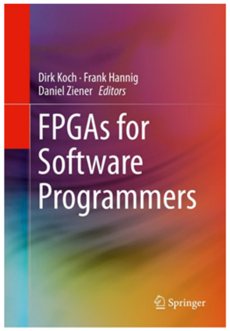
NI FPGA: Past, Present, Future

Terry Stratoudakis

"FPGAs." — "What?"

"Field programmable gate arrays
that are cool programmable chips." — "I see! Programmable. Great.
I have programming skills.
So, this should be an easy task for me."

"Well, not really,
but I can help you."



Dirk Koch · Frank Hannig
Daniel Ziener *Editors*

FPGAs for Software Programmers

Springer

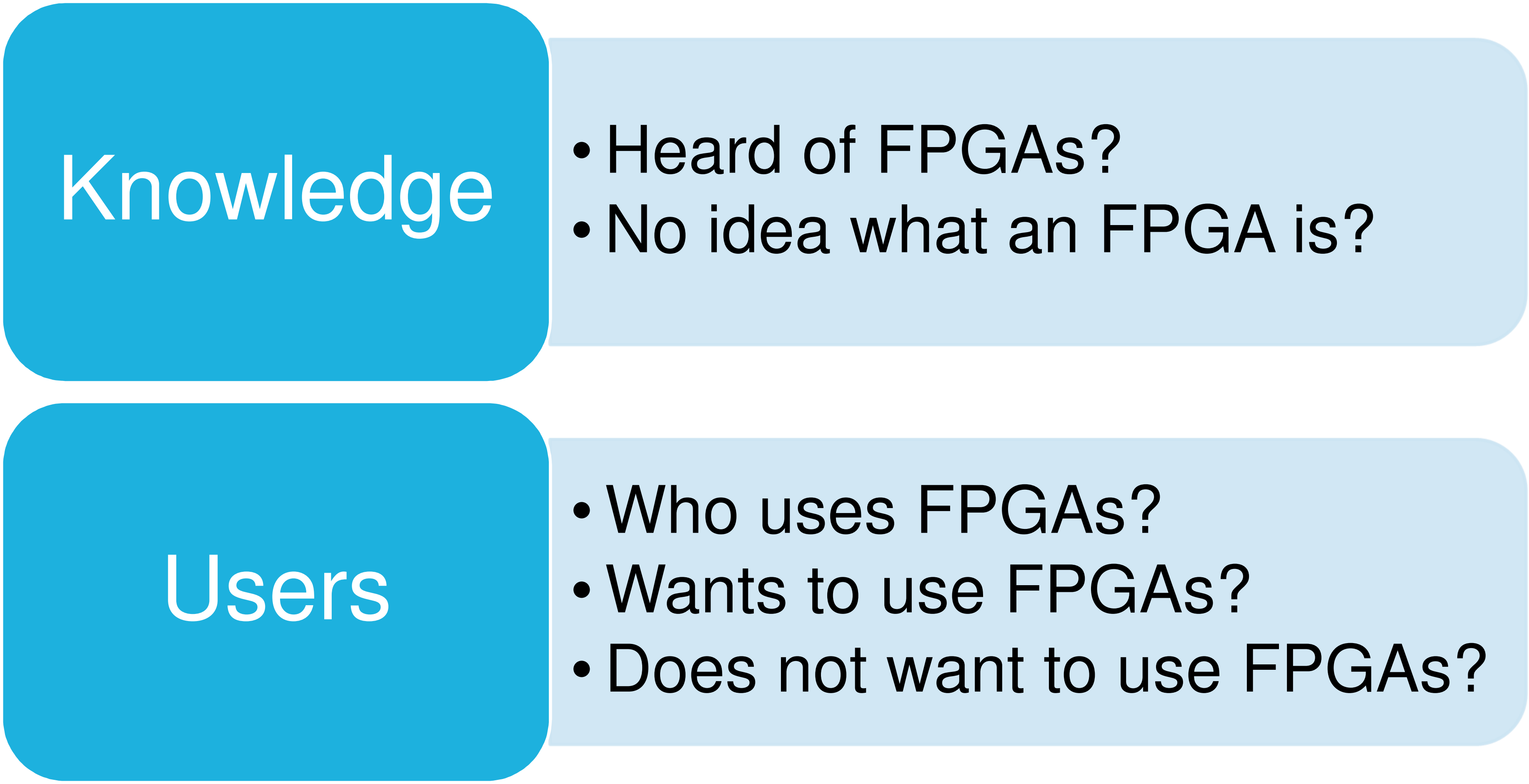
https://www.amazon.com/FPGAs-Software-Programmers-Dirk-Koch/dp/3319264060/ (page V)

Agenda

- Introductions
- FPGA Background
- NI FPGA Platform
- Workflows
- Case Studies
- Future Trends

Introductions

Audience?



Knowledge

- Heard of FPGAs?
- No idea what an FPGA is?

Users

- Who uses FPGAs?
- Wants to use FPGAs?
- Does not want to use FPGAs?

About Me

- Electrical Engineering degrees
- LabVIEW since 1998
- LabVIEW FPGA since 2008

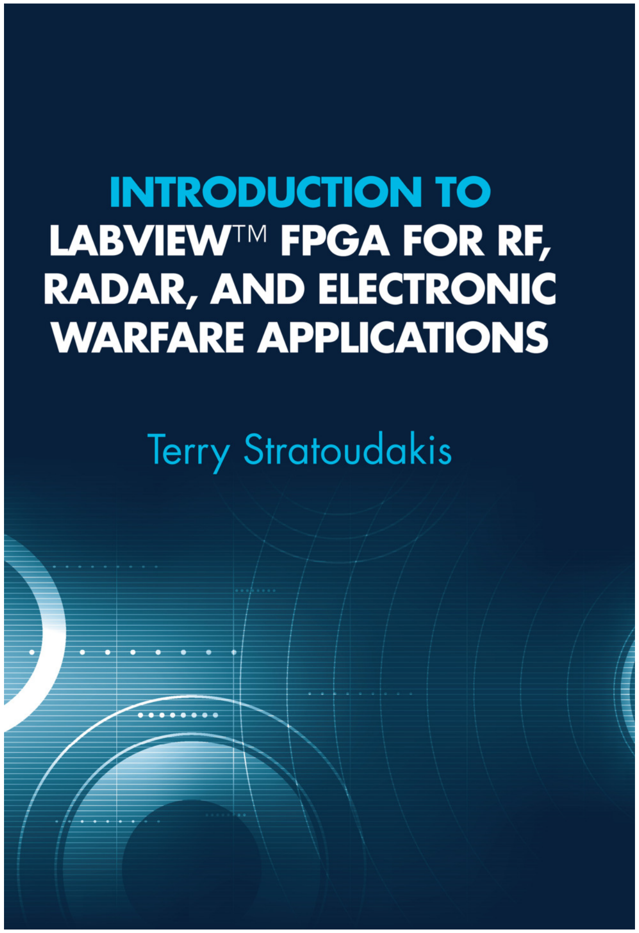
- Wrote book on LabVIEW FPGA in 2020

a) Training/Consulting

b) Engineering management

c) Architect/Developer

Systems Engineering approach



INTRODUCTION TO LABVIEW™ FPGA FOR RF, RADAR, AND ELECTRONIC WARFARE APPLICATIONS

Terry Stratoudakis

Background

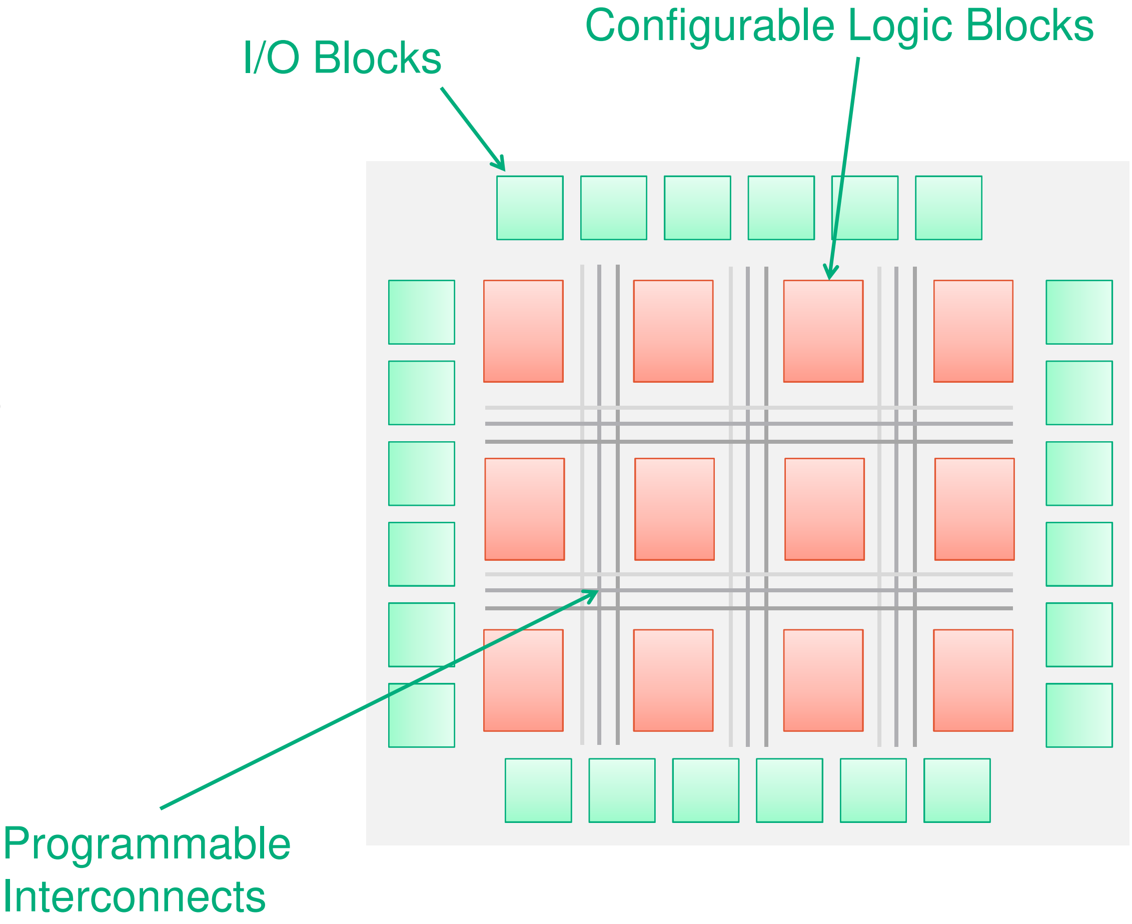
FPGAs: Why and what?

Why FPGAs?

Determinism	Low Latency	High Throughput	Custom Hardware
Low jitter	Quick (output) response to an input Microseconds	Massively parallel Ability to process gigabytes per second FPGAs go wide	Replace obsolete hardware Develop hardware that is not available on the market

What are FPGAs?

- Field Programmable Gate Arrays (FPGAs)
- Software defined hardware
- No operating system
- Configurable Integrated Circuit
- Programmed using Hardware Description Languages (HDL)



FPGAs compared to CPUs, GPUs, ASICs

	FPGA	CPU	GPU	ASIC
Latency	LOW	HIGH	HIGH	LOW
Throughput	HIGH	LOW	HIGH	HIGH
Development Time	MEDIUM	LOW	MEDIUM	HIGH
Custom Hardware	HIGH	LOW	LOW	HIGH
Reconfigurability	HIGH	NONE	NONE	NONE
Multi-core	HIGH	LOW	HIGH	HIGH
Developer Skill	Hardware	Software	Software, Parallel	Hardware

FPGA Challenges

Specialized skillset

- Electrical Engineers with specific background
- Different than software development

Long compile times – minutes, hours, overnight

- Mitigated with simulation

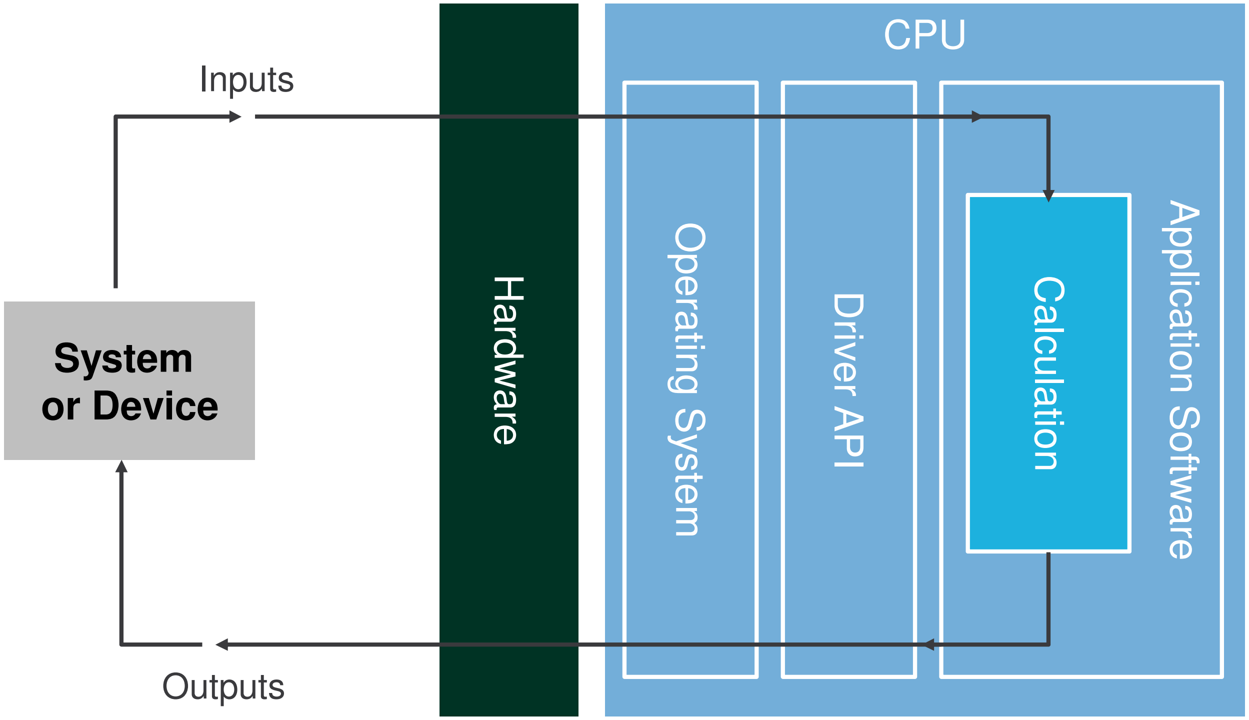
Vendor lock-in

Slow to adopt software engineering practices

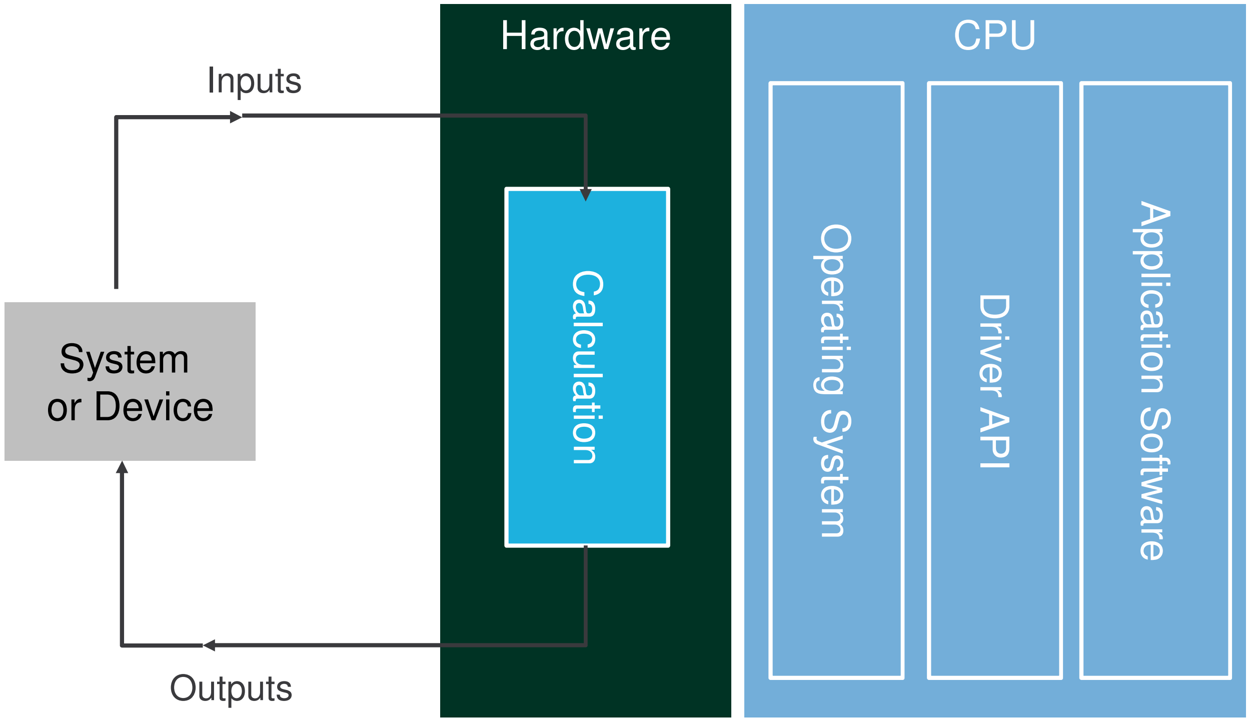
NI FPGA Platform

Software and Hardware review

Processor Based Approach



Decision Making in FPGA Hardware



NI FPGA-based Hardware

CompactRIO

FlexRIO

USRP RIO

Multifunction RIO

Modular Instruments

RF Instruments

FlexRIO with Integrated I/O

High-Speed Serial

25 Gbps per lane

16 Gbps per lane

PXIe-7903

Digitizers

4 ch. 500 MS/s
16-bit AI

4 ch. 1 GS/s
16-bit AI

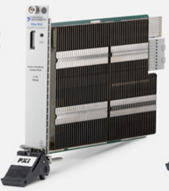
2 ch. 6.4 GS/s
12-bit AI
DC and AC Coupled Variants

Transceivers

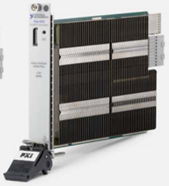
2x2 ch. 6.4 GS/s
12-bit AI & AO

Coprocessors

50,780 FPGA
Slices



60,600 FPGA
Slices



82,920 FPGA
Slices

Signal Generators

2 ch. 6.4 GS/s
12-bit AO

35+ Modules

High Speed Serial and Coprocessors

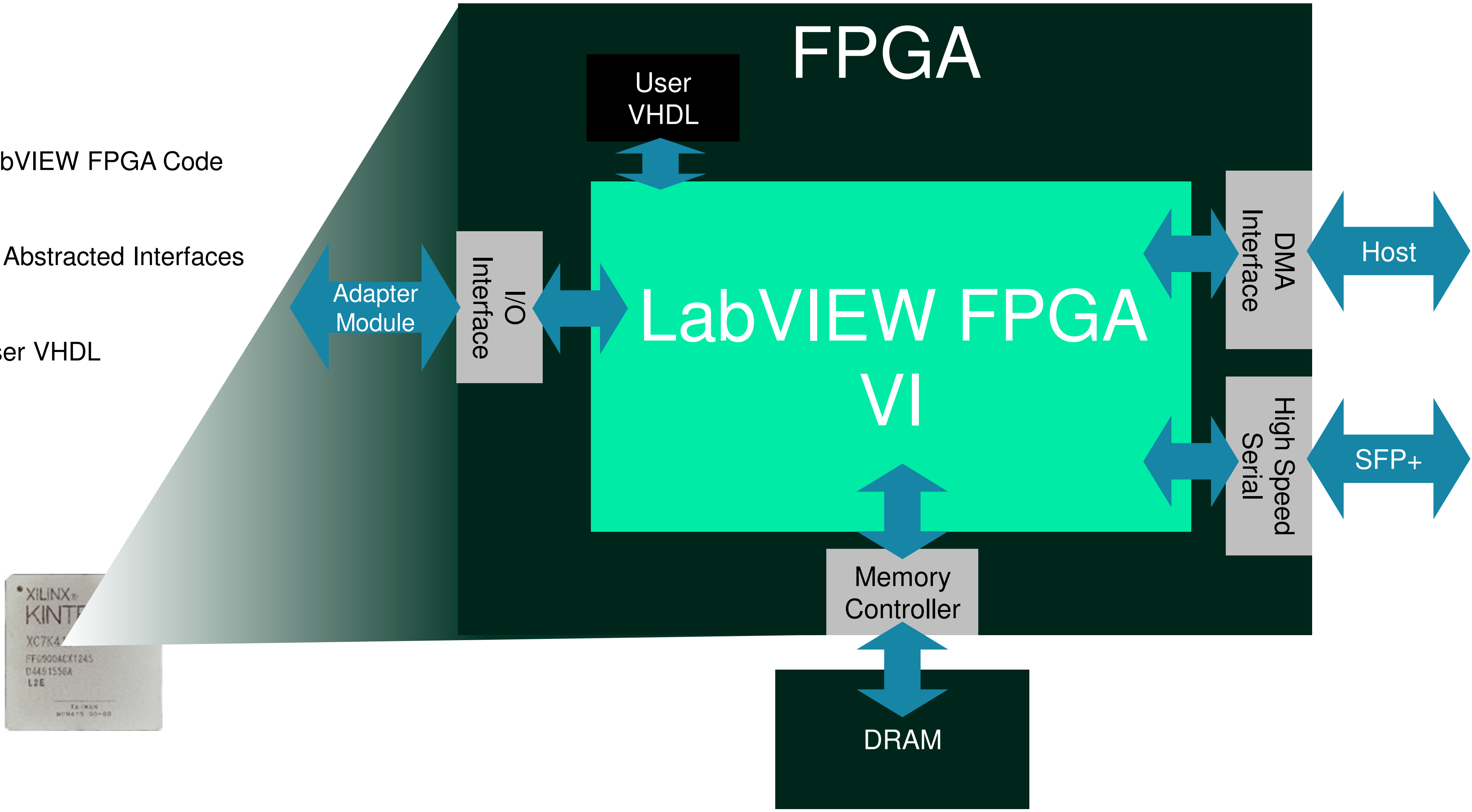
Model Name	PXIe-6594	PXIe-7902	PXIe-7915	PXIe-7903
I/O	8 RX/TX (MGTs) 8 DIO	24 RX/TX (MGTs)	4 RX/TX (MGTs) 8 DIO	48 RX/TX (MGTs)
Maximum Serial Data Rate (per channel)	28 Gb/s	12.5 Gb/s	16.4 Gb/s	28.2 Gb/s
FPGA	Kintex UltraScale+ KU15P	Virtex-7 485T	Kintex UltraScale KU060	Virtex UltraScale+ VU11P
Dynamic RAM	8 GB	2 GB	4 GB	20 GB
Block RAM	34.6 Mb	37.1 Mb	38.0 Mb	341 Mb
DSP Slices	1968	2800	2760	9216
PXI Backplane Link	PCIe Gen3 x8	PCIe Gen2 x8	PCIe Gen3 x8	PCIe Gen3 x8

Focus On Your Algorithm

🟩 = LabVIEW FPGA Code

▨ = NI Abstracted Interfaces

⬛ = User VHDL



LabVIEW FPGA

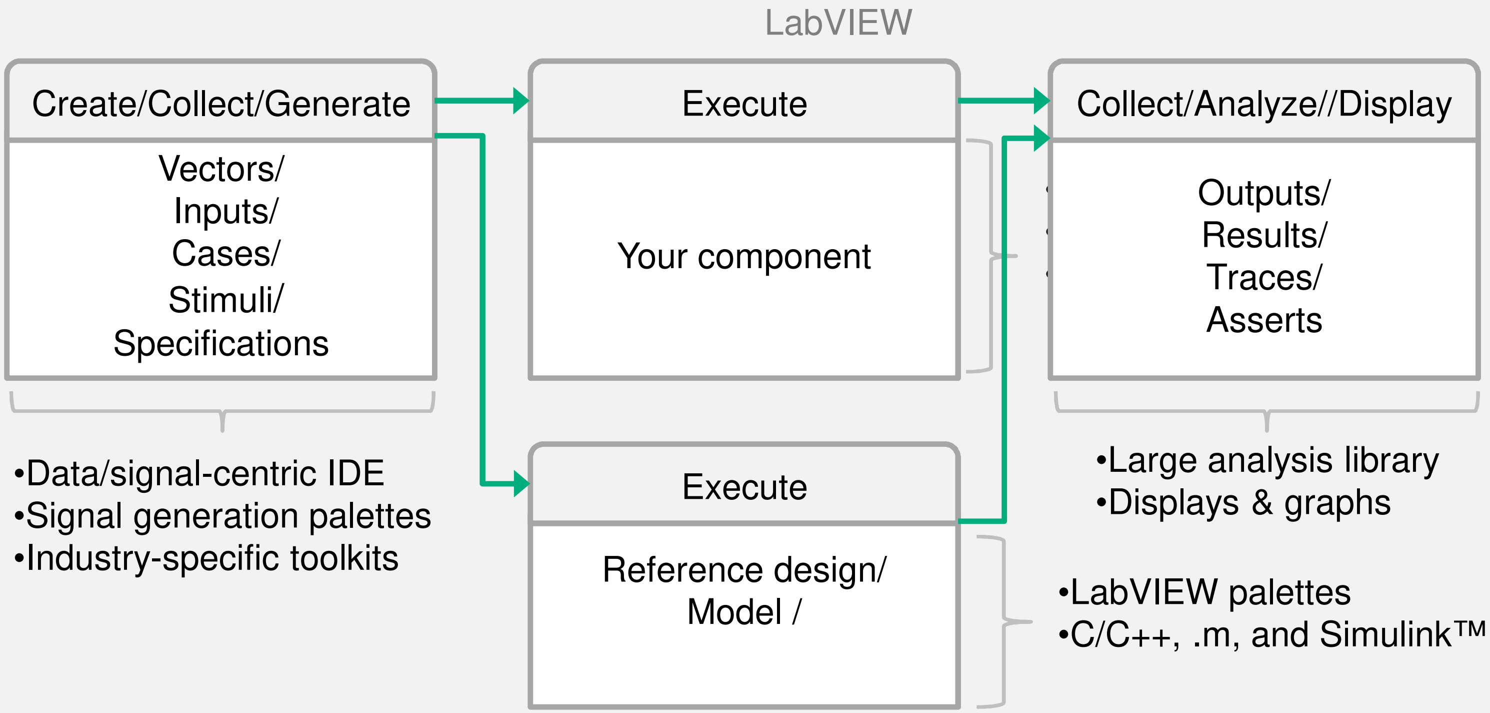
Add-on to LabVIEW

Since 2003

Works with AMD/Xilinx Vivado

Verifying Your Components with LabVIEW

- Create, execute, analyze, and present test results from one environment



LabVIEW

Create/Collect/Generate

Vectors/
Inputs/
Cases/
Stimuli/
Specifications

•Data/signal-centric IDE
•Signal generation palettes
•Industry-specific toolkits

Execute

Your component

Execute

Reference design/
Model /

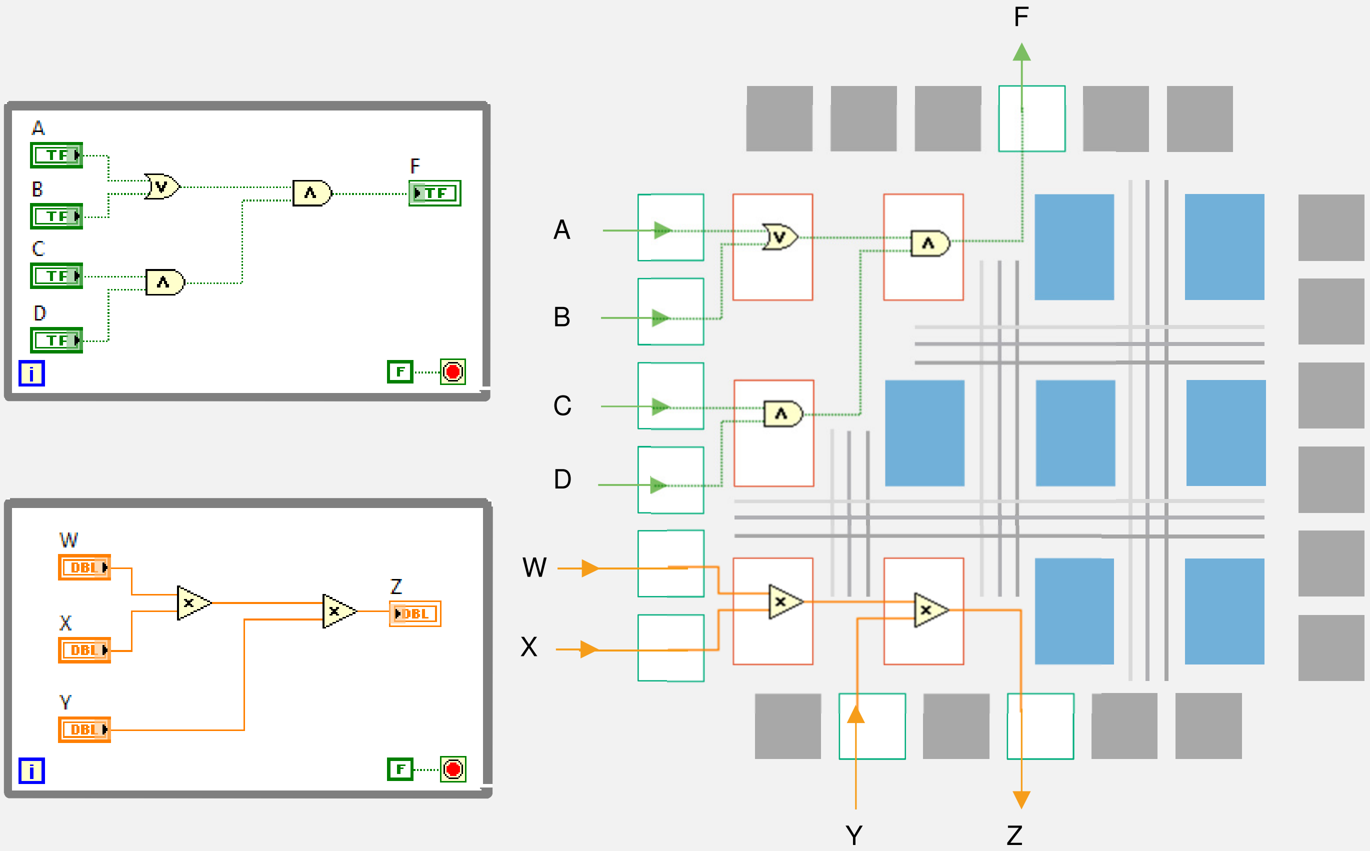
•LabVIEW palettes
•C/C++, .m, and Simulink™

Collect/Analyze//Display

Outputs/
Results/
Traces/
Asserts

•Large analysis library
•Displays & graphs

Mapping LabVIEW to an FPGA



Public Service Announcement

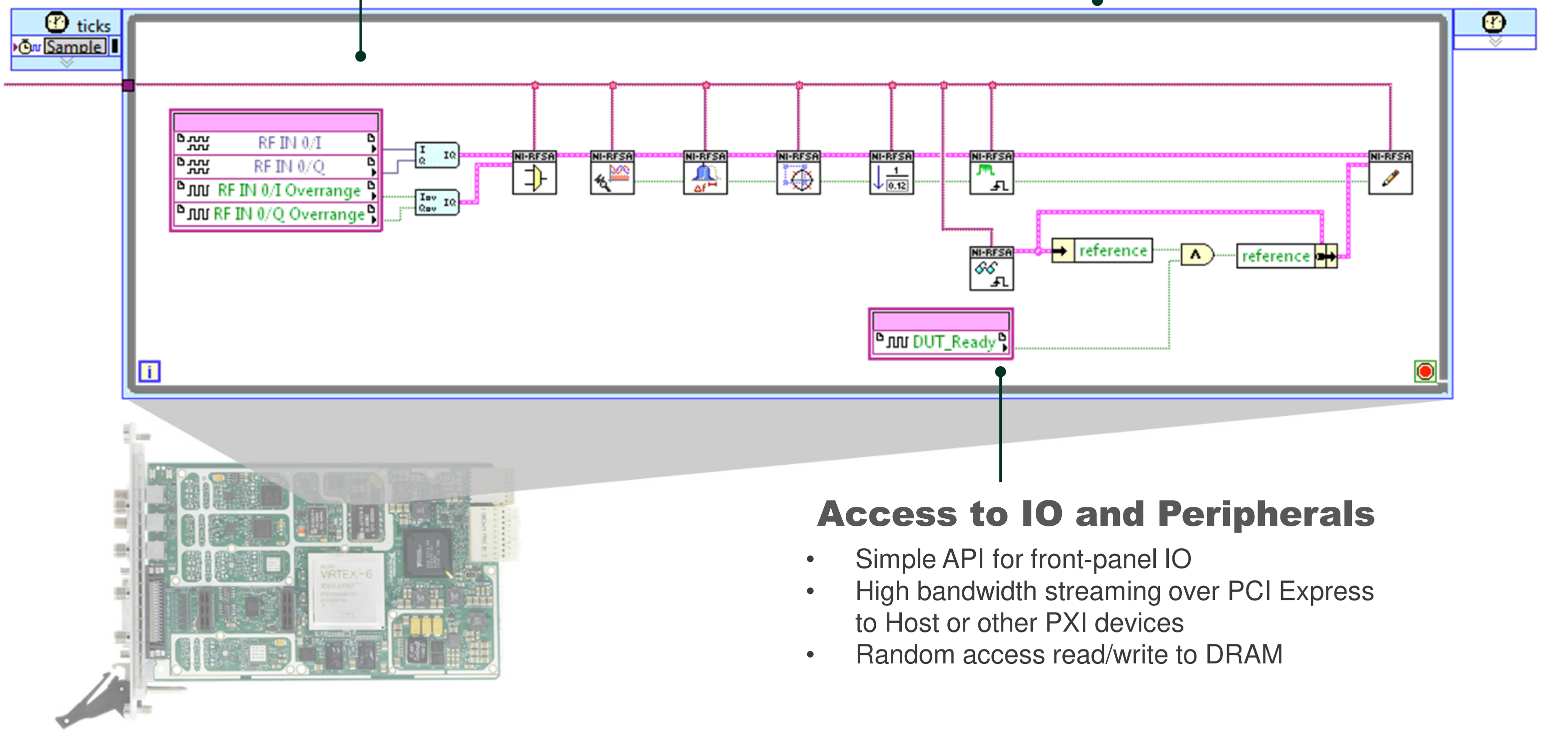
Easy is a four-letter word

Program with LabVIEW FPGA

- Familiar LabVIEW programming elements
- Develop, simulate, debug, compile and deploy through LabVIEW
- Integrate external FPGA IP

High-Performance Features

- High-throughput math functions
- Advanced timing control
- Access to optimized DSP Cores

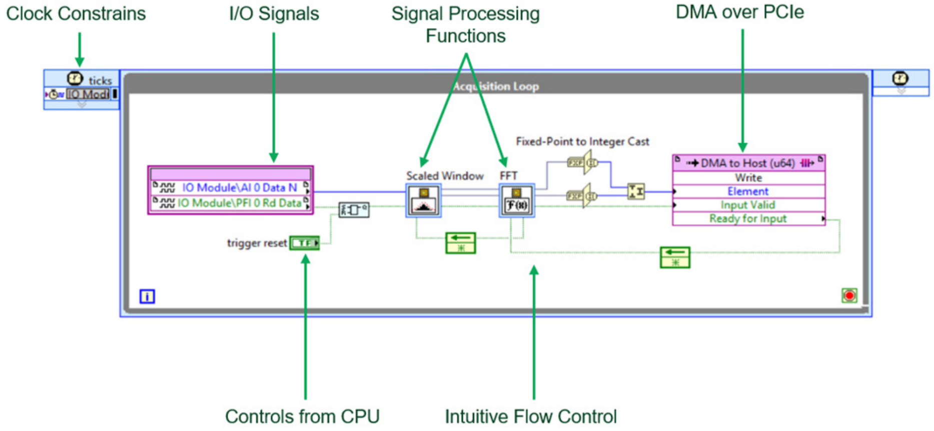


Access to IO and Peripherals

- Simple API for front-panel IO
- High bandwidth streaming over PCI Express to Host or other PXI devices
- Random access read/write to DRAM

Programming FPGAs with LabVIEW

Focus on your algorithms, not infrastructure



Clock Constrains

I/O Signals

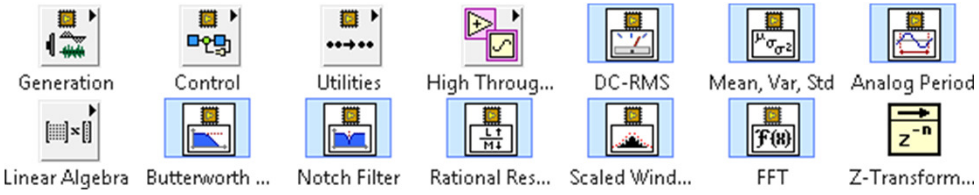
Signal Processing Functions

DMA over PCIe

Controls from CPU

Intuitive Flow Control

Save time with extensive libraries of FPGA IP



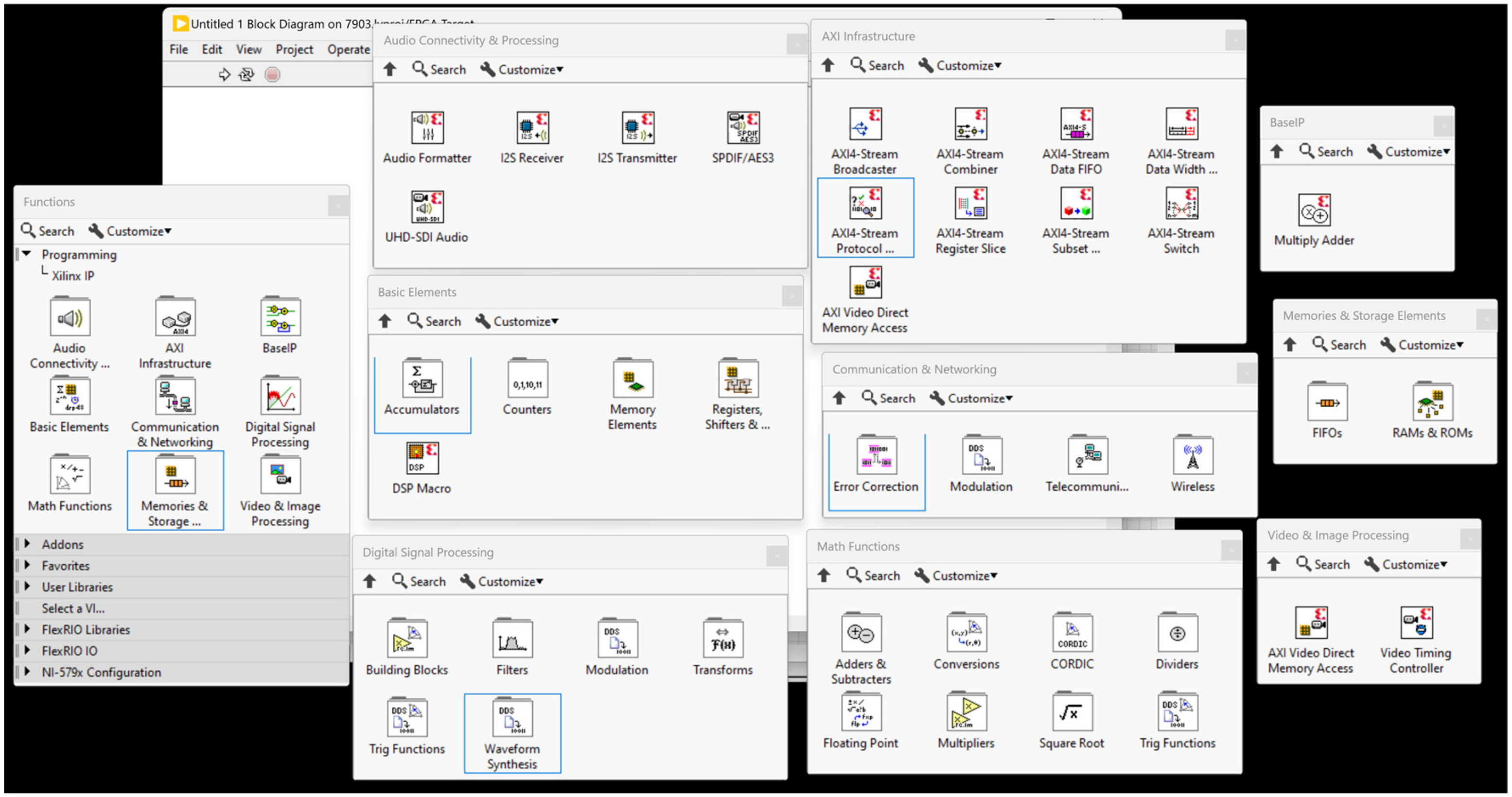
LabVIEW FPGA IP

10 Gigabit Ethernet UDP
3-Phase PLL
Accumulator
All-digital PLL
Area measurements
Bayer decoding
Binary morphology
Binary object detection
BRAM delay
BRAM FIFO
BRAM packetizer
Butterworth filter
Centroid calculation
Channel emulation
Channel power
CIC compiler
Color extraction
Color space conversion
Complex multiply
Corner detection
Counters
D latch
Delay
Digital gain
Digital pre-distortion
Digital pulse processing filter
Discrete delay
Discrete normalized integrator
Divide
Dot product
DPO
DRAM FIFO IDL
DRAM packetizer
DSP48 node
DUC/DDC compiler

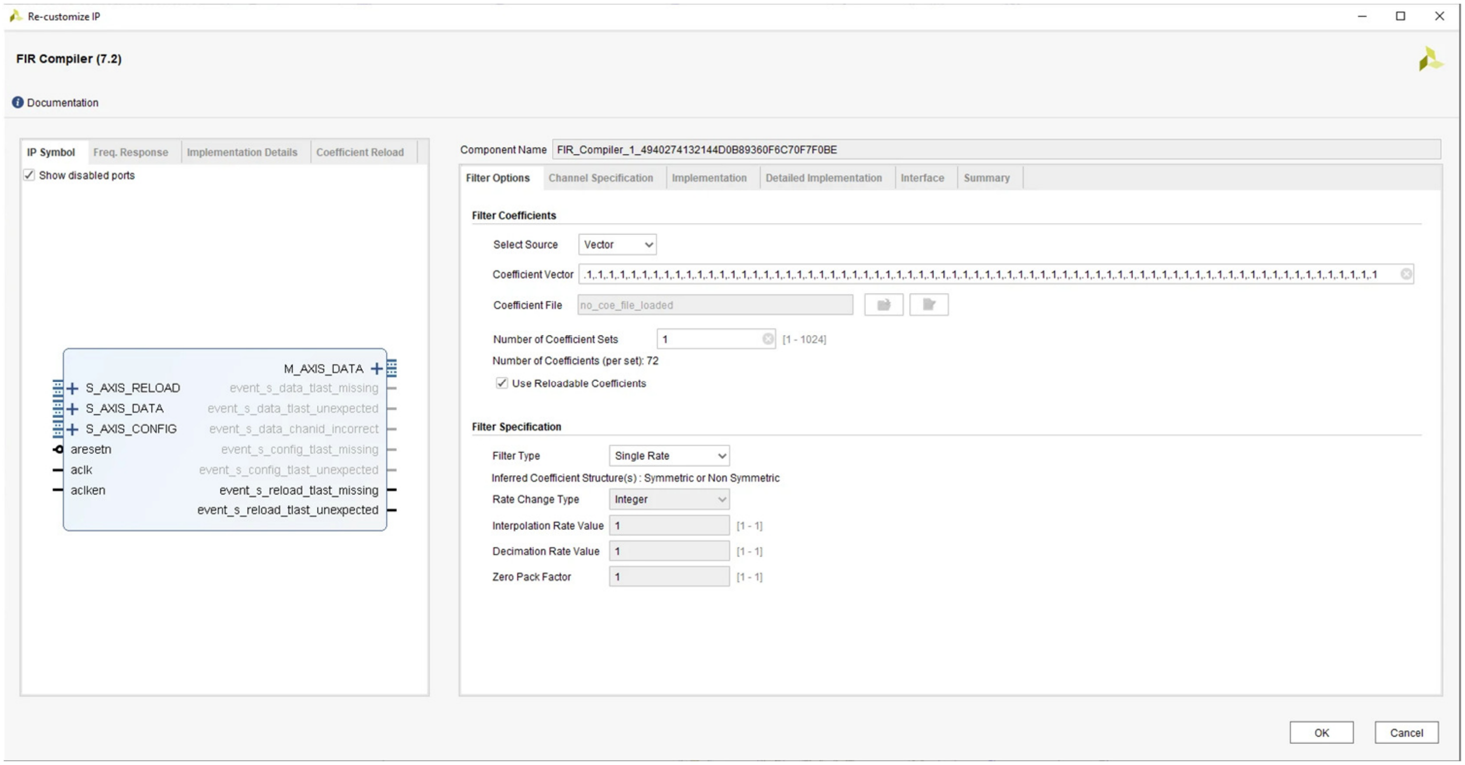
Edge detection
Equalization
Exponential
FFT
Filtering
FIR compiler
Fixed-point filter design
Fractional interpolator
Fractional resampler
Frequency domain measurements
Frequency mask trigger
Frequency shift
Halfband decimator
Handshake
Hardware test sequencer
I2C
Image operators
Image transforms
Instruction sequencer
IQ impairment correction
Line detection
Linear interpolation
Lock-in amplifier filter
Log
Matrix multiply
Matrix transpose
Mean, Var, Std deviation
Memory IDL
Moving average
N channel DDC
Natural log
Noise generation
Normalized square
Notch filter

Persistence display
PFT channelizer
PID
Pipeline frequency transform (PFT)
Polar to X/Y conversion
Power level trigger
Power servoing
Power spectrum
Programmable filter
Pulse measurements
Reciprocal
RFFE
Rising/falling edge detect
RS-232
Scaled window
Shading correction
Sin & Cos
Spectrogram
SPI
Square root
Streaming controller
Streaming IDL
Synchronous latch
Trigger IDL
Unit delay
VITA-49 data packing
Waveform generation
Waveform match trigger
Waveform math
X/Y to polar conversion
Xilinx Aurora
Zero crossing
Zero order hold
Z-Transform delay

Xilinx IP available through LabVIEW FPGA



Xilinx IP – FIR Compiler – Filter Options



Re-customize IP

FIR Compiler (7.2)

Documentation

☑ Show disabled ports

M_AXIS_DATA
S_AXIS_RELOAD
S_AXIS_DATA
S_AXIS_CONFIG
aresetn
aclk
aclken
event_s_data_tlast_missing
event_s_data_tlast_unexpected
event_s_data_chanid_incorrect
event_s_config_tlast_missing
event_s_config_tlast_unexpected
event_s_reload_tlast_missing
event_s_reload_tlast_unexpected

Component Name: FIR_Compiler_1_4940274132144D0B89360F6C70F7F0BE

Filter Coefficients

Select Source: Vector

Coefficient Vector: .1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1

Coefficient File: no_coe_file_loaded

Number of Coefficient Sets: 1 [1 - 1024]

Number of Coefficients (per set): 72

☑ Use Reloadable Coefficients

Filter Specification

Filter Type: Single Rate

Inferred Coefficient Structure(s) : Symmetric or Non Symmetric

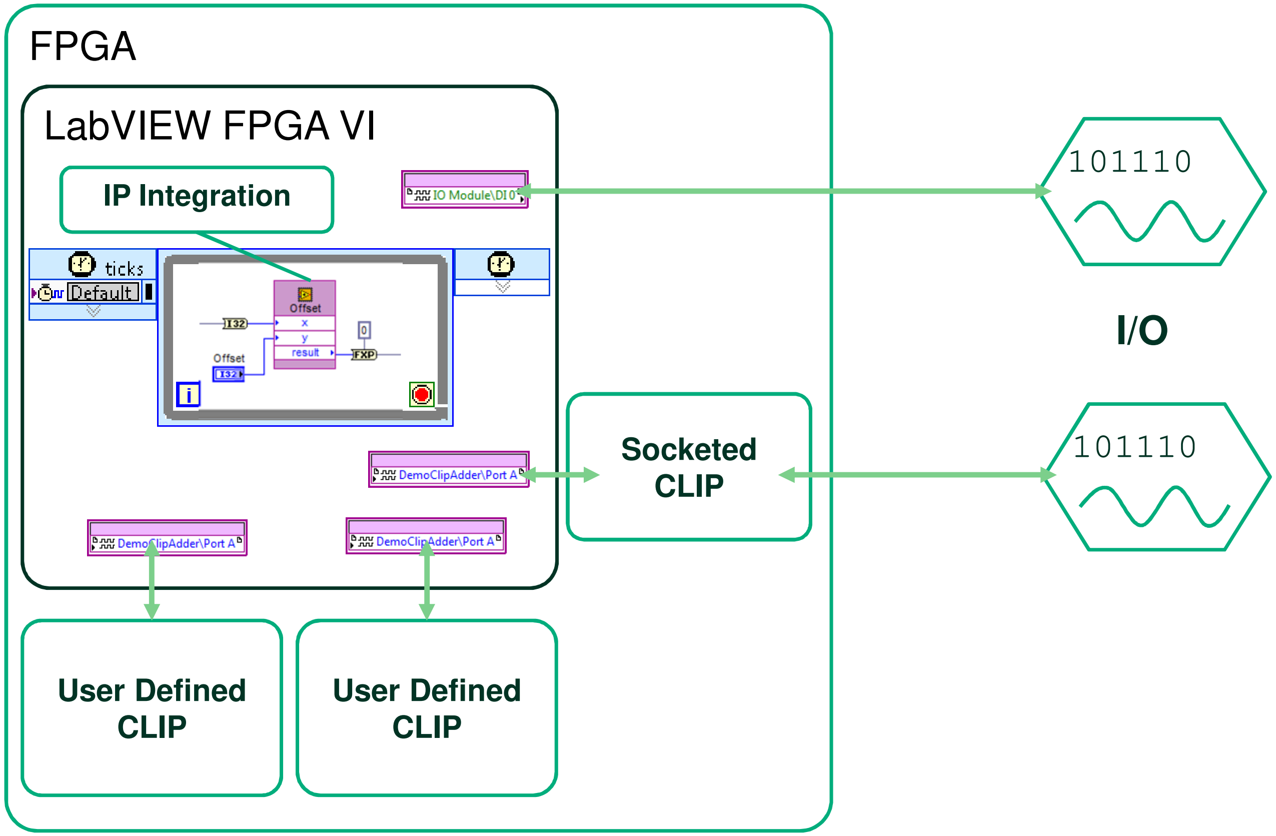
Rate Change Type: Integer

Interpolation Rate Value: 1 [1 - 1]

Decimation Rate Value: 1 [1 - 1]

Zero Pack Factor: 1 [1 - 1]

HDL Integration Mechanisms



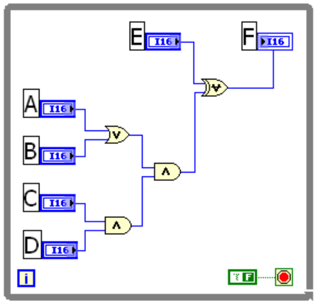
Leveraging Existing HDL Code

	CLIP	IPIN
Supported Execution Modes	• Inside SCTL • Outside SCTL	Inside SCTL
Support for Simulation	**No**	Yes
Support for 3rd Party Simulation	Yes	Yes
Support for multiple clock domains	Maximum number of clocks defined by FPGA	**Maximum of two clocks: an SCTL clock and an FPGA-derived clock, where the derived clock executes at a rate that is an integer multiple of the SCTL clock**
Execution mode with LabVIEW FPGA	Asynchronously to LabVIEW FPGA block diagram	Inline with LabVIEW FPGA block diagram

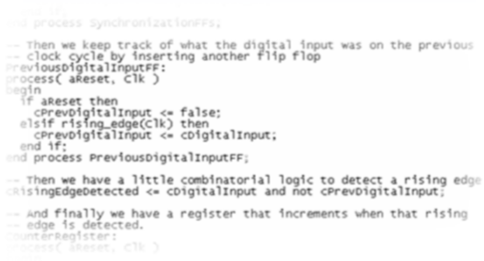
Additional information can be found in the LabVIEW Help

Compilation Process

LabVIEW FPGA Code

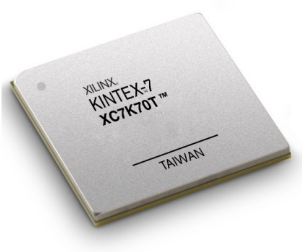


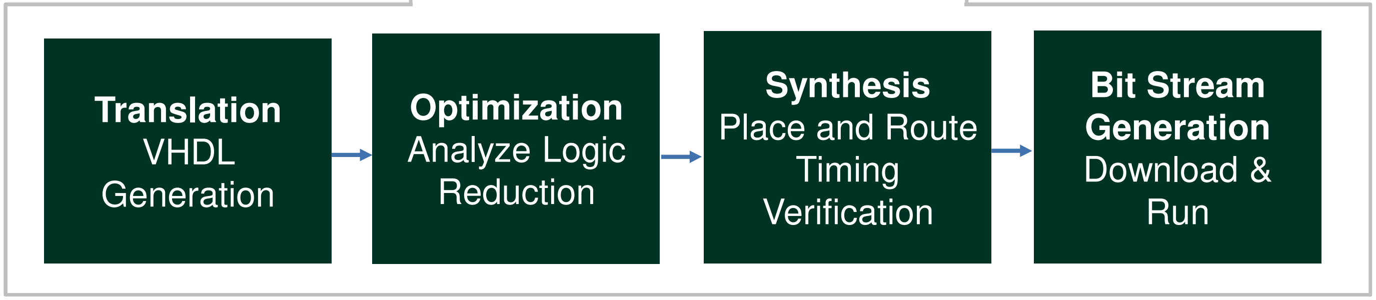
Compile VHDL through Xilinx



```
  end if;
end process SynchronizationFFs;

-- Then we keep track of what the digital input was on the previous
-- clock cycle by inserting another flip flop
PreviousDigitalInputFF:
process( aReset, Clk )
begin
  if aReset then
    cPrevDigitalInput <= false;
  elsif rising_edge(Clk) then
    cPrevDigitalInput <= cDigitalInput;
  end if;
end process PreviousDigitalInputFF;

-- Then we have a little combinatorial logic to detect a rising edge
cRisingEdgeDetected <= cDigitalInput and not cPrevDigitalInput;

-- And finally we have a register that increments when that rising
-- edge is detected.
CounterRegister:
process( aReset, Clk )
begin
```

FPGA Logic Implementation





Translation VHDL Generation

Optimization Analyze Logic Reduction

Synthesis Place and Route Timing Verification

Bit Stream Generation Download & Run

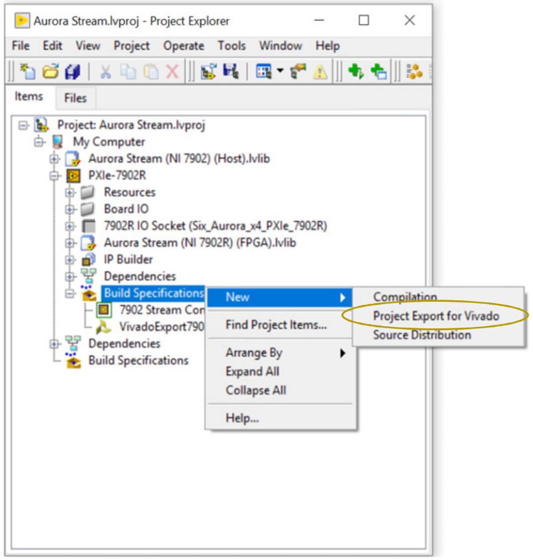
Public Service Announcement

Done is a four-letter word

LabVIEW FPGA Vivado Project Export

Using Vivado IDE – Vivado Project Export

- Enables development for NI FPGA-enabled hardware (e.g. FlexRIO) using Xilinx Vivado
- Maintains benefits of hardware abstraction
  - Retains hardware-specific “Board Support IP”
  - Interfaces (ADC, DAC, PCIe, DRAM)
  - Driver support (NI-RIO)
  - Provide “low-level” access
  - Intermediate files (design checkpoints, all reports)
  - Vivado tools
    - Timing Closure and Design Analysis, Applying Design Constraints, Design Analysis and Floorplanning, etc
  - 3rd party tools
- Enable optional use of existing LabVIEW FPGA IP

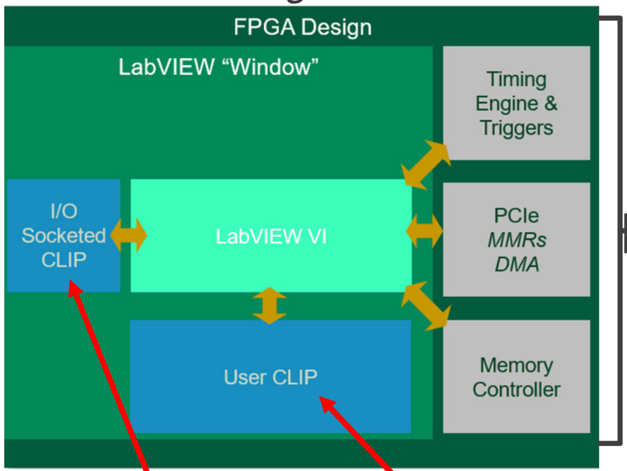


Aurora Stream.lvproj - Project Explorer
File Edit View Project Operate Tools Window Help
Items
Files
New
Find Project Items...
Arrange By
Expand All
Collapse All
Help...
Compilation
Project Export for Vivado
Source Distribution

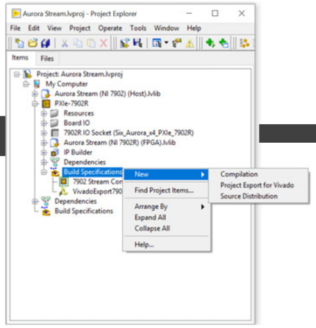
- Project: Aurora Stream.lvproj
  - My Computer
    - Aurora Stream (NI 7902) (Host).lvlib
    - PXIe-7902R
      - Resources
      - Board IO
      - 7902R IO Socket (Six_Aurora_x4_PXIe_7902R)
      - Aurora Stream (NI 7902R) (FPGA).lvlib
      - IP Builder
      - Dependencies
      - Build Specifications
        - 7902 Stream Cor
        - VivadoExport790
    - Dependencies
    - Build Specifications

How Does Vivado Project Export Work?

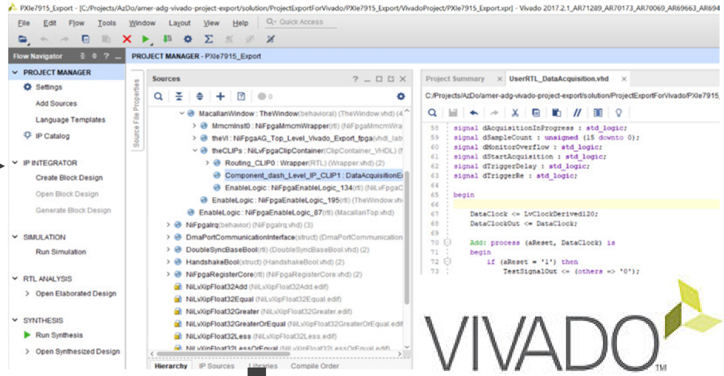
FPGA Design



Export Design

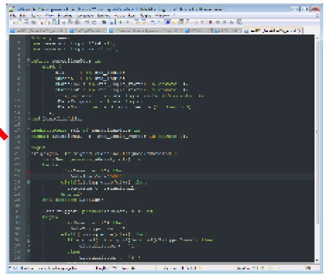

Aurora Stream.lvproj - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Aurora Stream.lvproj
My Computer
Aurora Stream (NI 7902) (Host).lvlib
PXIe-7902R
Resources
Board IO
7902R IO Socket (Six_Aurora_x4_PXIe_7902R)
Aurora Stream (NI 7902R) (FPGA).lvlib
IP Builder
Dependencies
Build Specifications
7902 Stream Con
VivadoExport790
Dependencies
Build Specifications
New
Find Project Items...
Arrange By
Expand All
Collapse All
Help...
Compilation
Project Export for Vivado
Source Distribution

Develop, Simulate, Compile

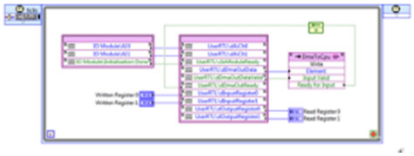

PXIe7915_Export - [C:/Projects/AzDo/amer-adg-vivado-project-export/solution/ProjectExportForVivado/PXIe7915_Export/VivadoProject/PXIe7915_Export.xpr] - Vivado 2017.2.1_AR71289_AR70173_AR70069_AR69663_AR694
File Edit Flow Tools Window Layout View Help
Quick Access
Flow Navigator
PROJECT MANAGER - PXIe7915_Export
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
Sources
MacallanWindow : TheWindow(behavioral) (TheWindow.vhd) (4
MmcmInst0 : NiFpgaMmcmWrapper(rtl) (NiFpgaMmcmWra
theVI : NiFpgaAG_Top_Level_Vivado_Export_fpga(vhdl_lab
theCLIPs : NiLvFpgaClipContainer(ClipContainer_VHDL) (f
Routing_CLIP0 : Wrapper(RTL) (Wrapper.vhd) (2)
Component_dash_Level_IP_CLIP1 : DataAcquisitionE
EnableLogic : NiFpgaEnableLogic_134(rtl) (NiLvFpgaC
EnableLogic : NiFpgaEnableLogic_195(rtl) (TheWindow.vh
EnableLogic : NiFpgaEnableLogic_87(rtl) (MacallanTop.vhd)
NiFpgaIrq(behavior) (NiFpgaIrq.vhd) (3)
DmaPortCommunicationInterface(struct) (DmaPortCommunication
DoubleSyncBaseBool(rtl) (DoubleSyncBaseBool.vhd) (2)
HandshakeBool(struct) (HandshakeBool.vhd) (2)
NiFpgaRegisterCore(rtl) (NiFpgaRegisterCore.vhd) (2)
NiLvXipFloat32Add (NiLvXipFloat32Add.edif)
NiLvXipFloat32Equal (NiLvXipFloat32Equal.edif)
NiLvXipFloat32Greater (NiLvXipFloat32Greater.edif)
NiLvXipFloat32GreaterOrEqual (NiLvXipFloat32GreaterOrEqual.edif
NiLvXipFloat32Less (NiLvXipFloat32Less.edif)
Hierarchy IP Sources Libraries Compile Order
Project Summary
UserRTL_DataAcquisition.vhd
C:/Projects/AzDo/amer-adg-vivado-project-export/solution/ProjectExportForVivado/PXIe7915
VIVADO™

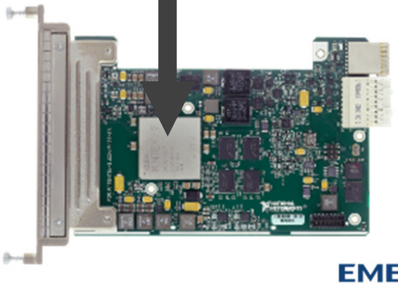
```
signal dAcquisitionInProgress : std_logic;
signal dSampleCount : unsigned (15 downto 0);
signal dMonitorOverflow : std_logic;
signal dStartAcquisition : std_logic;
signal dTriggerDelay : std_logic;
signal dTriggerRe : std_logic;

begin

   DataClock <= LvClockDerived120;
   DataClockOut <= DataClock;

   Add: process (aReset, DataClock) is
   begin
      if (aReset = '1') then
         TestSignalOut <= (others => '0');
```

User CLIP



“Board Support” IP



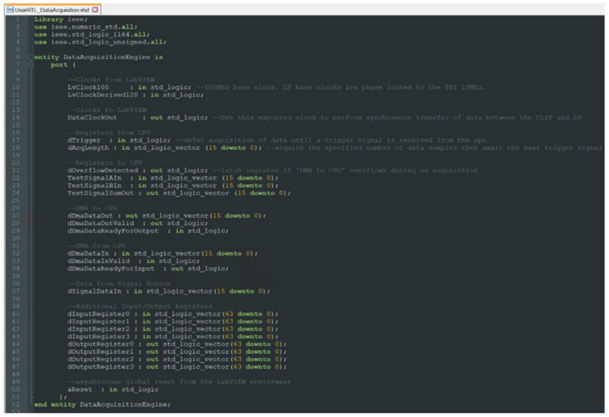


Deploy to Hardware

EMERSON. | NI

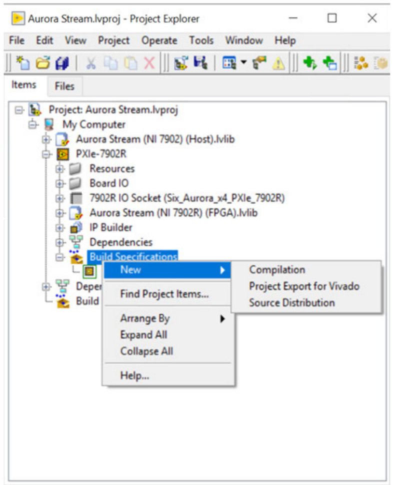
Creating a Vivado Project Export

Create User CLIP shell code

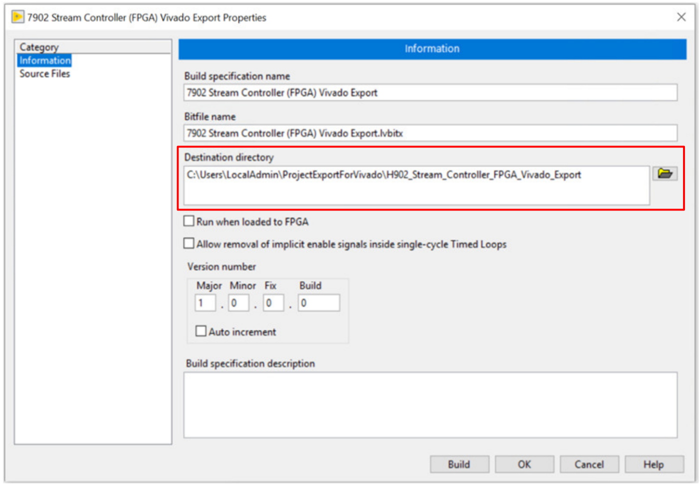

UserRTL_DataAcquisition.vhd

```vhdl
Library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity DataAcquisitionEngine is
    port (

        --Clocks from LabVIEW
        LvClock100     : in std_logic; --100MHz base clock. LV base clocks are phase locked to the PXI 10MHz.
        LvClockDerived120 : in std_logic;

        --Clocks to LabVIEW
        DataClockOut      : out std_logic; --Use this exported clock to perform synchronous transfer of data between the CLIP and LV

        --Registers from CPU
        dTrigger  : in std_logic; --defer acquisition of data until a trigger signal is received from the cpu
        dAcqLength : in std_logic_vector (15 downto 0); --acquire the specified number of data samples then await the next trigger signal

        --Registers to CPU
        dOverflowDetected : out std_logic; --latch register if 'DMA to CPU' overflows during an acquisition
        TestSignalAIn  : in std_logic_vector (15 downto 0);
        TestSignalBIn  : in std_logic_vector (15 downto 0);
        TestSignalSumOut : out std_logic_vector (15 downto 0);

        --DMA to CPU
        dDmaDataOut : out std_logic_vector(15 downto 0);
        dDmaDataOutValid  : out std_logic;
        dDmaDataReadyForOutput  : in std_logic;

        --DMA from CPU
        dDmaDataIn : in std_logic_vector(15 downto 0);
        dDmaDataInValid  : in std_logic;
        dDmaDataReadyForInput  : out std_logic;

        --Data from Signal Source
        dSignalDataIn : in std_logic_vector(15 downto 0);

        --Additional Input/Output Registers
        dInputRegister0 : in std_logic_vector(63 downto 0);
        dInputRegister1 : in std_logic_vector(63 downto 0);
        dInputRegister2 : in std_logic_vector(63 downto 0);
        dInputRegister3 : in std_logic_vector(63 downto 0);
        dOutputRegister0 : out std_logic_vector(63 downto 0);
        dOutputRegister1 : out std_logic_vector(63 downto 0);
        dOutputRegister2 : out std_logic_vector(63 downto 0);
        dOutputRegister3 : out std_logic_vector(63 downto 0);

        --asychronous global reset from the LabVIEW enviorment
        aReset  : in std_logic
      );
end entity DataAcquisitionEngine;
```

Create Build Specification



Configure and Build


7902 Stream Controller (FPGA) Vivado Export Properties

Build specification name: 7902 Stream Controller (FPGA) Vivado Export

Bitfile name: 7902 Stream Controller (FPGA) Vivado Export.lvbitx

Destination directory: C:\Users\LocalAdmin\ProjectExportForVivado\H902_Stream_Controller_FPGA_Vivado_Export

□ Run when loaded to FPGA

□ Allow removal of implicit enable signals inside single-cycle Timed Loops

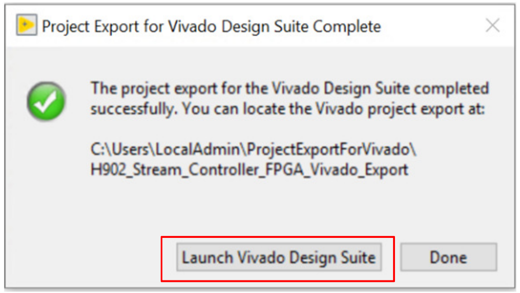
Version number

Major	Minor	Fix	Build
1	0	0	0

□ Auto increment

Build specification description

Launch Vivado


Project Export for Vivado Design Suite Complete

The project export for the Vivado Design Suite completed successfully. You can locate the Vivado project export at:

C:\Users\LocalAdmin\ProjectExportForVivado\
H902_Stream_Controller_FPGA_Vivado_Export

LabVIEW FPGA IP Export Utility

Overview

- Allow export of LabVIEW FPGA IP to netlist/VHDL, and reuse IP on **non-NI platform**.
- Full support for this functionality was started in LabVIEW 2020.

Features

Two ways to export the IP

Export to Encrypted Netlist (.dcp)
Export to Plaintext (.vhd, and RTL style)

Supported on all NI (Vivado) hardware by default

Workflow

1. Research, design, and develop IP in LabVIEW FPGA

2. Validate results (simulation and HW) using LabVIEW FPGA

3. Export Design via LabVIEW FPGA IP Export Utility

4. Incorporate the exported netlist into Vivado design
   - Run Synthesis
   - Run Implementation
   - Generate Bitstream

5. Leverage exported IP in overall design

Considerations

- Support first introduced in LabVIEW 2020.
- No automatic 4-wire/AXI support
- The NI non-FPGA target family needs to match what the IP Export was built against.
  - E.g., NI-7915 has an FPGA from the Kintex-Ultrascale family

LabVIEW FPGA Observations

LabVIEW FPGA – Observations

Strengths

- Domain experts can program FPGAs
- LabVIEW skills expand into FPGAs
- Graphical environment matches spatial aspect of FPGAs
- Some skill portability across NI FPGA product lines
- Peer to Peer (P2P) to/from other NI instruments
- Specialized and diverse applications

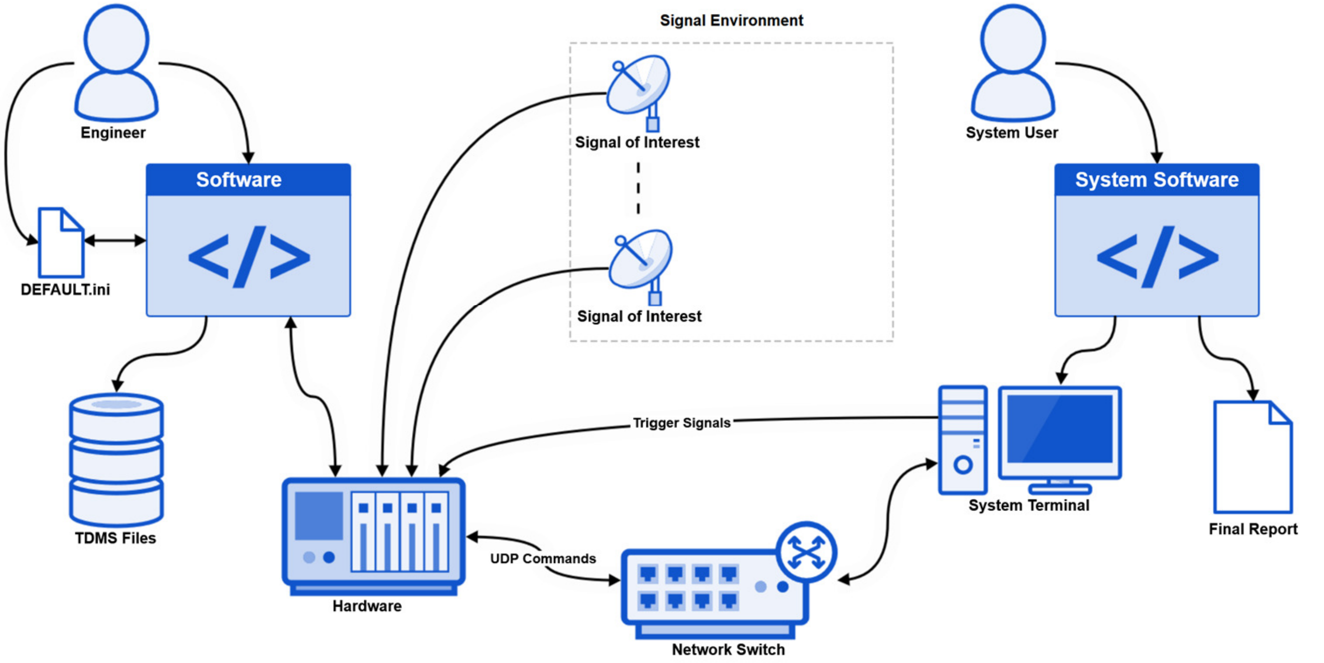
Weaknesses

- Community has not hit critical mass
- Training is challenging
  - User background, NI FPGA product, Application
- Primarily for NI FPGA hardware
- No System on Chip (SoC) support
- Some solutions can have high complexity

Case Studies

Reconfigurable IQ Digitizer

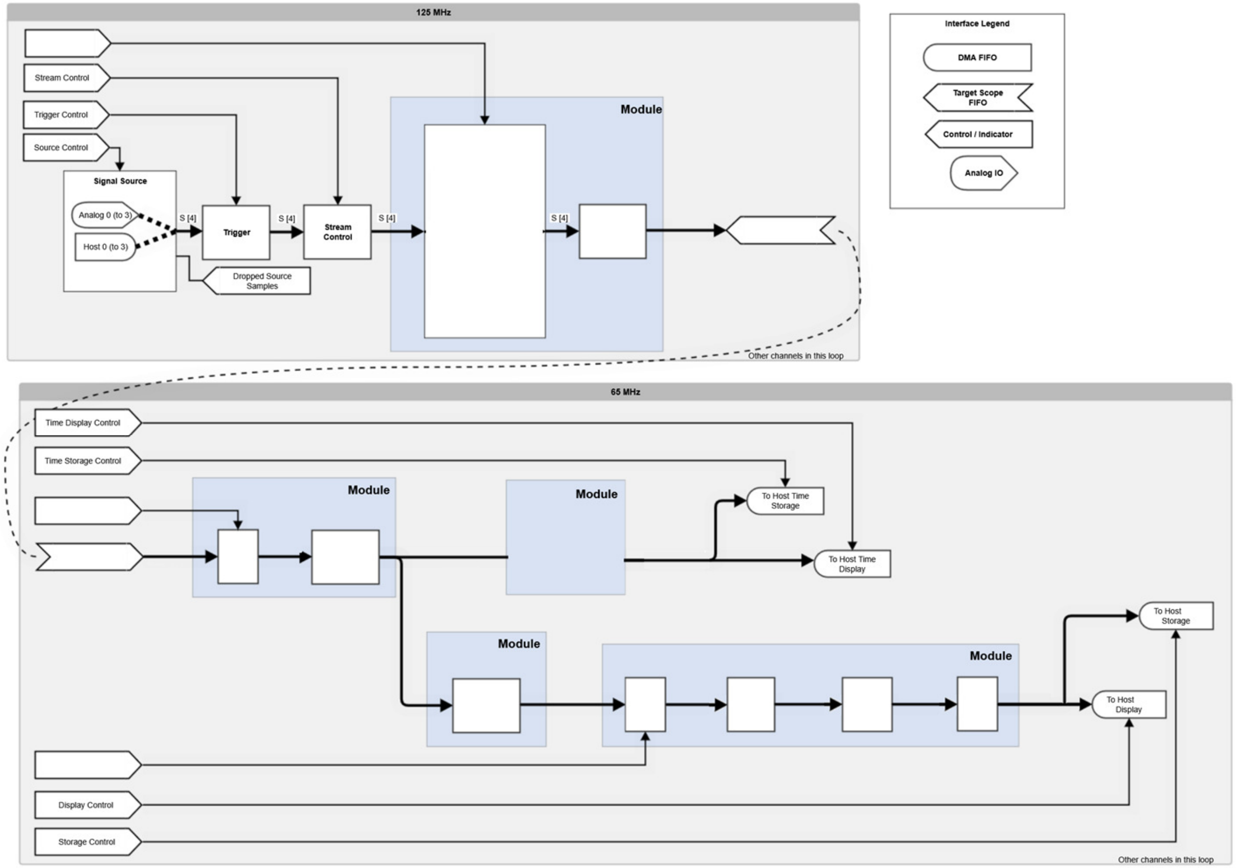
Reconfigurable IQ Digitizer – System Diagram



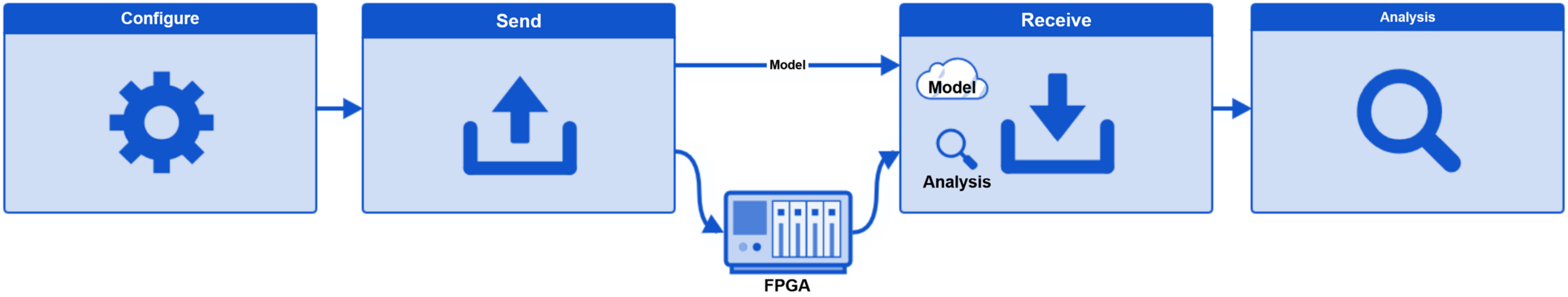
Reconfigurable IQ Digitizer – Excerpt 1

Situation	The functional data flow through DSP components is known. The specific DSP configuration and technical trade-offs are unknown and require evaluation.
Action	Develop template to support DSP IP component research.
Result	Multiple models as a simulated digital twin and FPGA bitfile could be studied independently. DSP IP component integration risk reduced.

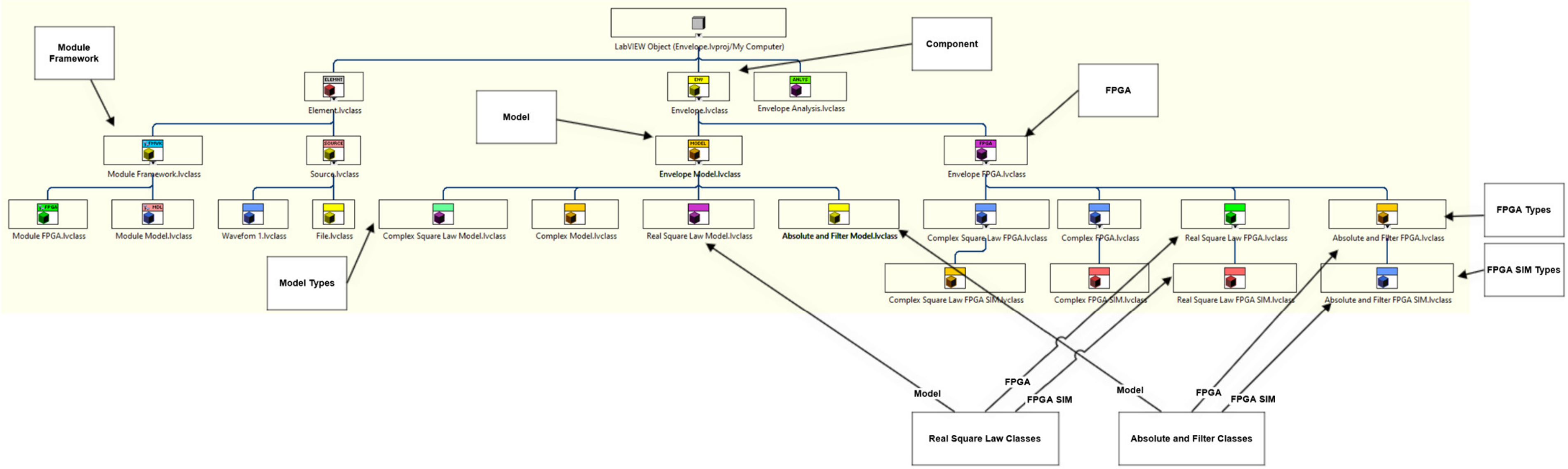
FPGA Functional Block Diagram – Situation



IP Development Template – Action



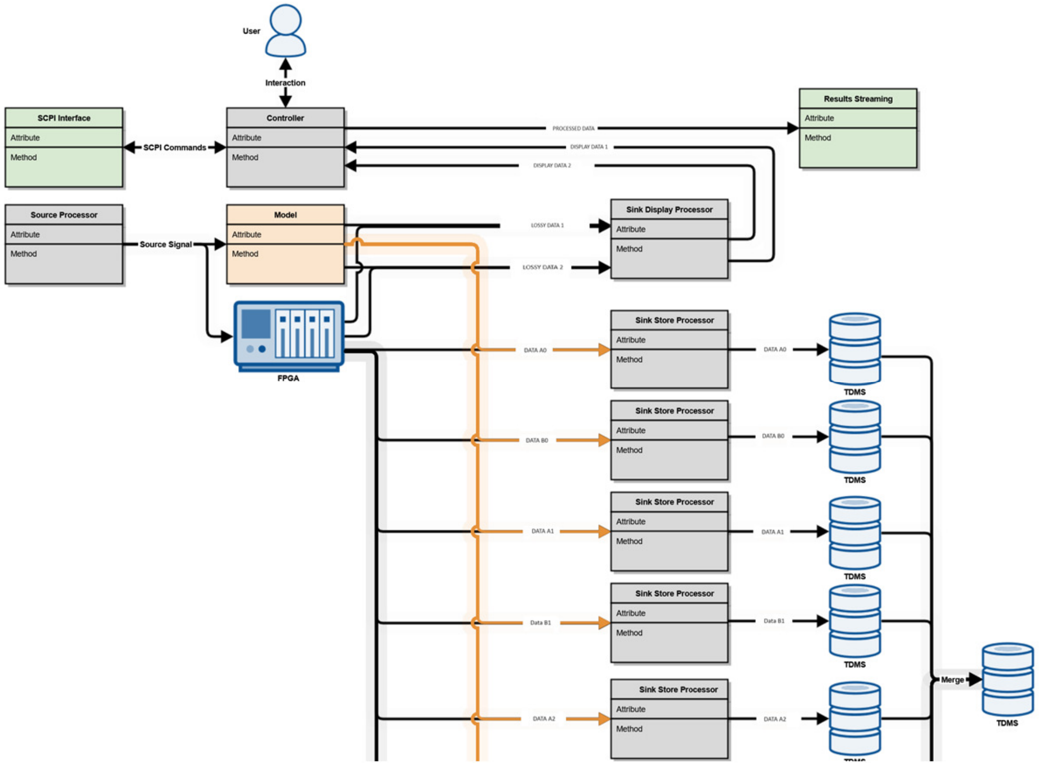
IP Development Template – Class Hierarchy



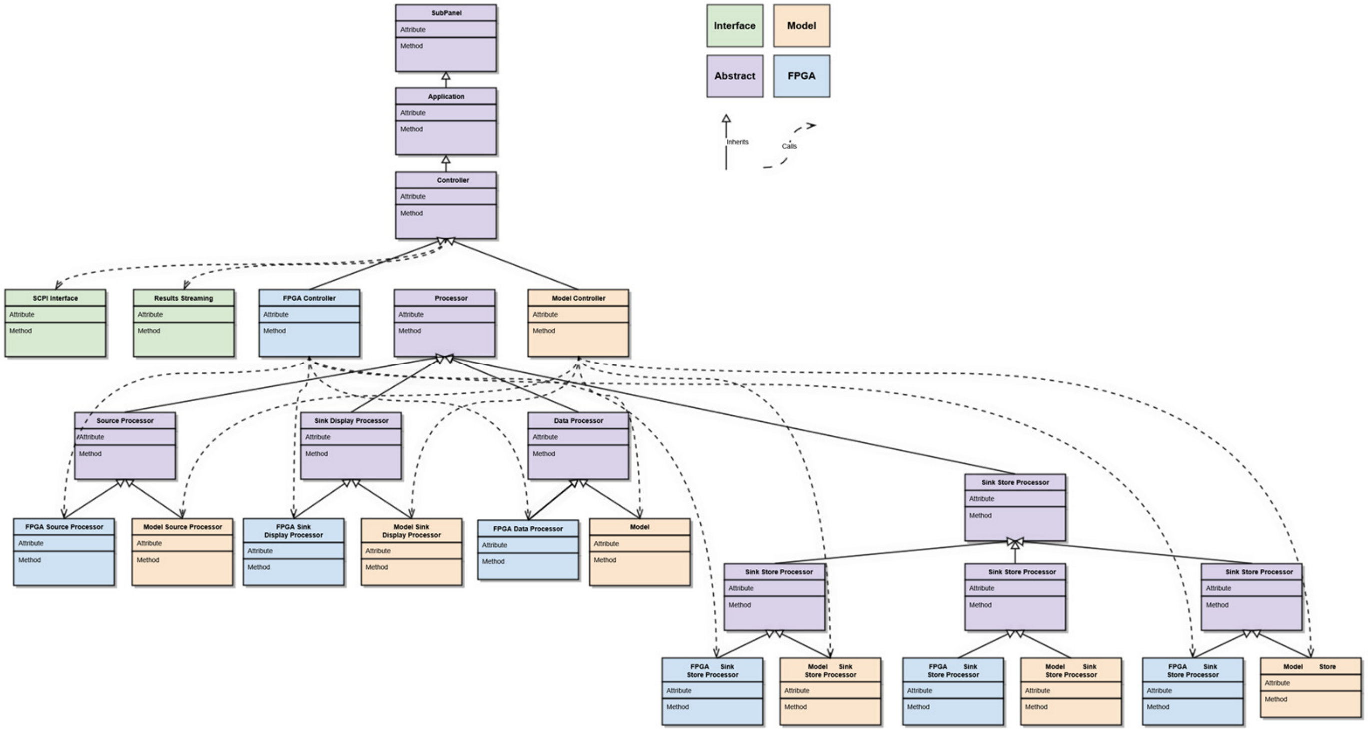
Reconfigurable IQ Digitizer – Excerpt 2

Situation	As part of a larger test system, there is a need for a multi-channel data acquisition system with custom DSP data reduction.
Action	Use NI Actor Framework to support substitution of simulated digital twin and hardware processes.
Result	Ability to integrate product into system prior to full FPGA implementation.

Software Architecture & Dataflow

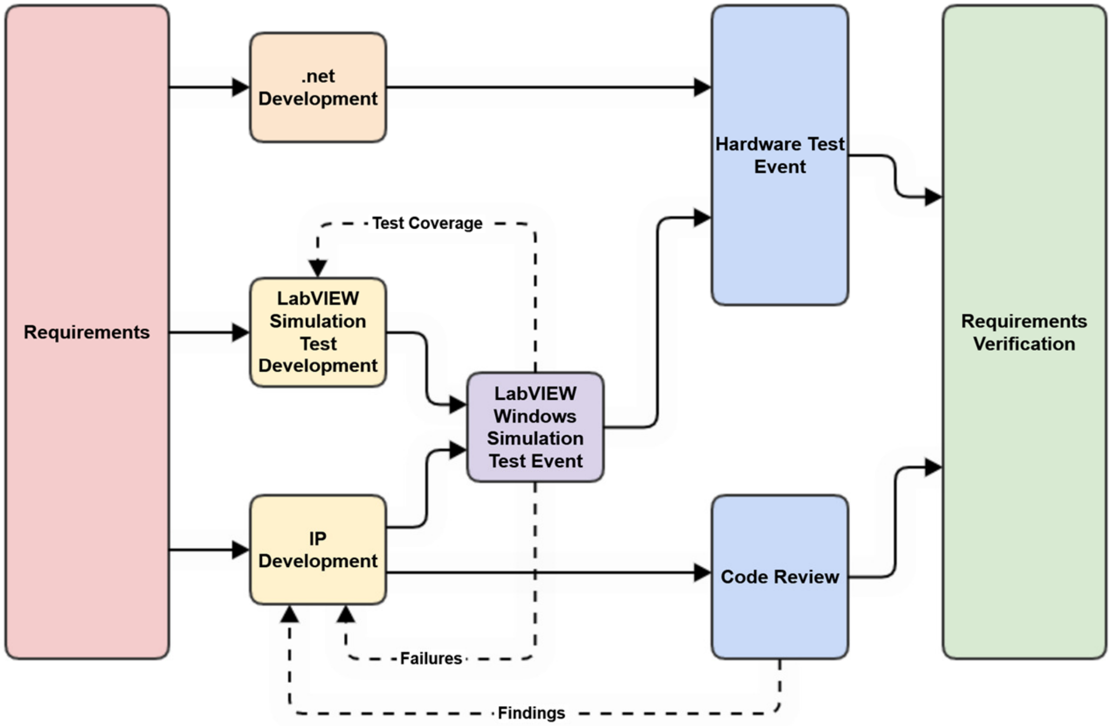


Actor Call Chain and Hierarchy



Digital Product Tester

Digital Product Tester – Workflow



Requirements

.net Development

LabVIEW Simulation Test Development

IP Development

LabVIEW Windows Simulation Test Event

Hardware Test Event

Code Review

Requirements Verification

Test Coverage

Failures

Findings

Digital Product Tester – Excerpt

Situation	Firmware is needed to support over 350 product requirements.
Action	Test plan written. Developed 50 simulated tests that covered 270 of the requirements.
Result	Simulated tests saved test time and hardware costs. Significant reduction of integration risk. High pass rate of tests when running with hardware.

Product IP Test Workflow Optimization

EMERSON. | ni

Product IP Test Workflow Optimization

Situation	Product IP test team is looking for optimal workflow.
Action	Reviewed workflow options including: 1. Simulation Export 2. Vivado Export 3. Simulation (Simulated I/O)
Result	Changes to VHDL can be added by any Test Engineer

Product IP Test Workflow Analysis

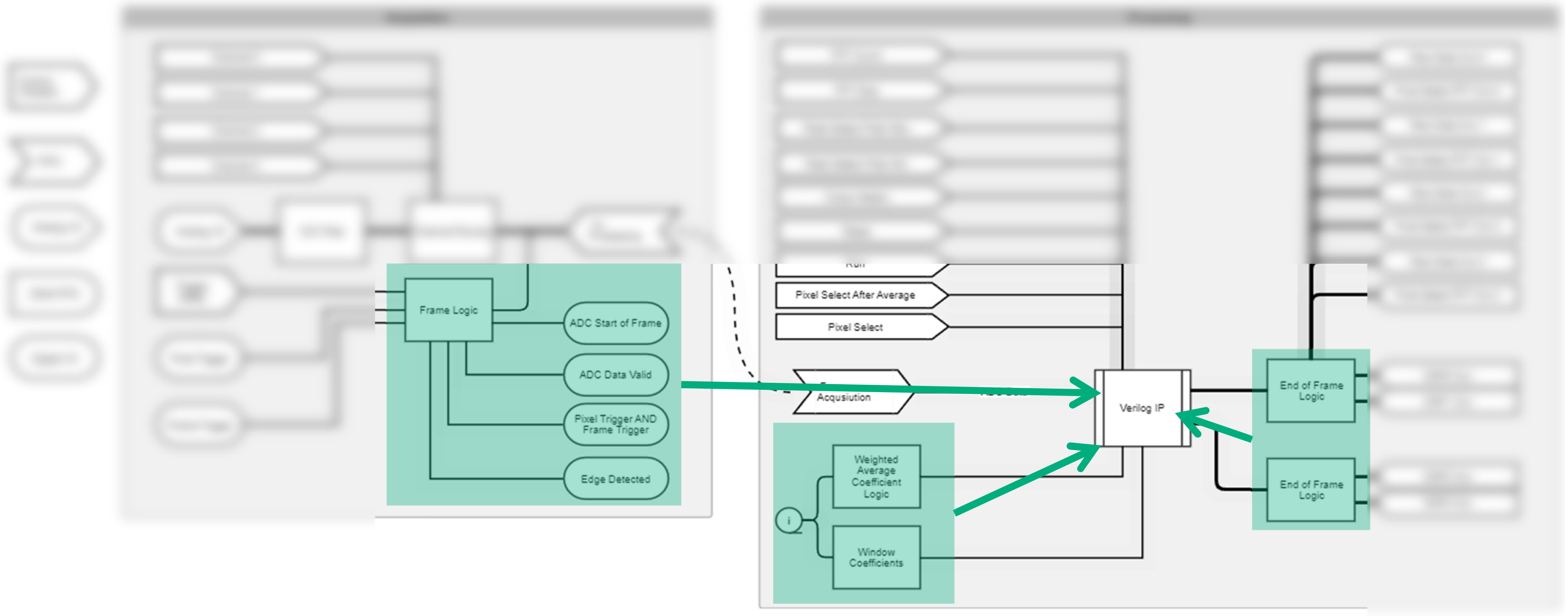
Workflow	Test Engineer Required?	Firmware Engineer Required?	
Simulation Export	Yes	Yes	☒
Vivado Export	Yes	Yes	☒
Simulation (Simulated I/O)	Yes	No	☑

LabVIEW FPGA CLIP Interface Simplification

LabVIEW FPGA CLIP Interface Simplification

Situation	LabVIEW FPGA that wrapped the Component Level IP running Verilog had excess interface logic. This made Verilog integration is labor intensive.
Action	Work with Verilog developer to update Interface Control Document (ICD) to encapsulated the interface logic. Developed Verilog integration procedure.
Results	Reduced Verilog LabVIEW FPGA CLIP integration risks.

LabVIEW FPGA CLIP Interface Simplification



Case Studies Review

- Reconfigurable IQ Digitizer
- Digital Product Tester
- Product IP Test Workflow Optimization
- LabVIEW FPGA CLIP Simplification

Trends

NI FPGA Trends

Developer Background of NI FPGAs

Scenario	More Vivado/HDL developers using NI FPGAs
Proposed Action	Make sure interfaces are defined • What goes into LabVIEW FPGA, CLIP, and/or IPIN?
Forecast	Integration risk is reduced • Access to more open, licensed, and proprietary IP

More High-Speed Serial on NI FPGA Products

Scenario	More HSS/MGT interfacing on NI products
Proposed Action	Development of tooling and examples
Forecast	More NI FPGA products in systems • Better developer experience (e.g., NI FPGA P2P)

CI/CD in NI FPGA workflows

Scenario	CI/CD needed in NI FPGA workflows
Proposed Action	Development of tooling and examples
Forecast	More efficient and standardized workflows

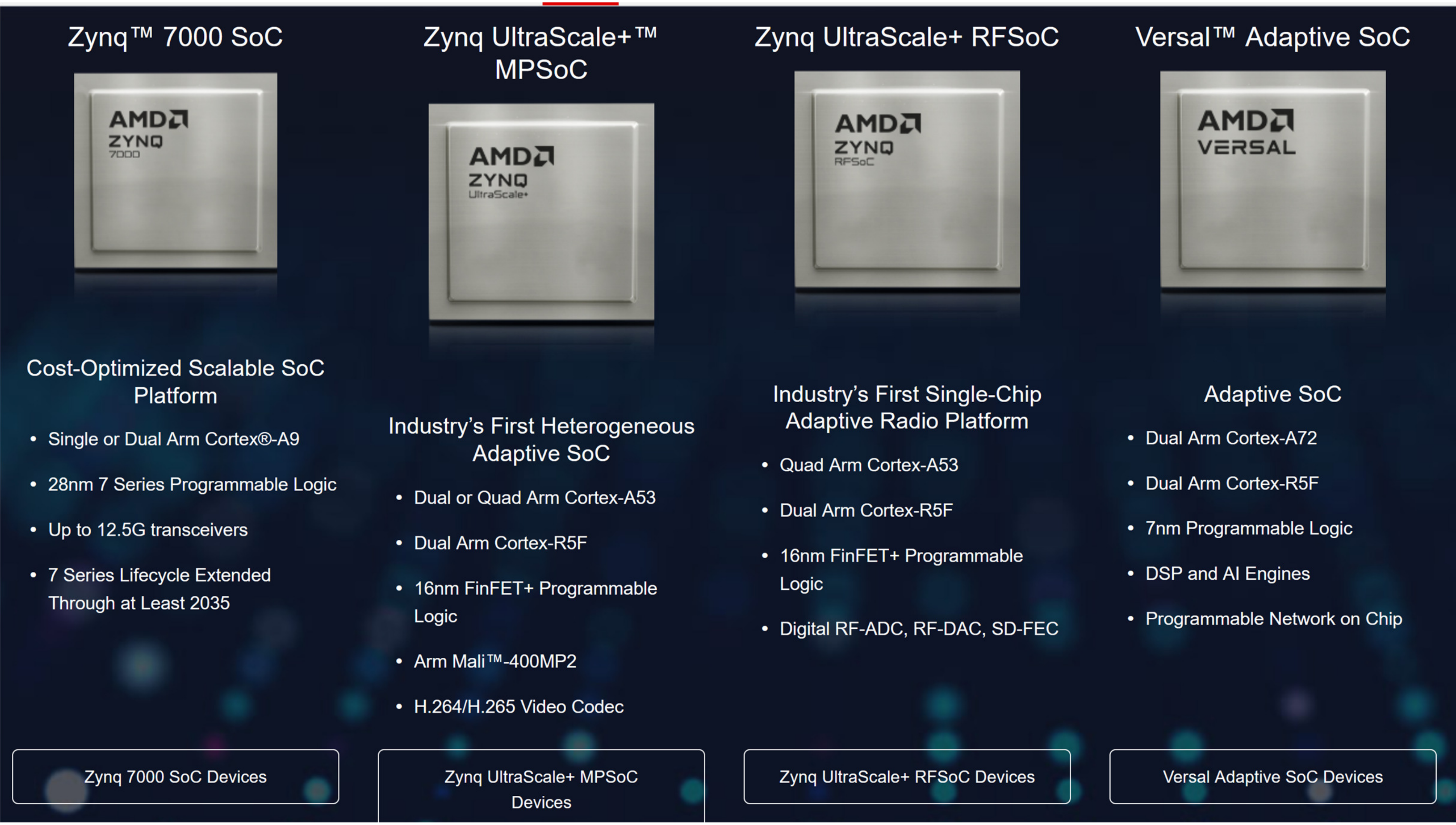
Varied Host Interface Languages for NI FPGA

Scenario	Increase in non-LabVIEW host interfacing • e.g., C, C#, Python
Proposed Action	Advocating of best practices in workflows
Forecast	Wider use of NI FPGA products • More accessible for developers

FPGA Trends

Chips Getting Larger and More Complex

Scenario	Chips getting larger and more complex • e.g., System on Chip (SoC)
Proposed Action	More standard tools & techniques • Systems Engineering • Software Engineering
Forecast	Larger teams could benefit from more structure



Zynq™ 7000 SoC

Cost-Optimized Scalable SoC Platform

- Single or Dual Arm Cortex®-A9
- 28nm 7 Series Programmable Logic
- Up to 12.5G transceivers
- 7 Series Lifecycle Extended Through at Least 2035

Zynq 7000 SoC Devices

Zynq UltraScale+™ MPSoC

Industry's First Heterogeneous Adaptive SoC

- Dual or Quad Arm Cortex-A53
- Dual Arm Cortex-R5F
- 16nm FinFET+ Programmable Logic
- Arm Mali™-400MP2
- H.264/H.265 Video Codec

Zynq UltraScale+ MPSoC Devices

Zynq UltraScale+ RFSoC

Industry's First Single-Chip Adaptive Radio Platform

- Quad Arm Cortex-A53
- Dual Arm Cortex-R5F
- 16nm FinFET+ Programmable Logic
- Digital RF-ADC, RF-DAC, SD-FEC

Zynq UltraScale+ RFSoC Devices

Versal™ Adaptive SoC

Adaptive SoC

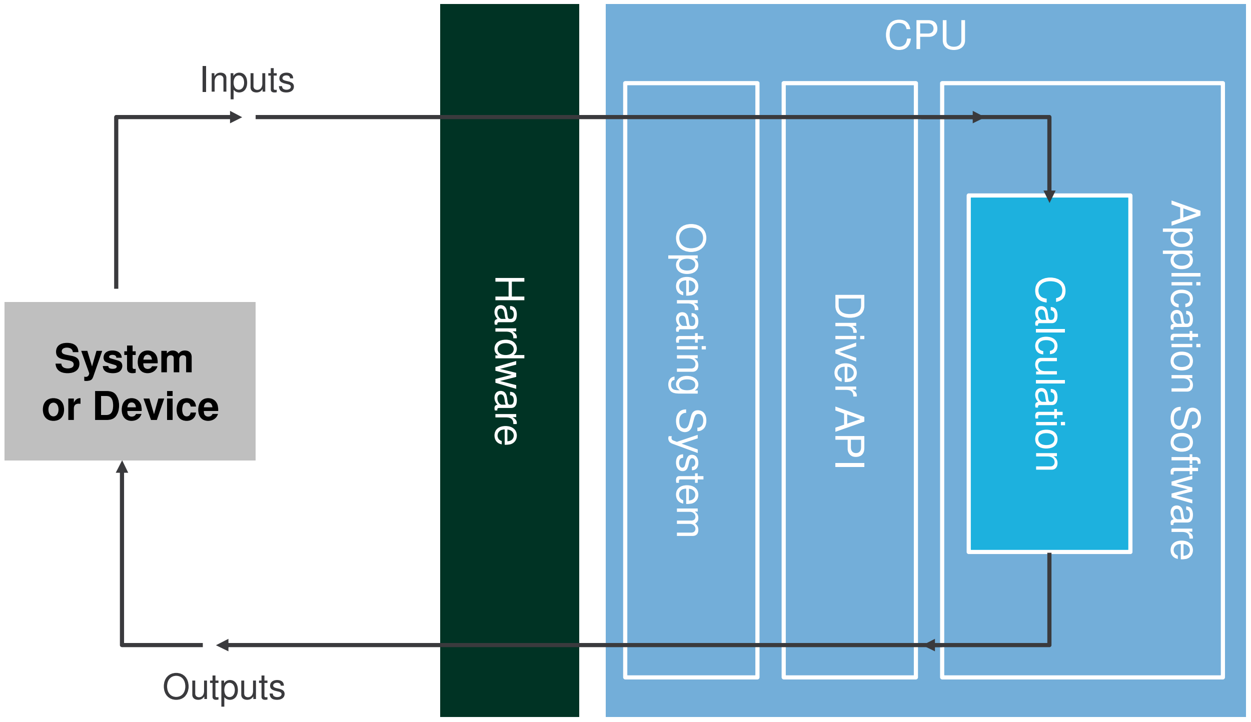
- Dual Arm Cortex-A72
- Dual Arm Cortex-R5F
- 7nm Programmable Logic
- DSP and AI Engines
- Programmable Network on Chip

Versal Adaptive SoC Devices

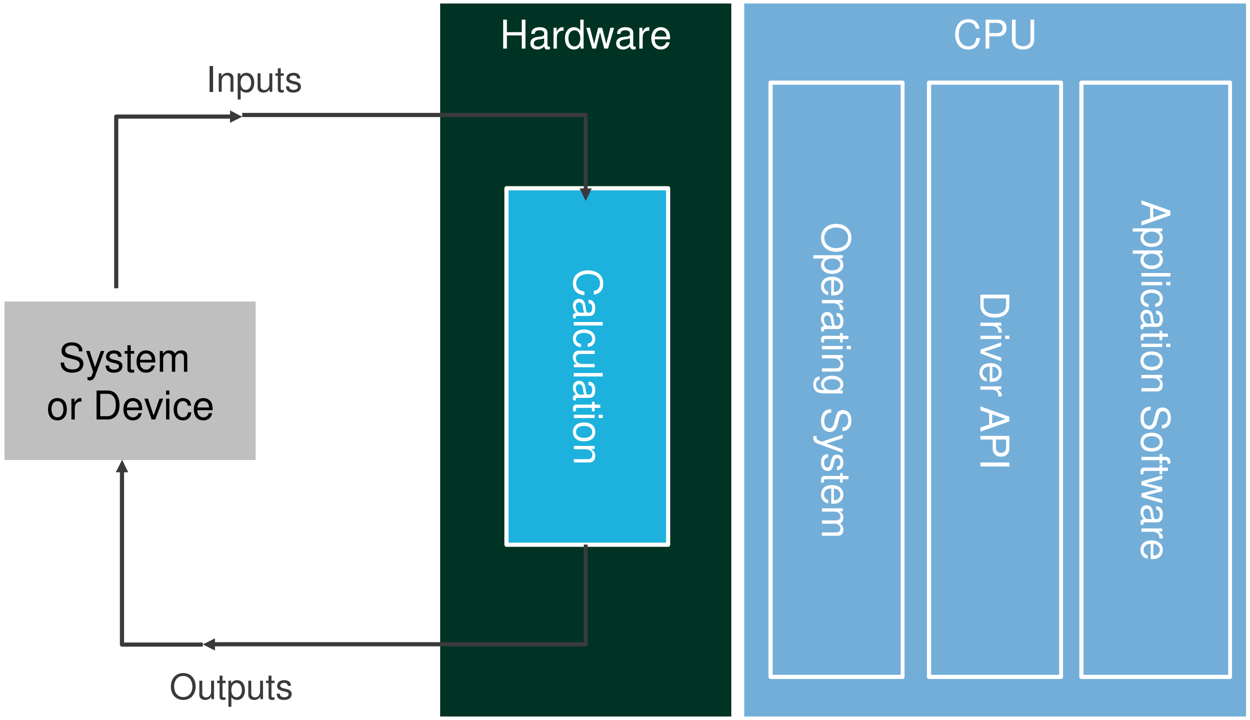
Datacenters Power Utilization

Scenario	Datacenters utilizing CPUs and GPUs in power hungry algorithms such as LLMs
Proposed Action	Development of domain specific architectures Find areas where FPGAs could be utilized
Forecast	More efficient computing

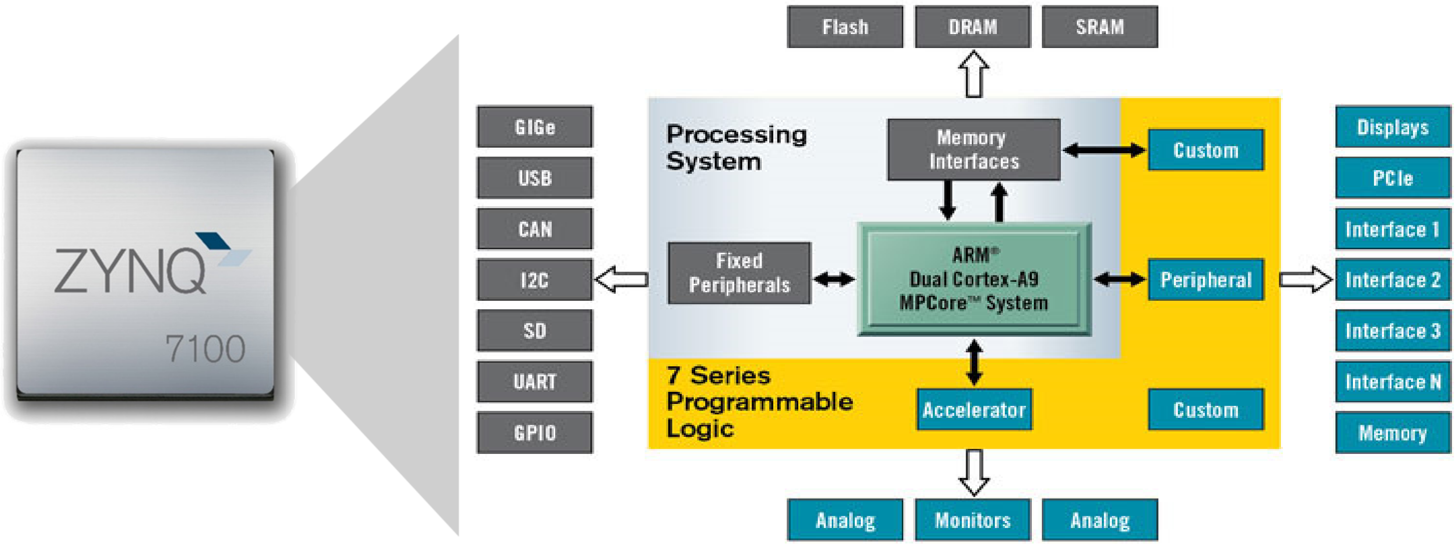
Processor Based Approach



Decision Making in FPGA Hardware



The Future of FPGAs: Heterogeneous, Massively Parallel SOCs



- Reduced power consumption
- Smaller overall footprint
- Improved re-configurability
- Lower Cost

Image Source Xilinx: Xilinx_Zynq-7000_AP_SoC.jpg

Future of NI FPGA

NI

HSS is near on all new NI FPGA products – it is like network connectivity on a computer – see 7903, see

Veristand and Matlab and LabVIEW FPGA - https://www.ni.com/en/support/documentation/supplemental/20/matlab---simulink---and-labview-fpga--importing-hdl-coder--expor.html

See 7890/1 (note the QSP+)

https://github.com/ni/hdlcoder-support-package-for-nifpga-hardware#hdl-coder-support-package-for-ni-fpga-hardware
https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q0000019fsLCAQ&l=en-US

https://www.mathworks.com/help/hdlcoder/gs/generate-hdl-code-from-matlab-code-using-the-command-line-interface.html;jsessionid=e486e1592f8ba9ae6604c29efa8e

FlexRIO

- Target Applications
  - High speed data converters
  - Custom digital interfacing
  - Real time digital signal processing

- Key Benefits
  - Fully configurable
  - High-speed analog, digital, and RF I/O
  - Uses timing and synchronization capabilities of PXI
    - Synchronize Multiple Modules

FlexRIO Adapter Modules

Digital

100 Mbps
SE DIO

300 Mbps
LVDS DIO

300 Mbps
SE/LVDS DIO

1 Gbps
LVDS DIO

Camera Link

RS-485/422

Digitizers

2 ch. 3 GS/s
8-bit AI

2 ch. 1.6 GS/s
12-bit AI

4 ch. 250 MS/s
14-bit AI

2 ch. 250 MS/s
16-bit AI

16 ch. 120 MS/s
16-bit AI

4 ch. 120 MS/s
16-bit AI

32 ch. 50 MS/s
12-bit AI

2 ch. 80 MS/s
14-bit AI

2 ch. 120 MS/s
16-bit AI

2 ch. 40 MS/s
12-bit AI

16 ch. 50 MS/s,
14-bit AI

RF

100MHz BW
4.4 GHz RF I/O

200MHz BW
4.4 GHz RF Tx

200MHz BW
4.4 GHz RF Rx

Transceivers

2 ch. 100 MS/s
14-bit AI
16-bit AO

2 ch. 250 MS/s
14-bit AI
16-bit AO

4 ch. 100 MS/s
16-bit AI
16-bit AO

Signal Generators

2 ch. 1.25 GS/s
14-bit AO

1 ch. 2 GS/s
14-bit AO

32 ch. 1MS/s
16-bit AO

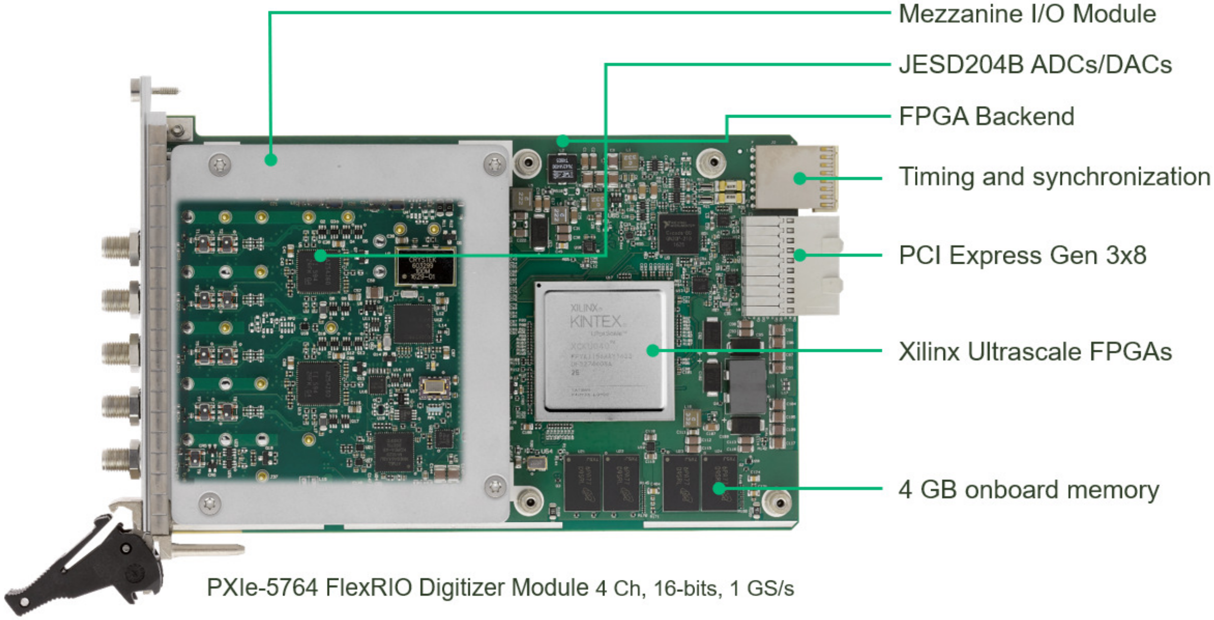
16 ch. 1MS/s
16-bit AO

FlexRIO Architectures

High-Speed Serial Converters, Integrated I/O

- Second FlexRIO architecture
- Mezzanine I/O module communicates with FPGA via high-speed serial communication
- Xilinx UltraScale FPGAs

- JESD204B interface standard
  - Supports high bandwidth, high performance, high speed, and multi-channel applications



PXIe-5764 FlexRIO Digitizer Module 4 Ch, 16-bits, 1 GS/s

NI Reconfigurable Oscilloscopes

Model	Channels	Max Bandwidth	Max. Sample Rate	AI Voltage Range	Onboard Memory
PXIe-5164	2	400MHz	1GS/s	-50V-50V	1.5GB
PXIe-5170	4/8	100MHz	250MS/s	-2.5V-2.5V	0.75/1.5GB
PXIe-5171	8	250MHz	250MS/s	-2.5V-2.5V	1.5GB
PXIe-5172	4/8	100MHz	250MS/s	-40V-40V	0.75/1.5GB

PXIe-5164

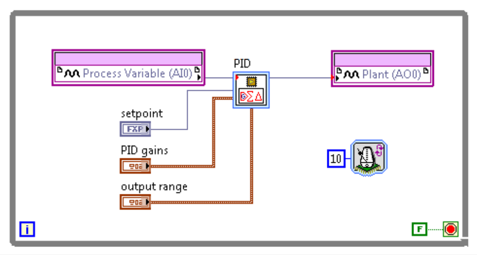
PXIe-5170

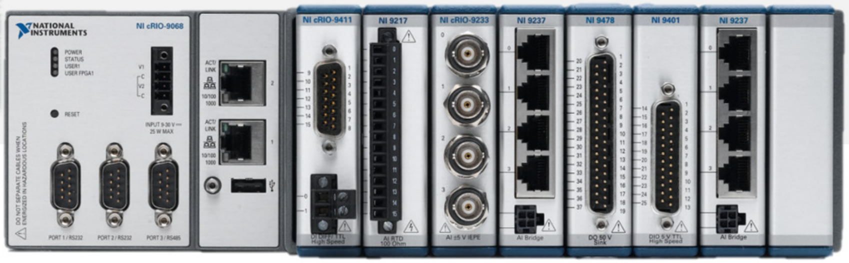
PXIe-5171

PXIe-5172

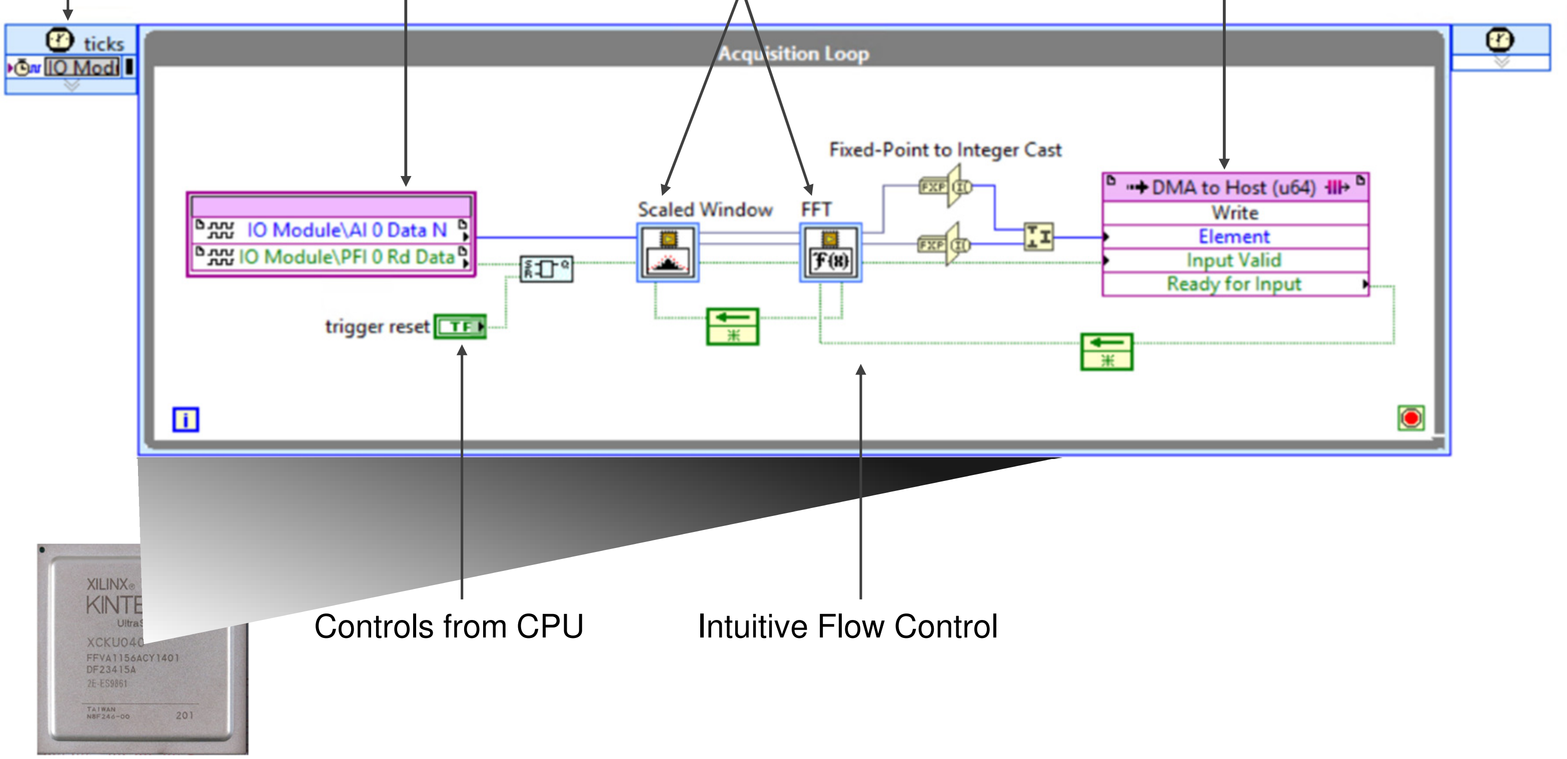
LabVIEW FPGA Module

- Use LabVIEW to design hardware
- Offload the most critical pieces of your application
  - High speed control
  - Inline signal processing
  - Custom protocols
  - Custom timing, triggering, and synchronization
  - Fast stimulus/response testing





A *Quick* Look at LabVIEW FPGA



Clock Constrains

I/O Signals

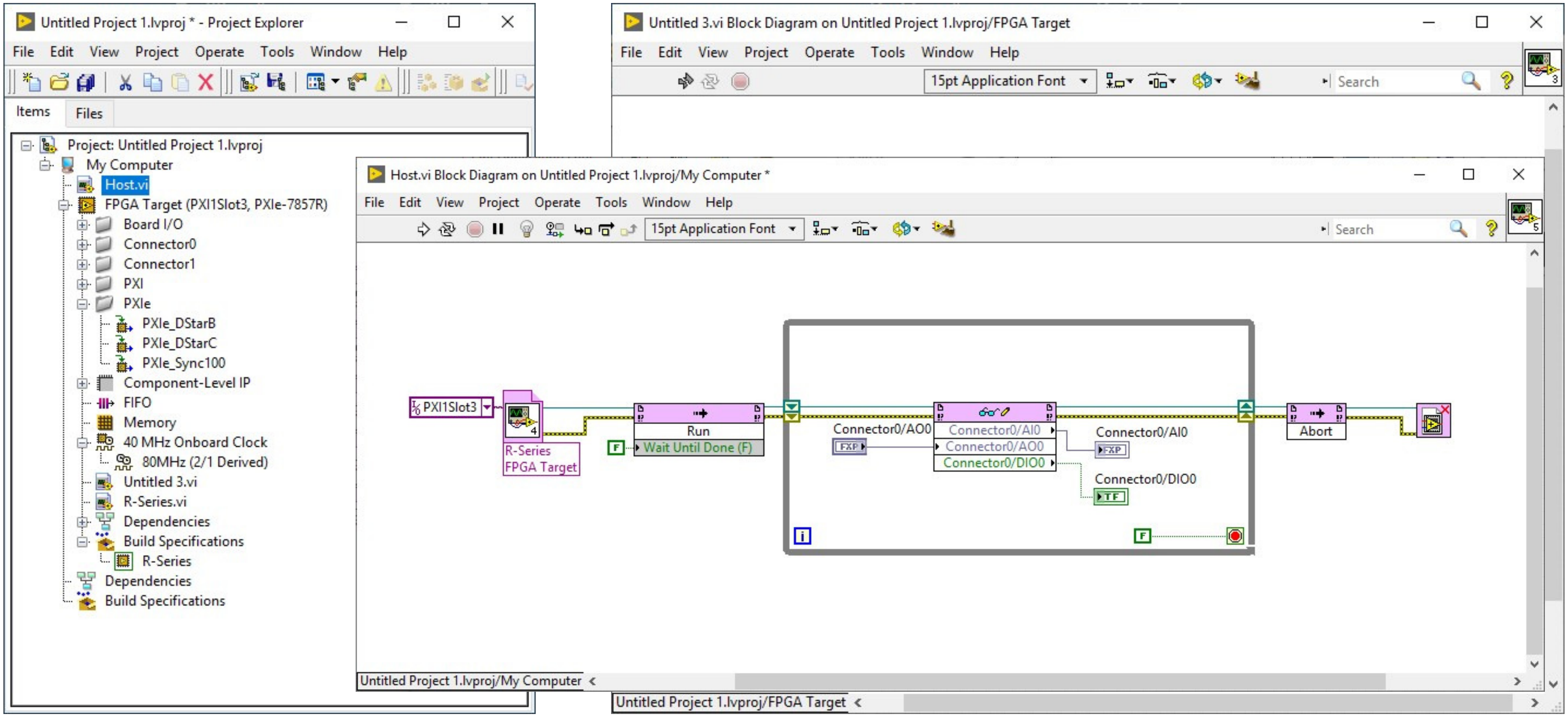
Signal Processing Functions

DMA over PCIe

Controls from CPU

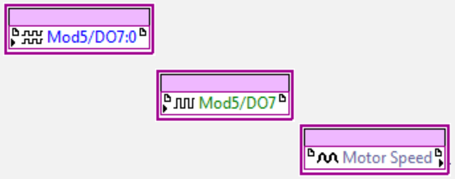
Intuitive Flow Control

LabVIEW Interface

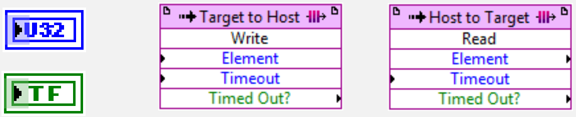


LabVIEW FPGA Elements

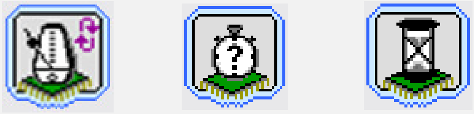
I/O Interface



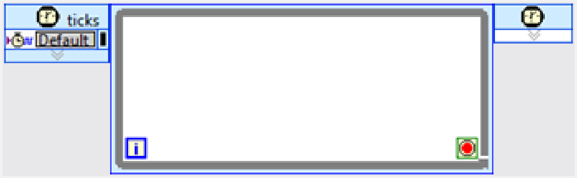
Data Communication



Timing

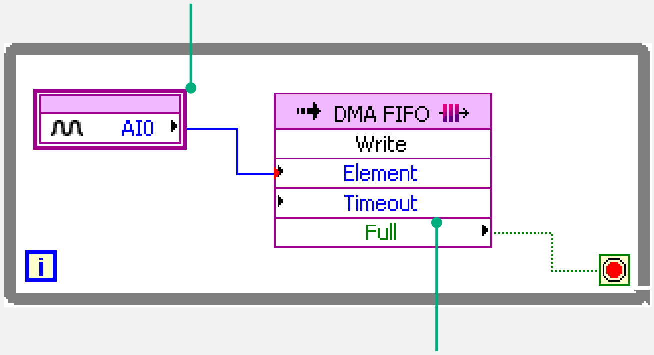


Control

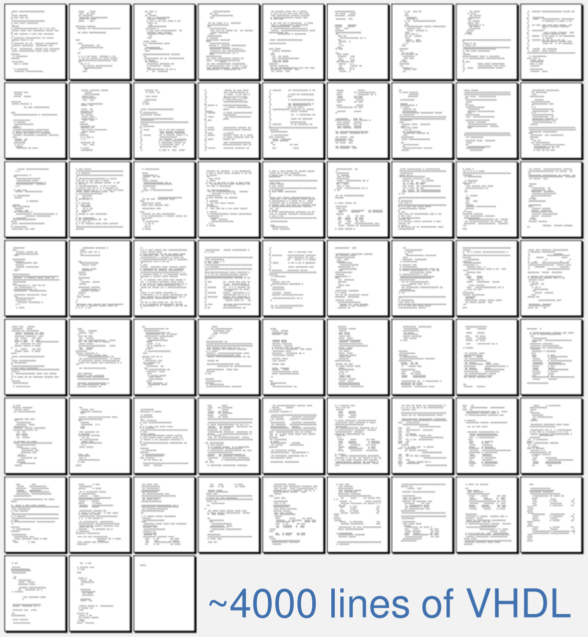


Abstraction of Hardware Complexities

Acquire analog data point-by-point



Directly transfer analog data to processor memory via FIFO for data logging, display, etc.



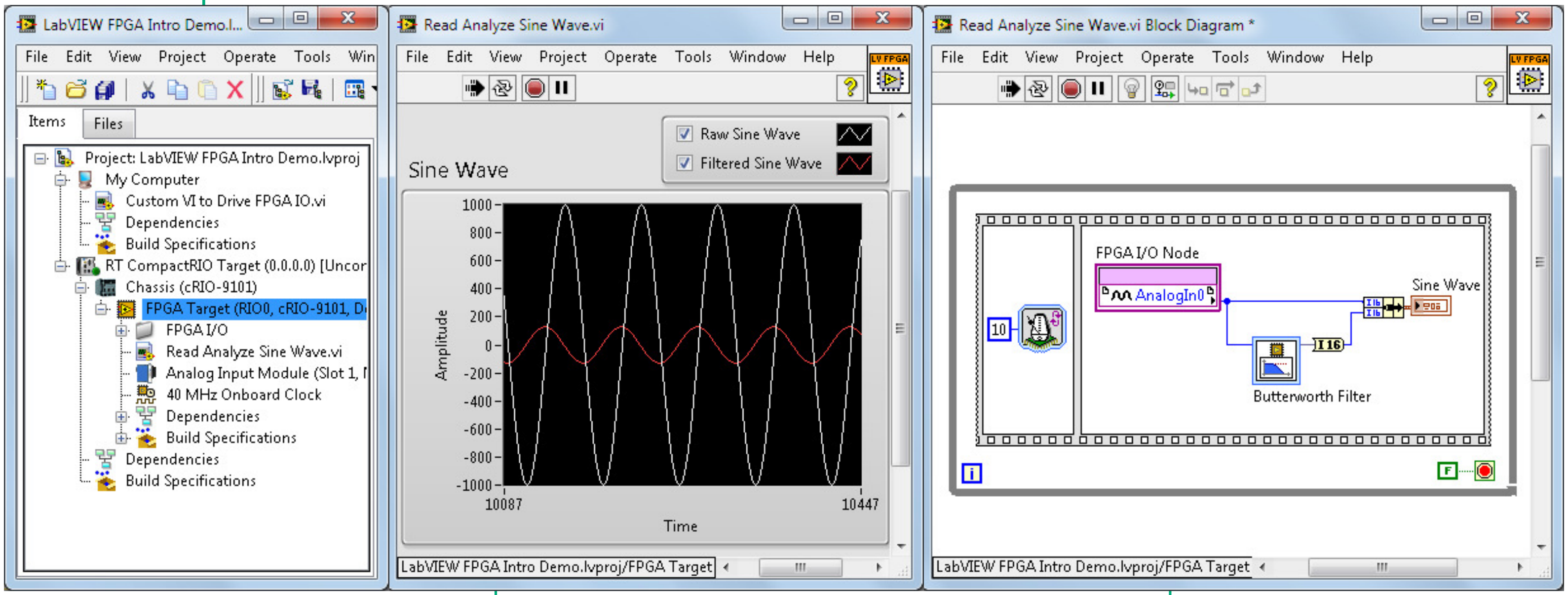
~4000 lines of VHDL

LabVIEW FPGA vs. VHDL

LabVIEW Graphical Development Environment

“Project” = System Configuration

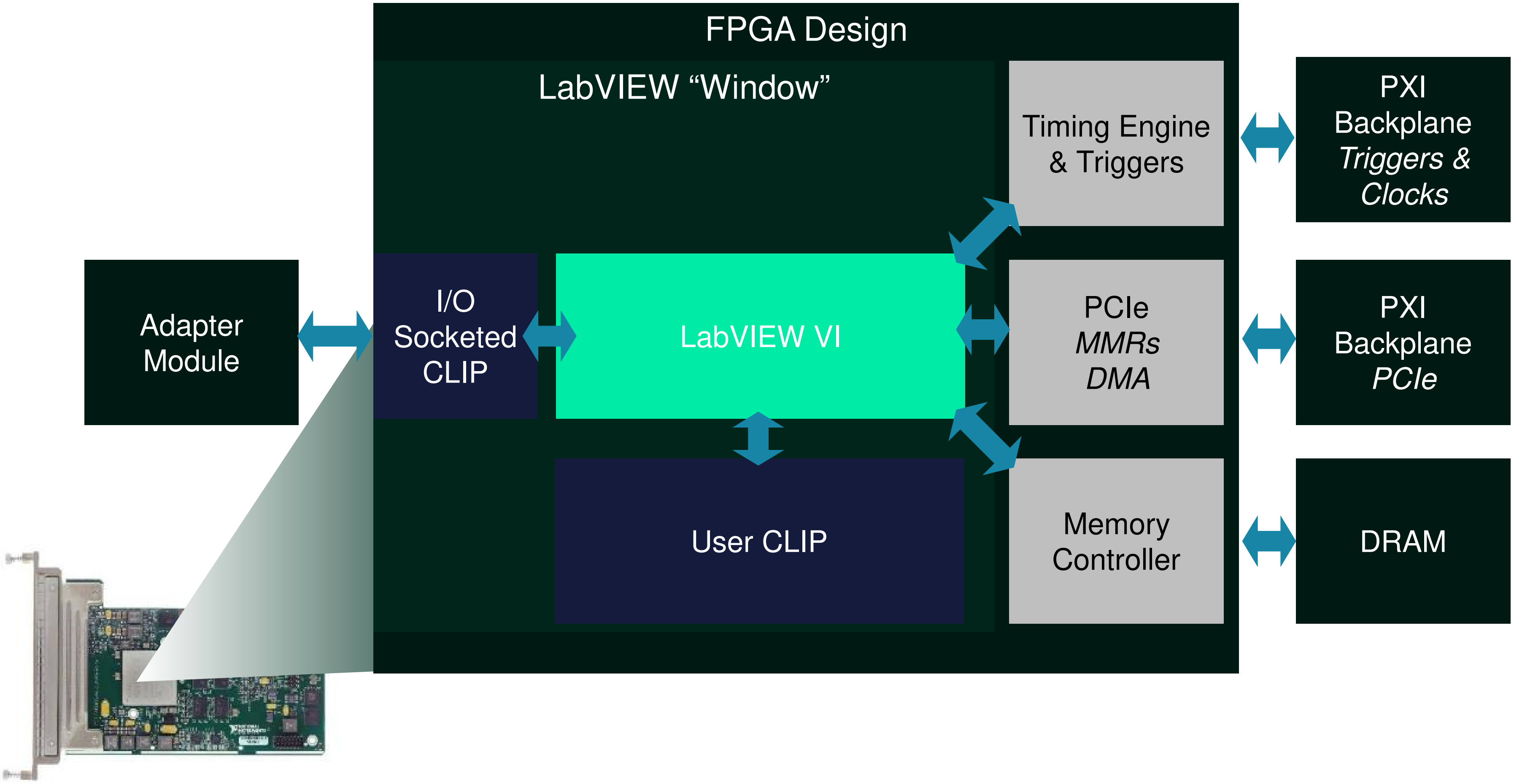
“VI” = Application


LabVIEW FPGA Intro Demo.l...
File Edit View Project Operate Tools Win
Items Files
Project: LabVIEW FPGA Intro Demo.lvproj
My Computer
Custom VI to Drive FPGA IO.vi
Dependencies
Build Specifications
RT CompactRIO Target (0.0.0.0) [Uncor
Chassis (cRIO-9101)
FPGA Target (RIO0, cRIO-9101, D
FPGA I/O
Read Analyze Sine Wave.vi
Analog Input Module (Slot 1, I
40 MHz Onboard Clock
Dependencies
Build Specifications
Dependencies
Build Specifications
Read Analyze Sine Wave.vi
File Edit View Project Operate Tools Window Help
Sine Wave
Raw Sine Wave
Filtered Sine Wave
Amplitude
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
10087
10447
Time
LabVIEW FPGA Intro Demo.lvproj/FPGA Target
Read Analyze Sine Wave.vi Block Diagram *
File Edit View Project Operate Tools Window Help
FPGA I/O Node
AnalogIn0
10
Sine Wave
I16
Butterworth Filter
F
i
LabVIEW FPGA Intro Demo.lvproj/FPGA Target

“Front Panel” = Interface Elements

“Block Diagram” = Code

FlexRIO FPGA Design Hierarchy

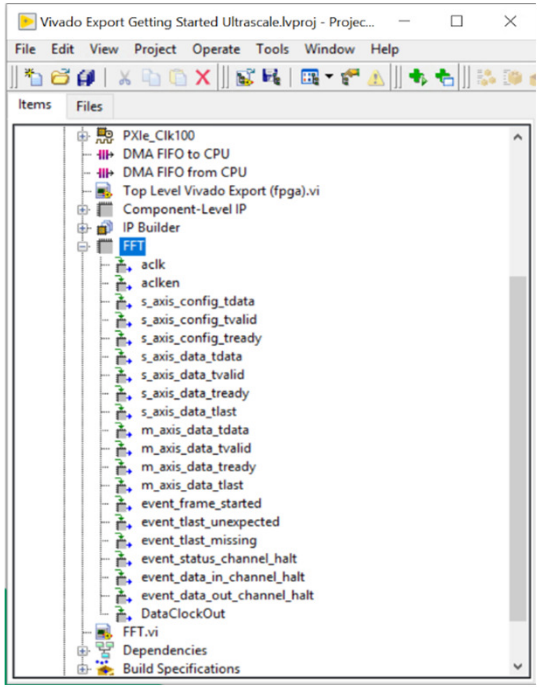


User-Defined CLIP

User CLIP

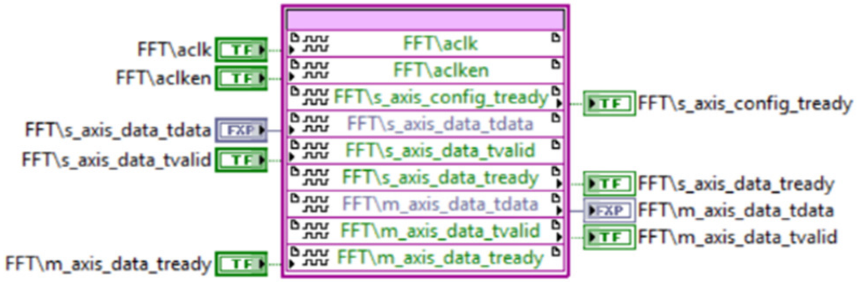
- Runs completely parallel to LabVIEW FPGA VI
- User defines the entire interface to LV FPGA Block Diagram
- Supports different netlist types

CLIP in Project

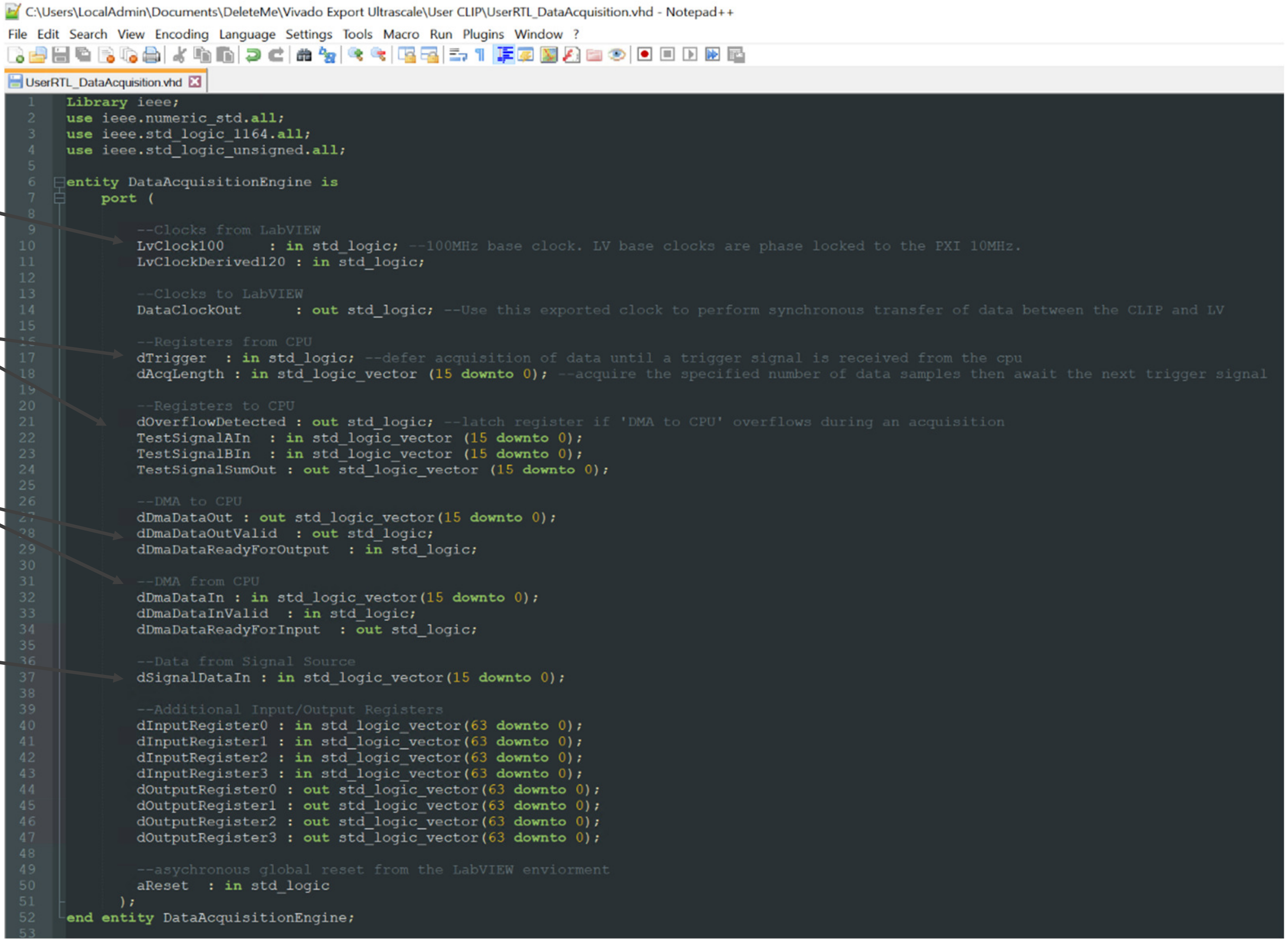

Vivado Export Getting Started Ultrascale.lvproj - Projec...

- PXIe_Clk100
- DMA FIFO to CPU
- DMA FIFO from CPU
- Top Level Vivado Export (fpga).vi
- Component-Level IP
- IP Builder
- FFT
  - aclk
  - aclken
  - s_axis_config_tdata
  - s_axis_config_tvalid
  - s_axis_config_tready
  - s_axis_data_tdata
  - s_axis_data_tvalid
  - s_axis_data_tready
  - s_axis_data_tlast
  - m_axis_data_tdata
  - m_axis_data_tvalid
  - m_axis_data_tready
  - m_axis_data_tlast
  - event_frame_started
  - event_tlast_unexpected
  - event_tlast_missing
  - event_status_channel_halt
  - event_data_in_channel_halt
  - event_data_out_channel_halt
  - DataClockOut
- FFT.vi
- Dependencies
- Build Specifications

CLIP on Block Diagram



Example CLIP


C:\Users\LocalAdmin\Documents\DeleteMe\Vivado Export Ultrascale\User CLIP\UserRTL_DataAcquisition.vhd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
UserRTL_DataAcquisition.vhd

```vhdl
Library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity DataAcquisitionEngine is
    port (

        --Clocks from LabVIEW
        LvClock100      : in std_logic; --100MHz base clock. LV base clocks are phase locked to the PXI 10MHz.
        LvClockDerived120 : in std_logic;

        --Clocks to LabVIEW
        DataClockOut       : out std_logic; --Use this exported clock to perform synchronous transfer of data between the CLIP and LV

        --Registers from CPU
        dTrigger  : in std_logic; --defer acquisition of data until a trigger signal is received from the cpu
        dAcqLength : in std_logic_vector (15 downto 0); --acquire the specified number of data samples then await the next trigger signal

        --Registers to CPU
        dOverflowDetected : out std_logic; --latch register if 'DMA to CPU' overflows during an acquisition
        TestSignalAIn  : in std_logic_vector (15 downto 0);
        TestSignalBIn  : in std_logic_vector (15 downto 0);
        TestSignalSumOut : out std_logic_vector (15 downto 0);

        --DMA to CPU
        dDmaDataOut : out std_logic_vector(15 downto 0);
        dDmaDataOutValid  : out std_logic;
        dDmaDataReadyForOutput  : in std_logic;

        --DMA from CPU
        dDmaDataIn : in std_logic_vector(15 downto 0);
        dDmaDataInValid  : in std_logic;
        dDmaDataReadyForInput  : out std_logic;

        --Data from Signal Source
        dSignalDataIn : in std_logic_vector(15 downto 0);

        --Additional Input/Output Registers
        dInputRegister0 : in std_logic_vector(63 downto 0);
        dInputRegister1 : in std_logic_vector(63 downto 0);
        dInputRegister2 : in std_logic_vector(63 downto 0);
        dInputRegister3 : in std_logic_vector(63 downto 0);
        dOutputRegister0 : out std_logic_vector(63 downto 0);
        dOutputRegister1 : out std_logic_vector(63 downto 0);
        dOutputRegister2 : out std_logic_vector(63 downto 0);
        dOutputRegister3 : out std_logic_vector(63 downto 0);

        --asychronous global reset from the LabVIEW enviorment
        aReset  : in std_logic
    );
end entity DataAcquisitionEngine;
```

Clocks to/from LabVIEW

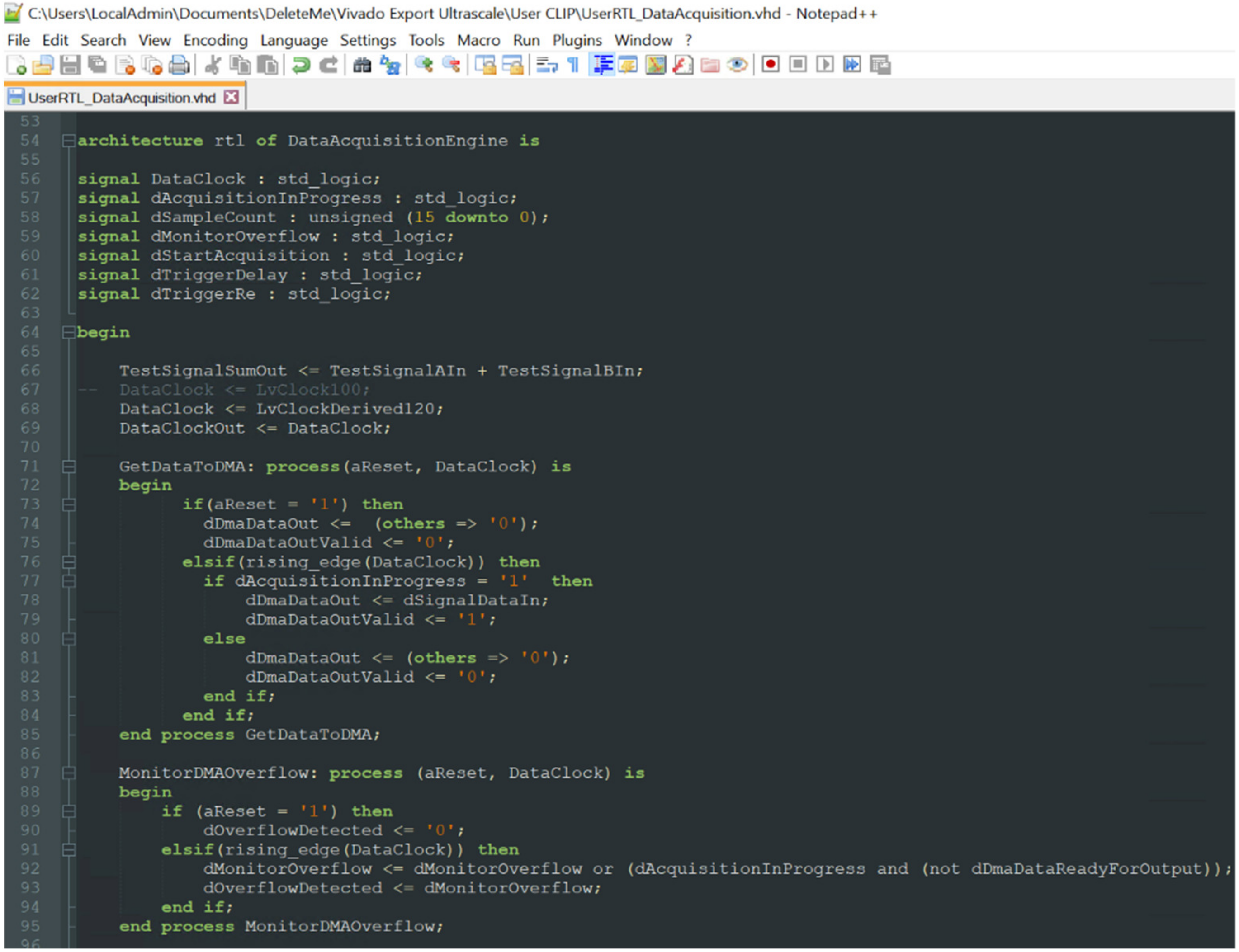
Memory mapped registers to/from host

DMA to/from Host

Signals to/from LabVIEW

Example CLIP

Application Specific User Code


C:\Users\LocalAdmin\Documents\DeleteMe\Vivado Export Ultrascale\User CLIP\UserRTL_DataAcquisition.vhd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
UserRTL_DataAcquisition.vhd

```vhdl

architecture rtl of DataAcquisitionEngine is

signal DataClock : std_logic;
signal dAcquisitionInProgress : std_logic;
signal dSampleCount : unsigned (15 downto 0);
signal dMonitorOverflow : std_logic;
signal dStartAcquisition : std_logic;
signal dTriggerDelay : std_logic;
signal dTriggerRe : std_logic;

begin

    TestSignalSumOut <= TestSignalAIn + TestSignalBIn;
--  DataClock <= LvClock100;
    DataClock <= LvClockDerived120;
    DataClockOut <= DataClock;

    GetDataToDMA: process(aReset, DataClock) is
    begin
          if(aReset = '1') then
            dDmaDataOut <=  (others => '0');
            dDmaDataOutValid <= '0';
          elsif(rising_edge(DataClock)) then
            if dAcquisitionInProgress = '1'  then
                dDmaDataOut <= dSignalDataIn;
                dDmaDataOutValid <= '1';
            else
                dDmaDataOut <= (others => '0');
                dDmaDataOutValid <= '0';
            end if;
          end if;
    end process GetDataToDMA;

    MonitorDMAOverflow: process (aReset, DataClock) is
    begin
        if (aReset = '1') then
            dOverflowDetected <= '0';
        elsif(rising_edge(DataClock)) then
            dMonitorOverflow <= dMonitorOverflow or (dAcquisitionInProgress and (not dDmaDataReadyForOutput));
            dOverflowDetected <= dMonitorOverflow;
        end if;
    end process MonitorDMAOverflow;
```

Socketed CLIP

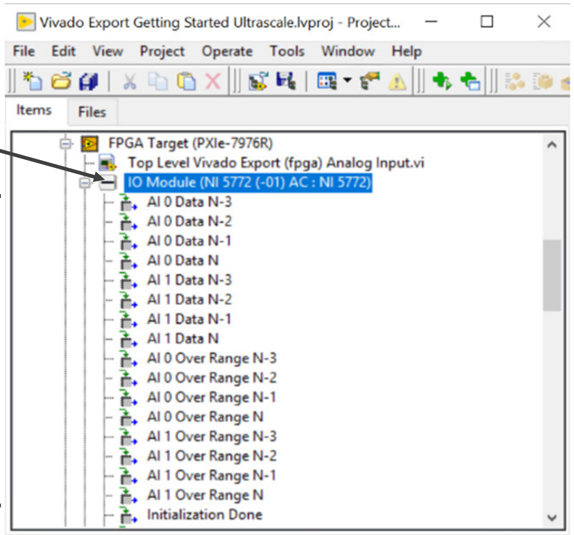
I/O Socket CLIP

- Interface between FPGA GPIO/MGTs and LabVIEW
- Runs completely parallel to LabVIEW code
- Often passes clocks to LabVIEW for synchronous data movement
- I/O developer defines interface to LabVIEW FPGA

Socketed CLIP in Project

I/O CLIP

CLIP Signals



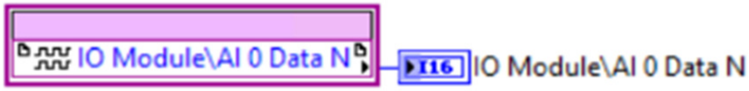
Vivado Export Getting Started Ultrascale.lvproj - Project...

File Edit View Project Operate Tools Window Help

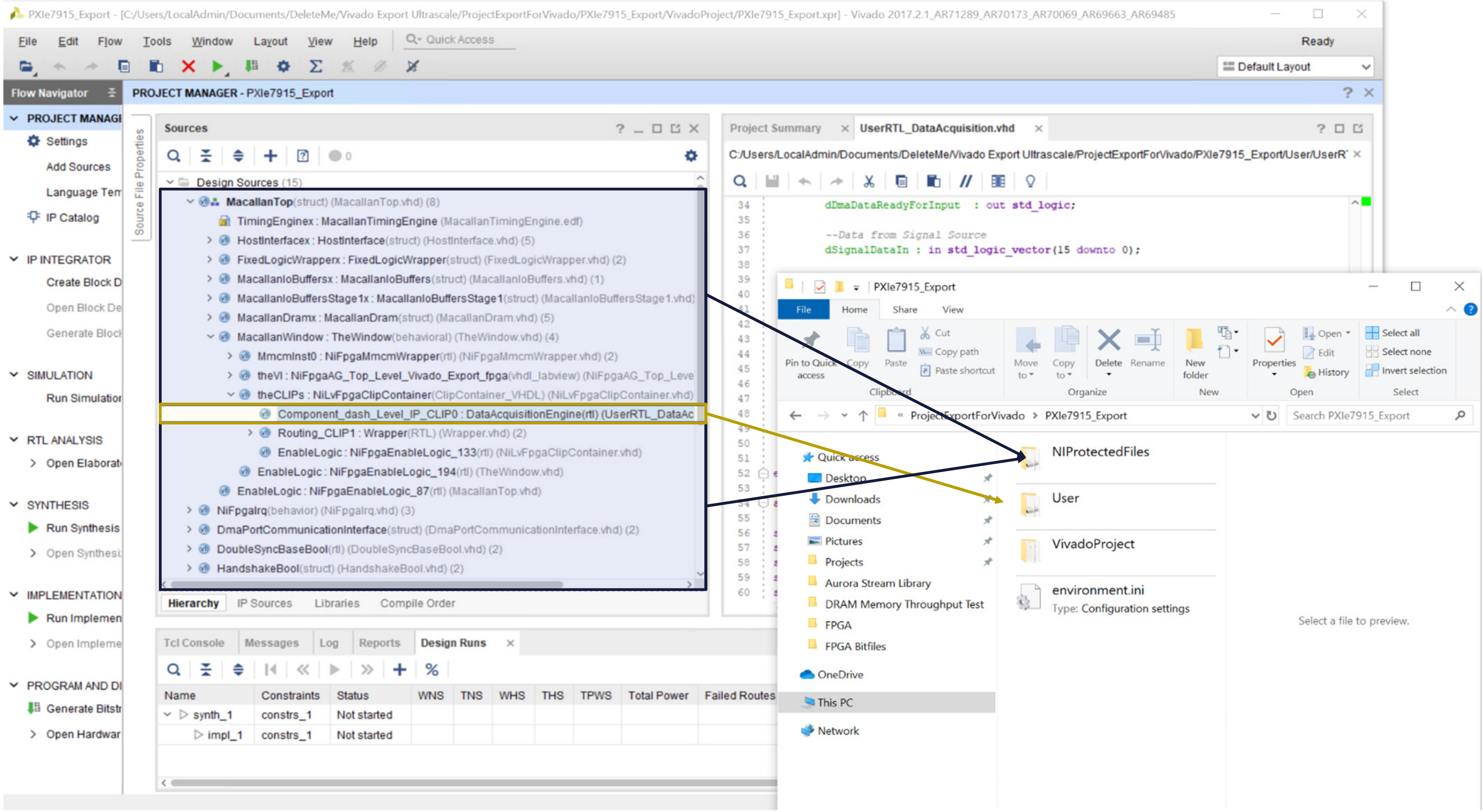
Items Files

- FPGA Target (PXIe-7976R)
  - Top Level Vivado Export (fpga) Analog Input.vi
  - IO Module (NI 5772 (-01) AC : NI 5772)
    - AI 0 Data N-3
    - AI 0 Data N-2
    - AI 0 Data N-1
    - AI 0 Data N
    - AI 1 Data N-3
    - AI 1 Data N-2
    - AI 1 Data N-1
    - AI 1 Data N
    - AI 0 Over Range N-3
    - AI 0 Over Range N-2
    - AI 0 Over Range N-1
    - AI 0 Over Range N
    - AI 1 Over Range N-3
    - AI 1 Over Range N-2
    - AI 1 Over Range N-1
    - AI 1 Over Range N
    - Initialization Done

Socketed CLIP on Block Diagram



Vivado – RTL Hierarchy



PXIe7915_Export - [C:/Users/LocalAdmin/Documents/DeleteMe/Vivado Export Ultrascale/ProjectExportForVivado/PXIe7915_Export/VivadoProject/PXIe7915_Export.xpr] - Vivado 2017.2.1_AR71289_AR70173_AR70069_AR69663_AR69485

PROJECT MANAGER - PXIe7915_Export

Sources

- Design Sources (15)
  - MacallanTop(struct) (MacallanTop.vhd) (8)
    - TimingEnginex : MacallanTimingEngine (MacallanTimingEngine.edf)
    - HostInterfacex : HostInterface(struct) (HostInterface.vhd) (5)
    - FixedLogicWrapperx : FixedLogicWrapper(struct) (FixedLogicWrapper.vhd) (2)
    - MacallanIoBuffersx : MacallanIoBuffers(struct) (MacallanIoBuffers.vhd) (1)
    - MacallanIoBuffersStage1x : MacallanIoBuffersStage1(struct) (MacallanIoBuffersStage1.vhd)
    - MacallanDramx : MacallanDram(struct) (MacallanDram.vhd) (5)
    - MacallanWindow : TheWindow(behavioral) (TheWindow.vhd) (4)
      - MmcmInst0 : NiFpgaMmcmWrapper(rtl) (NiFpgaMmcmWrapper.vhd) (2)
      - theVI : NiFpgaAG_Top_Level_Vivado_Export_fpga(vhdl_labview) (NiFpgaAG_Top_Leve
      - theCLIPs : NiLvFpgaClipContainer(ClipContainer_VHDL) (NiLvFpgaClipContainer.vhd)
        - Component_dash_Level_IP_CLIP0 : DataAcquisitionEngine(rtl) (UserRTL_DataAc
        - Routing_CLIP1 : Wrapper(RTL) (Wrapper.vhd) (2)
        - EnableLogic : NiFpgaEnableLogic_133(rtl) (NiLvFpgaClipContainer.vhd)
      - EnableLogic : NiFpgaEnableLogic_194(rtl) (TheWindow.vhd)
    - EnableLogic : NiFpgaEnableLogic_87(rtl) (MacallanTop.vhd)
  - NiFpgaIrq(behavior) (NiFpgaIrq.vhd) (3)
  - DmaPortCommunicationInterface(struct) (DmaPortCommunicationInterface.vhd) (2)
  - DoubleSyncBaseBool(rtl) (DoubleSyncBaseBool.vhd) (2)
  - HandshakeBool(struct) (HandshakeBool.vhd) (2)

UserRTL_DataAcquisition.vhd

C:/Users/LocalAdmin/Documents/DeleteMe/Vivado Export Ultrascale/ProjectExportForVivado/PXIe7915_Export/User/UserR

```
        dDmaDataReadyForInput  : out std_logic;

        --Data from Signal Source
        dSignalDataIn : in std_logic_vector(15 downto 0);
```

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes
synth_1	constrs_1	Not started							
impl_1	constrs_1	Not started							

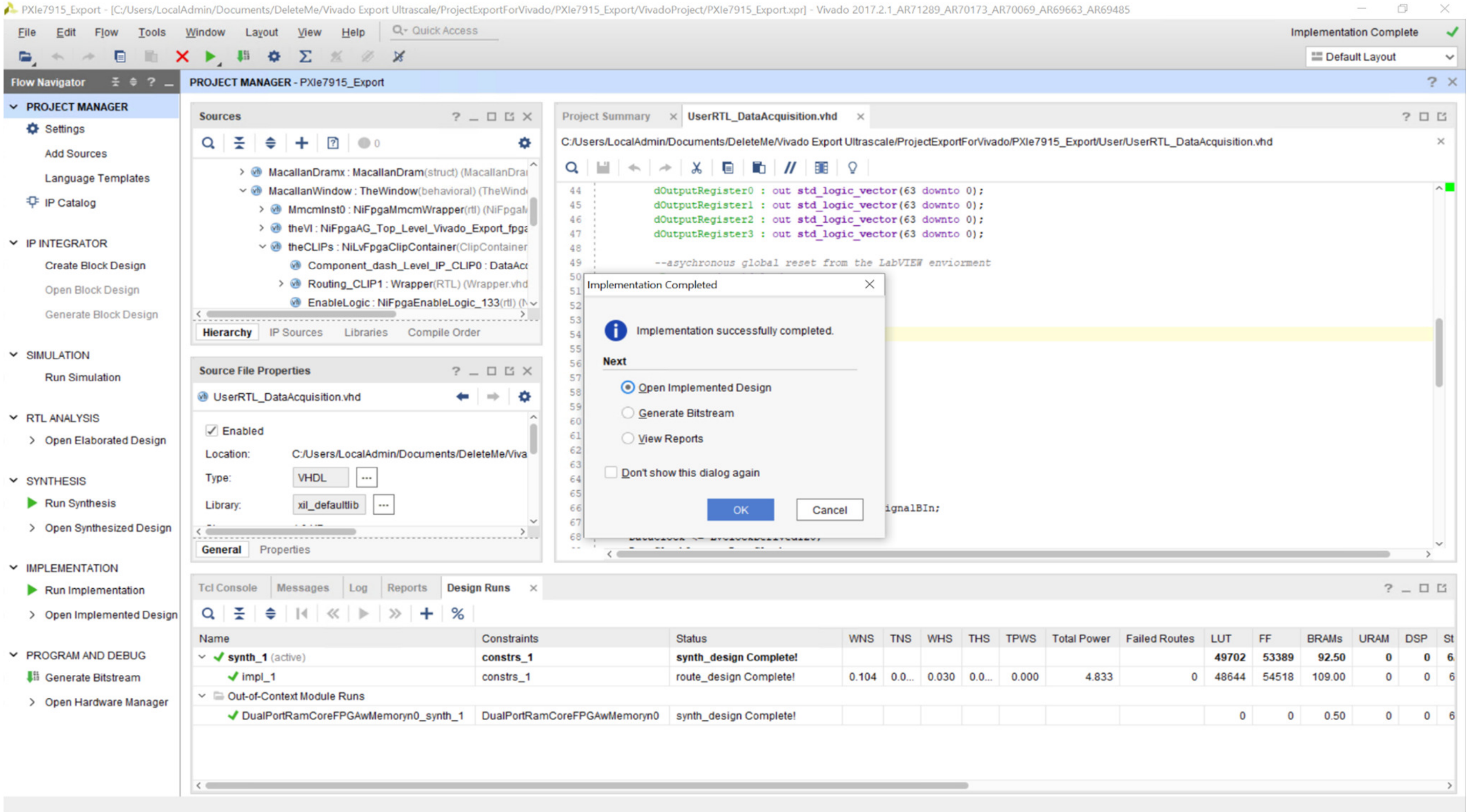
PXIe7915_Export

ProjectExportForVivado › PXIe7915_Export

- NIProtectedFiles
- User
- VivadoProject
- environment.ini
Type: Configuration settings

Select a file to preview.

Vivado – RTL Hierarchy



PXIe7915_Export - [C:/Users/LocalAdmin/Documents/DeleteMe/Vivado Export Ultrascale/ProjectExportForVivado/PXIe7915_Export/VivadoProject/PXIe7915_Export.xpr] - Vivado 2017.2.1_AR71289_AR70173_AR70069_AR69663_AR69485

Sources:

- MacallanDramx : MacallanDram(struct) (MacallanDra
- MacallanWindow : TheWindow(behavioral) (TheWind
  - MmcmInst0 : NiFpgaMmcmWrapper(rtl) (NiFpgaM
  - theVI : NiFpgaAG_Top_Level_Vivado_Export_fpga
  - theCLIPs : NiLvFpgaClipContainer(ClipContainer
    - Component_dash_Level_IP_CLIP0 : DataAc
    - Routing_CLIP1 : Wrapper(RTL) (Wrapper.vhd
    - EnableLogic : NiFpgaEnableLogic_133(rtl) (N

Source File Properties: UserRTL_DataAcquisition.vhd

☑ Enabled

Location: C:/Users/LocalAdmin/Documents/DeleteMe/Viva

Type: VHDL

Library: xil_defaultlib

C:/Users/LocalAdmin/Documents/DeleteMe/Vivado Export Ultrascale/ProjectExportForVivado/PXIe7915_Export/User/UserRTL_DataAcquisition.vhd

```
        dOutputRegister0 : out std_logic_vector(63 downto 0);
        dOutputRegister1 : out std_logic_vector(63 downto 0);
        dOutputRegister2 : out std_logic_vector(63 downto 0);
        dOutputRegister3 : out std_logic_vector(63 downto 0);

        --asychronous global reset from the LabVIEW enviorment
```

Implementation Completed

Implementation successfully completed.

Next

● Open Implemented Design

○ Generate Bitstream

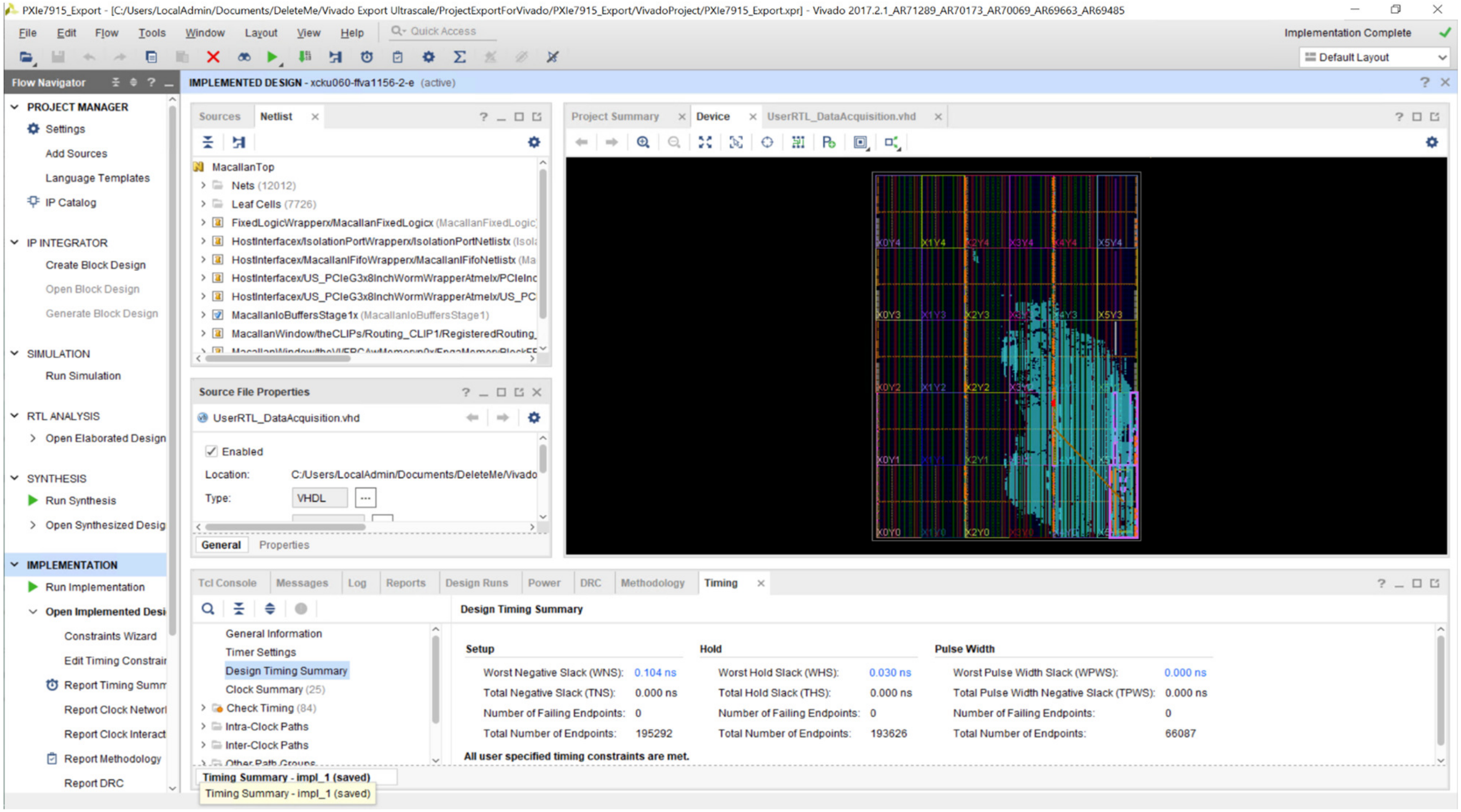
○ View Reports

□ Don't show this dialog again

OK Cancel

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	St
✓ **synth_1** (active)	**constrs_1**	**synth_design Complete!**								**49702**	**53389**	**92.50**	**0**	**0**	**6.**
✓ impl_1	constrs_1	route_design Complete!	0.104	0.0...	0.030	0.0...	0.000	4.833	0	48644	54518	109.00	0	0	6
Out-of-Context Module Runs															
✓ DualPortRamCoreFPGAwMemoryn0_synth_1	DualPortRamCoreFPGAwMemoryn0	synth_design Complete!								0	0	0.50	0	0	6

Vivado – RTL Hierarchy



PXIe7915_Export - [C:/Users/LocalAdmin/Documents/DeleteMe/Vivado Export Ultrascale/ProjectExportForVivado/PXIe7915_Export/VivadoProject/PXIe7915_Export.xpr] - Vivado 2017.2.1_AR71289_AR70173_AR70069_AR69663_AR69485

Implementation Complete ✓

IMPLEMENTED DESIGN - xcku060-ffva1156-2-e (active)

```
MacallanTop
  > Nets (12012)
  > Leaf Cells (7726)
  > FixedLogicWrapperx/MacallanFixedLogicx (MacallanFixedLogic
  > HostInterfacex/IsolationPortWrapperx/IsolationPortNetlistx (Isol
  > HostInterfacex/MacallanFifoWrapperx/MacallanFifoNetlistx (Ma
  > HostInterfacex/US_PCIeG3x8InchWormWrapperAtmelx/PCIeInc
  > HostInterfacex/US_PCIeG3x8InchWormWrapperAtmelx/US_PC
  > MacallanIoBuffersStage1x (MacallanIoBuffersStage1)
  > MacallanWindow/theCLIPs/Routing_CLIP1/RegisteredRouting_
```

Source File Properties

UserRTL_DataAcquisition.vhd

☑ Enabled

Location: C:/Users/LocalAdmin/Documents/DeleteMe/Vivado

Type: VHDL

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.104 ns	Worst Hold Slack (WHS):	0.030 ns	Worst Pulse Width Slack (WPWS):	0.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	195292	Total Number of Endpoints:	193626	Total Number of Endpoints:	66087

All user specified timing constraints are met.

Timing Summary - impl_1 (saved)