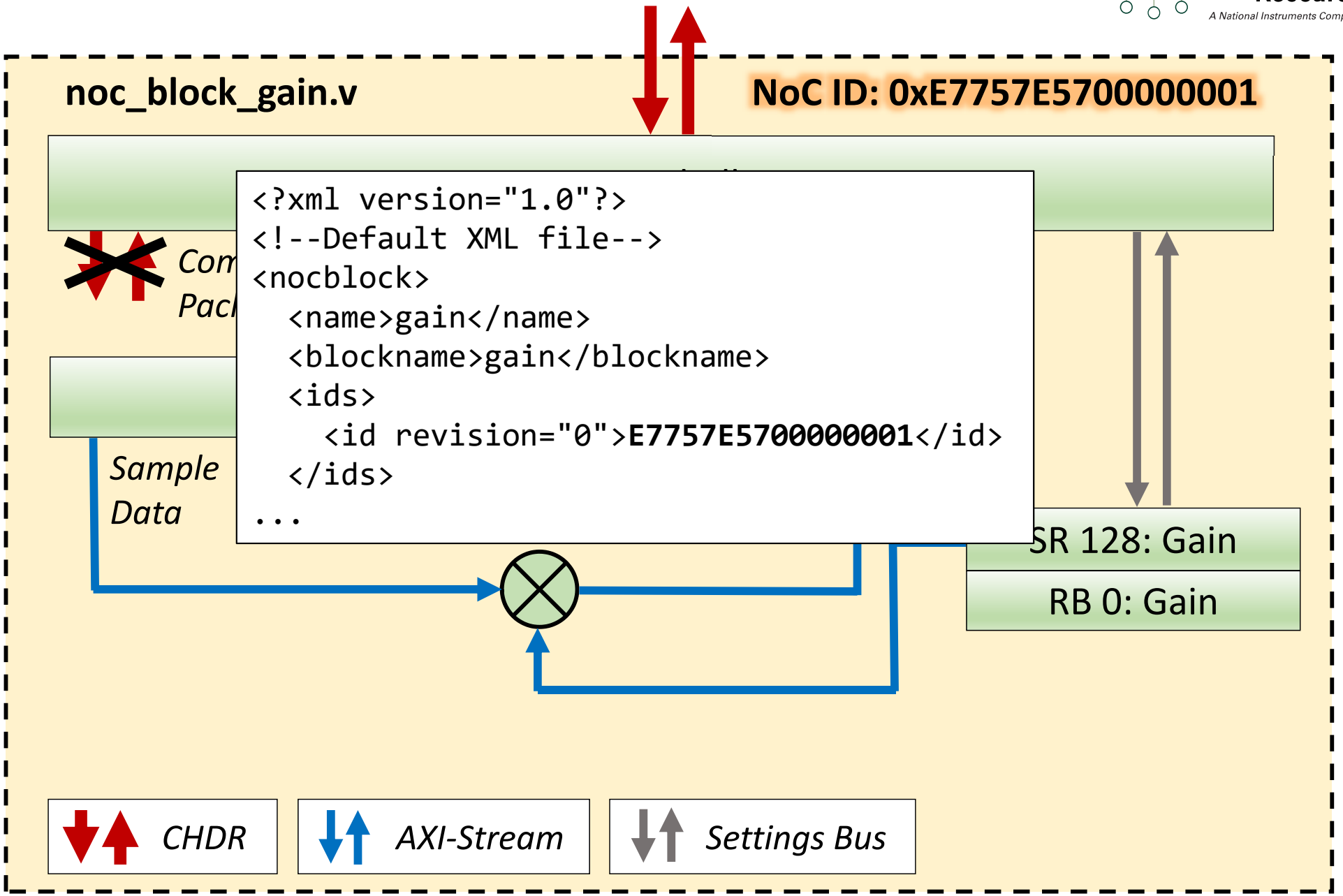
Ettus Research™
A National Instruments Company

```xml
<?xml version="1.0"?>
<!--Default XML file-->
<nocblock>
  <name>gain</name>
  <blockname>gain</blockname>
  <ids>
    <id revision="0">E7757E5700000001</id>
  </ids>

  <registers>
    <setreg>
      <name>GAIN</name>
      <address>128</address>
    </setreg>
  </registers>
  <args>
    <arg>
      <name>gain</name>
      <type>double</type>
      <value>1.0</value>
      <check>GE($gain, 0.0) AND LE($gain, 32767.0)</check>
      <check_message>Gain must be in the range [0, 32767]</check_message>
      <action>SR_WRITE("GAIN", IROUND($gain))</action>
    </arg>
  </args>

  <!--One input, one output. If this is used, better have all the info the C++ file.-->
  <ports>
    <sink>
      <name>in</name>
    </sink>
    <source>
      <name>out</name>
    </source>
  </ports>
</nocblock>
```

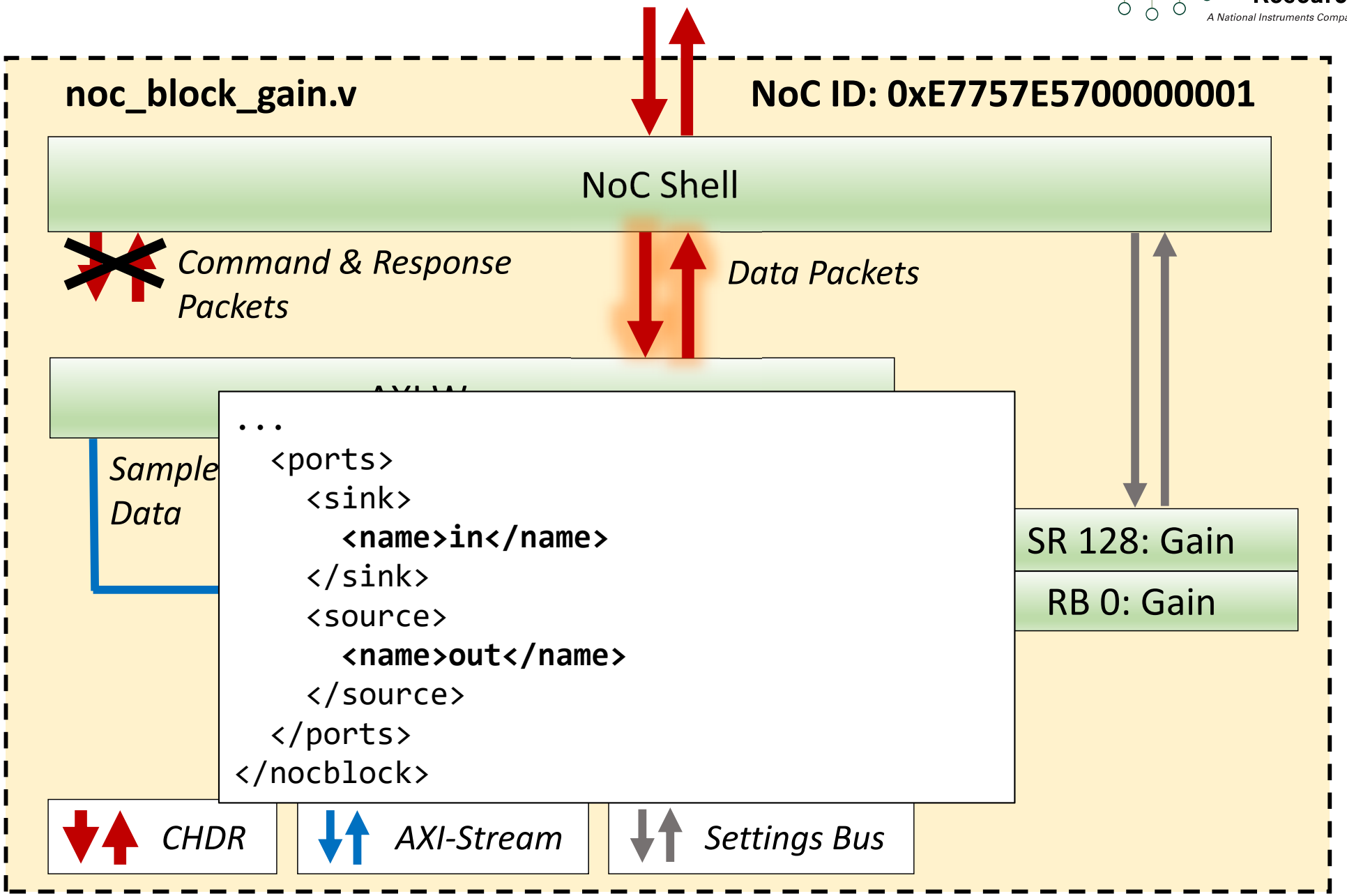
Noc Script


noc_block_gain.v
NoC ID: 0xE7757E5700000001
NoC Shell
Command & Response Packets
Data Packets
AXI Wrapper
Sample Data
SR 128: Gain
RB 0: Gain
CHDR
AXI-Stream
Settings Bus

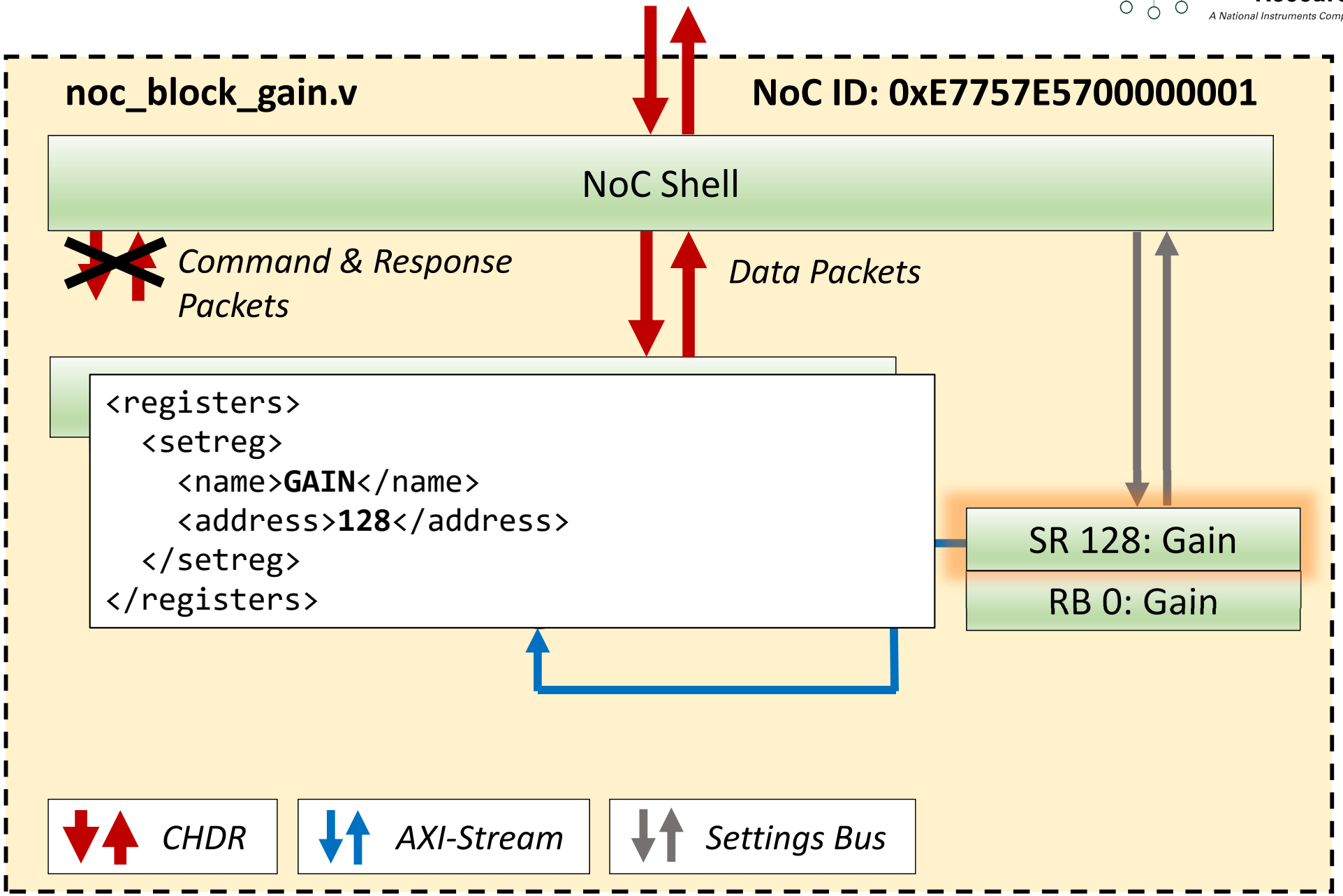
Noc Script


noc_block_gain.v
NoC ID: 0xE7757E5700000001
<?xml version="1.0"?>
<!--Default XML file-->
<nocblock>
<name>gain</name>
<blockname>gain</blockname>
<ids>
<id revision="0">E7757E5700000001</id>
</ids>
...
Sample Data
SR 128: Gain
RB 0: Gain
CHDR
AXI-Stream
Settings Bus

Noc Script

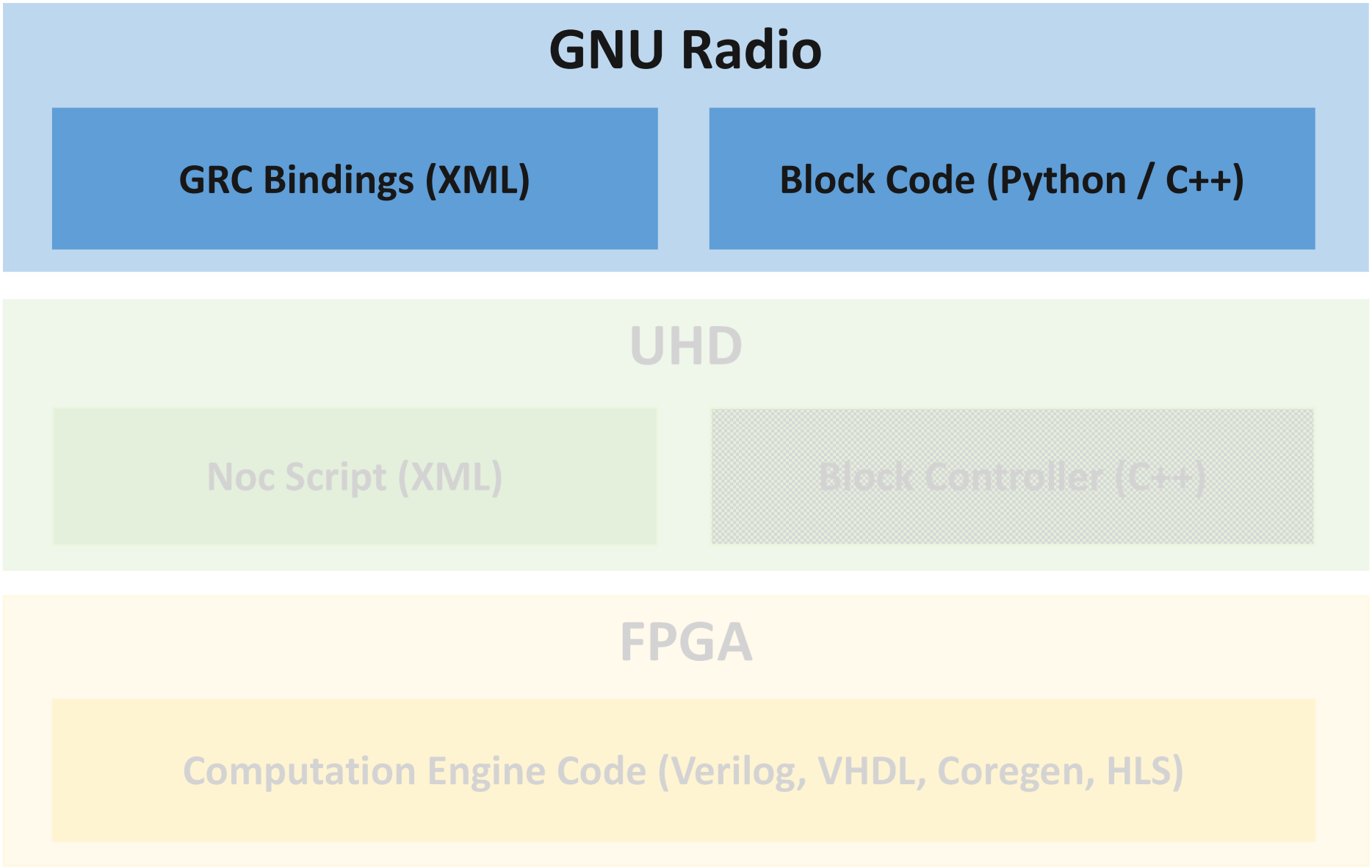

noc_block_gain.v
NoC ID: 0xE7757E5700000001
NoC Shell
Command & Response Packets
Data Packets
Sample Data
...
<ports>
<sink>
<name>in</name>
</sink>
<source>
<name>out</name>
</source>
</ports>
</nocblock>
SR 128: Gain
RB 0: Gain
CHDR
AXI-Stream
Settings Bus

Noc Script


noc_block_gain.v
NoC ID: 0xE7757E5700000001
NoC Shell
Command & Response Packets
Data Packets
<registers>
<setreg>
<name>GAIN</name>
<address>128</address>
</setreg>
</registers>
SR 128: Gain
RB 0: Gain
CHDR
AXI-Stream
Settings Bus

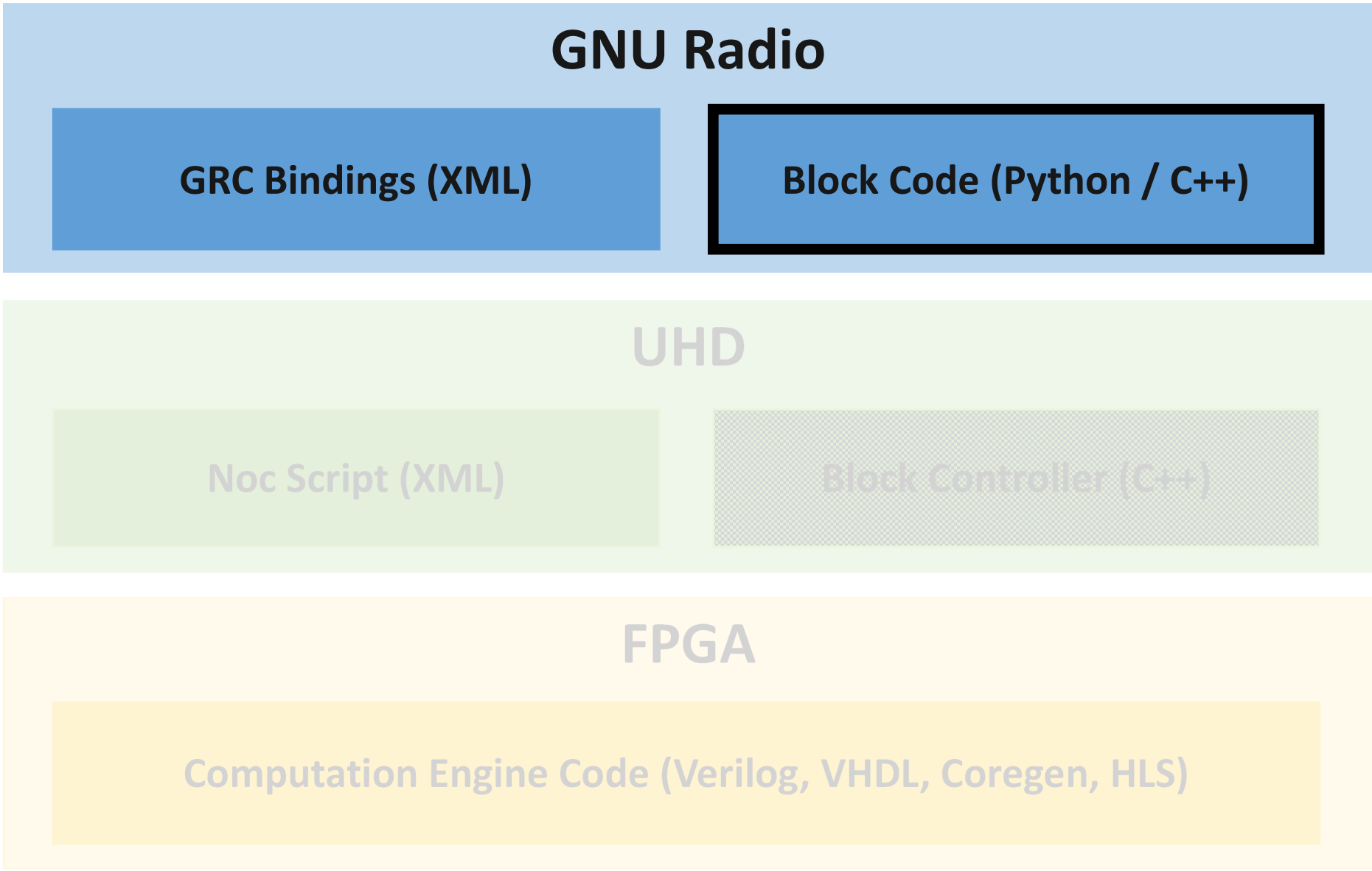
RFNoC Framework

Ettus Research™
A National Instruments Company



RFNoC Framework

Ettus Research™
A National Instruments Company



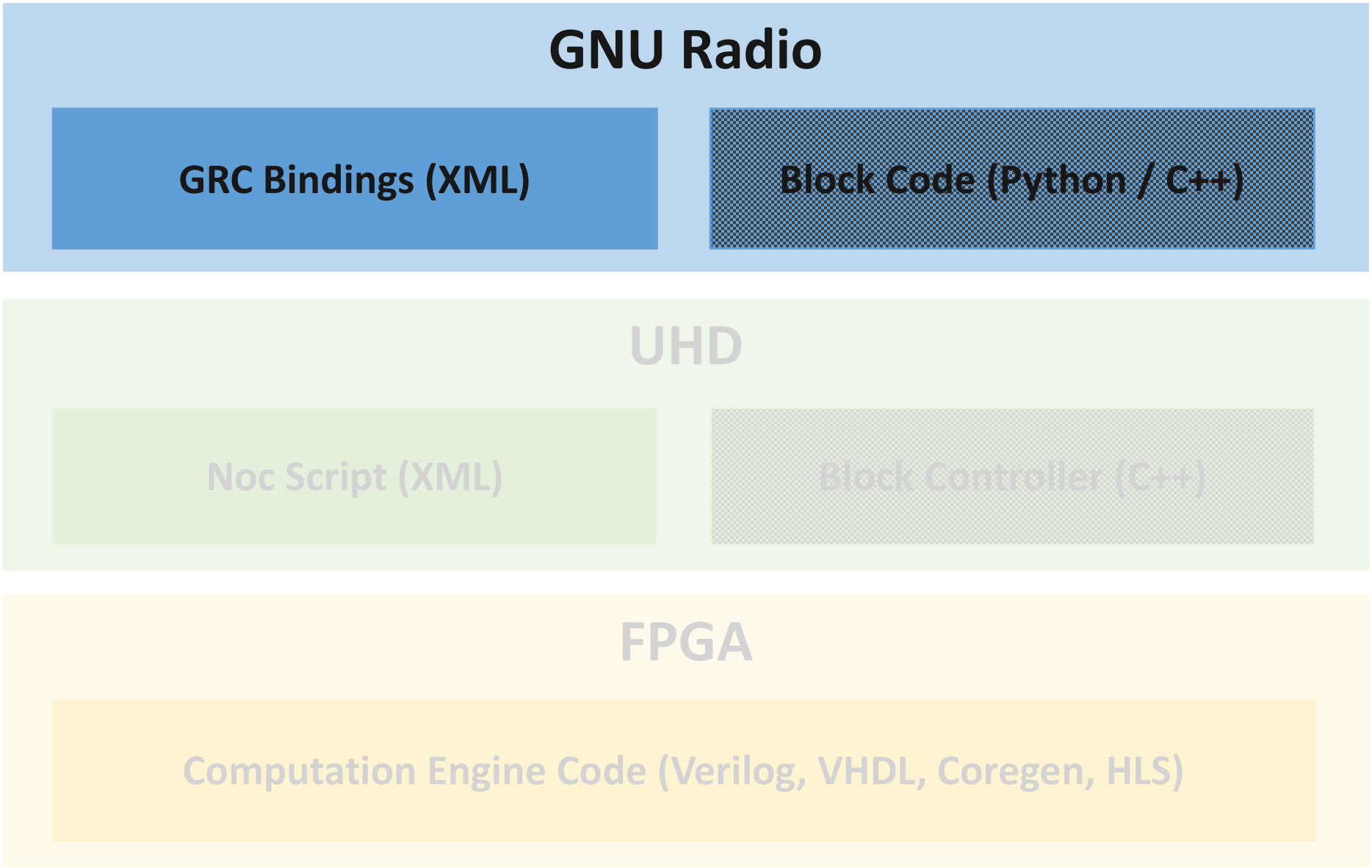
GNU Radio Block Code

Ettus Research™
A National Instruments Company

- C++ or Python
- How does GNU Radio interface to RFNoC?
  - via C++ infrastructure code in gr-ettus
  - Gr-ettus provides a base RFNoC block class
  - Users extend base class for their RFNoC blocks
  - Many blocks can use base class “as is”
  - No C++ or Python code!
- rfnoc-tutorial/lib/gain_impl.cc
  - Our simple block does not need to do anything!

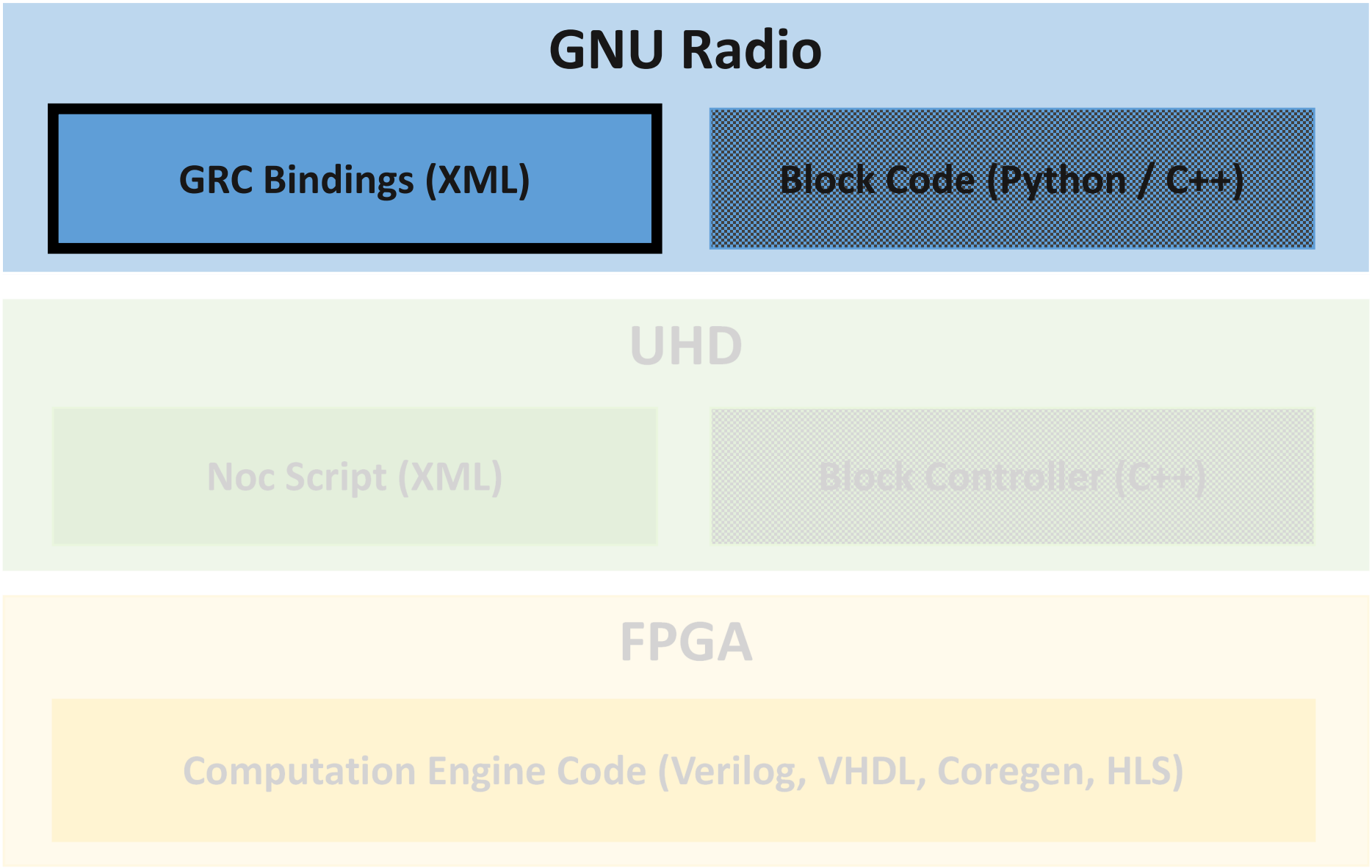
RFNoC Framework

Ettus Research™
A National Instruments Company



RFNoC Framework

Ettus Research™
A National Instruments Company



GNU Radio Companion Bindings

Ettus Research™ A National Instruments Company

- XML
- Describes GNU Radio blocks to GRC
- No recompilation!
- Requirement of GNU Radio Companion
- Not strictly necessary for GNU Radio
- Tutorial on how to write them:
  gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion
- Skeleton file generated by RFNoC modtool!

GRC XML

Ettus Research™
A National Instruments Company

- Add ‘Gain’ parameter
- Add callback to set gain arg
- Edit **rfnoc-tutorial/grc/tutorial_gain.xml**:

```
self.$(id).set_arg("gain", $gain)
</make>
<callback>set_arg("gain", $gain)</callback>

<param>
  <name>Gain</name>
  <key>gain</key>
  <value>1.0</value>
  <type>real</type>
</param>
```

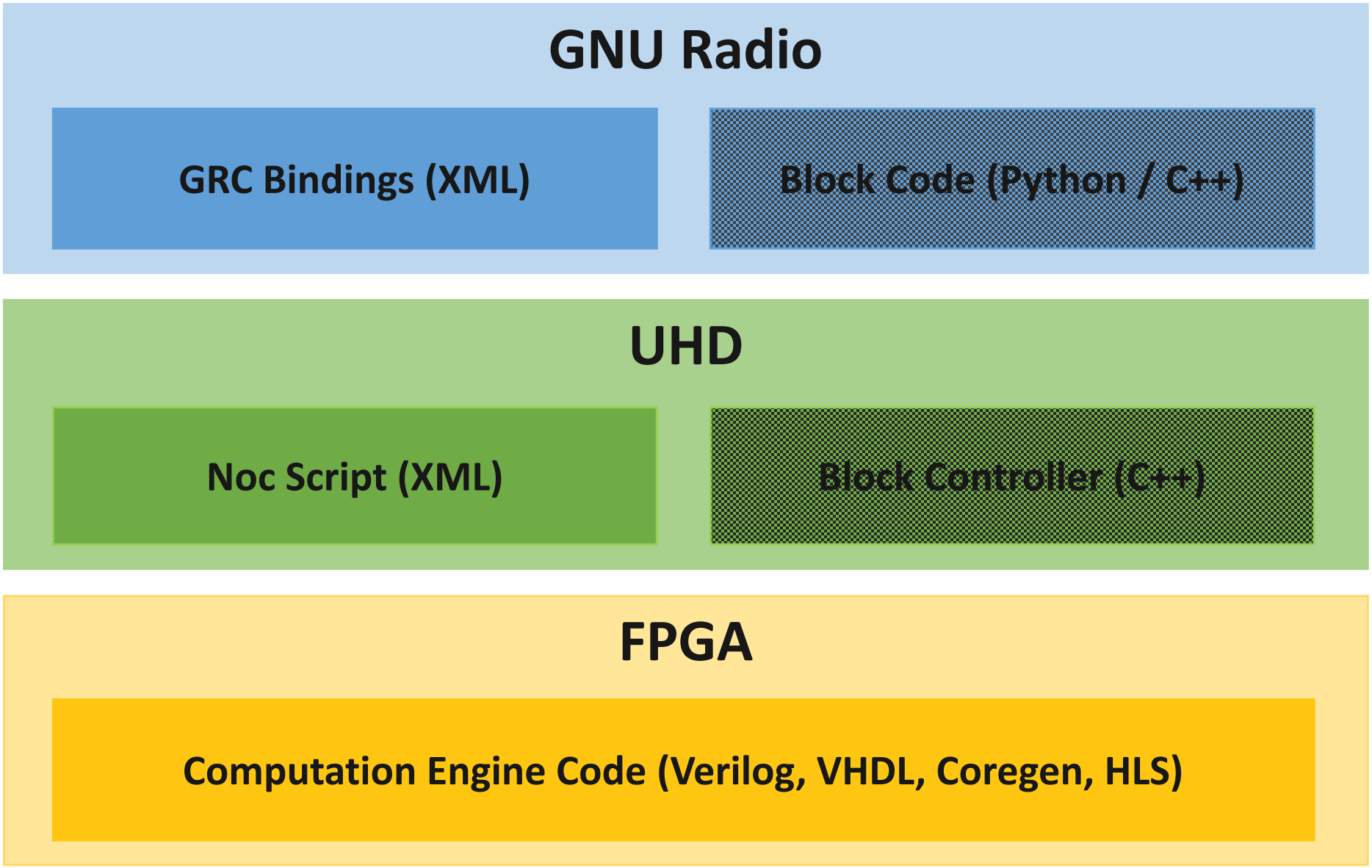
GRC XML

Ettus Research™
A National Instruments Company

```
<?xml version="1.0"?>
<block>
  <name>RFNoC: gain</name>
  <key>tutorial_gain</key>
  <category>tutorial</category>
  <import>import tutorial</import>
  <make>tutorial.gain(
          self.device3,
          uhd.stream_args( # TX Stream Args
                cpu_format="$type",
                otw_format="$otw",
                args="gr_vlen={0},{1}".format(${grvlen}, "" if $grvlen == 1 else "spp={0}".format($grvlen)),
          ),
          uhd.stream_args( # RX Stream Args
                cpu_format="$type",
                otw_format="$otw",
                args="gr_vlen={0},{1}".format(${grvlen}, "" if $grvlen == 1 else "spp={0}".format($grvlen)),
          ),
          $block_index,
          $device_index
    )
self.$(id).set_arg("gain", $gain)
</make>
<callback>set_arg("gain", $gain)</callback>

  <param>
    <name>Gain</name>
    <key>gain</key>
    <value>1.0</value>
    <type>real</type>
  </param>

```

RFNoC Framework

Ettus Research™
A National Instruments Company



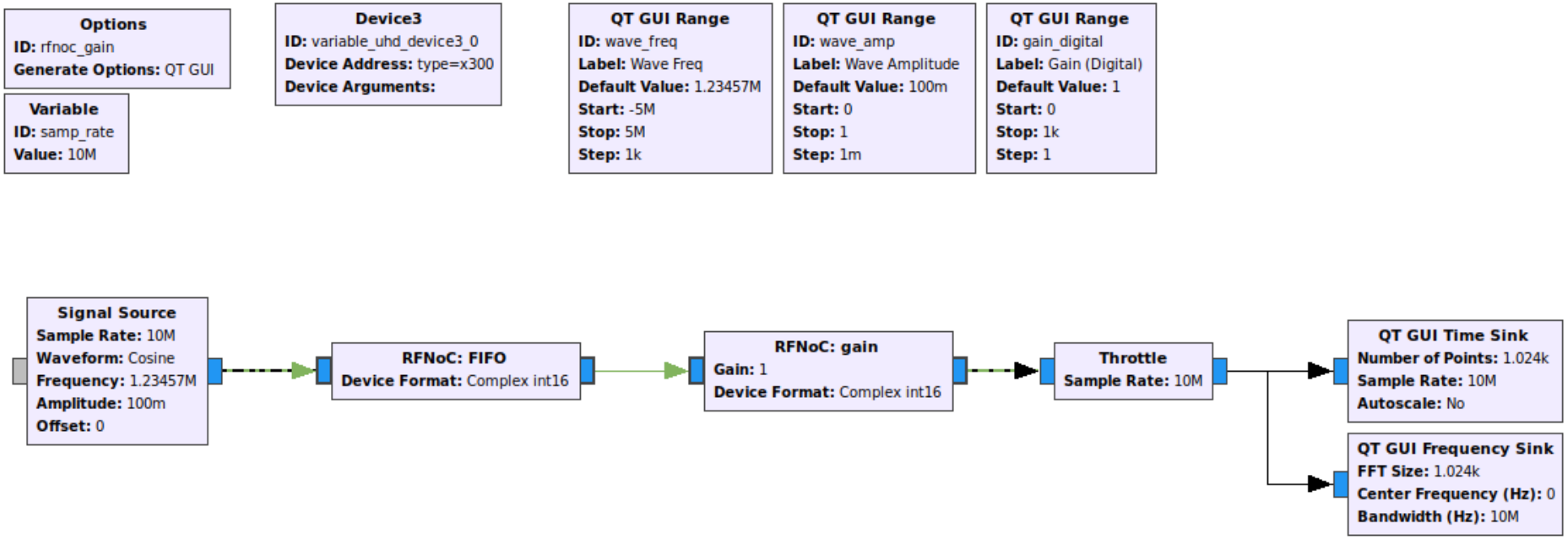
Installation

Ettus Research™ A National Instruments Company

- Install:
  - **source ~/rfnoc-workshop/setup_env.sh**
  - **cd ~/rfnoc-workshop/src/rfnoc-tutorial/build**
  - **make install**
- Program bitstream:
  - Turn on X310
  - **source ~/rfnoc-workshop/src/uhd-fpga/usrp3/top/x300/setupenv.sh**
  - **viv_jtag_program ~/rfnoc-workshop/src/usrp_x310_fpga_RFNOC_HG_2.bit**

Testing in HW

- Open rfnoc_gain flowgraph
  - **source ~/rfnoc-workshop/setup_env.sh**
  - **gnuradio-companion**
  - Open ~/rfnoc-workshop/src/rfnoc-tutorial-complete/examples/rfnoc_gain.grc



Advanced Topics

- UHD RFNoC C++ API application
- How to use command packets
  - Tune DSP offset in DDC / DUC

UHD RFNoC C++ App

Ettus Research™
A National Instruments Company

- RFNoC apps can use only UHD's C++ API
- Design flow:
  - Create device3 usrp object
  - Grab block controller objects
  - Setup graph
  - Setup blocks either with block controller methods, block args, and/or writing registers
    - Creating a block controller method is more complex, but allows using C++ syntax
    - Block arg is an <arg> defined in Noc Script XML
    - Directly writing to a register is the simplest
  - Setup streamers
  - Run app with send() / recv()
- There is some overlap with existing API

Gain RFNoC C++ App

Ettus Research™
A National Instruments Company

- RFNoC C++ app for the gain block:
  - Transmits sine wave through FIFO and Gain block
  - Receives amplified sine wave and sends samples to GNU Radio for plotting via UDP
  - User can change gain via console
- **cp ~/rfnoc-workshop/src/gain-app-example/* ~/rfnoc-workshop/src/rfnoc-tutorial/apps/**
- Edit ~/rfnoc-workshop/src/rfnoc-tutorial/apps/rfnoc_gain_example.cpp
- Most of the code is already filled out, we will fill in some of the important RFNoC related parts

Gain RFNoC C++ App

Ettus Research™ A National Instruments Company

- Add code to:
  - Setup device3 usrp object with device args
  - Grab block controller objects for fifo and gain blocks
    - Name string comes from Gain block's Noc Script XML, see tag <blockname>
  - Setup graph and connect blocks
- C++11 auto keyword makes this easier

```
auto usrp = uhd::device3::make(std::string(usrp_args));

auto fifo_block_ctrl = usrp->get_block_ctrl(
    uhd::rfnoc::block_id_t(0, "FIFO"));
auto gain_block_ctrl = usrp->get_block_ctrl(
    uhd::rfnoc::block_id_t(0, "gain"));

auto graph = usrp->create_graph("gain_graph");
graph->connect(fifo_block_ctrl->get_block_id(),
    gain_block_ctrl->get_block_id());
```

Gain RFNoC C++ App

Ettus Research™ A National Instruments Company

- Add code to setup RX and TX streamers
- Streamers have built in args such as “block_id” and “spp” that have special meaning
- Streamer connections are defined with “block_id”

```
uhd::stream_args_t rx_stream_args("fc32", "sc16");
rx_stream_args.args["block_id"] = gain_block_ctrl->get_block_id();
rx_stream_args.args["spp"] = boost::lexical_cast<std::string>(spp);
auto rx_stream = usrp->get_rx_stream(rx_stream_args);

uhd::stream_args_t tx_stream_args("fc32", "sc16");
tx_stream_args.args["block_id"] = fifo_block_ctrl->get_block_id();
tx_stream_args.args["spp"] = boost::lexical_cast<std::string>(spp);
auto tx_stream = usrp->get_tx_stream(tx_stream_args);
```

Gain RFNoC C++ App

Ettus Research™
A National Instruments Company

```
158     po::variables_map vm;
159     po::store(po::parse_command_line(argc, argv, desc), vm);
160     po::notify(vm);
161
162     auto usrp = uhd::device3::make(std::string(usrp_args));
163
164     // Grab block controllers, block id name comes from block descript XML tag <blockname>
165     auto fifo_block_ctrl = usrp->get_block_ctrl(uhd::rfnoc::block_id_t(0, "FIFO"));
166     auto gain_block_ctrl = usrp->get_block_ctrl(uhd::rfnoc::block_id_t(0, "gain"));
167
168     // Create and connect
169     auto graph = usrp->create_graph("gain_graph");
170     graph->connect(fifo_block_ctrl->get_block_id(), gain_block_ctrl->get_block_id());
171
172     // Setup receive streamer
173     uhd::stream_args_t rx_stream_args("fc32", "sc16");
174     rx_stream_args.args["block_id"] = gain_block_ctrl->get_block_id();
175     rx_stream_args.args["spp"] = boost::lexical_cast<std::string>(spp);
176     auto rx_stream = usrp->get_rx_stream(rx_stream_args);
177
178     // Setup transmit streamer
179     uhd::stream_args_t tx_stream_args("fc32", "sc16");
180     tx_stream_args.args["block_id"] = fifo_block_ctrl->get_block_id();
181     tx_stream_args.args["spp"] = boost::lexical_cast<std::string>(spp);
182     auto tx_stream = usrp->get_tx_stream(tx_stream_args);
183
184     auto tx_thread_inst = std::thread(&tx_thread, tx_stream, wave_table_length, wave_ampl, spp);
185     auto rx_thread_inst = std::thread(&rx_thread, rx_stream, udp_addr, udp_port, spp);
186     auto set_gain_thread_inst = std::thread(&set_gain_thread, gain_block_ctrl);
187     tx_thread_inst.join();
188     rx_thread_inst.join();
189     set_gain_thread_inst.detach();
190
191     std::cout << std::endl << "Done!" << std::endl << std::endl;
192     return EXIT_SUCCESS;
193 }
```

Gain RFNoC C++ App

Ettus Research™ A National Instruments Company

- Add code to control gain at runtime
- This code could have also been set_arg() or a call to a custom block controller method

```
if ((s.length() > 0) && (not stop)) {
    try {
        uint16_t gain = boost::lexical_cast<uint16_t>(s);
        gain_block_ctrl->sr_write("GAIN", gain);
    }
    catch (boost::bad_lexical_cast) {
         std::cout << "Invalid gain." << std::endl;
    }
}
```

Gain RFNoC C++ App

Ettus Research™
A National Instruments Company

```
120
121 void set_gain_thread(uhd::rfnoc::block_ctrl_base::sptr gain_block_ctrl) {
122     while (not stop) {
123         std::string s;
124
125         std::cout << "\nEnter gain (ctrl-c to exit): ";
126         std::cin >> s;
127
128         if ((s.length() > 0) && (not stop)) {
129             try {
130                 uint16_t gain = boost::lexical_cast<uint16_t>(s);
131                 gain_block_ctrl->sr_write("GAIN", gain);
132             }
133             catch (boost::bad_lexical_cast) {
134                 std::cout << "Invalid gain." << std::endl;
135             }
136         }
137     }
138 }
139
```

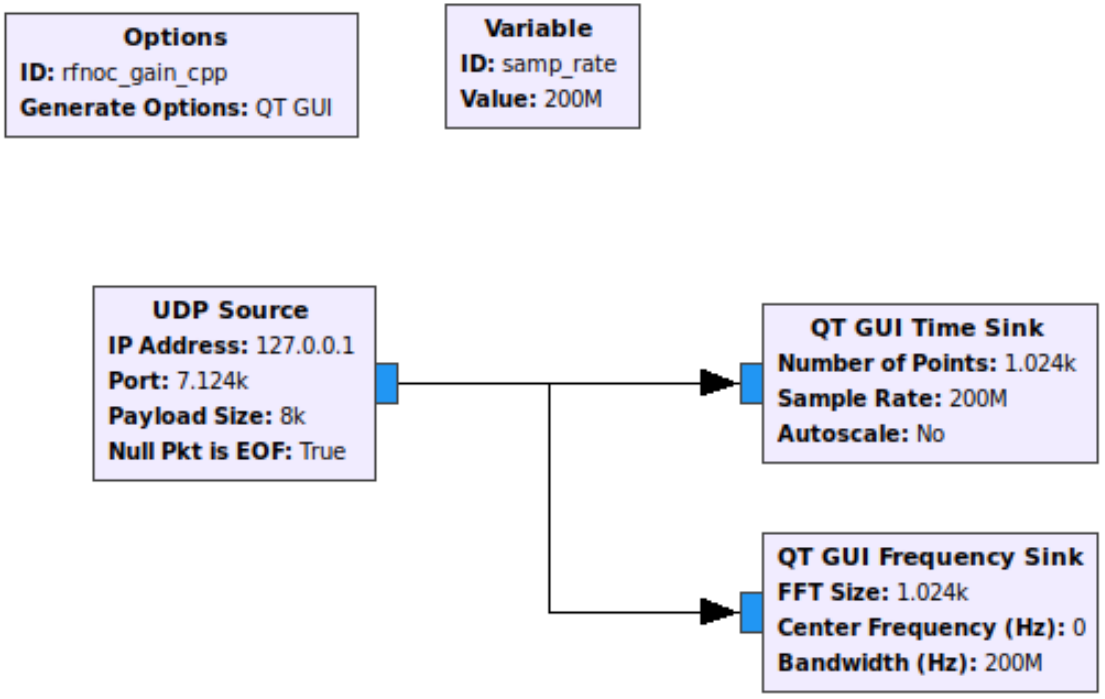
Gain RFNoC C++ App

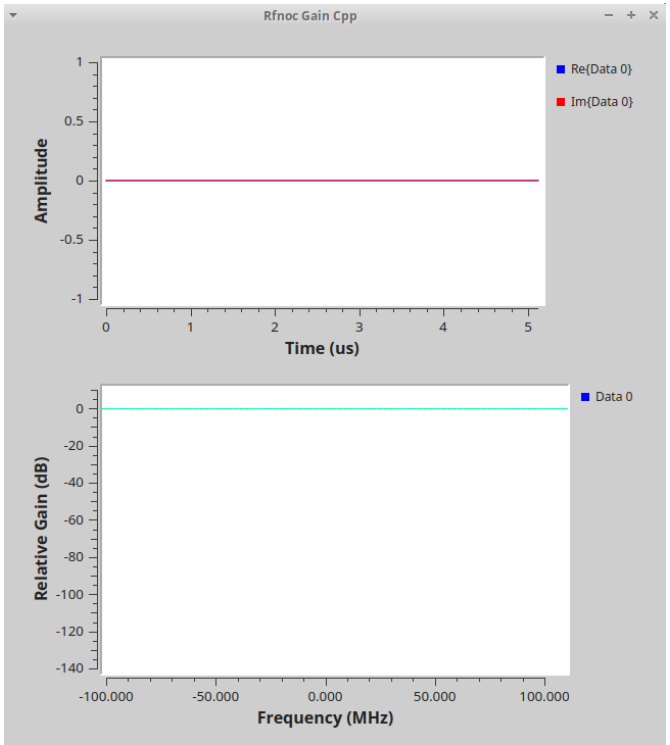
Ettus Research™
A National Instruments Company

- Rerun cmake and build
- **source ~/rfnoc-workshop/setup_env.sh**
- **cd ~/rfnoc-workshop/src/rfnoc-tutorial/build**
- **cmake ..**
- **make install**
- Now rfnoc_gain_example should be in path
- Need to start GNU Radio flowgraph first

Gain RFNoC C++ App

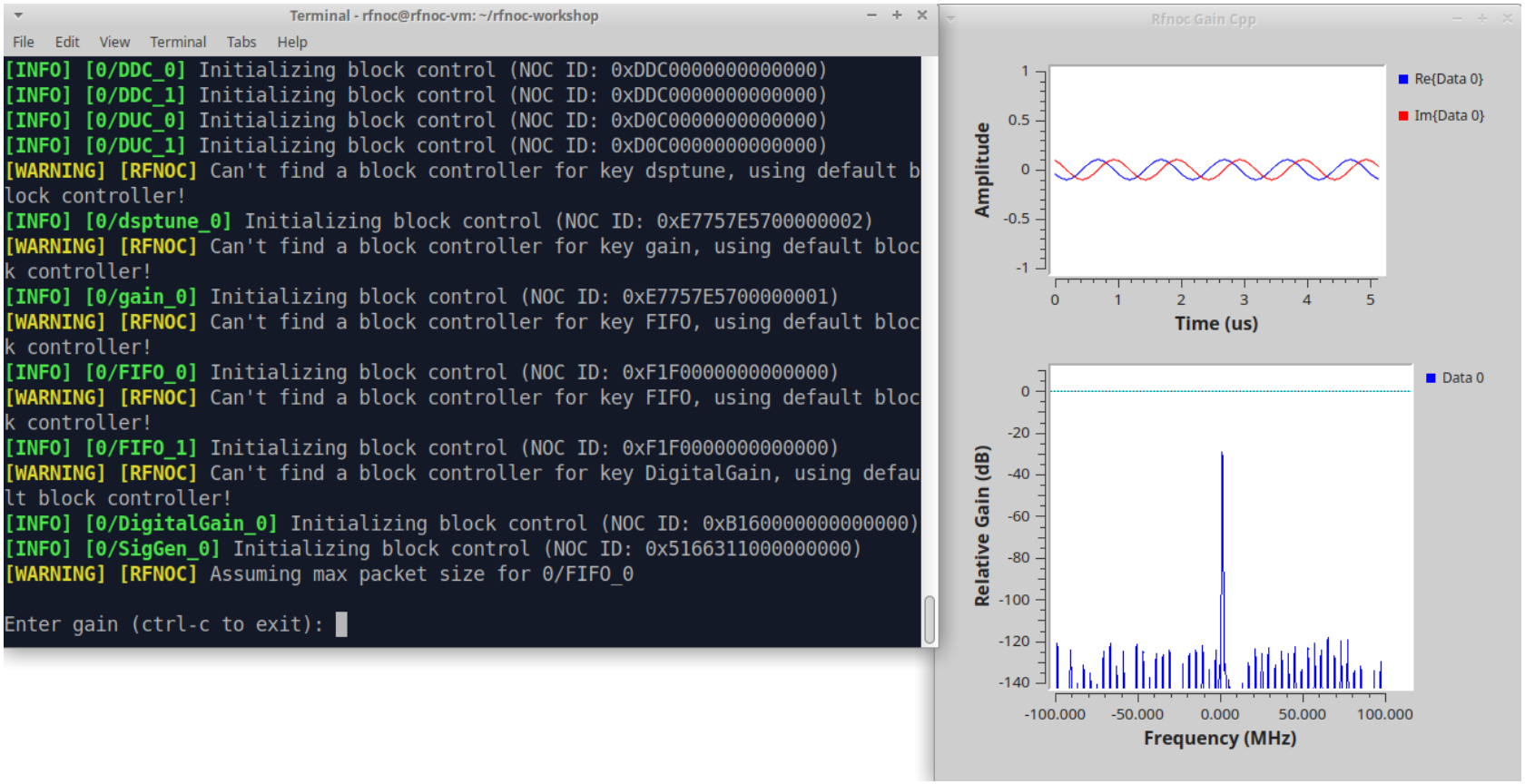
- Open gnuradio-companion
- Run ~/rfnoc-workshop/src/rfnoc-tutorial-complete/examples/rfnoc_gain_cpp.grc





Gain RFNoC C++ App

- In new terminal run rfnoc_gain_example
- **source ~/rfnoc-workshop/setup_env.sh**
- **rfnoc_gain_example**



Gain RFNoC C++ App

Ettus Research™ A National Instruments Company

- If you see an error, make sure the GNU Radio flowgraph is running first!

```
lock controller!
[INFO] [0/dsptune_0] Initializing block control (NOC ID: 0xE7757E5700000002)
[WARNING] [RFNOC] Can't find a block controller for key gain, using default bloc
k controller!
[INFO] [0/gain_0] Initializing block control (NOC ID: 0xE7757E5700000001)
[WARNING] [RFNOC] Can't find a block controller for key FIFO, using default bloc
k controller!
[INFO] [0/FIFO_0] Initializing block control (NOC ID: 0xF1F0000000000000)
[WARNING] [RFNOC] Can't find a block controller for key FIFO, using default bloc
k controller!
[INFO] [0/FIFO_1] Initializing block control (NOC ID: 0xF1F0000000000000)
[WARNING] [RFNOC] Can't find a block controller for key DigitalGain, using defau
lt block controller!
[INFO] [0/DigitalGain_0] Initializing block control (NOC ID: 0xB160000000000000)
[INFO] [0/SigGen_0] Initializing block control (NOC ID: 0x5166311000000000)
[WARNING] [RFNOC] Assuming max packet size for 0/FIFO_0

Enter gain (ctrl-c to exit):
Sending UDP packet failed: "send: Connection refused"

Done!

rfnoc@rfnoc-vm ~/rfnoc-workshop $
```

RFNoC Block Command Packets

Ettus Research™
A National Instruments Company

- Registers are set via CHDR command packets
- Noc Shell's cmdout port allows users to send command packets
  - Users can configure other blocks without host in FPGA
  - Example: send commands to Radio core to set hardware gain, retune frontend without host latency
- dsptune: example block to show how to use command packets to offset tune with DDC & DUC
  - Also shows how to create Out-of-Tree C++ block controllers for UHD and GNU Radio
  - Code located in rfnoc-tutorial-complete

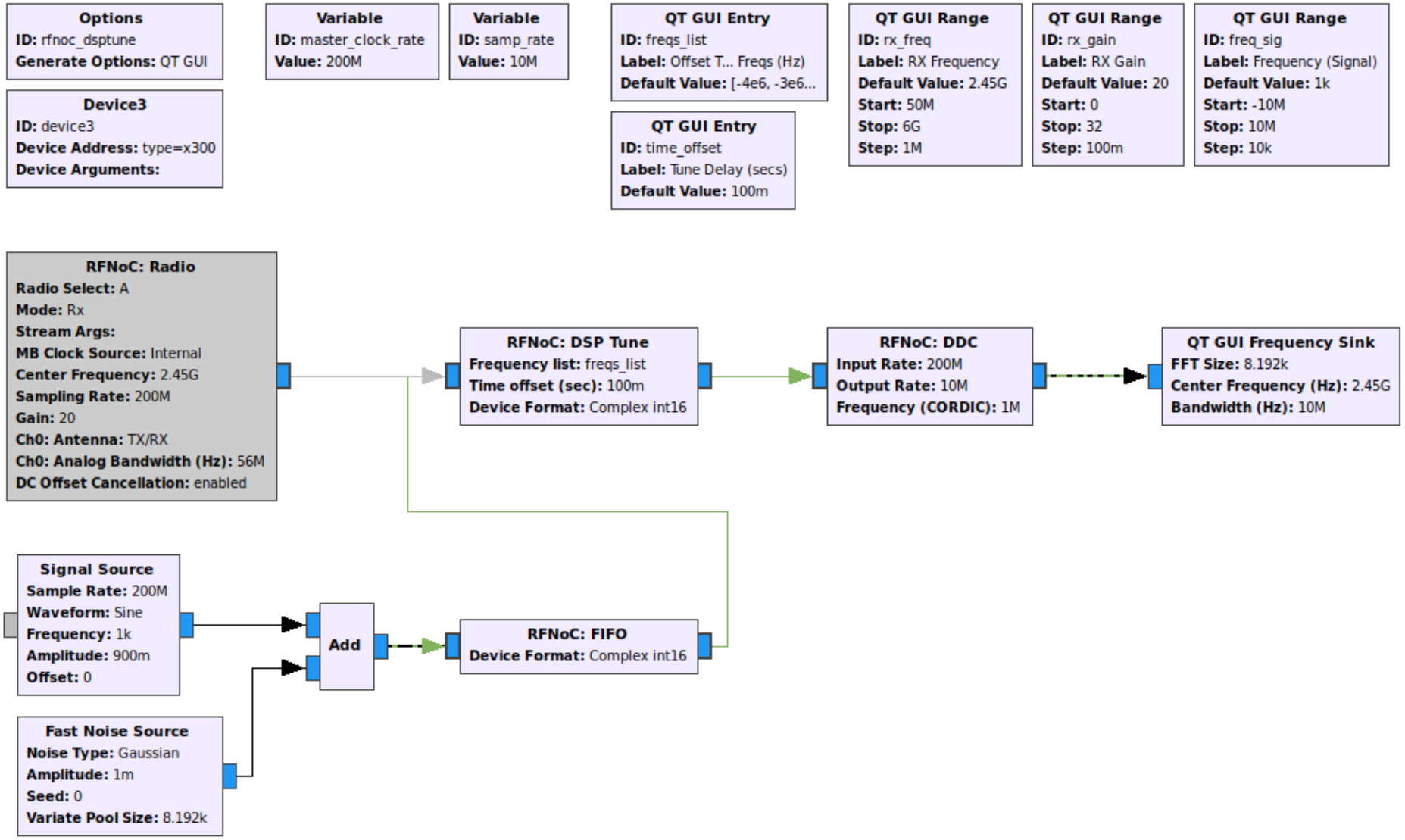
dsptune RFNoC Block Walkthrough

Ettus Research™
A National Instruments Company

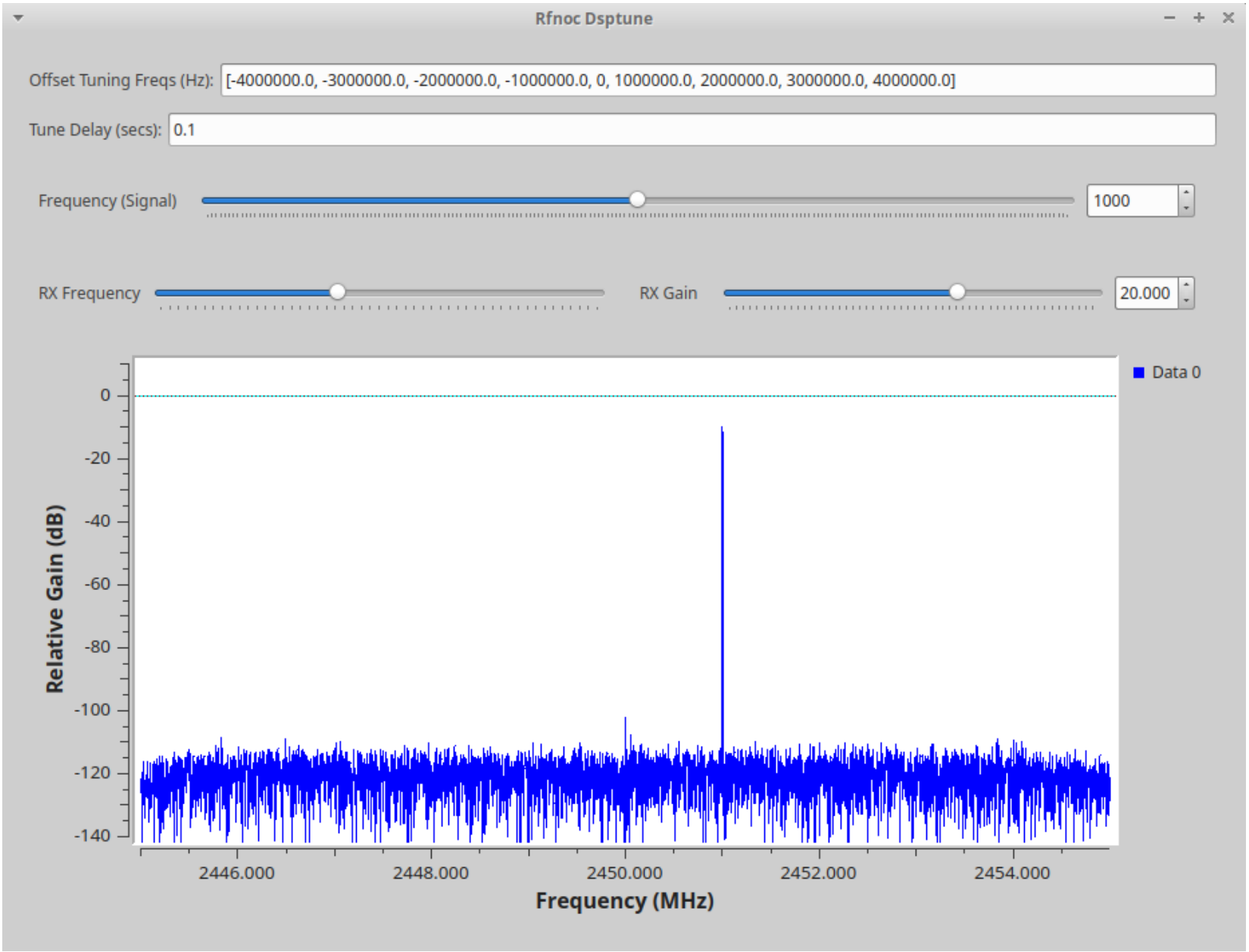
- Program X310 with second bitstream
  - **source ~/rfnoc-workshop/src/uhd-fpga/usrp3/top/x300/setupenv.sh**
  - **viv_jtag_program ~/rfnoc-workshop/src/usrp_x310_RFNOC_HG_2.bit**
- Run dsptune GNU Radio Companion flowgraph at ~/rfnoc-workshop/src/rfnoc-tutorial-complete/examples/tutorial_dsptune.grc

dsptune RFNoC Block Walkthrough

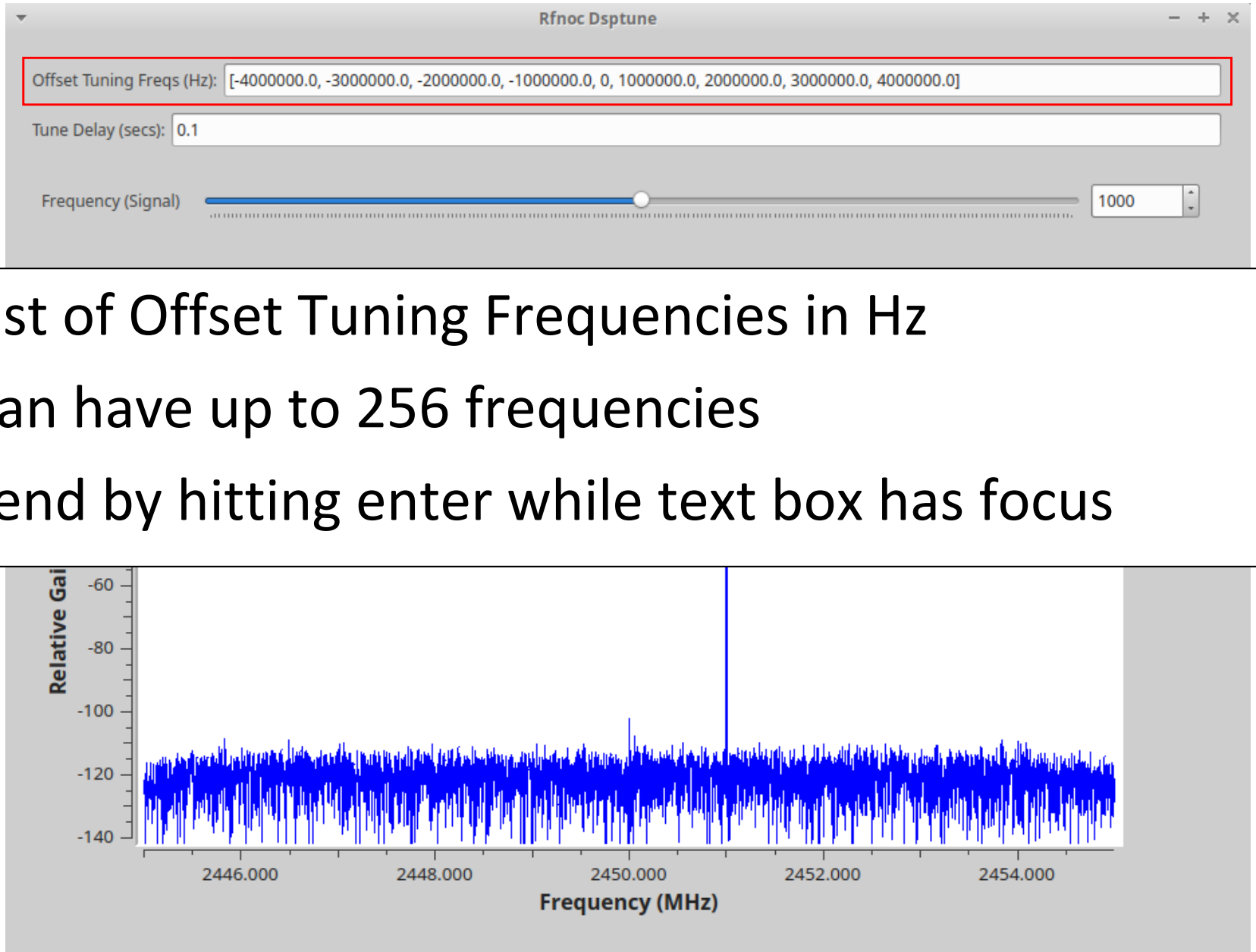
Ettus Research™
A National Instruments Company



dsptune RFNoC Block Walkthrough

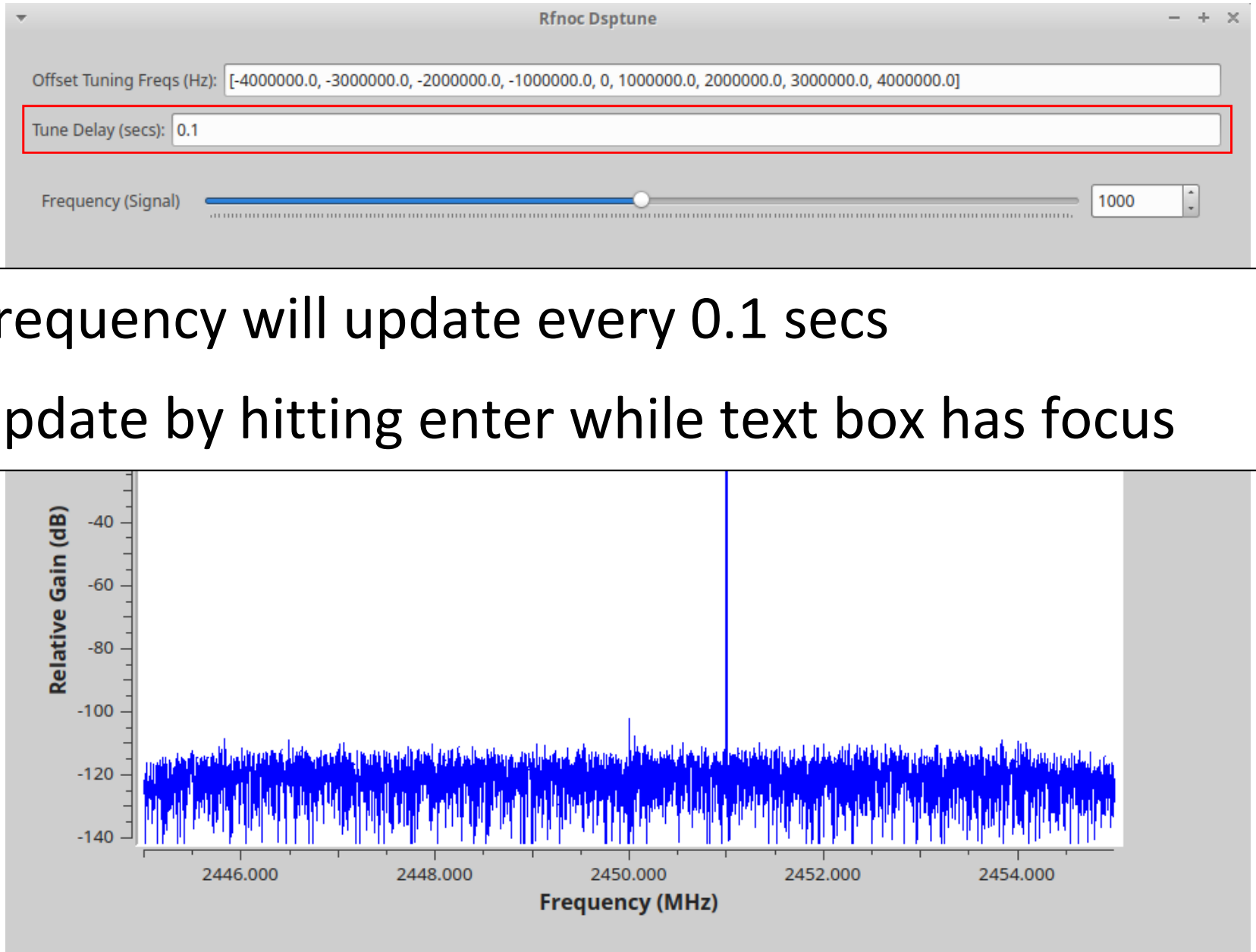


dsptune RFNoC Block Walkthrough



- List of Offset Tuning Frequencies in Hz
- Can have up to 256 frequencies
- Send by hitting enter while text box has focus

dsptune RFNoC Block Walkthrough



- Frequency will update every 0.1 secs
- Update by hitting enter while text box has focus

dsptune RFNoC Block Walkthrough

Ettus Research™ A National Instruments Company

- Open ~/rfnoc-workshop/src/rfnoc-tutorial-complete/rfnoc/fpga-src/noc_block_dsptune.v
- Design:
  - Frequency list loaded into FIFO via settings bus
    - axi_setting_reg handles all of this
  - Tune delay setting register
  - State machine to create and send command packets
  - How do we know what Stream ID to send to?
    - Uses next_dst_sid. This requires that the DDC or DUC must be the next block in the flowgraph. This is one of many valid approaches. Another approach would be to get the SID from UHD and program a setting register. We could have also tried to find the DDC / DUC by using command packets to read the NOC ID of every block connected to the crossbar.

dsptune RFNoC Block Walkthrough

Ettus Research™ A National Instruments Company

```
136
137  wire [31:0] freq_tdata;
138  wire freq_tvalid, freq_tready;
139  axi_setting_reg #(
140    .ADDR(SR_FREQ), .AWIDTH(8), .WIDTH(32), .USE_FIFO(1), .FIFO_SIZE(8))
141  axi_setting_reg_freq (
142    .clk(ce_clk), .reset(ce_rst),
143    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
144    .o_tdata(freq_tdata), .o_tlast(), .o_tvalid(freq_tvalid), .o_tready(freq_tready));
145
146  wire [31:0] time_offset;
147  setting_reg #(
148    .my_addr(SR_TIME_OFFSET), .awidth(8), .width(32))
149  sr_time_delta (
150    .clk(ce_clk), .rst(ce_rst),
151    .strobe(set_stb), .addr(set_addr), .in(set_data), .out(time_offset), .changed());
152
```

- Frequency list stored in AXI Setting Register:
  - Has AXI stream interface and internal FIFO
  - List is consumed so it must be reprogrammed

Dsptune RFNoC Block Walkthrough

Ettus Research™
A National Instruments Company

```
136
137  wire [31:0] freq_tdata;
138  wire freq_tvalid, freq_tready;
139  axi_setting_reg #(
140    .ADDR(SR_FREQ), .AWIDTH(8), .WIDTH(32), .USE_FIFO(1), .FIFO_SIZE(8))
141  axi_setting_reg_freq (
142    .clk(ce_clk), .reset(ce_rst),
143    .set_stb(set_stb), .set_addr(set_addr), .set_data(set_data),
144    .o_tdata(freq_tdata), .o_tlast(), .o_tvalid(freq_tvalid), .o_tready(freq_tready));
145
146  wire [31:0] time_offset;
147  setting_reg #(
148    .my_addr(SR_TIME_OFFSET), .awidth(8), .width(32))
149  sr_time_delta (
150    .clk(ce_clk), .rst(ce_rst),
151    .strobe(set_stb), .addr(set_addr), .in(set_data), .out(time_offset), .changed());
152
```

- Time Offset, called Tune Delay in the flowgraph, is a normal settings register

Dsptune RFNoC Block Walkthrough

Ettus Research™
A National Instruments Company

```
184
185  assign freq_tready = cmdout_tvalid & cmdout_tready & cmdout_tlast;
186
187  always @(posedge ce_clk) begin
188    if (ce_rst | clear_tx_seqnum) begin
189      seqnum       <= 0;
190      state        <= S_IDLE;
191    end else begin
192      case (state)
193        S_IDLE : begin
194          cnt <= 1;
195          if (freq_tvalid) begin
196            if (time_offset == 0) begin
197              state <= S_HDR;
198            end else begin
199              state <= S_WAIT;
200            end
201          end
202        end
203        // Wait until time to send freq
204        S_WAIT : begin
205          if (cnt == time_offset) begin
206            state <= S_HDR;
207          end else begin
208            cnt   <= cnt + 1;
209          end
210        end
211        // Send header
212        S_HDR : begin
213          if (cmdout_tready) begin
214            state  <= S_CMD;
215          end
216        end
217        // Send command
218        S_CMD : begin
219          if (cmdout_tready) begin
220            seqnum <= seqnum + 1;
221            state  <= S_IDLE;
222          end
223        end
224        default : state <= S_IDLE;
225      endcase
226    end
227  end
```

- State machine:
- If FIFO has a frequency
- Reads FIFO
- Waits for time offset
- Sends command packet
- Repeat

Dsptune RFNoC Block Walkthrough

Ettus Research™
A National Instruments Company

```
178
179  assign cmdout_tdata  = (state == S_HDR) ? {2'd2, 2'd0, seqnum, 16'd16, src_sid, next_dst_sid} :
180                                            {SR_FREQ_ADDR, freq_tdata}; // {address[31:0], data[31:0]}
181  assign cmdout_tvalid = (state == S_HDR || state == S_CMD) ? 1'b1 : 1'b0;
182  assign cmdout_tlast  = (state == S_CMD) ? 1'b1 : 1'b0;
183  assign ackin_tready  = 1'b1; // Accept all response packets
184
```

- Command packets controlled by State Machine
- Payload is frequency register address and frequency offset

```
228
229  // Passthru
230  assign m_axis_data_tready = s_axis_data_tready;
231  assign s_axis_data_tvalid = m_axis_data_tvalid;
232  assign s_axis_data_tlast  = m_axis_data_tlast;
233  assign s_axis_data_tdata  = m_axis_data_tdata;
234
```

- Samples pass through unchanged

dsptune RFNoC Block Walkthrough

Ettus Research™
A National Instruments Company

- dsptune sends a list or vector of frequencies
- But Noc Script XML does not support vectors
- Need to create C++ UHD block controller and GNU Radio block code
- GNU Radio dsptune block code: ~/rfnoc-workshop/src/rfnoc-tutorial-complete/lib/dsptune_impl.cc
- UHD dsptune block controller: ~/rfnoc-workshop/src/rfnoc-tutorial-complete/lib/dsptune_block_ctrl_impl.cpp

dsptune GR Block Code

Ettus Research™ A National Instruments Company

```
53  /*
54   * The private constructor
55   */
56  dsptune_impl::dsptune_impl(
57       const gr::ettus::device3::sptr &dev,
58       const ::uhd::stream_args_t &tx_stream_args,
59       const ::uhd::stream_args_t &rx_stream_args,
60       const int block_select,
61       const int device_select
62  )
63    : gr::ettus::rfnoc_block("dsptune"),
64      gr::ettus::rfnoc_block_impl(
65          dev,
66          gr::ettus::rfnoc_block_impl::make_block_id("dsptune",  block_select, device_select),
67          tx_stream_args, rx_stream_args
68          )
69  {}
70
71  void dsptune_impl::set_freqs(const std::vector<float> &freqs)
72  {
73      get_block_ctrl_throw< ::uhd::rfnoc::dsptune_block_ctrl >()->set_freqs(freqs);
74  }
75
```

- Added set_freqs() method that simply calls block controller's set_freqs() and passes freqs vector

dsptune UHD Block Controller

Ettus Research™ A National Instruments Company

```
42      //! Set window coefficients and length
43      void set_freqs(const std::vector<float> &freqs)
44      {
45          UHD_LOGGER_TRACE(unique_id())
46              << "dsptune_block::set_freqs()" << std::endl;
47          if (freqs.size() > MAX_NUM_FREQS) {
48              throw uhd::value_error(str(
49                  boost::format("dsptune_block::set_freqs(): Too many frequencies! "
50                                "Provided %d, can only store up to %d.\n")
51                  % freqs.size() % size_t(MAX_NUM_FREQS)
52              ));
53          }
54
55          // Check frequencies are valid
56          float sample_rate = _tree->access<double>("tick_rate").get();
57          float max_freq_upper = sample_rate/2.0;
58          float max_freq_lower = -sample_rate/2.0;
59          for (size_t i = 0; i < freqs.size(); i++) {
60              if (std::abs(freqs[i]/2.0) >= max_freq_upper) {
61                  throw uhd::value_error(str(
62                      boost::format("dsptune_block::set_freqs(): Frequency out of range! "
63                                    "Provided %f, valid range is (%f, %f).\n")
64                      % freqs[i] % max_freq_lower % max_freq_upper
65                  ));
66              }
67          }
68
69          // Send frequencies
75
76      private:
77
78      };
79
80      UHD_RFNOC_BLOCK_REGISTER(dsptune_block_ctrl,"dsptune");
```

- Error checking

Dsptune UHD Block Controller

Ettus Research™ A National Instruments Company

```
42      //! Set window coefficients and length
43      void set_freqs(const std::vector<float> &freqs)
44      {
45          UHD_LOGGER_TRACE(unique_id())
46              << "dsptune_block::set_freqs()" << std::endl;
47          if (freqs.size() > MAX_NUM_FREQS) {
48              throw uhd::value_error(str(
49                  boost::format("dsptune_block::set_freqs(): Too many frequencies! "
50                                "Provided %d, can only store up to %d.\n")
51                  % freqs.size() % size_t(MAX_NUM_FREQS)
52              ));
53          }
54
55          // Check frequencies are valid
56          float sample_rate = _tree->access<double>("tick_rate").get();
57          float max_freq_upper = sample_rate/2.0;
58          float max_freq_lower = -sample_rate/2.0;
59          for (size_t i = 0; i < freqs.size(); i++) {
60              if (std::abs(freqs[i]/2.0) >= max_freq_upper) {
61                  throw uhd::value_error(str(
62                      boost::format("dsptune block::set freqs(): Frequency out of range! "
67          }
68
69          // Send frequencies
70          for (size_t i = 0; i < freqs.size(); i++) {
71              uint32_t actual_freq = uint32_t(std::round((freqs[i] / sample_rate) * pow(2.0, 32)));
72              sr_write(SR_FREQS, actual_freq);
73          }
74      }
75
76  private:
77
78  };
79
80  UHD_RFNOC_BLOCK_REGISTER(dsptune_block_ctrl,"dsptune");
```

- Round frequencies and send via direct register write

Dsptune UHD Block Controller

Ettus Research™ A National Instruments Company

```
42    //! Set window coefficients and length
43    void set_freqs(const std::vector<float> &freqs)
44    {
45        UHD_LOGGER_TRACE(unique_id())
46            << "dsptune_block::set_freqs()" << std::endl;
47        if (freqs.size() > MAX_NUM_FREQS) {
48            throw uhd::value_error(str(
49                boost::format("dsptune_block::set_freqs(): Too many frequencies! "
50                              "Provided %d, can only store up to %d.\n")
51                % freqs.size() % size_t(MAX_NUM_FREQS)
52            ));
53        }
54
55        // Check frequencies are valid
56        float sample_rate = _tree->access<double>("tick_rate").get();
57        float max_freq_upper = sample_rate/2.0;
58        float max_freq_lower = -sample_rate/2.0;
```

- This is how UHD is aware of the Out-of-Tree block
- Macro registers block with UHD
- Name string must match block's Noc Script XML
- rfnocmodtool generates this code automatically

```
76 private:
77
78 };
79
80 UHD_RFNOC_BLOCK_REGISTER(dsptune_block_ctrl,"dsptune");
```

dsptune GRC XML

Ettus Research™ A National Instruments Company

- Setting “time_offset” is similar to “gain” in gain block
- Notice callback for set_freqs()
- SWIG bindings allow Python to directly call GNU Radio block method

```
2 <block>
21   )
22 self.$(id).set_arg("time_offset", $time_offset/(1/(125.0e6*(12.0/7.0))))
23 </make>
24
25   <callback>set_freqs($freqs)</callback>
26   <!-- Number of clock ticks relative to X310 ce_clk rate which is exactly 125.0e6*(12.0/7.0) -->
27   <callback>set_arg("time_offset", $time_offset/(1/(125.0e6*(12.0/7.0))))</callback>
28
29   <param>
30     <name>Frequency list</name>
31     <key>freqs</key>
32     <value>[1e6, 2e6, 3e6, 4e6]</value>
33     <type>real_vector</type>
34   </param>
35
36   <param>
37     <name>Time offset (sec)</name>
38     <key>time_offset</key>
39     <value>0.0</value>
40     <type>real</type>
41   </param>
42
```

Questions?

Final Takeaway

Ettus Research™ A National Instruments Company

RFNoC is for FPGAs is what GNU Radio is for GPPs

	RFNoC	GNU Radio
Infrastructure for SDR applications	✓	✓
Handles data movement between blocks	✓ (AXI-Based)	✓ (Circular Buffers)
Takes care of boring and recurring tasks	✓ (Flow control, addressing, routing)	✓ (R/W pointer up-dating, tag handling)
Provides library of blocks	✓ (Growing)	✓ (Huge and well-tested)
Has a graphical front end	✓ (gr-ettus)	✓ (GRC)
Open Source	✓	✓
Writes your blocks for you	✗	✗