SMAP 240MHz Digital Filter Demo

April 7, 2009

Kayla Nguyen, Jason Zheng, Charles Le,
Jason Chu, Yutao He, Mohammad Ashtijou,
Mimi Paller

JPL DISCREET - not for distribution

Acknowledgement

- We wish to thank the following people for their help during the period of the task
  - Louise Veilleux
  - Kevin Wheeler
  - Mark Fischman
  - Samuel Chan
  - Mike Spencer
  - Vu Duong
  - Duane Clark
  - Andy Berkun
  - Other ISAAC team members
  - Jason Hyon

Outline

- Task Overview
- Algorithm Design
- FPGA Design & Implementation
- Prototype Board
- Testbench Construction & Demonstration
- GUI Interface
- Conclusion

JPL DISCREET - not for distribution

Task Overview

Task Objective

- Investigate whether it is feasible to implement the 240 MHz digital filter design on Xilinx Virtex II 3000 devices.
- Work duration: Jan – April 2009

Approach

- Inherit and extend the ISAAC 60 MHz digital filter design
  - Algorithm Development and Simulation (ADS)
    - Based on the ISAAC 60 MHz digital filter design
  - FPGA Design and Implementation (FDI)
    - Based on the ISAAC 60 MHz digital filter core
  - Testbed Construction and Demonstration (TCD)
    - Based on the ISAAC iBench

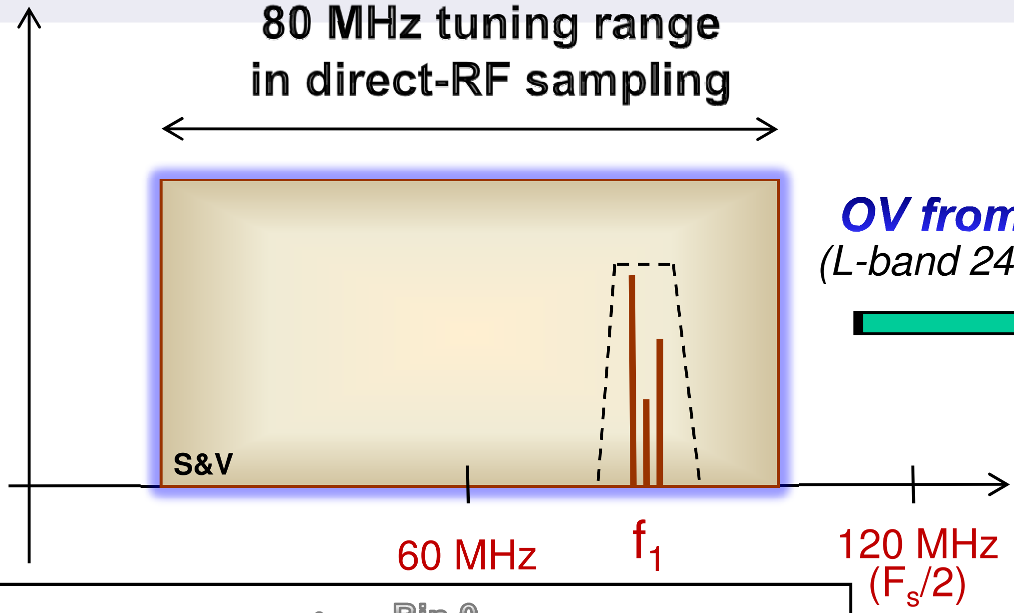
Demo Agenda

- Demonstrate technical feasibility of 240 MHz digital filter on Xilinx Virtex-II 3000
- Demonstrate the ability to tune the center frequency with 1MHz spacing

JPL DISCREET - not for distribution

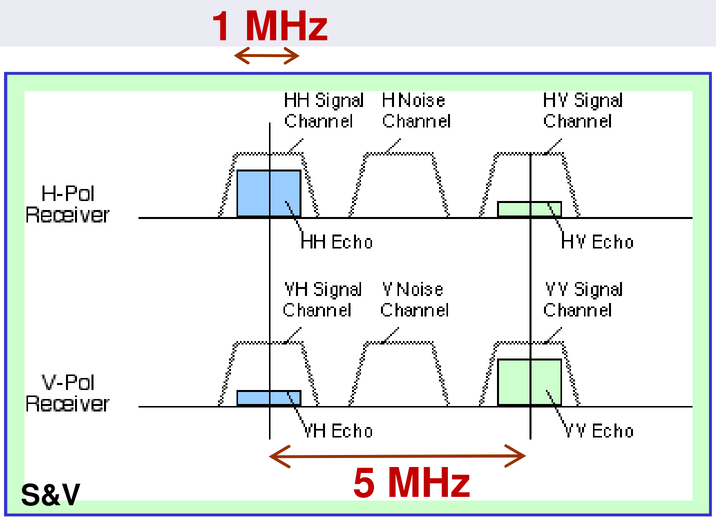
Instrument Digital Electronics

Radar Onboard Processor Requirements



OV from ADC

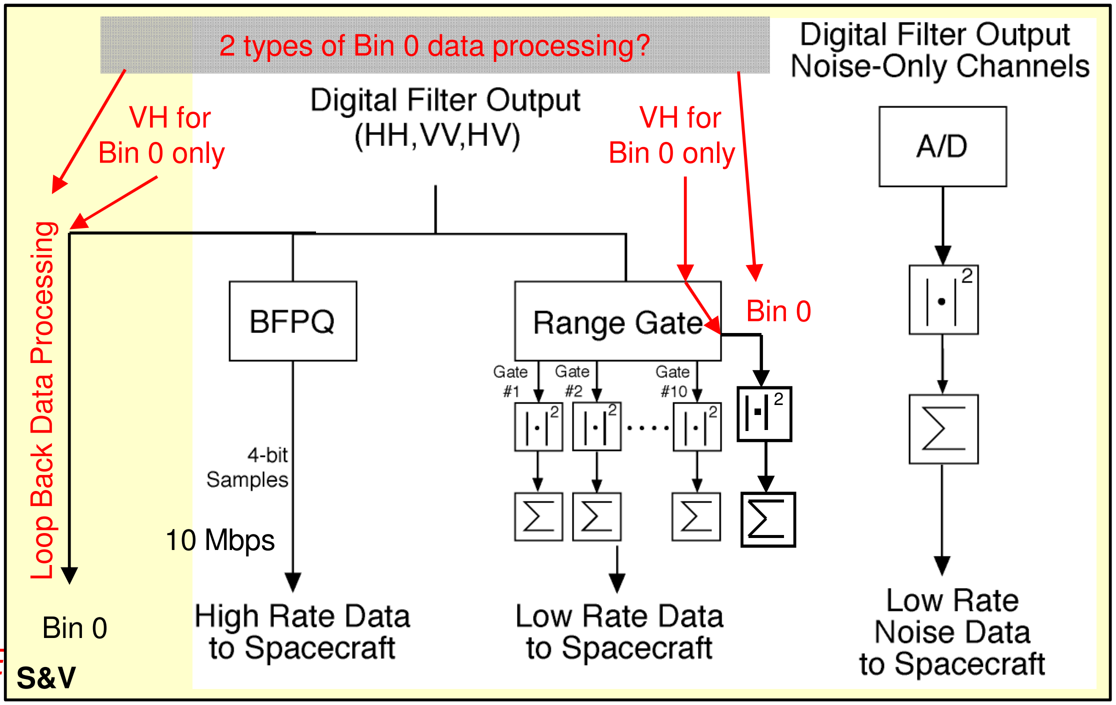
(L-band 240 MHz)



from Digital Filter

(narrowband channelizer)

- **Bin 0**
  - filtered ADC output; no data compression; generated every 45 IPPs (15 msec) ; special representation for loopback samples.
- **High-Rate Data**
  - 12:4 BFPQ; collected over land only; over ½ scan only; over AM half orbit only.
- **Low-Rate Data**
  - always collected; squared; range gating into 10 range bins; integrated over 45 IPPs; 16-bit output for each range bin.
- **Low-Rate Noise Data**
  - always collected, except during bin 0; integrated over 45 IPPs; 16-bit output.

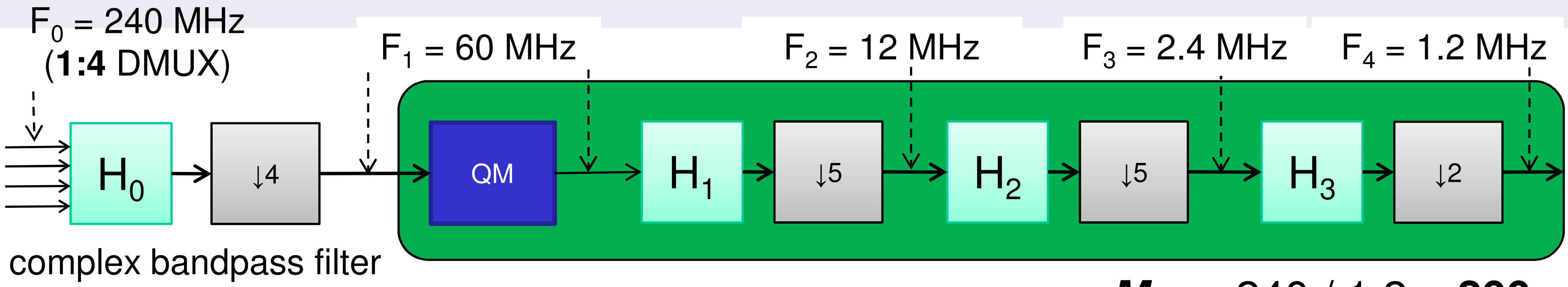

2 types of Bin 0 data processing?
Digital Filter Output Noise-Only Channels
VH for Bin 0 only
Digital Filter Output (HH,VV,HV)
VH for Bin 0 only
A/D
Loop Back Data Processing
BFPQ
Range Gate
Bin 0
Gate #1
Gate #2
Gate #10
4-bit Samples
10 Mbps
Bin 0
High Rate Data to Spacecraft
Low Rate Data to Spacecraft
Low Rate Noise Data to Spacecraft
S&V

SMAP 240MHz Digital Filter Demo

JPL DISCREE

Algorithm Development

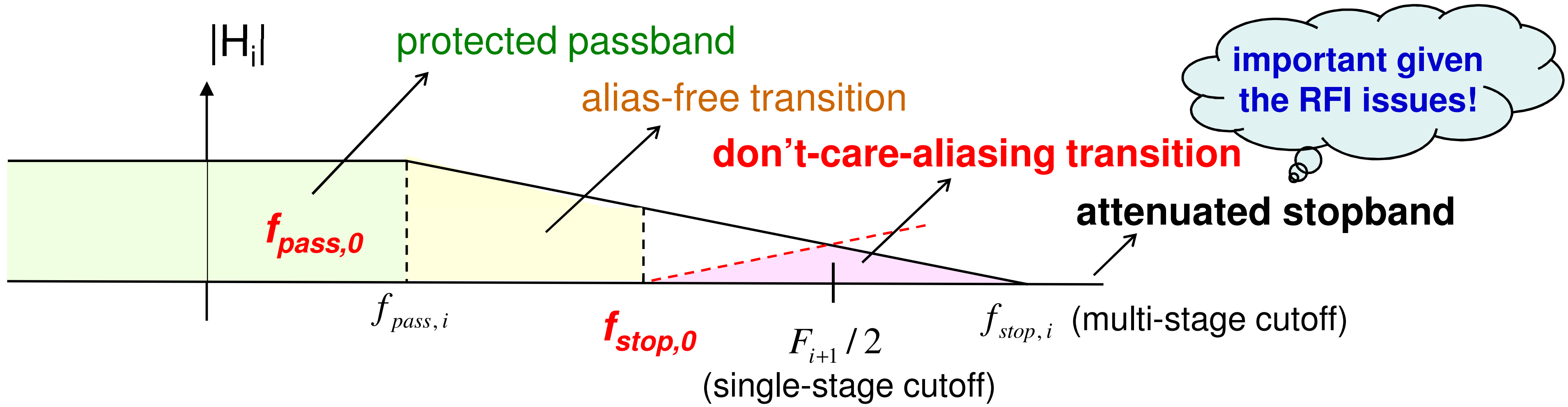
Multi-Stage Decimation Filter Design



$\boldsymbol{M_{tot}}$ **=** 240 / 1.2 **= 200**

high decimation ratio

Individual Lowpass Prototype Specifications:



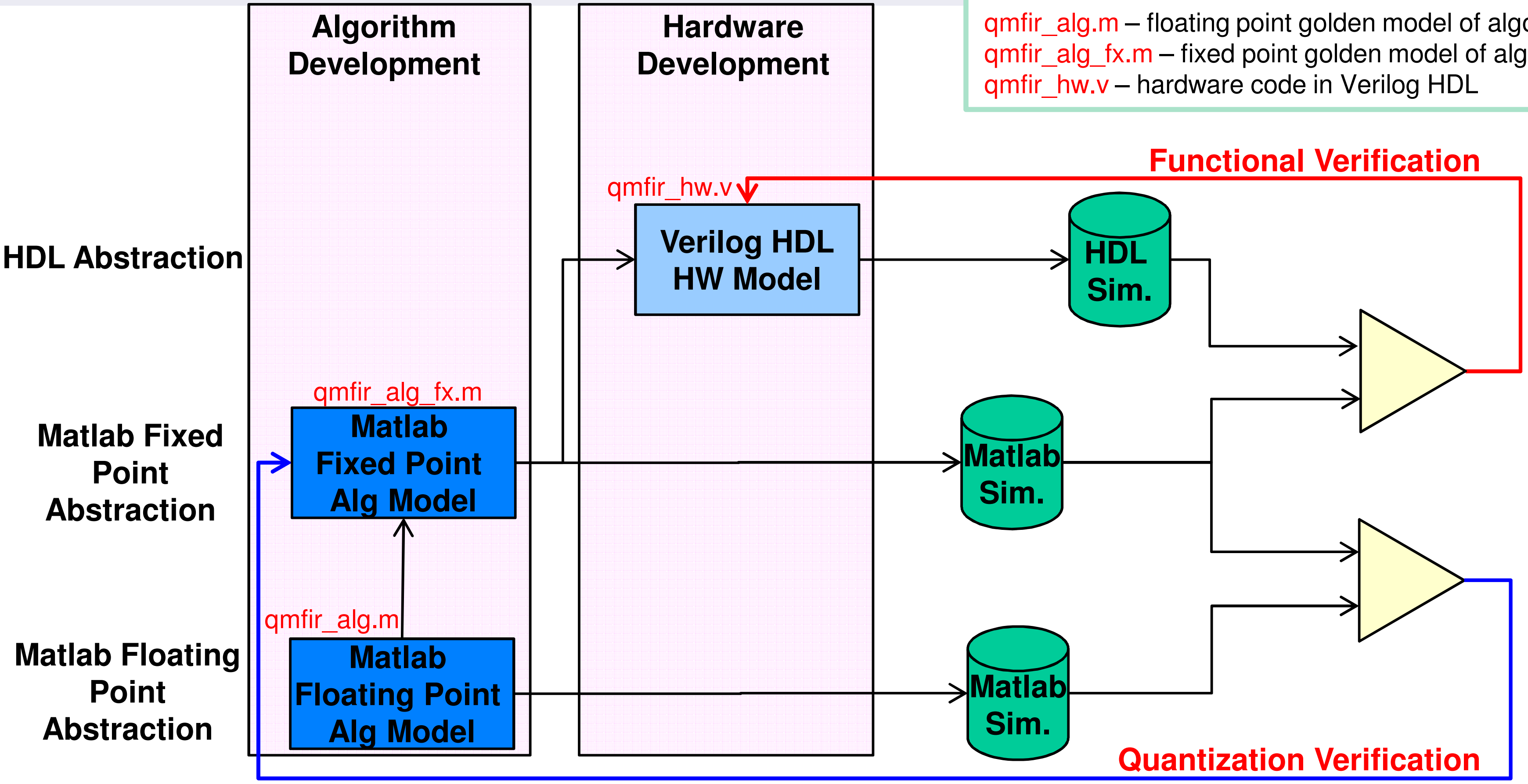
- $f_{pass,i} = f_{pass,0}$
- $f_{stop,i} = F_{i+1} - f_{stop,0}$
- $\delta_{pass,i} = (1/4)\delta_{pass,0}$
- $\delta_{stop,i} = \delta_{stop,0}$

- $f_{pass,0}$ = *0.5 MHz, desired passband freq.*
- $f_{stop,0}$ = *0.6 MHz, (required) stopband freq.*
- $\delta_{pass,0}$ < *0.1 dB, (required) passband ripple.*
- $\delta_{stop,0}$ < *-40 dB, (required) stopband attenuation.*

- F_0 = *240 MHz, ADC sampling rate, slowed down to 60 MHz by 1:4 DMUX.*
- $F_{i+1} = F_i / M_i$, *i^{th} stage decimated output sampling rate.*
- i = *[0,1,2,3], total 4 stages.*
- M_i = *[4,5,5,2], decimation ratios.*

FPGA Design & Implementation
Traditional Design Methodology

qmfir_alg.m – floating point golden model of algorithm
qmfir_alg_fx.m – fixed point golden model of algorithm
qmfir_hw.v – hardware code in Verilog HDL



Functional Verification - Comparison between HDL H/W Implementation model and Matlab Algorithmic Fixed-Point model
- NOT AN EASY FIX! Abstraction level is NOT the same: Matlab fixed point vs. HDL

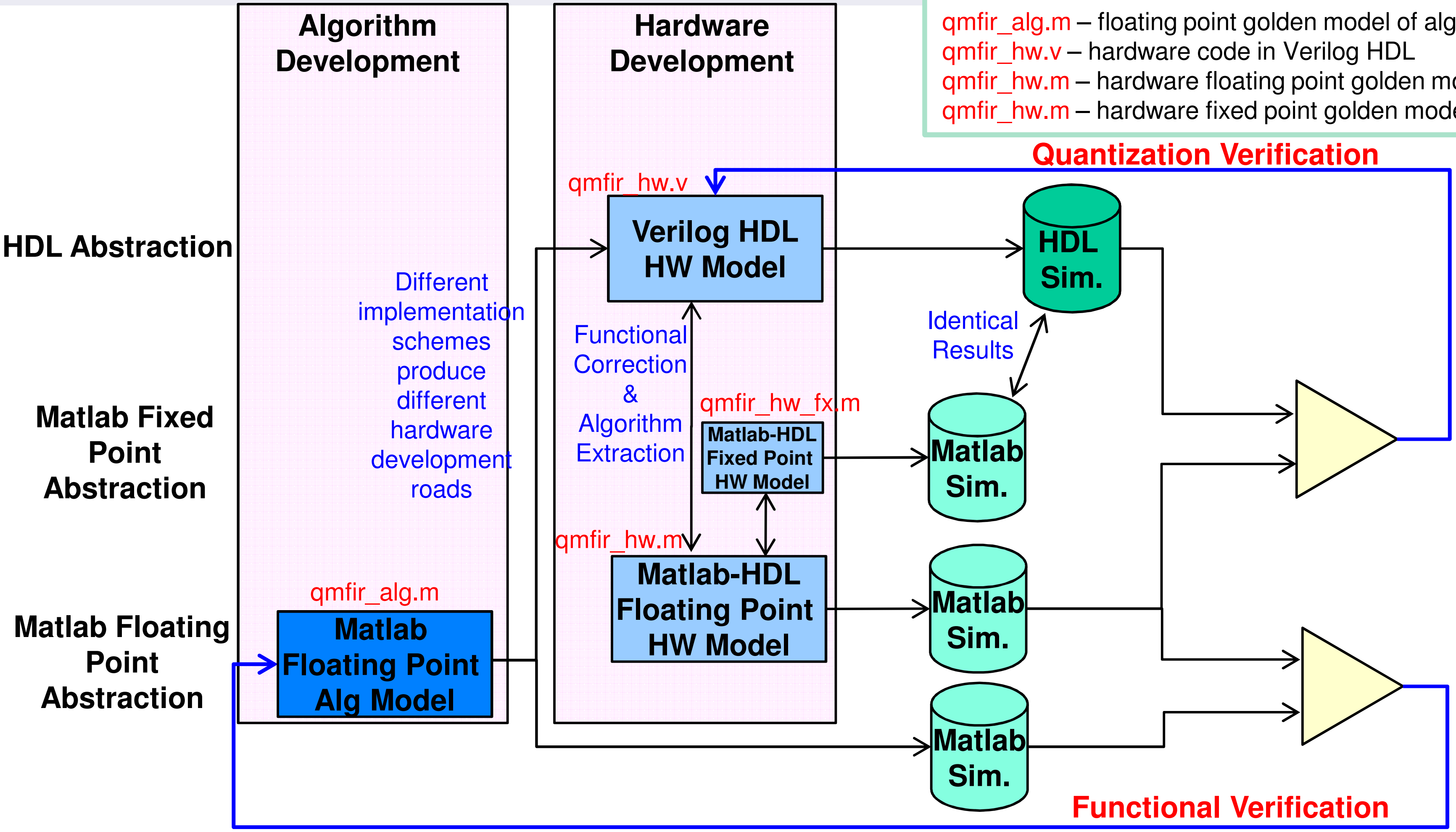
Quantization Verification – Comparison between Matlab Algorithmic Floating-Point and Fixed-Point models
- Comparison done in algorithm development stage

JPL DISCREET - not for distribution

FPGA Design & Implementation

Current ISAAC Design Methodology

qmfir_alg.m – floating point golden model of algorithm
qmfir_hw.v – hardware code in Verilog HDL
qmfir_hw.m – hardware floating point golden model of algorithm
qmfir_hw.m – hardware fixed point golden model of algorithm



Algorithm Development
Hardware Development
Quantization Verification
qmfir_hw.v
HDL Abstraction
Verilog HDL HW Model
HDL Sim.
Different implementation schemes produce different hardware development roads
Functional Correction & Algorithm Extraction
Identical Results
Matlab Fixed Point Abstraction
qmfir_hw_fx.m
Matlab-HDL Fixed Point HW Model
Matlab Sim.
qmfir_hw.m
Matlab-HDL Floating Point HW Model
Matlab Sim.
qmfir_alg.m
Matlab Floating Point Abstraction
Matlab Floating Point Alg Model
Matlab Sim.
Functional Verification

Functional Verification - Comparison between Algorithm and H/W Implementation floating-point models, both in Matlab

- EASIER FIX! Abstraction level is the same: Matlab floating point

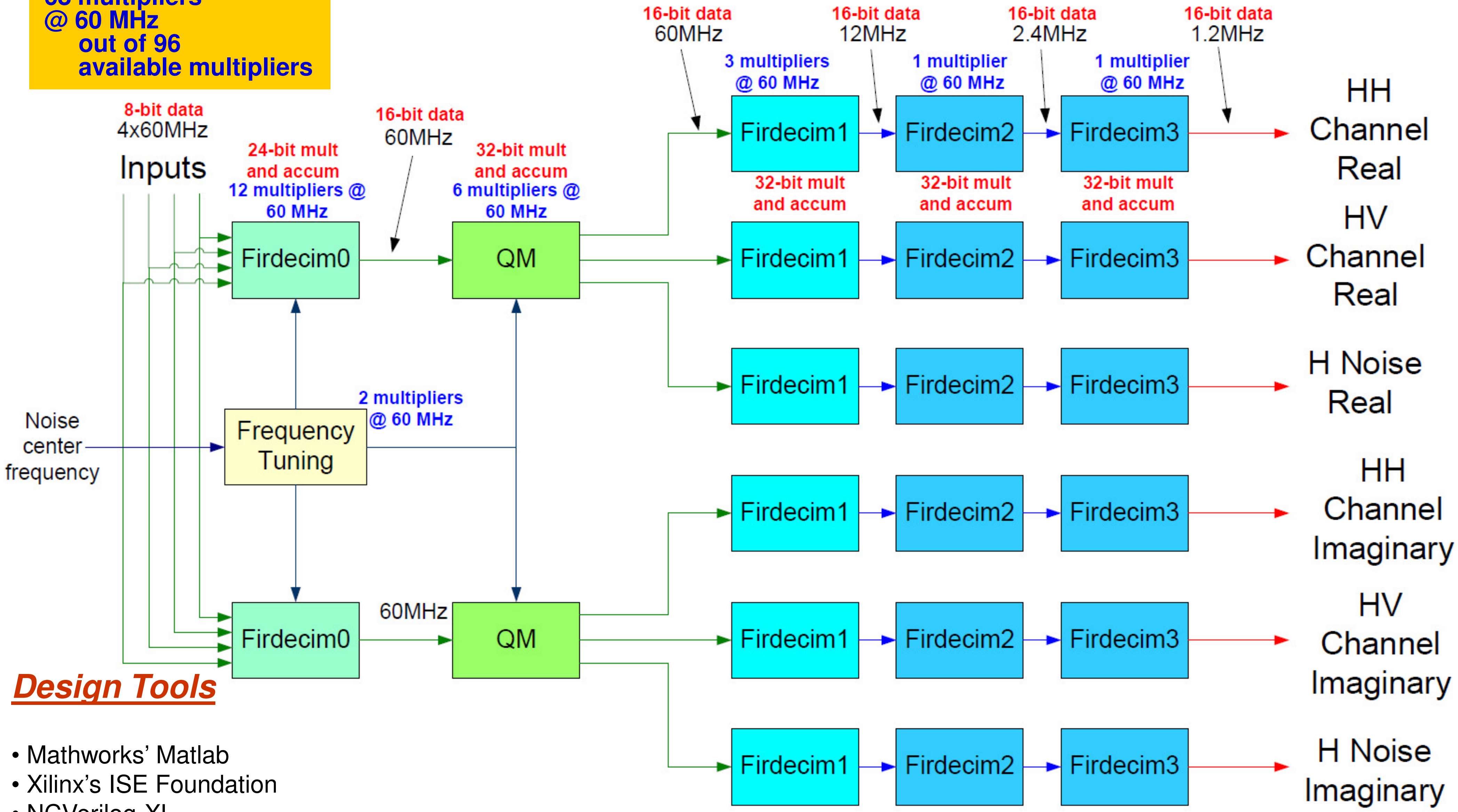
Quantization Verification – Comparison between Matlab fixed-point and HDL models of H/W Implementation

- Comparison done in hardware development stage

FPGA Design & Implementation

QM & Polyphase Decimation Filter

Total
68 multipliers
@ 60 MHz
out of 96
available multipliers



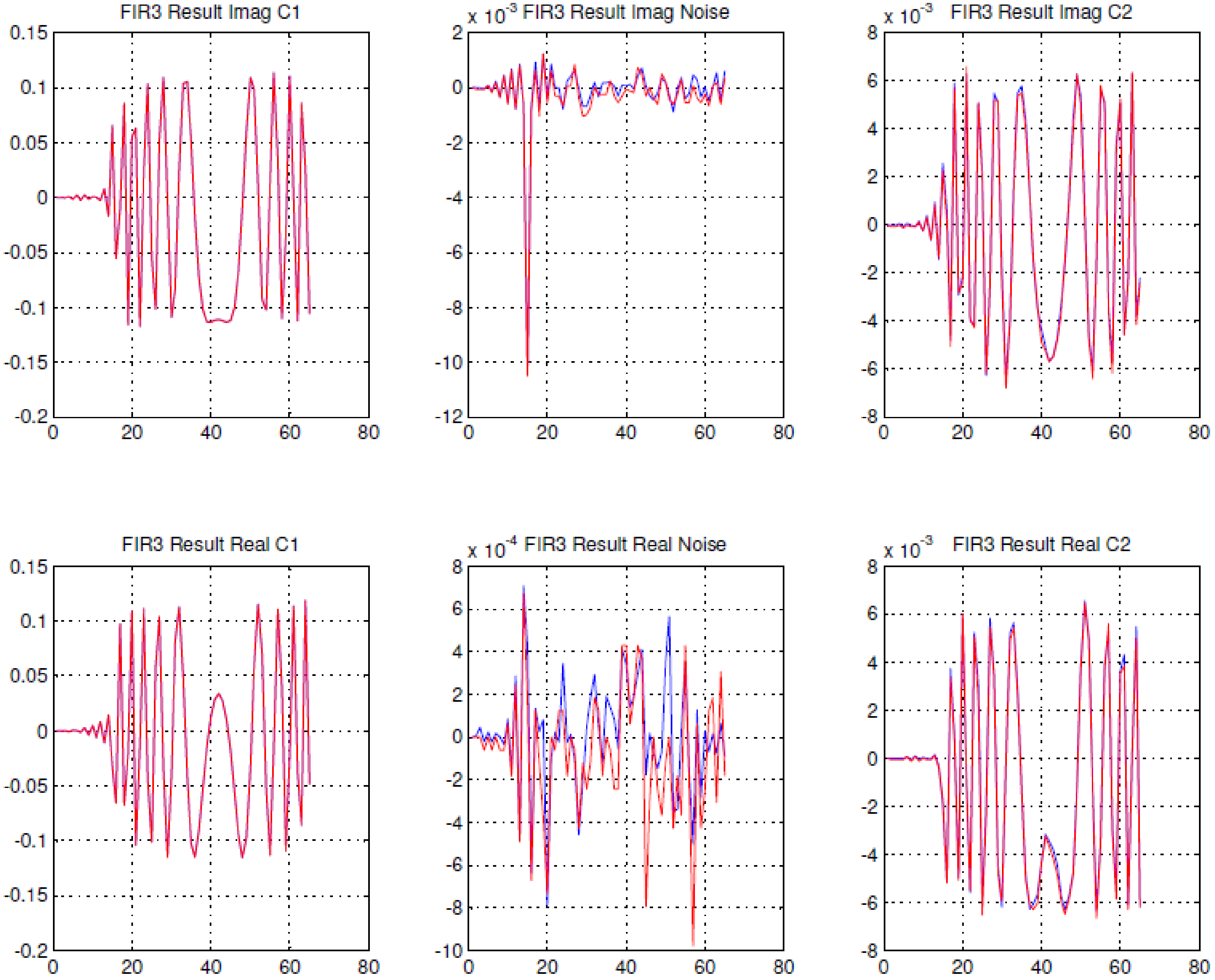
Design Tools

- Mathworks' Matlab
- Xilinx's ISE Foundation
- NCVerilog-XL

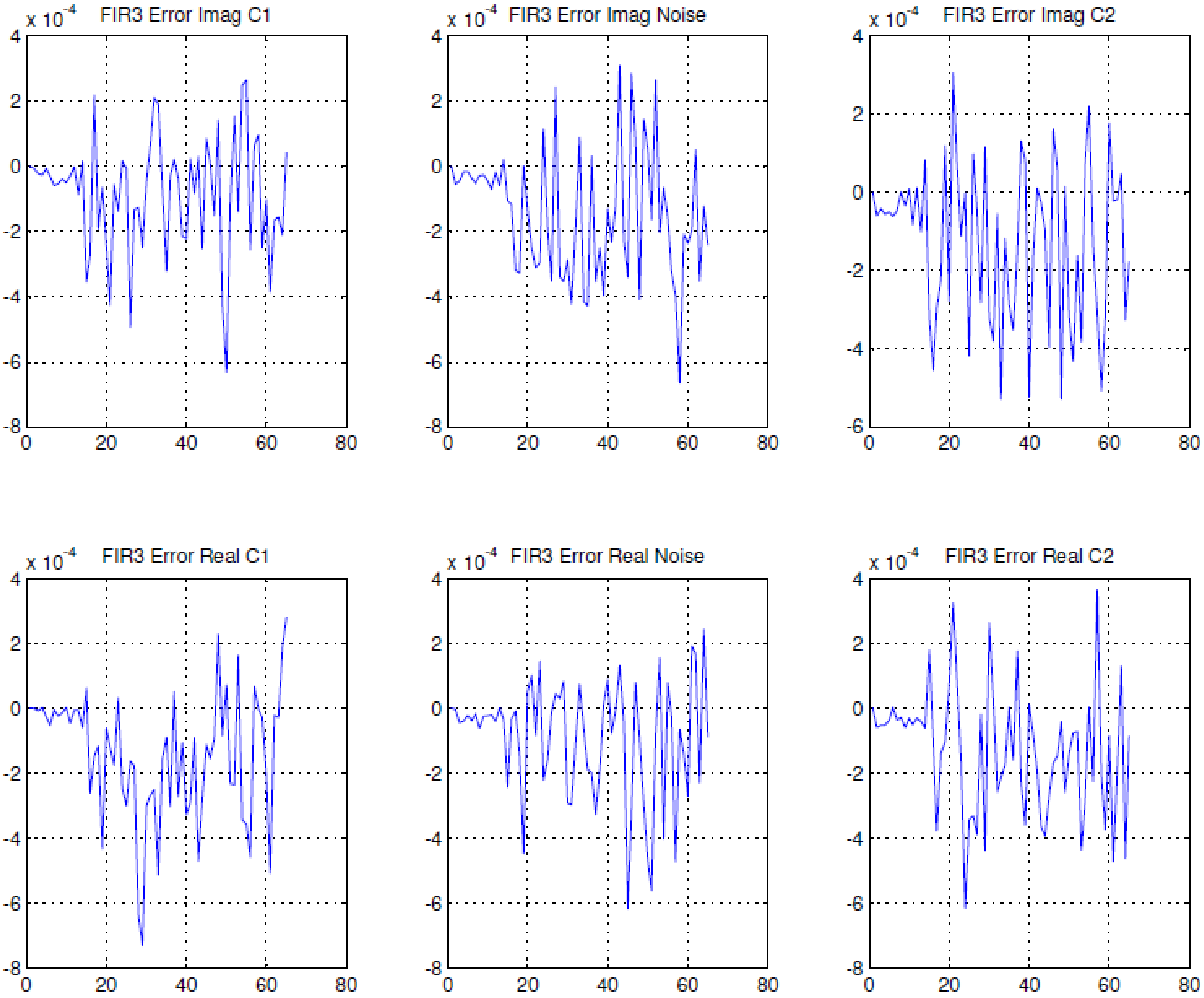
JPL DISCREET - not for distribution

FPGA Design & Implementation
Functional Verification – Outputs Comparison

Blue – floating point Matlab
(qmfir_hw.m)
Red – Hardware output
(qmfir_hw.v)
8-bit input
16-bit output



FPGA Design & Implementation
Functional Verification – Error



Functional Verification

- *Meets expectation of fixed-point error analysis*
- Additional items to be done
  - Gate Level netlist simulation
  - Test coverage
  - Real science and echo data simulation

FPGA Design & Implementation
Performance Validation

Performance Validation

- *Timing closure - 29% timing margin with space qualified part*
- *Power consumption – no requirement yet*
- Signal integrity – need to wait for integration
- *Device usage – more than 40% margin*
- SEU Mitigation – no requirement yet

PARAMETER:	**Xilinx XC2V3000 (speed grade 4)**		**Xilinx XQR2V3000 (space qualified part) (speed grade 4)**	
Speed	87.2 MHz		85.4 MHz	
	Used / Total	**%**	**Used / Total**	**%**
# of Multipliers	68 / 96	70%	68 / 96	70%
# of Slices	8,224 / 14,336	57%	8,278 / 14,336	57%
# of Flip Flops	9,680 / 28,672	33%	9,680 / 28,672	33%
# of 4 input LUTs	13,944 / 28,672	48%	14,064/ 28,672	49%
Power Consumption	DYNAMIC: 0.87W, QUIESCENT: 0.08W, TOTAL: 0.95 W (Estimate from XPower, Industrial temperature range)		DYNAMIC: 0.89W, QUIESCENT: 0.38W, TOTAL: 1.26 W (Estimate from XPower, Industrial temperature range)	

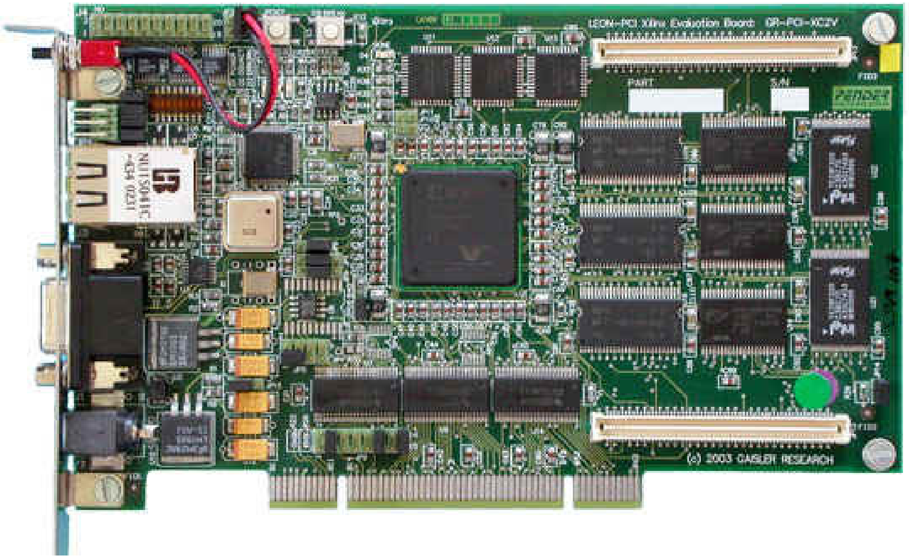
V2 Prototype Board Options

Our Choice

	Option 1 **AFX-FF1152-200**	Option 2 **GVA-325**	Option 3 **XC2V3000-4FF1152C**	Option 4 **XC2V3000-4FF1152C**	Option 5 **GR-PCI-XC2V**
Vendor	Xilinx	GV & Associates	ERST	Acromag	HighTech Global
Key Features	• XC2V3000 in FF1152 •. Onboard programmable oscillator •systemACE port	2 XC2v3000, and 2 SpartanT-II FPGAs for interface and configuration control	XC2V3000 in FF1152	XC2V3000 in FF1152	. LEON V8 processor . XC2V3000 . 64MB SDRAM . 10/100 Ethernet
Price ($k)	2.3	13.5	3.3	$4.2+$750	4.9
Lead Time	discontinued	4 weeks	7~9 weeks	1 week	1 week

V2 Prototype Board

GR-PCI-XC2V Evaluation Board


LEON-PCI Xilinx Evaluation Board GR-PCI-XC2V
(c) 2003 GAISLER RESEARCH

- Virtex-II XC2V3000-FG676-4 FPGA
- On-board FPGA configuration PROMs
- 8 Mbyte Flash PROM
- 1 Mbyte SRAM
- 64 Mbyte PC133 SDRAM
- Ethernet PHY 10/100 Mbit transceiver
- 33 MHz, 32-bit PCI interface
- Standard RS-232 UART port for DSU
- 120-pins memory and custom I/O expansion connectors
- JTAG and slave-serial FPGA programming capability
- Supports LEON2 core frequencies up to 65 MHz

Testbench Construction & Demonstration

Testbench & Data Flow Descriptions

FPGA CORES

Input BRAM

- Size of 6KB
- Holds the input data from Host Machine until the digital filter core is ready to process the data

Output BRAM

- Size of 200B each
- Holds the processed data until the Test Station is ready to retrieve the data

UART I/F

- Serves to collect commands and data from test station

240MHz Digital Filter

- 4 streams input data at 60MHz
- 6 streams output data at 1.2MHz
- Filter running at 60MHz

COMPONENTS

PROM – each 4MB

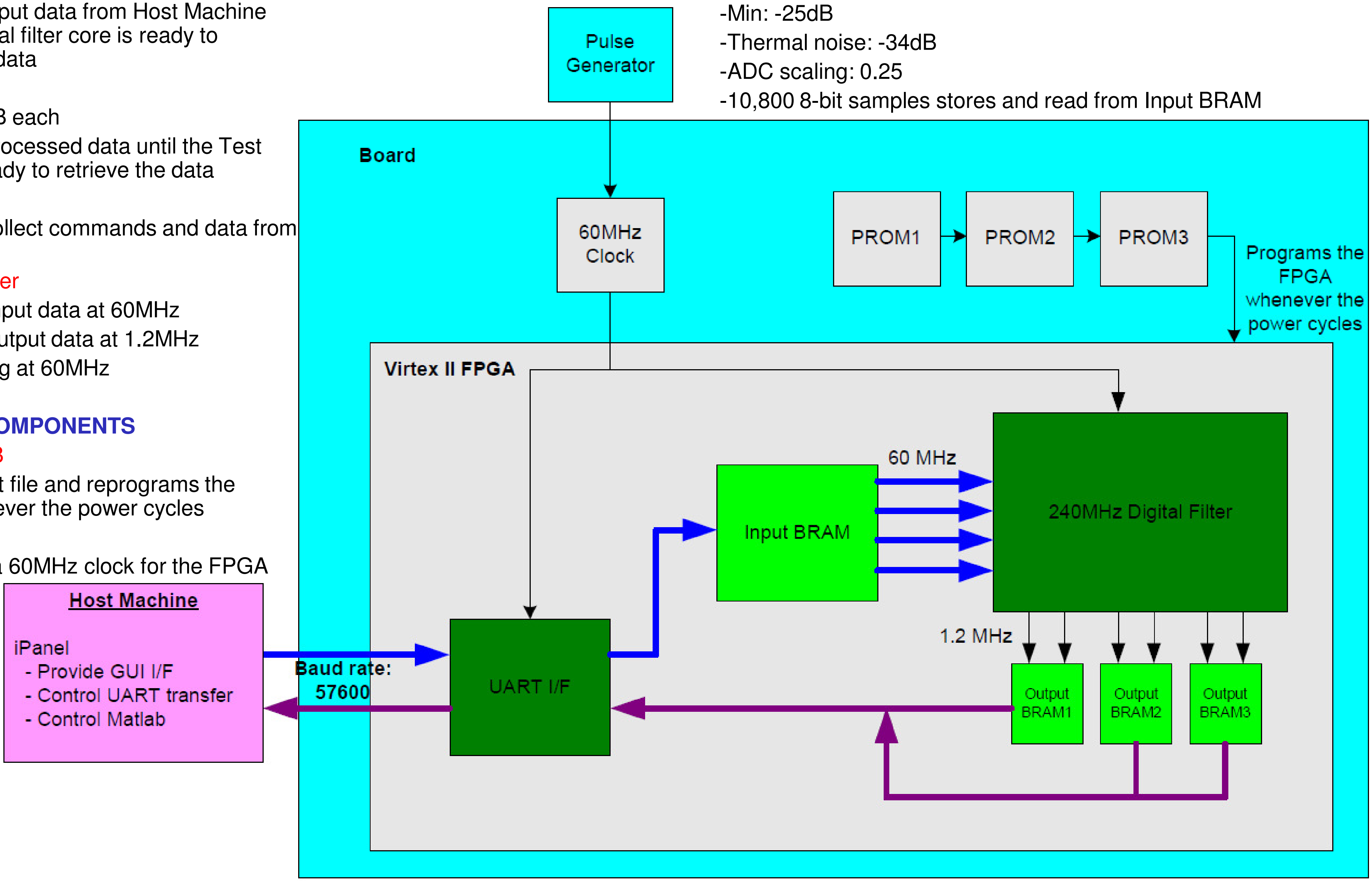
- Holds the bit file and reprograms the FPGA whenever the power cycles

Pulse Generator

- Generates a 60MHz clock for the FPGA

Test Data

- Synthetic data generated by Matlab
- Max: 0dB
- Min: -25dB
- Thermal noise: -34dB
- ADC scaling: 0.25
- 10,800 8-bit samples stores and read from Input BRAM



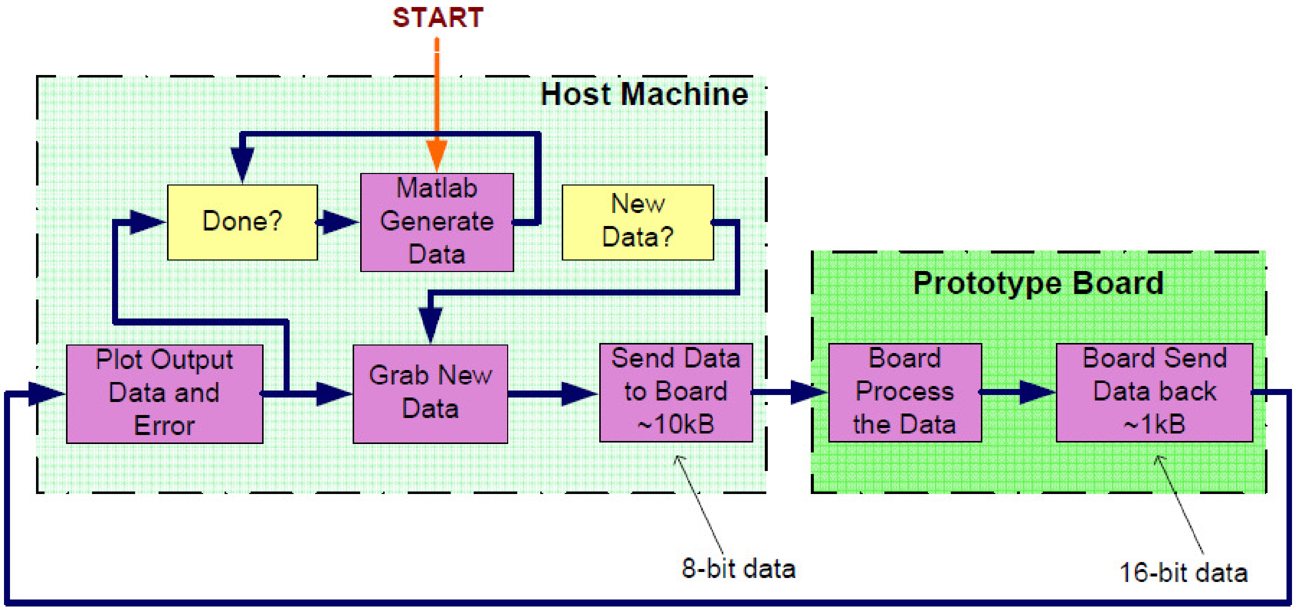
Host Machine

iPanel
- Provide GUI I/F
- Control UART transfer
- Control Matlab

Testbench Construction & Demonstration

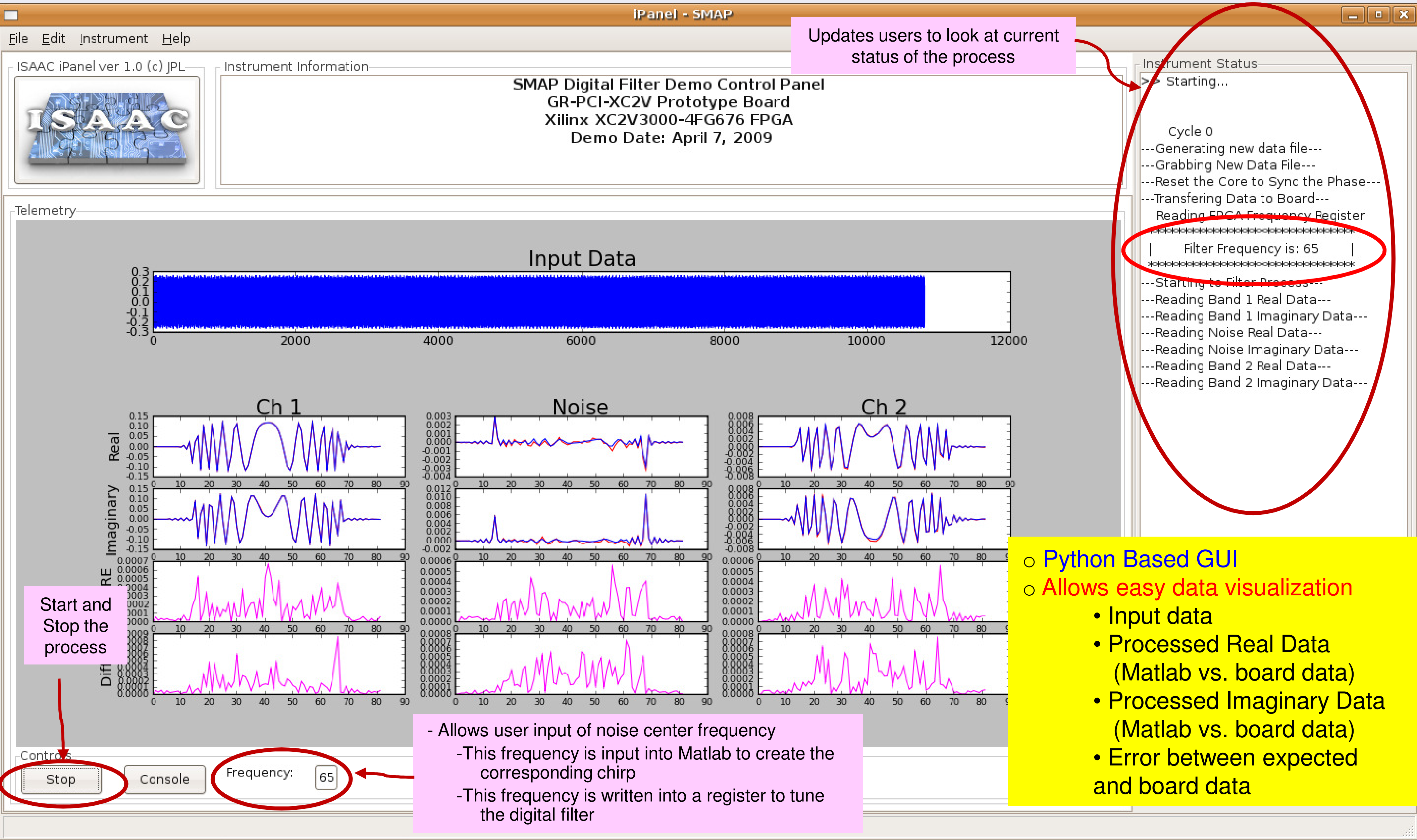
Demo Flow Operations

- Host Machine
  - Runs on Linux with the iPanel in Python
  - Calls Matlab to generate synthetic data
  - Sends data to the board over UART to be stored on the Input BRAM inside the FPGA
  - Starts the Digital Filter core by writing to a register
- Prototype Board
  - When the register is written, it sends a signal to the Digital Filter Core to grab data from the Input BRAM to process the data
  - Digital Filter core process all of the data and store the outputs in the Output BRAM
- Host Machine
  - Grabs the data from the Output BRAM through UART
  - Plots the output data vs. the expected output data generated by Matlab
  - Plots the error between the output data and the expected data



JPL DISCREET - not for distribution

Testbench Construction & Demonstration
iPanel



SMAP Digital Filter Demo Control Panel
GR-PCI-XC2V Prototype Board
Xilinx XC2V3000-4FG676 FPGA
Demo Date: April 7, 2009

```
>> Starting...


   Cycle 0
---Generating new data file---
---Grabbing New Data File---
---Reset the Core to Sync the Phase---
---Transfering Data to Board---
   Reading FPGA Frequency Register
*******************************
|      Filter Frequency is: 65      |
*******************************
---Starting to Filter Process---
---Reading Band 1 Real Data---
---Reading Band 1 Imaginary Data---
---Reading Noise Real Data---
---Reading Noise Imaginary Data---
---Reading Band 2 Real Data---
---Reading Band 2 Imaginary Data---
```

Updates users to look at current status of the process

Start and Stop the process

- Allows user input of noise center frequency
  - This frequency is input into Matlab to create the corresponding chirp
  - This frequency is written into a register to tune the digital filter

- ○ Python Based GUI
- ○ Allows easy data visualization
  - Input data
  - Processed Real Data (Matlab vs. board data)
  - Processed Imaginary Data (Matlab vs. board data)
  - Error between expected and board data

JPL DISCREET - not for distribution

Lessons Learned

- Lessons Learned
  - Positive Lesson: enhanced productivity due to IP reuse in an ISAAC framework based design methodology (9 months for 60MHz design down to 3 months for 240MHz design)
  - Negative Lesson: insufficient vendor support for discontinued parts (board, FPGA)
  - Focus on scope
  - Effective requirement communication between algorithm and hardware designers
  - Better synergy between technology development and project needs

JPL DISCREET - not for distribution

Conclusion & Future Study

- The Testbench Construction and Demonstration work has
  - Verified and validated the algorithm design and FPGA implementation
  - Successfully shown that the 240MHz digital filter design is feasible on the Xilinx Virtex-II FPGA
  - Successfully shown that the 240MHz digital filter is able to tune to different center frequencies with spacing of 1MHz

- Suggestions for further study (as applied to SMAP)
  - Expand frequency tuning design to include frequency spacing of 0.5MHz
  - Design FPGA interfaces with ADC, CTU, BFPQ, and Data Formatter
  - Work on test plan and implement full test coverage
  - Further reduce multiplier count with VKCMs (constant multipliers implemented in LUTs)
    - FIR0 can save 12x2 full multipliers by replacing with VKCMs
  - How to detect error in computation
  - SEU mitigation