

---

# FPGA Based Digital Signal Processing – Applications & Techniques

---

Nathan Eddy

Fermilab

BIW12 Tutorial

---

# Outline

- Digital Signal Processing Basics
- Modern FPGA Overview
- Instrumentation Examples

---

# Advantages of Digital Signal Processing (DSP)

- NO DRIFT – due to temperature or age
- ACCURACY – defined by number of bits
- PREDICTABILITY – from simulation
- PERFORMANCE
  - Linear Phase Response possible
  - Adaptability in terms of resources
- PRODUCTION – identical units, no tuning
- FLEXIBILITY – via firmware modifications
- DYNAMIC RANGE

---

# For Beam Instrumentation

- Need to work with analog input signals
  - Beam pickups, Schottky detectors, Torroids, etc
  - Requires Analog to Digital Converters (ADCs)
- Need to produce analog output signals
  - To act on the beam – RF, kick signals, etc
  - Require Digital to Analog Converters (DACs)
- The effectiveness of FPGA solutions is largely dominated by the performance of the converters

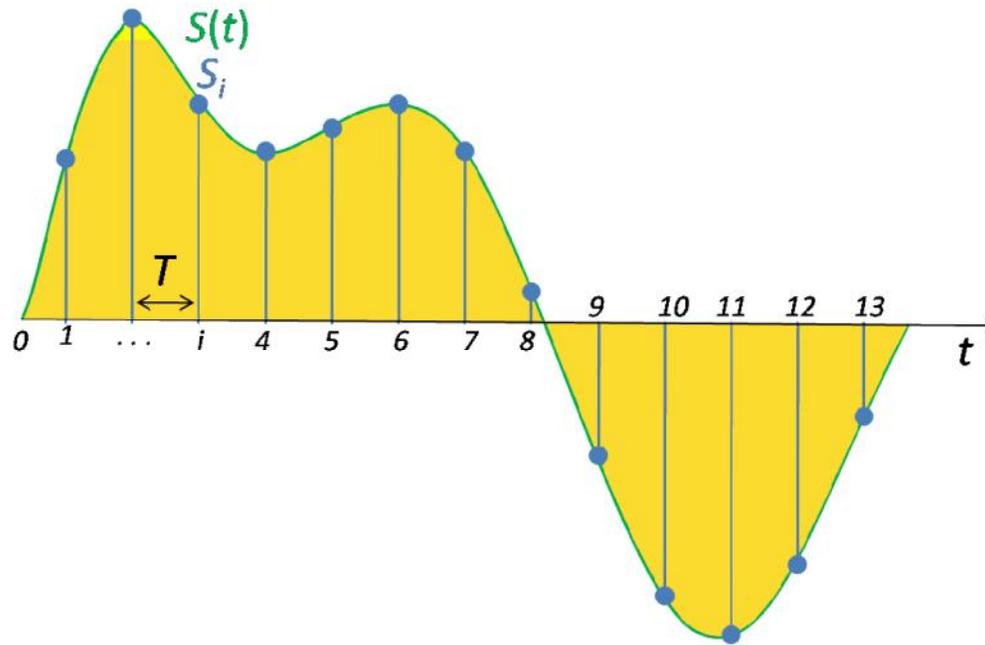
---

# Practical Limitations of Sampling

- Common sources of sampling error
  - Aliasing
  - Quantization
  - Sample Clock Jitter
- Can be characterized by their impact on the Signal to Noise Ratio (**SNR**) of the sampled signal
  - $SNR \sim \log(S_a/N_a)$
  - Typically expressed in decibels (**db**)

# Discrete Time Sampling

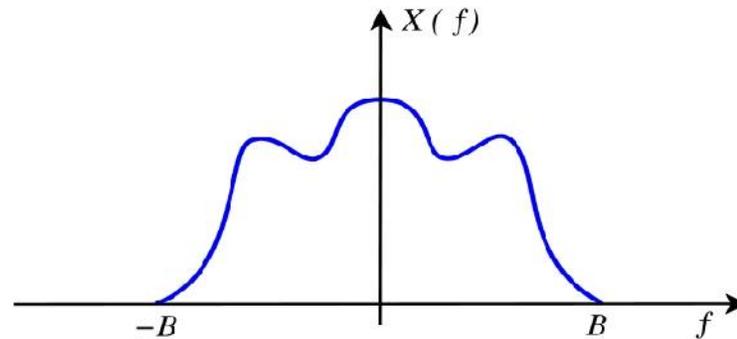
- $S_i = S(t) * \delta'(t)$  where  $\delta'(t) = \sum \delta(t-nT)$



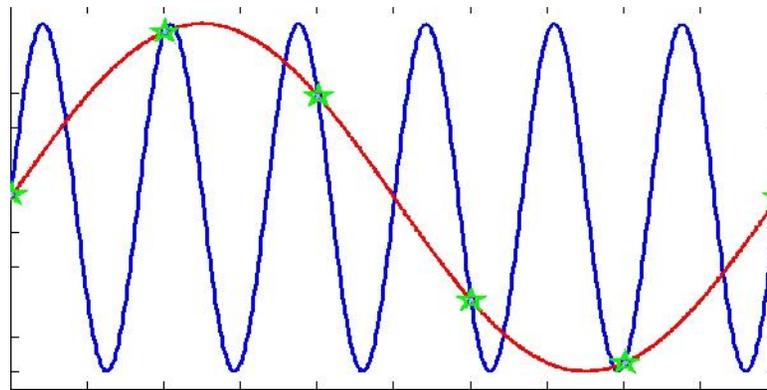
- The sequence  $S_i$  is the sampled version of  $S(t)$
- The sampling frequency  $F_s$  is  $1/T$

# Sampling Theorem

- For ideal reconstruction  $F_s > 2B$  where  $B$  is the highest frequency in the signal of interest

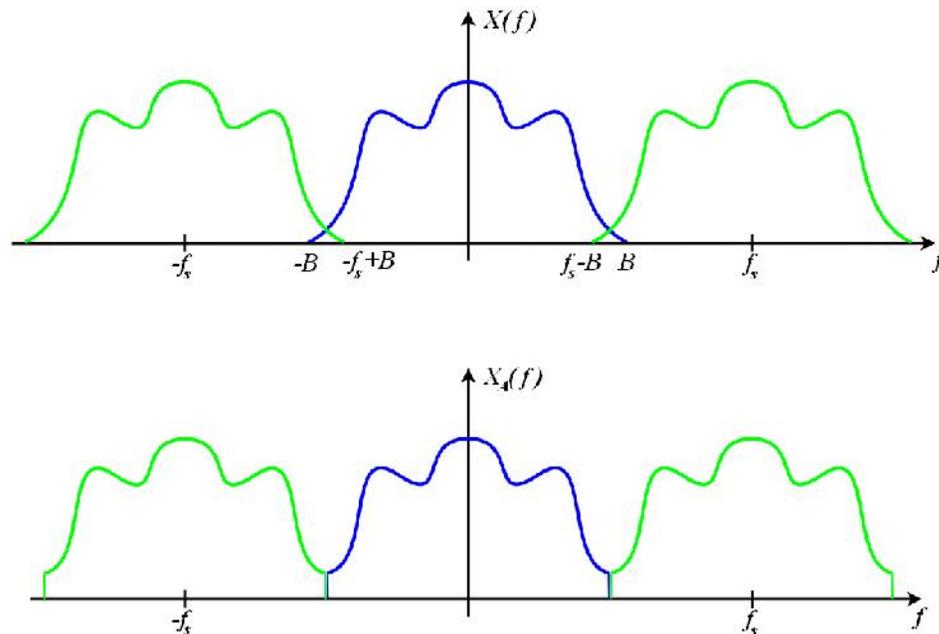


- Sampling ambiguity



# Aliasing

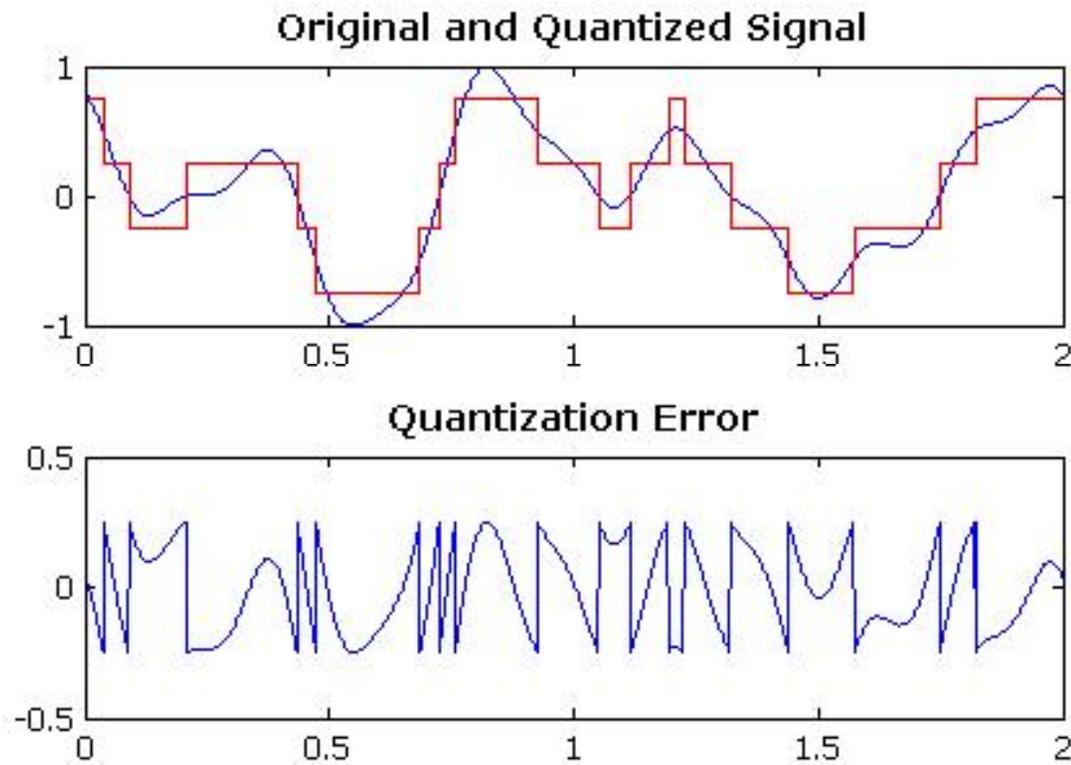
- Signals at frequencies larger  $F_s/2$  than will “alias” into the first Nyquist band



- Undersampling makes use of this effect

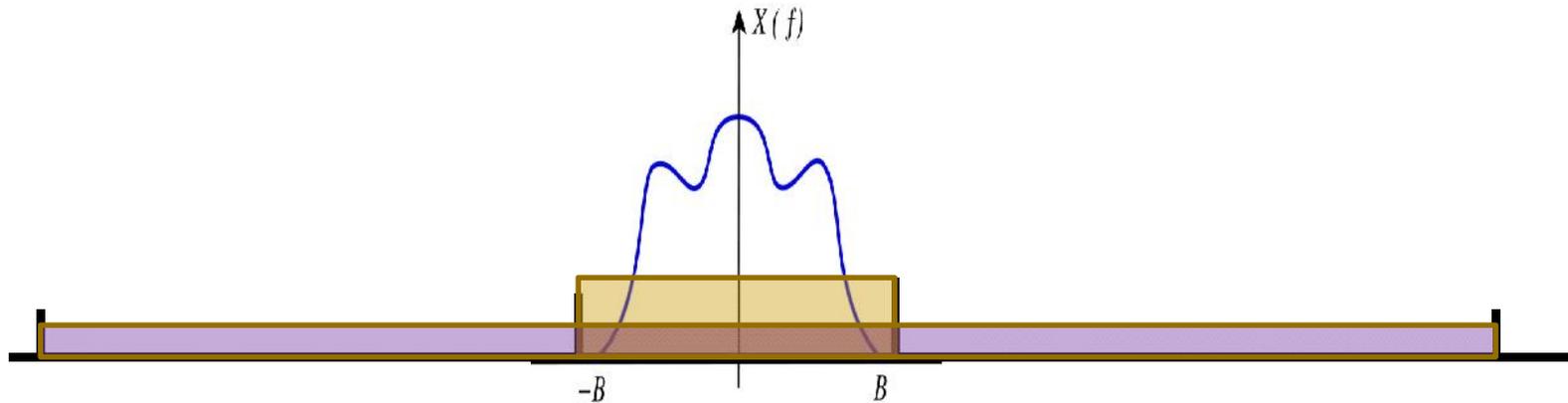
# Quantization

- Results in signal noise



- $SNR_{db} = 1.76 + 6.02N$ ,  $N$ =number of bits

# Oversampling Improves SNR



- Gain  $SNR_{db} = 10\log(R)$  where  $R = F_s/2B$ 
  - $F_s$  is the sampling Frequency
  - $B$  is signal bandwidth
- Also relaxes the requirements on the anti-aliasing analog filter

# Sampling Clock Jitter

- Another source of signal noise
  - Proportional to maximum signal frequency,  $f_{max}$
  - RMS sampling clock jitter,  $T_a$

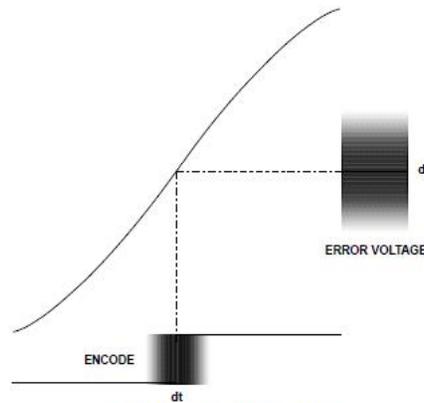
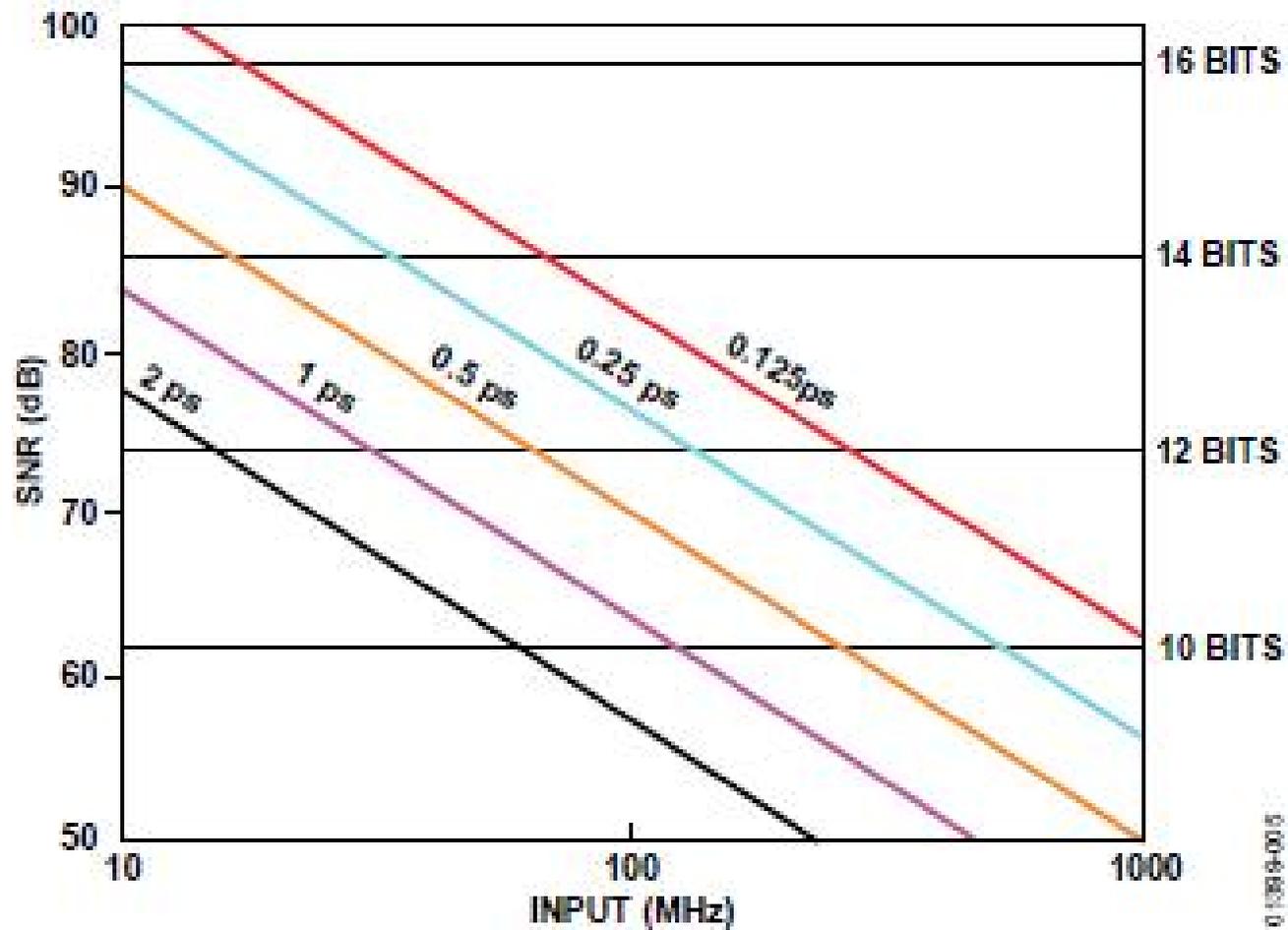


Figure 1. RMS Jitter vs. RMS Noise

- $SNR_{db} = 20 \log(1/(2\pi f_{max} T_a))$
- A significant effect for undersampling applications

# Theoretical SNR Comparison



---

# The Z domain

- A sampled sequence,  $x(n)$  can be represented

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n}$$

- Where  $z^{-1}$  is the unit delay related to the sample period (T)
- $X(z)$  is the z-transform of  $x(n)$

# Difference Eq & Transfer Function

- A constant coefficient difference equation is a recursive relationship where-by the output of a discrete time system can be calculated using a combination of past output values and past and present input values

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

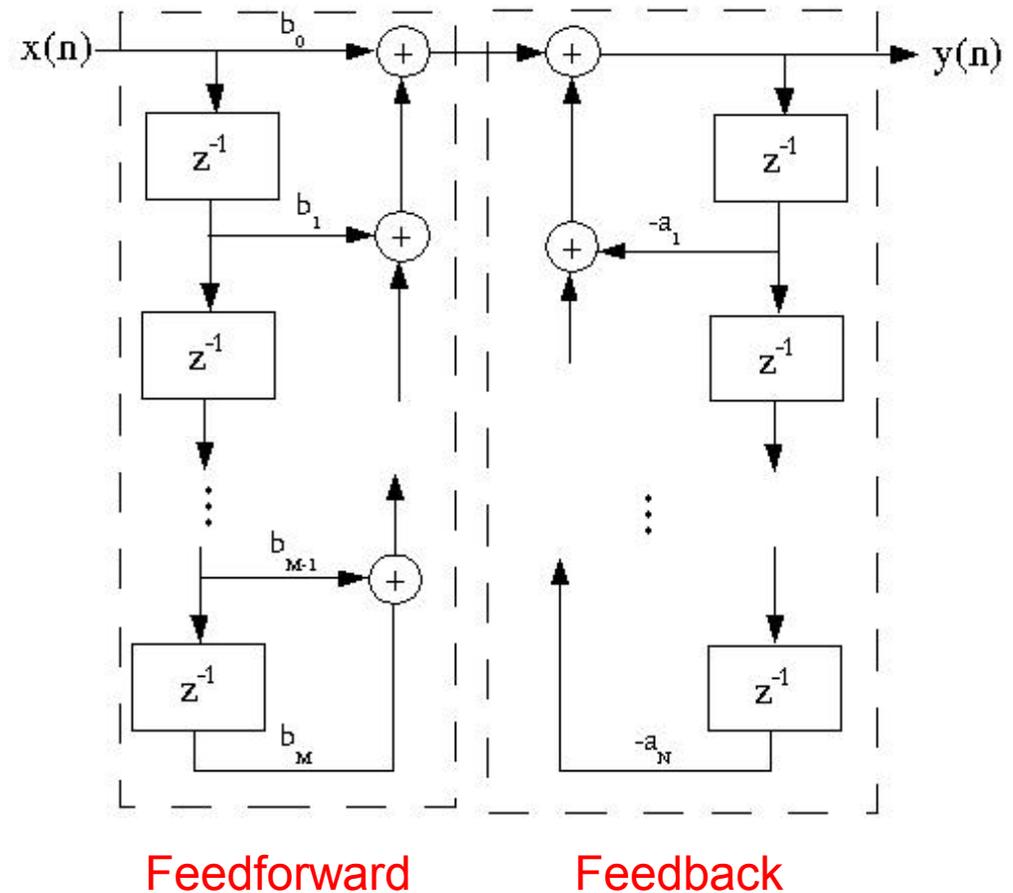
- Gives z-domain transfer function,  $H(z)$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^N b_k z^{-k}}{\sum_{k=0}^M a_k z^{-k}}$$

- Digital filters are specified by  $\mathbf{a}, \mathbf{b}$  coefficients

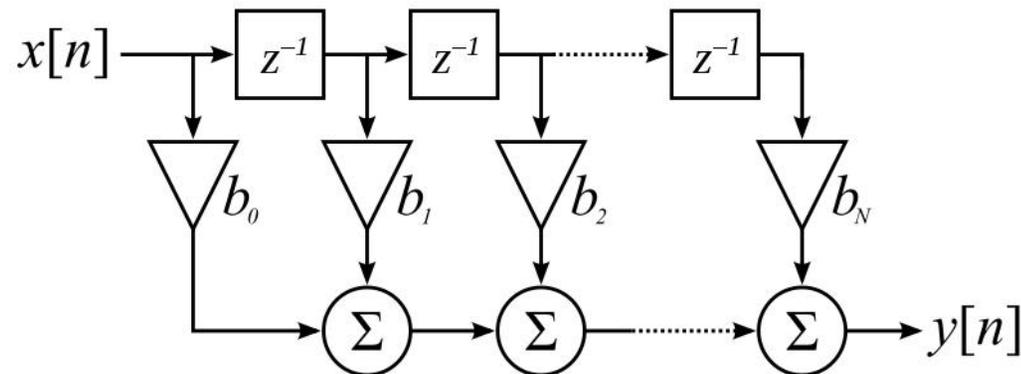
# Infinite Impulse Response (IIR) Filter

- Feedback provides infinite impulse response
  - $a_n$  non-zero for  $n \geq 1$
- Properties
  - Very efficient
  - Non-linear Phase
  - Can be unstable



# Finite Impulse Response (FIR) Filter

- Nth Order FIR filter



- Just the feedforward block

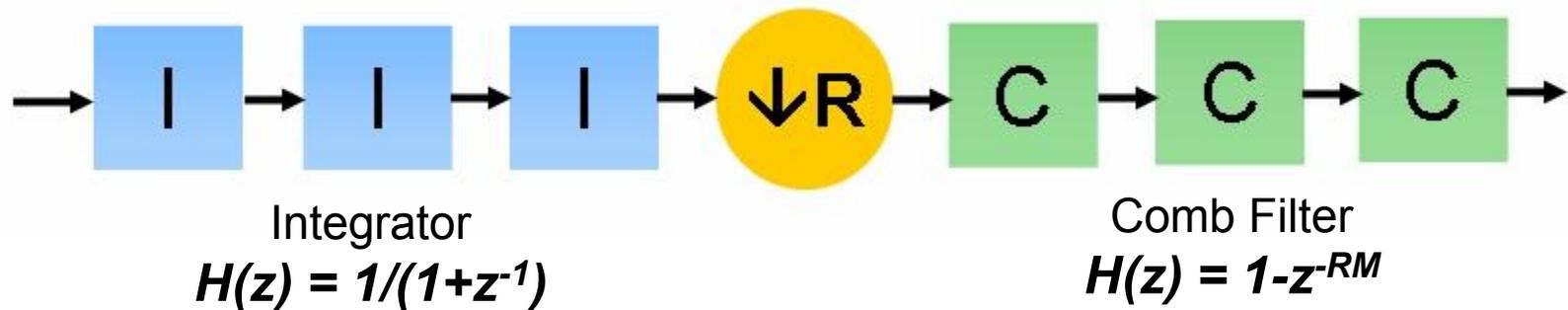
- $a_0=1$ , all others zero

- Has linear phase if coefficients are symmetric

- That is  $\{b_0, \dots, b_N\} = \{b_N, \dots, b_0\}$

- No analog equivalent

# Cascade Integrating Comb (CIC) Filter



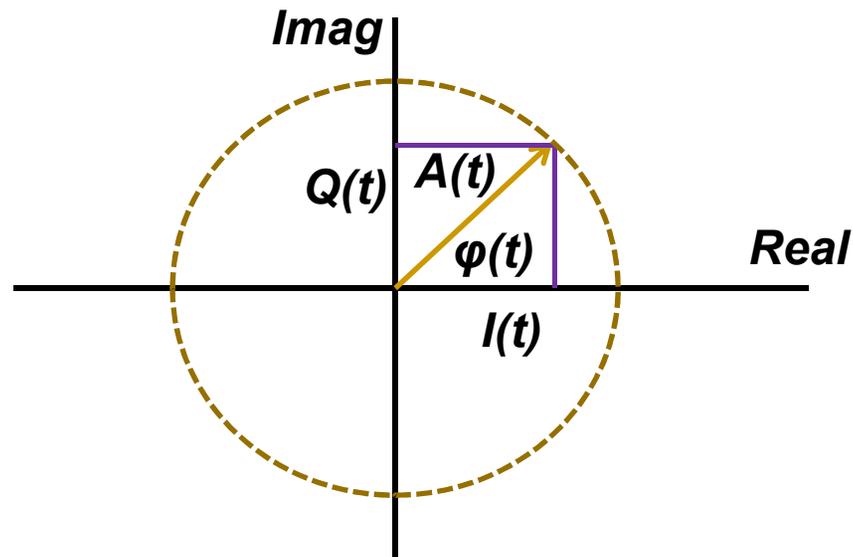
- For an Nth Order CIC Filter, the transfer function is

$$H(z) = \frac{(1-z^{-RM})^N}{(1+z^{-1})^N} = \left( \sum_{k=0}^{RM-1} z^{-k} \right)^N$$

- Equivalent to N FIR filters with unit coefficients -> symmetric
- Linear Phase even though it has infinite response filter sections
- Used as very efficient way to filter and change rate
  - Can be used as Interpolation filter by reversing I & C sections

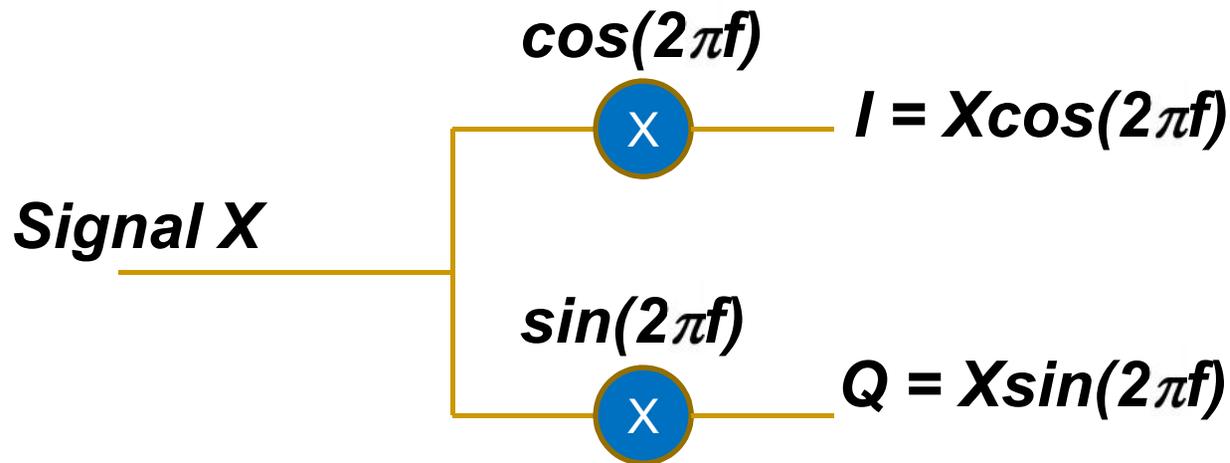
# Quadrature Signals, $I$ & $Q$

- Complex signal representation



- In-Phase (real) & Quadrature (imaginary)
  - $I(t) = A(t)\cos(\varphi(t))$
  - $Q(t) = A(t)\sin(\varphi(t))$

# Digital Down-Conversion



- Multiply a signal  $X$  by  $\sin(2\pi f)$  &  $\cos(2\pi f)$  where  $f$  is the the down-conversion frequency
  - Shifts the spectral components of  $X$  down in frequency by  $f$
- Commonly followed by decimating low pass filter
  - Remove sum frequency components
  - Process narrow bandwidth signals at lower rate in baseband

---

# Discrete Fourier Transform (DFT)

- Discrete Fourier Transform of length N

$$X(k) = \sum_{n=0}^{N-1} x(n) [\cos(2\pi nk/N) + i\sin(2\pi nk/N)]$$

- $X(k)$  determines the complex signal contribution of the frequency in the composition of  $x(n)$
- Phase or direction (forward, inverse) given by the sign of the imaginary term

---

# Fast Fourier Transform (FFT)

- Works as a bank of band-pass filters
    - The output magnitude from each filter is proportional to the input energy in each band
  - An  $N$  point DFT requires  $N^2$  operations
  - Can compute the  $N$  point DFT as two  $N/2$  point DFTs
    - Extrapolate to the limit where  $N$  is a power of 2 for an  $N$  point FFT
  - An  $N$  point FFT requires  $N/2 \log_2(N)$  operations
  - Note, FFT are almost always implemented as a power of 2 but can be any prime factor
-

---

# Outline

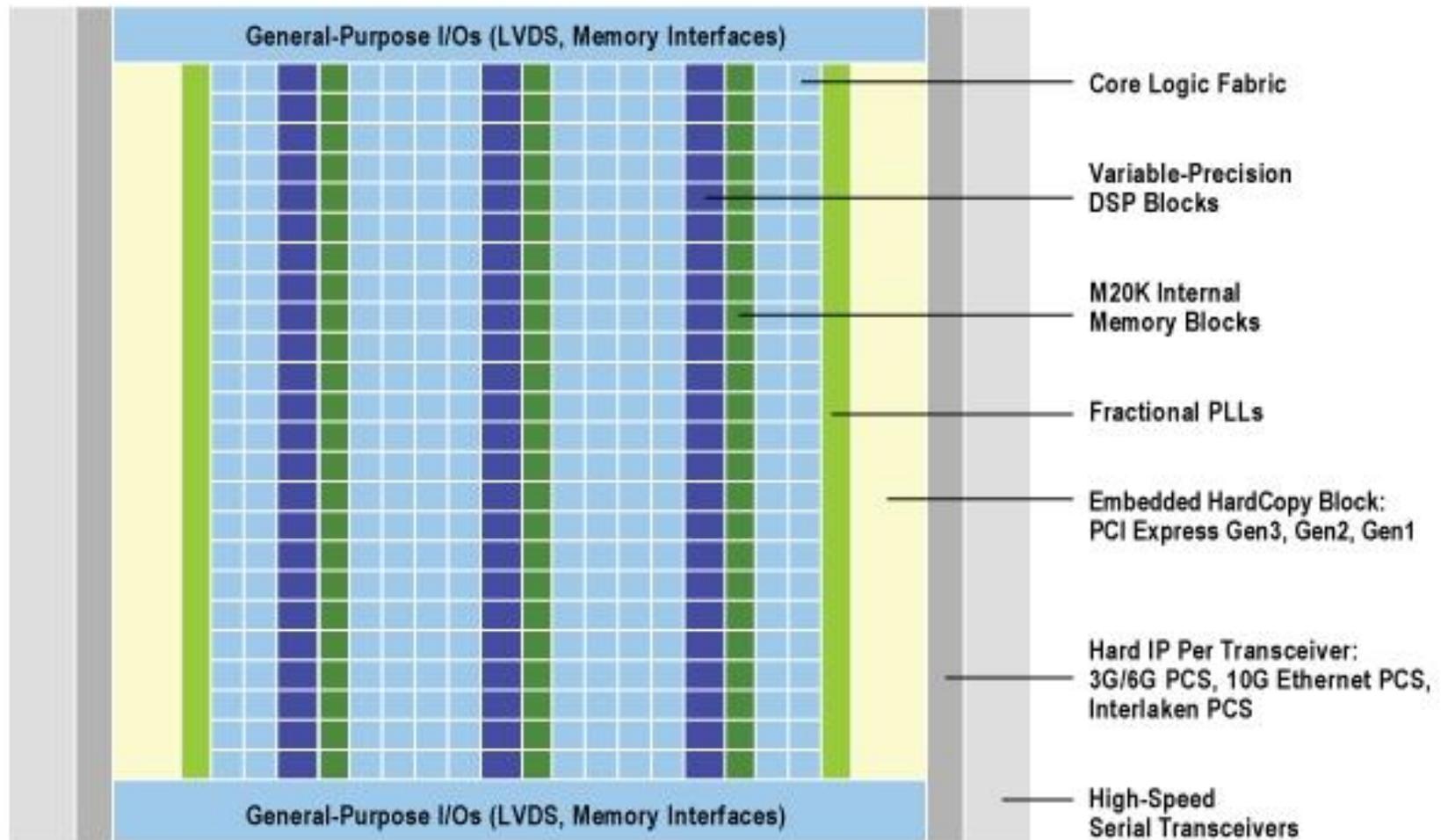
- Digital Signal Processing Basics
- **Modern FPGA Overview**
- Instrumentation Examples

---

# FPGA Advantages for DSP

- Parallel processing power (Speed)
    - Able to consume large quantities of data
    - Alleviate bandwidth bottlenecks in front-ends
  - Pipeline architecture
    - Efficient digital filter implementation
    - Perform operations under strict timing (latency) control
  - Dedicated DSP (multiplier) blocks
  - Flexibility
    - Clocking synchronous with accelerator
    - Easy integration with ADCs and DACs
    - Easy to modify algorithms and functionality
-

# High End FPGA Architecture



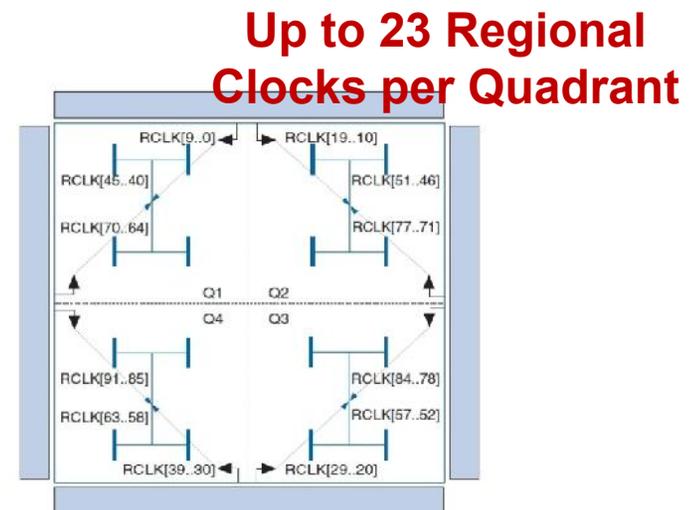
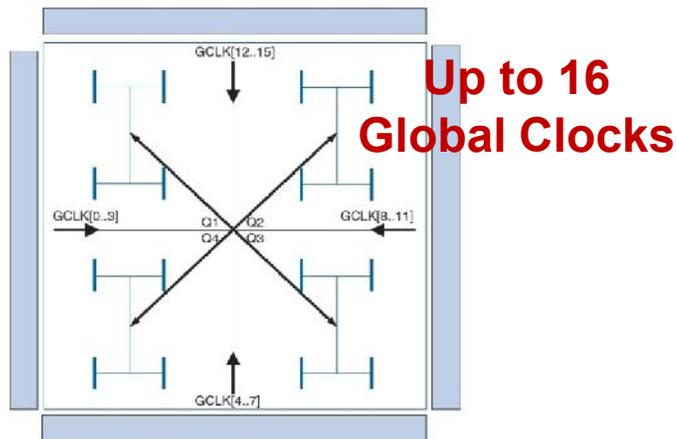
---

# Flexible I/O Options

- Dedicated hardware support for almost all standard I/O protocols
- Transceivers with equalization for fast differential serial I/O
  - PCI Express, RapidIO, etc
  - Support for over 20GBps links!
- External memory interface support
  - DDR, DDR2, DDR3
- Speed and quantity of available I/O -> \$\$

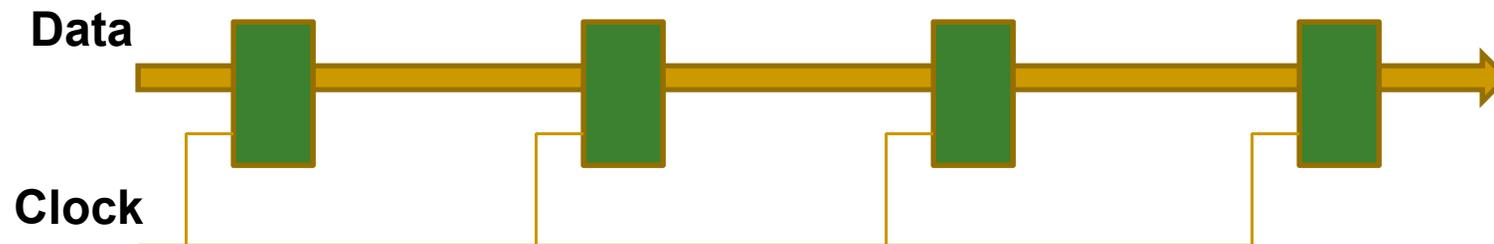
# Clock Domains and Distribution Networks

- Multiple PLLs and DLLs for sophisticated clock management
  - Support for run-time reconfiguration
  - Fractional PLLs for arbitrary clock synthesis
- Dedicated global and regional clock distribution networks
  - Support high rate synchronous design
  - Support parallel & pipelined design



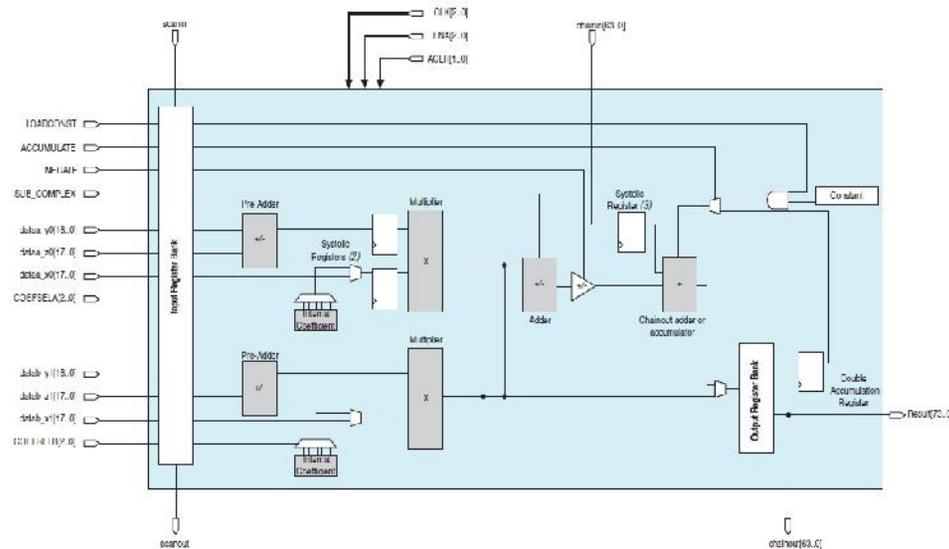
# Synchronous Design - Pipelining

- Reliable designs require synchronous design principles
  - Increase speed and bandwidth
  - Trade resources and latency
- Simple Delay Pipeline



# Digital Signal Processing (DSP) Blocks

- Based upon dedicated multiplier blocks



- Provide efficient flexibility and power
  - Fully customize, dedicated support for FIRs, FFT, etc to optimize for speed and efficiency
- Modern blocks provide capability for 32 or 64 bit floating point operations

# Integer Representation

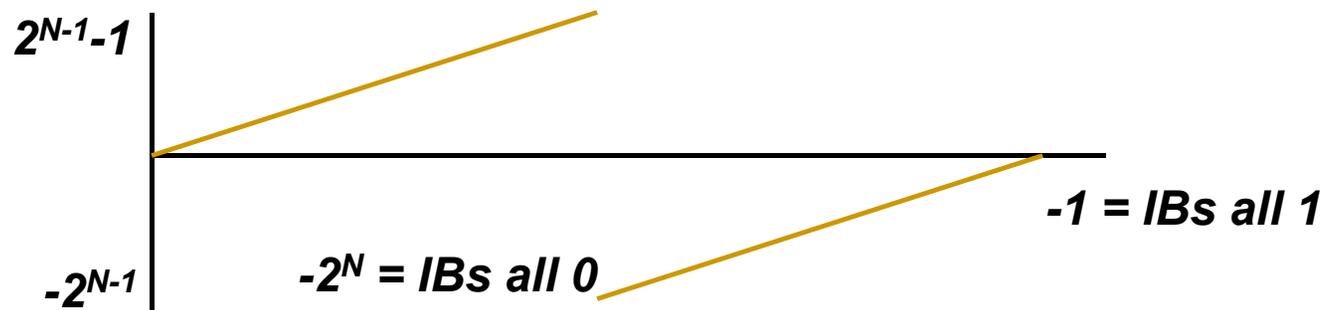
- 2s Complement  $N$  Bit representation



- Positive numbers just simple binary representation
- Negative numbers are binary number that when added to a positive number of the same magnitude equals zero

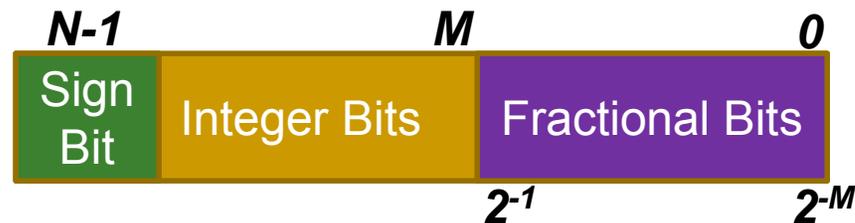
- Convenient Properties

- Simple arithmetic – addition & subtraction the same



# Fixed Point Signal Processing

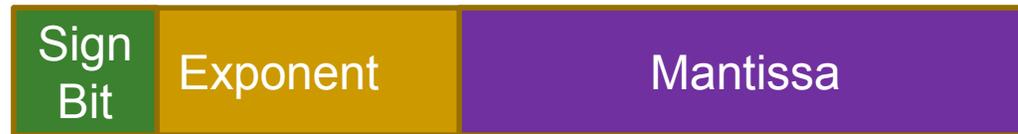
- Bit widths double on each multiply!
- Need to control bit widths for efficient resource use
  - Truncation or Rounding
  - Saturation or Roll-over
- Fixed Point N Bit Representation



- Can facilitate keeping track of bit widths

# Floating Point Representation

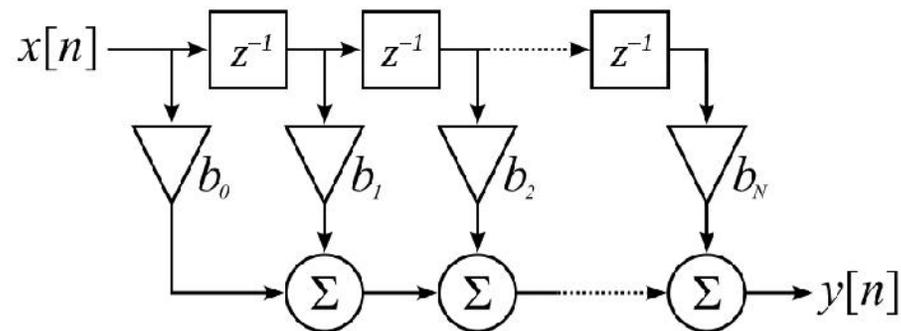
- IEEE-754 Standard Followed



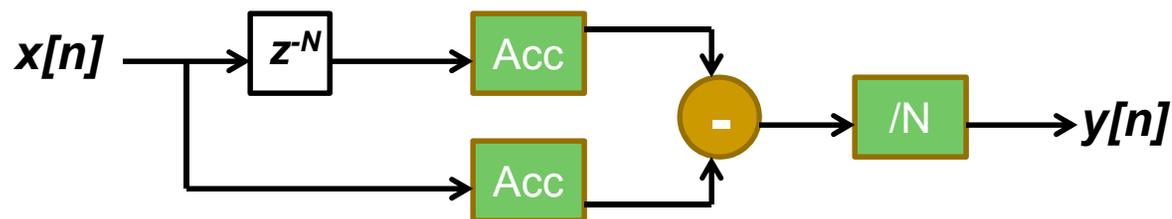
- 32 bit – Single Precision
  - 8 bit Exponent & 23 bit Mantissa
- 64 bit – Double Precision
  - 11 bit Exponent & 52 bit Mantissa
- Single-Extended Precision
  - Exponent and Mantissa widths are not fixed
  - Minimum 11 Exponent Bits (Exponent < Mantissa)
  - Minimum 31 Mantissa Bits
  - Total Bits at least 43 up to 64

# Running Average Example

- Can implement an  $N$  sample running average as an FIR with  $N$  taps with coefficients of  $1/N$



- Can be efficiently implemented using two accumulators and a delay line



---

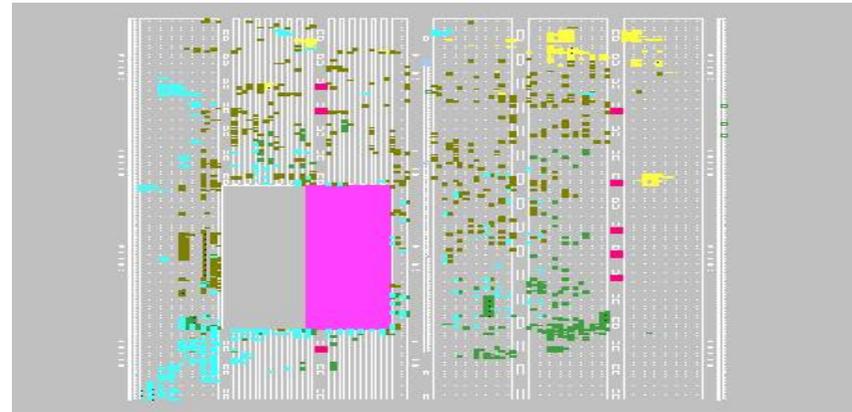
# Intellectual Property (IP) Cores

- Functional cores which can be used to greatly speed up and simplify the design process
- Each FPGA manufacturer provides cores for all basic components
  - PLLs, RAM, FIFOs, Flip-Flops, etc
  - Accumulators, Add, Sub, Multiply, Divide, etc
  - Take advantage of chip resources
- Advanced cores available for almost any task
  - Simple implementation via GUI parametrization

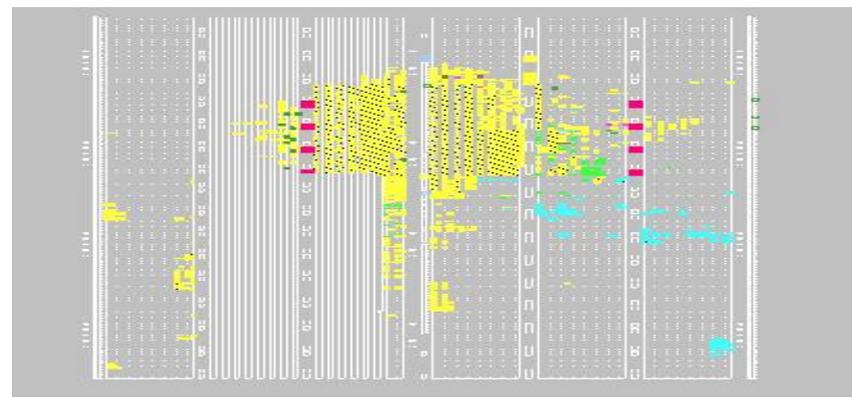
# Embedded Systems

- Hard Core Embedded processor is a dedicated physical component of the chip, separate from the programmable logic
  - 2-4 times faster than Soft Core
  - More efficient if you need a processor
- Soft Core Embedded processor is built out of the programmable logic on the chip
  - A 32 bit RISC processor uses about few percent of total resources
  - Have option to re-allocate resources if processor not needed

Hard Core Resource Allocation



Soft Core Resource Allocation



---

# System On a Programmable Chip (SOPC)

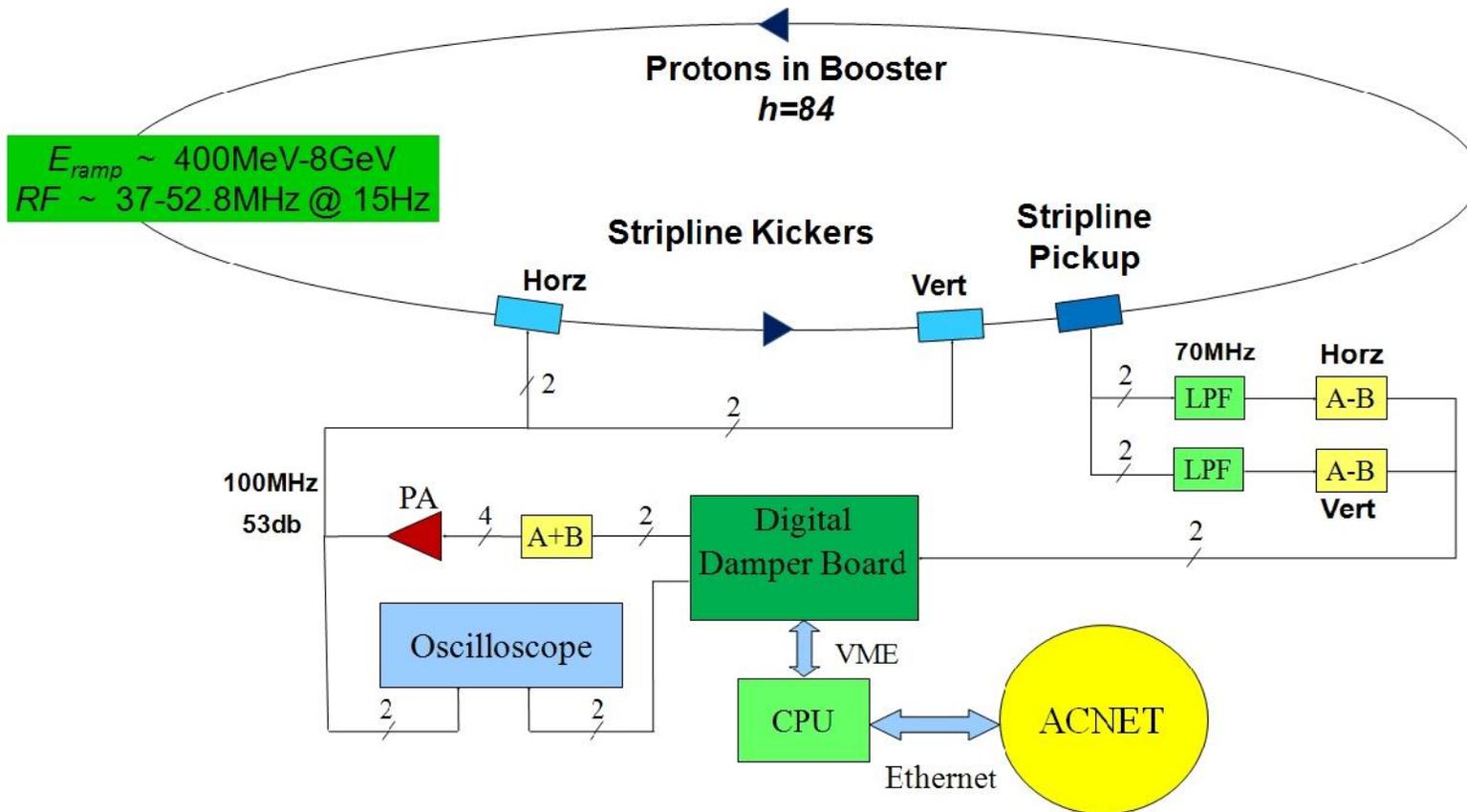
- A wide variety of design tools and options available
  - Pure HDL entry – still possible
- Focus upon developing System On a Programmable Chip
  - Aim to simplify the design process as chip architectures become more complicated
- Tightly couple design and simulation at all levels

---

# Outline

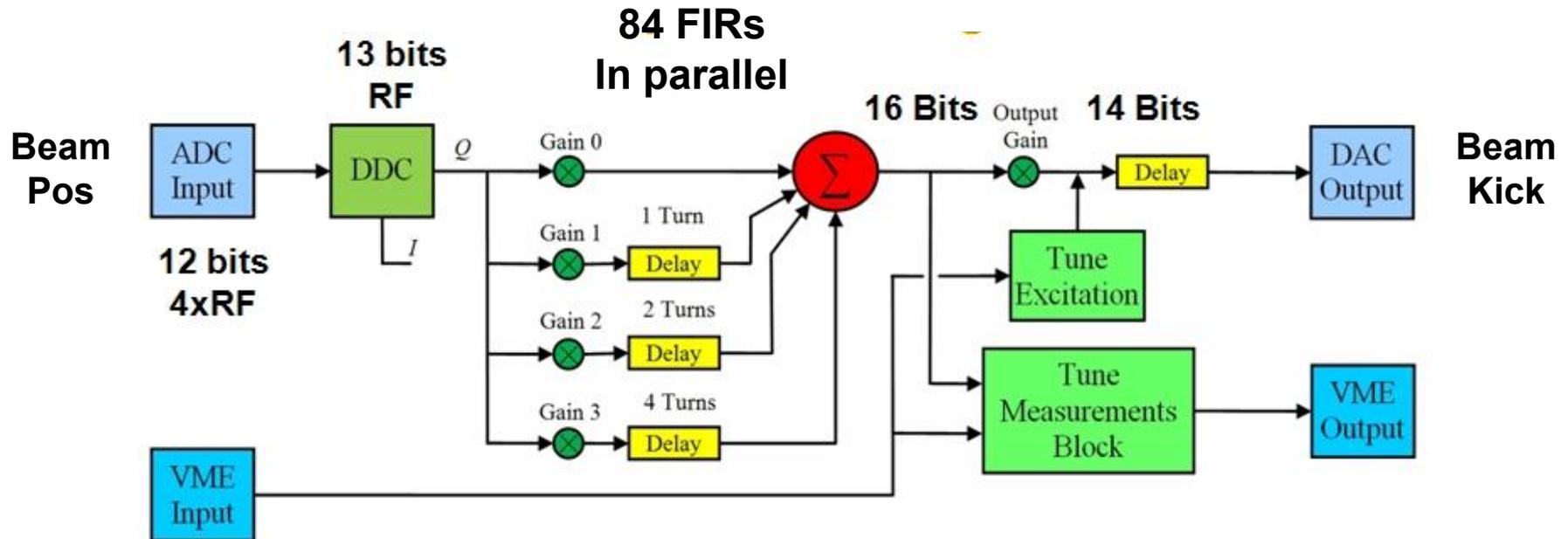
- Digital Signal Processing Basics
- Modern FPGA Overview
- **Instrumentation Examples**

# Fermilab Booster Digital Damper



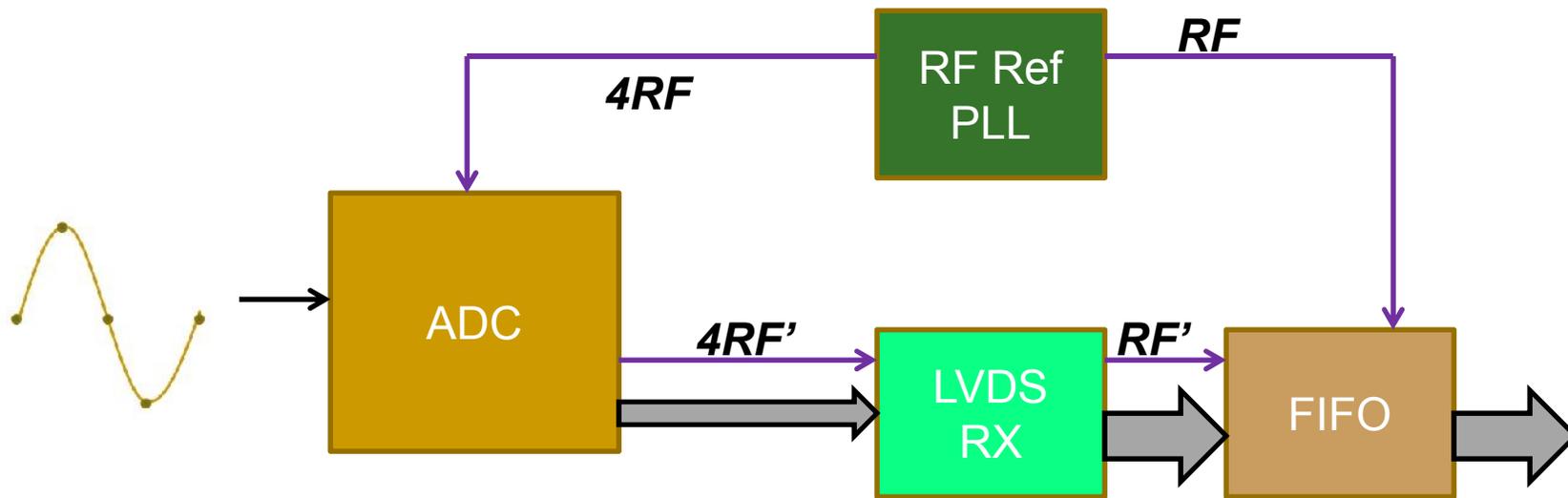


# Firmware Overview



- Design operates synchronous with Booster RF sweeping from 37MHz to 53MHz
- Provides bucket by bucket damping on all 84 Booster bunches

# ADC Input clocked at $4*RF$



- ADC clock is shifted to provide ideal sampling
- FIFO used to shift data from ADC clock to internal clock

---

# RF Reference PLL

- Use one of the Enhanced PLLs
  - Dedicated clock pin for RF Reference
  - Directly connected to clock output pins for each ADC clock
- Real-time reconfigurability
  - Use to adjust phase of ADC clock in  $15^\circ$  increments ( $\sim 200\text{ps}$  @ 50Mhz)
- Instantiated with parametrized IP

# PLL Instantiation

**ALTPLL**

Currently selected device family: Stratix II  
 Match project/default

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA | 5 Summary

General/Modes > Inputs/Lock > Bandwidth/SS > Clock switchover >

**pll\_rfclk1**

inclk0 frequency: 45.000 MHz  
Operation Mode: No Compensation  
PLL Type: Enhanced PLL

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	4/1	0.00	50.00
c2	2/1	0.00	50.00
c3	6/1	0.00	50.00
c4	4/1	0.00	50.00
c5	4/1	0.00	50.00

Inputs: jnclk0, areset, scanclk, scanread, scanwrite, scandata  
Outputs: c0, c1, c2, c3, c4, c5, scandataout, scandone, locked

Stratix II

Able to implement in Enhanced PLL

**General**

Target this configuration for migration to the HardCopy II device family (Any)

Which device speed grade will you be using? 5

Use military temperature range devices only

What is the frequency of the inclk0 input? 45.00 MHz

Set up PLL in LVD5 mode (Data rate: Not Available Mbps)

**PLL type**

Which PLL type will you be using?

Fast PLL  
 Enhanced PLL  
 Select the PLL type automatically

**Operation mode**

How will the PLL outputs be generated?

Use the feedback path inside the PLL

In Normal Mode  
 In Source-Synchronous Compensation Mode  
 In Zero Delay Buffer Mode  
 Connect the fbmimic port (bidirectional)

With no compensation  
 Create an 'fbmimic' input for an external feedback (External Feedback Mode)

Which output clock will be compensated for? c0

Cancel < Back Next > Finish

# PLL Instantiation

The screenshot displays the ALTPLL configuration tool interface. The main window is titled "ALTPLL" and has a navigation bar with tabs: 1 Parameter Settings, 2 PLL Reconfiguration, 3 Output Clocks (selected), 4 EDA, and 5 Summary. Below the navigation bar, there are sub-tabs for clocks: clk c0, clk c1, clk c2, clk c3, clk c4, and clk c5. The "clk c0" sub-tab is active, showing a block diagram of the PLL and its configuration parameters.

The block diagram shows the PLL configuration for "pll\_rfclk1" on a "Stratix II" device. The configuration includes:

- inclk0: inclk0 frequency: 45.000 MHz
- Operation Mode: No Compensation
- PLL Type: Enhanced PLL

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	4/1	0.00	50.00
c2	2/1	0.00	50.00
c3	6/1	0.00	50.00
c4	4/1	0.00	50.00
c5	4/1	0.00	50.00

The right-hand pane shows the configuration for "c0 - Core/External Output Clock". It is noted as "Able to implement in Enhanced PLL". The "Use this clock" checkbox is checked. The "Clock Tap Settings" section includes:

- Enter output clock frequency: Requested settings: 100.0000000 MHz, Actual settings: 45.000000
- Enter output clock parameters:
  - Clock multiplication factor: 1
  - Clock division factor: 1 (with "<< Copy" button)
  - Clock phase shift: 0.00 deg
  - Clock duty cycle (%): 50.00

At the bottom right, there are "Per Clock Feasibility Indicators" for c0, c1, c2, c3, c4, and c5, all of which are green, indicating they are feasible.

Navigation buttons at the bottom include "Cancel", "< Back", "Next >", and "Finish".

---

# LVDS Receiver

- Instantiated with parametrized IP
- Able to take full advantage of built in RX blocks
  - Dynamic Phase Alignment circuitry
  - Automatically de-serialize data from  $4 \cdot RF$  rate to 4 time multi-plexed samples at RF rate
- No need to worry about timing constraints!

# LVDS Receiver IP

**ALTLVDS**

Parameter Settings | EDA | Summary

General > Frequency/PLL settings > Receiver settings >

Currently selected device family: Stratix II

Match project/default

This module acts as an

LVDS transmitter  LVDS receiver

Implement Serializer/Deserializer circuitry in logic cells  
The receiver starts capturing the LVDS stream at the fast clock edge. This is intended for slow speeds and byte alignment may be different from the hard SERDES implementation.

Enable Dynamic Phase Alignment mode (receiver only)

What is the number of channels? 12 channels

What is the deserialization factor? 4

Use External PLL

Resource Usage  
1 clkctrl + 48 reg + 12 stratixii\_lvds\_receiver + 1 stratixii\_pll

Cancel < Back Next > Finish

# LVDS Receiver IP

The screenshot displays the ALTLVDS configuration tool interface. At the top, there is a logo and the title "ALTLVDS". Below the title are navigation tabs: "1 Parameter Settings", "2 EDA", and "3 Summary". The "Parameter Settings" tab is active, and within it, the "Frequency/PLL settings" sub-tab is selected. The main area shows a block diagram of the "lvds\_rx" component. The block is labeled "LVDS Receiver" and has the following parameters: "12 channels, x4", "212.00 MHz", "I/P data rate=212.00", and "Outclk Freq = 53.00". The block has several input and output ports: "rx\_in[11..0]", "rx\_inclock", "pll\_areset", "rx\_cda\_reset[11..0]", "rx\_channel\_data\_align[11..0]", "rx\_out[47..0]", "rx\_outclock", and "rx\_locked". The block is connected to a "Stratix II" device. To the right of the block diagram, there are configuration options for the input data rate and clock rate. The input data rate is set to "212" Mbps. The input clock rate is specified by "clock frequency" (212.00 MHz) and "clock period" (4.717 ns). There are also checkboxes for "Use shared PLL(s) for receivers and transmitters", "Use 'pll\_areset' input port", "Use 'rx\_pll\_enable' input port", and "Use 'rx\_locked' output port". The clock resource used for 'rx\_outclock' is set to "Auto selection". The phase alignment of 'rx\_in' with respect to the rising edge of 'rx\_inclock' is set to "0.00" degrees. At the bottom left, there is a "Resource Usage" section showing: "1 clkctrl + 48 reg + 12 stratixii\_lvds\_receiver + 1 stratixii\_pll". At the bottom right, there are navigation buttons: "Cancel", "< Back", "Next >", and "Finish".

**ALTLVDS**

About Documentation

1 Parameter Settings 2 EDA 3 Summary

General > Frequency/PLL settings > Receiver settings >

lvds\_rx

rx\_in[11..0]  
rx\_inclock  
pll\_areset  
rx\_cda\_reset[11..0]  
rx\_channel\_data\_align[11..0]

LVDS Receiver  
12 channels, x4  
212.00 MHz  
I/P data rate=212.00  
Outclk Freq = 53.00

rx\_out[47..0]  
rx\_outclock  
rx\_locked

Stratix II

What is the input data rate? 212 Mbps

Specify the input clock rate by

clock frequency 212.00 MHz

clock period 4.717 ns

Use shared PLL(s) for receivers and transmitters

Use 'pll\_areset' input port

Use 'rx\_pll\_enable' input port

Use 'rx\_locked' output port

What is the clock resource used for 'rx\_outclock'? Auto selection

What is the phase alignment of 'rx\_in' with respect to the rising edge of 'rx\_inclock'? (in degrees) 0.00

Resource Usage

1 clkctrl + 48 reg + 12 stratixii\_lvds\_receiver + 1 stratixii\_pll

Cancel < Back Next > Finish

# LVDS Receiver IP

**ALTLVDS**

Parameter Settings | EDA | Summary

General > Frequency/PLL settings > Receiver settings

**lvds\_rx**

LVDS Receiver  
12 channels, x4  
212.00 MHz  
I/P data rate=212.00  
Outclk Freq = 53.00

Inputs: rx\_in[11..0], rx\_inclock, pll\_areset, rx\_cda\_reset[11..0], rx\_channel\_data\_align[11..0]

Outputs: rx\_out[47..0], rx\_outclock, rx\_locked

Stratix II

Resource Usage  
1 clkctrl + 48 reg + 12 stratixii\_lvds\_receiver + 1 stratixii\_pll

Register outputs

Use 'rx\_channel\_data\_align' input port  
A pulse to this signal causes data alignment circuit to add one bit of latency into serial data stream

Use 'rx\_cda\_reset' input port  
Resets the data alignment circuitry

Use 'rx\_cda\_max' output port  
Indicates when the next rx\_channel\_data\_align pulse restores the serial data latency back to 0

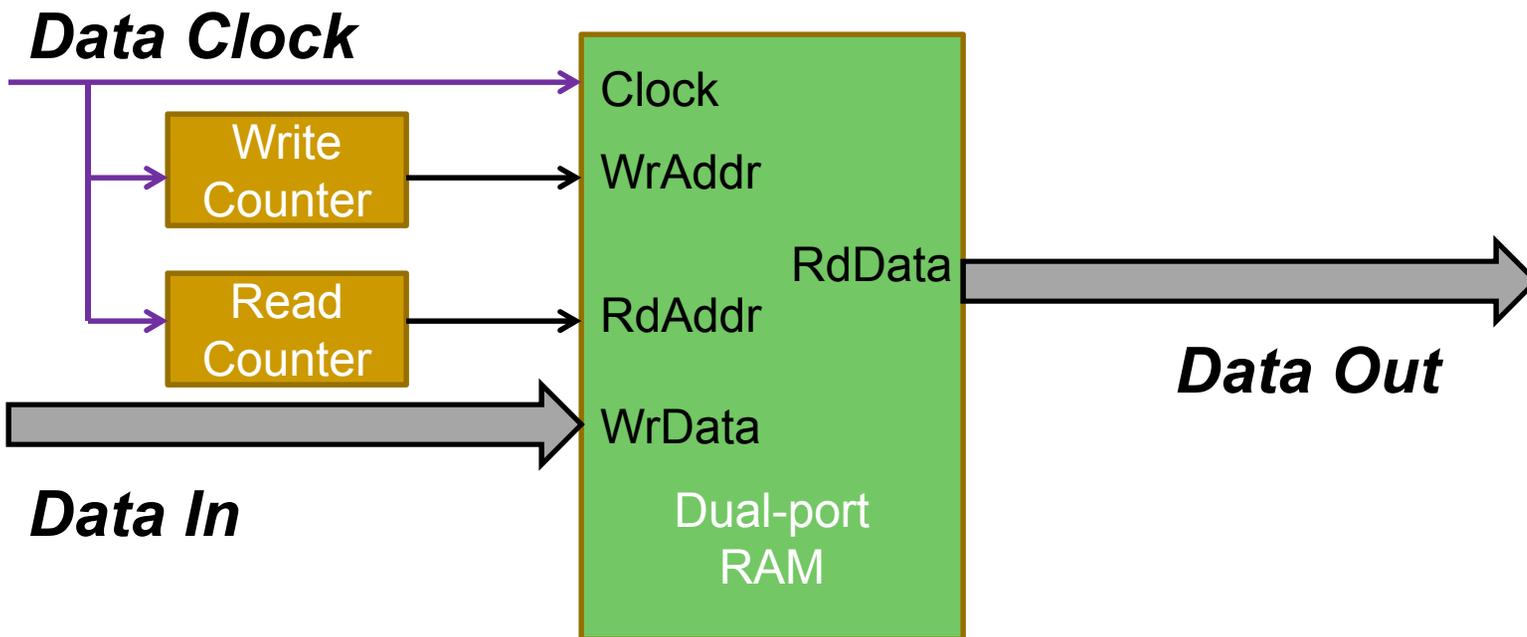
After how many pulses does the data alignment circuitry restore the serial data latency back to 0?  pulses

Align data to the rising edge of clock  
LVDS input data is aligned at the rising edge of the LVDS clock

Cancel < Back Next > Finish

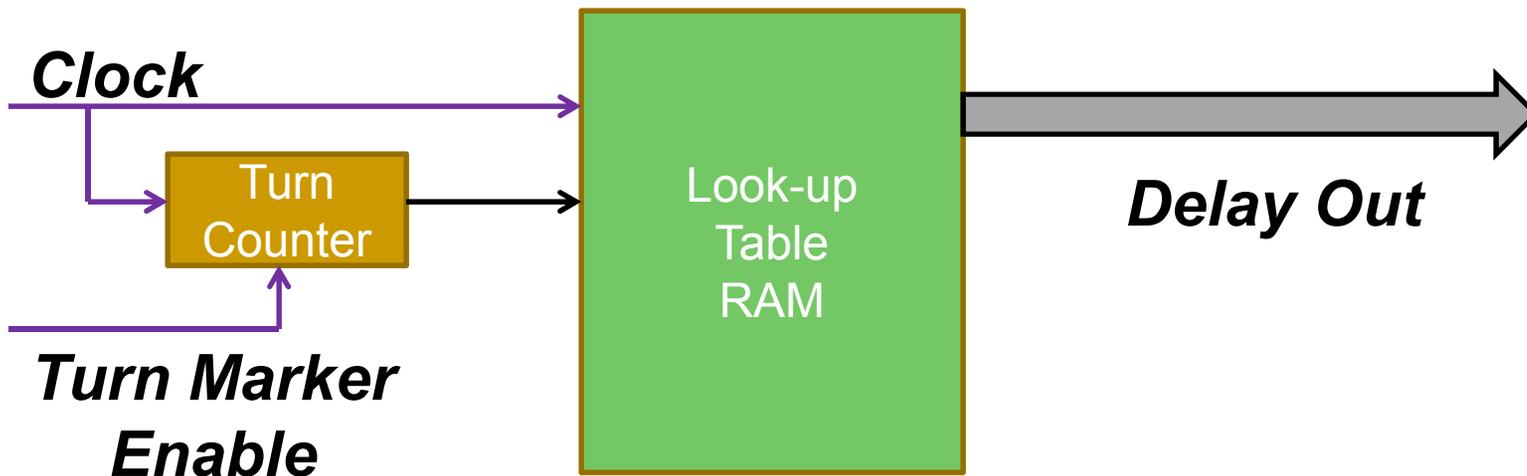
# Programmable Delay

- Dual-port RAM to provide programmable delay
  - Offset read & write pointer to requested delay value
    - Reset Read Counter to zero when Write Counter = Delay-1
  - Maximum delay set by the depth of the RAM



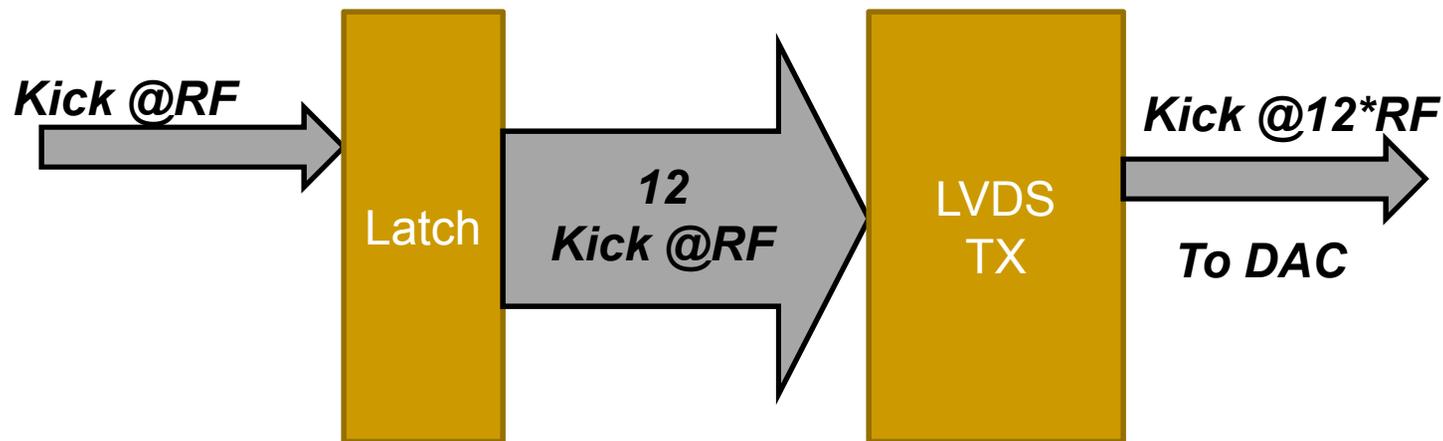
# Handling Frequency Sweep

- Digital Portion of design is operating locked to the Booster RF and sweeps along with it
- Need to adjust output delay to account for fixed input/output cable & amplifier delays
- Use lookup table to specify output delay



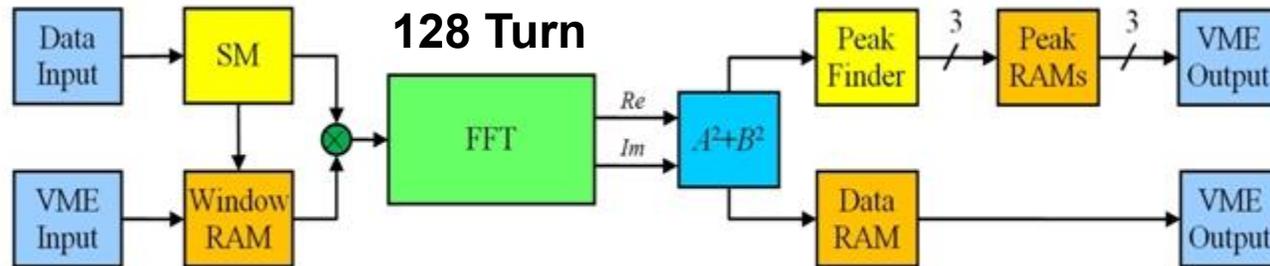
# Implementation of “Fine” Delay

- Filter produces kick for each RF bucket
- Operate DAC at  $12 \cdot \text{RF}$  clock rate



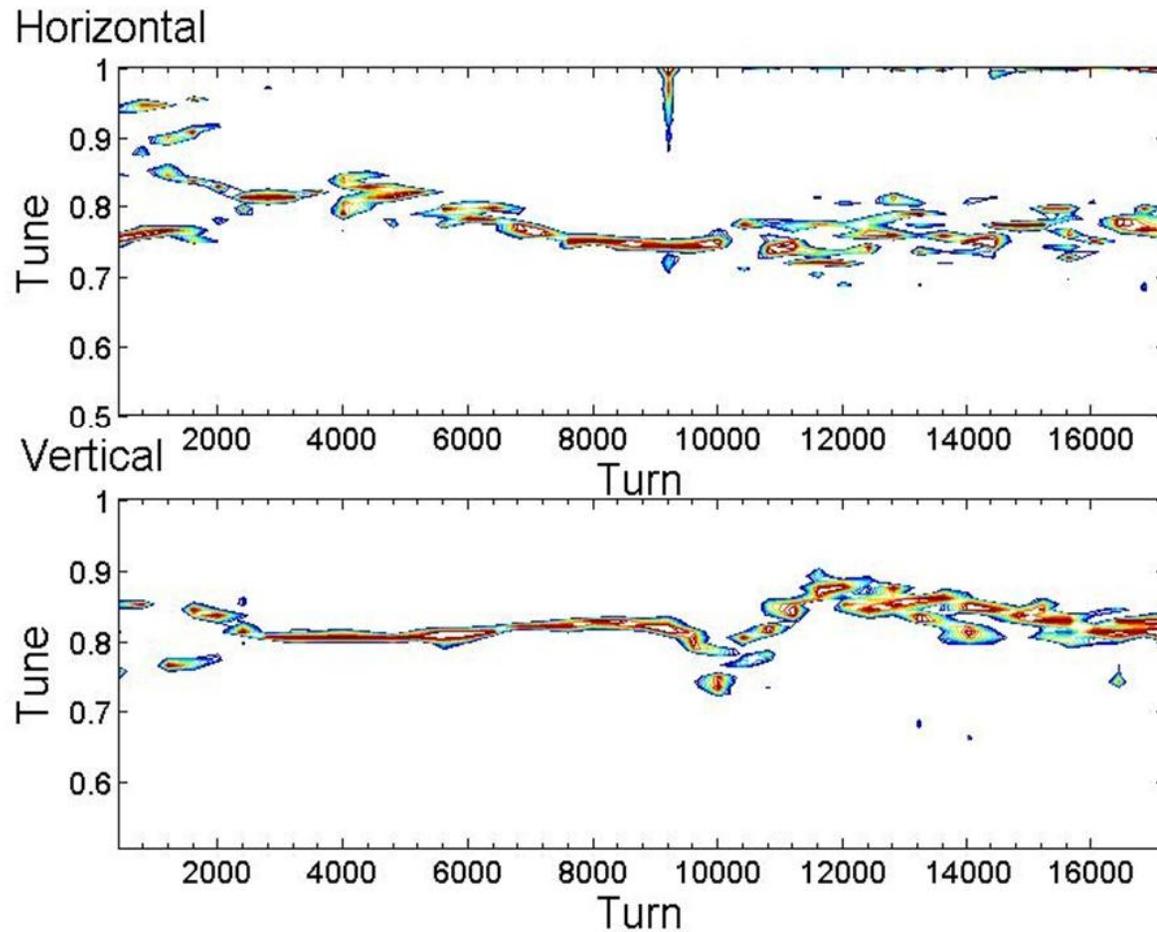
- Can shift data fed to LVDS Transmitter to provide  $\text{RF}/12$  delay resolution

# Tune Measurement

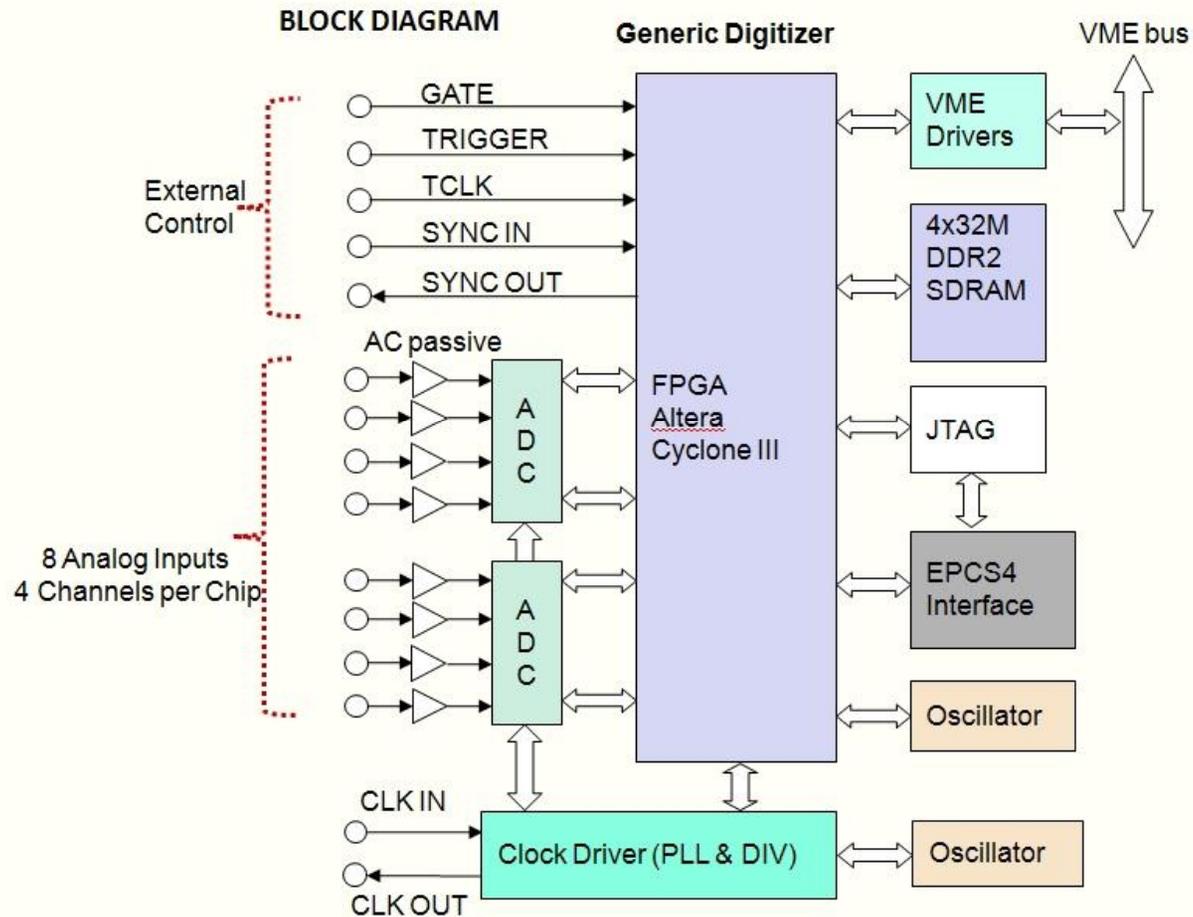


- Excite single bunch with noise via the damper
  - User programmable excitation, noise or anti-damping
  - Can vary gain and duration of excitation at each measurement point
- Simple State Machine to Control Measurement
  - Up to 64 measurements at selectable time (turn) within the machine cycle
  - Select bunch position to pass to FFT engine
- Measure response of single bunch over 128 turns
  - Instantiate fixed point FFT using commercial IP

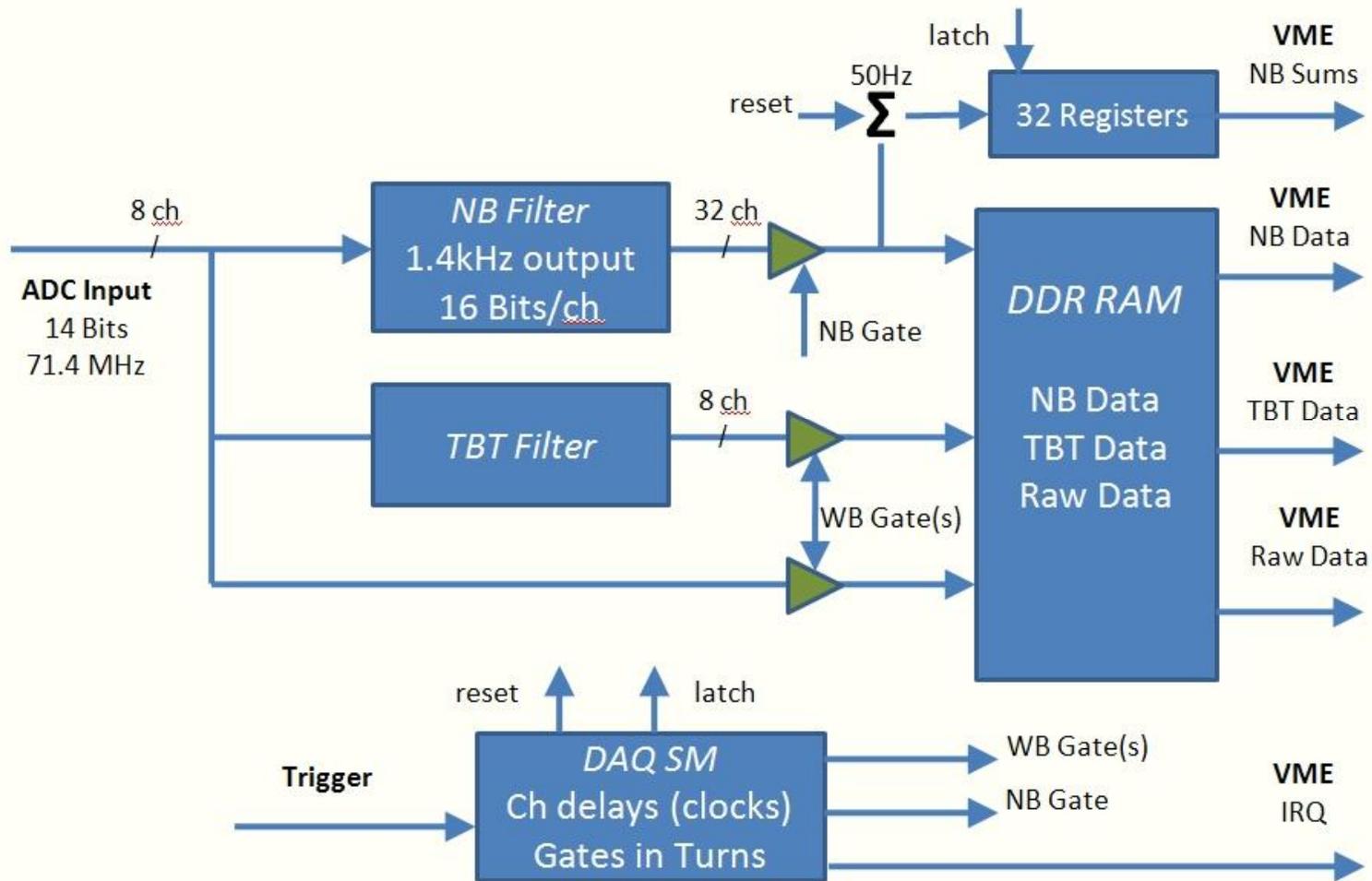
# Tune Measurement



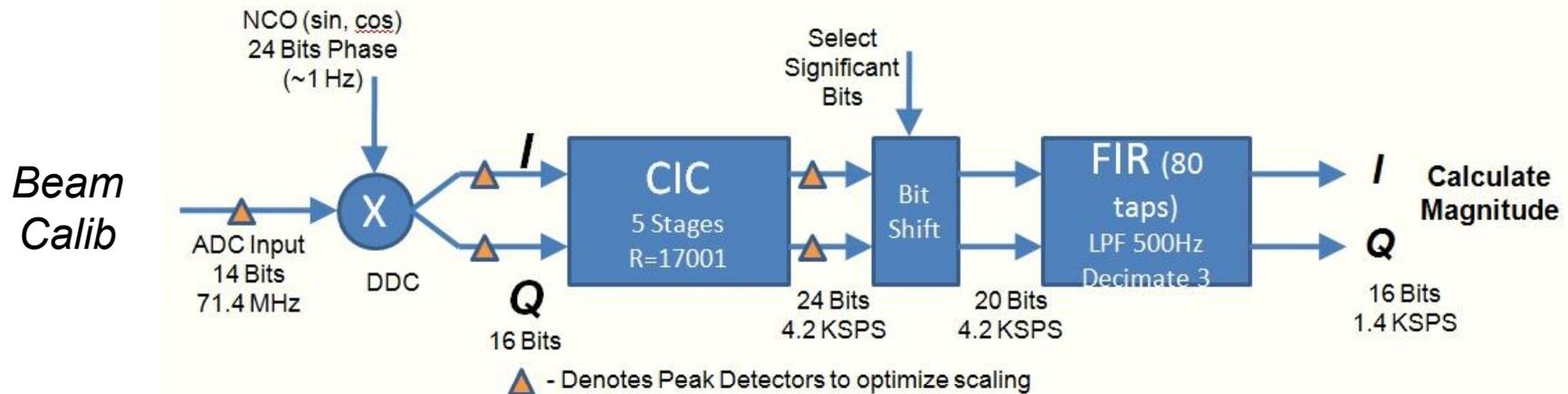
# 8 Channel 125MS/s Digitizer



# Digital BPM Receiver Firmware



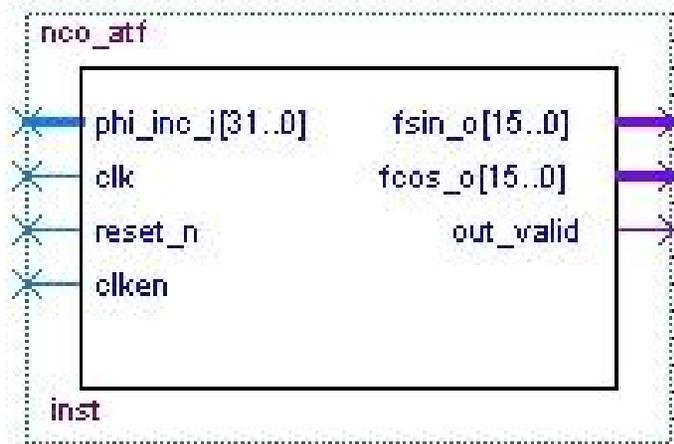
# Narrowband (Closed Orbit) Filter



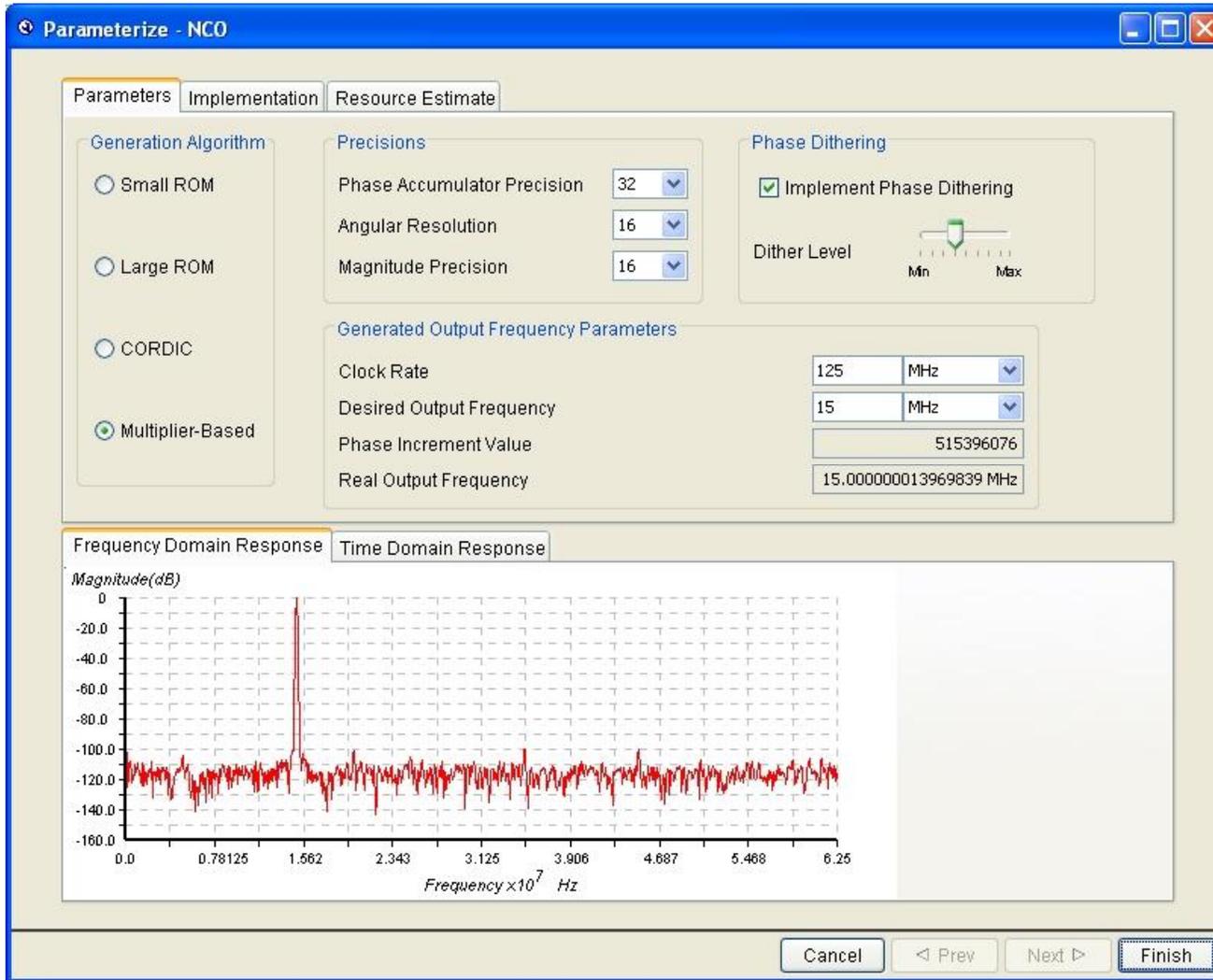
- Digital Down Converter (DDC) Section
  - 32 separate filter paths simultaneously
  - 8 channels, I&Q, 2 frequencies (beam, calibration)
- CIC filters operating in parallel at 71MHz
- Serial FIR filter at 4.2KHz

# Numerically Controlled Oscillator (NCO) IP

- Generate Sin & Cos signals at requested frequency
  - Frequency Output =  $F_{CLK}Phi/2^N$
  - Frequency Resolution =  $F_{CLK}/2^N$

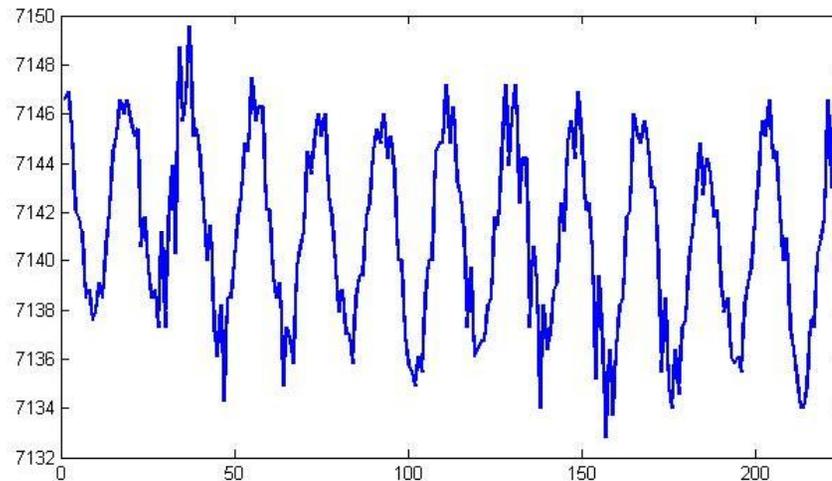


# NCO IP



# DDC Quantization Error

- Will always have a slight offset between the NCO frequency and the beam frequency



- Easy solution is to offset the NCO frequency to get an integer number of periods in the averaging window

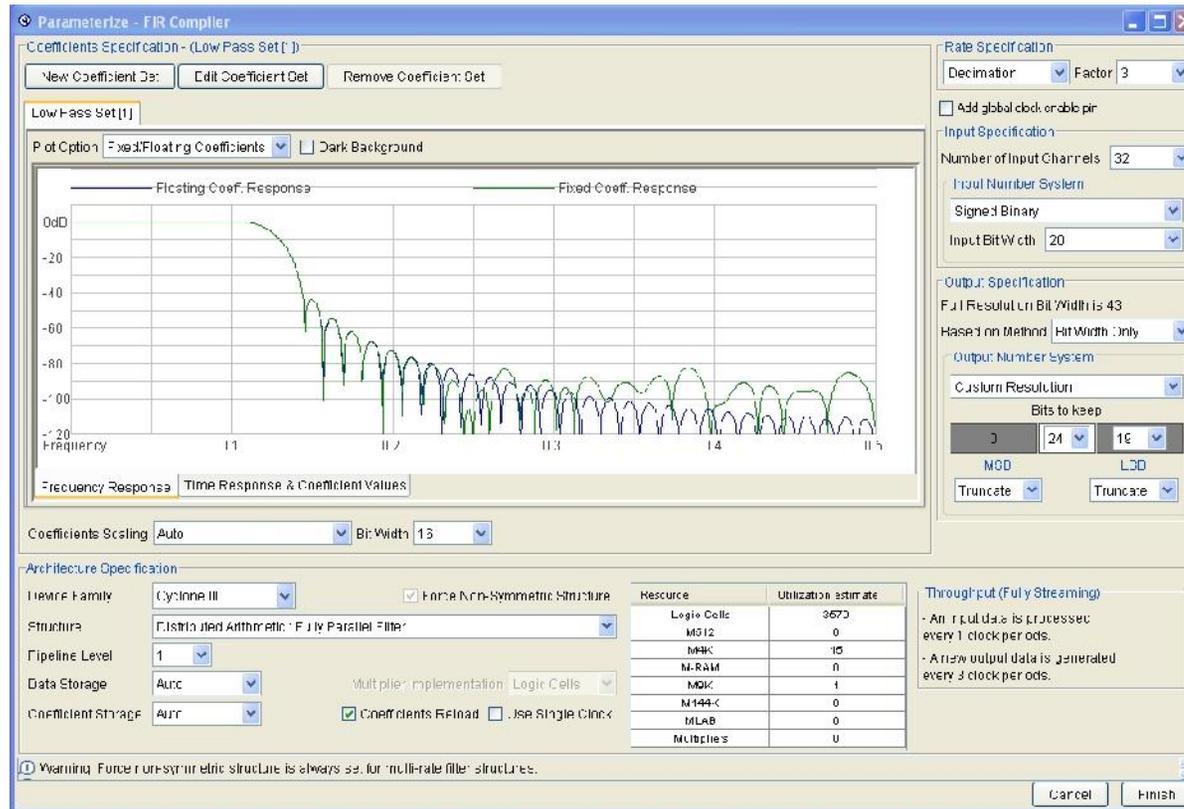
# CIC Filter IP

The screenshot shows the 'Parameter Settings' tab of the CIC Filter IP configuration tool. The interface includes a breadcrumb trail: 'Architecture' > 'Input/Output Options'. The 'Device Family' section has 'Target' set to 'Cyclone II'. The 'Filter Specifications' section includes: 'Filter type' set to 'Decimate', 'Number of stages' set to '5', 'Differential delay' set to '1', and 'Rate factor' set to '17001'. A 'Variable Rate Factor Options' section contains an unchecked checkbox for 'Enable variable rate factor', with 'Minimum' set to '128' and 'Maximum' set to '32000'. The 'Multi-channel Options' section has 'Number of interfaces' set to '32' and 'Number of channels per interface' set to '1'. The 'Data Storage Options' section includes: 'Integrator data storage' set to 'Logic Element', 'RAM type of integrator data storage' set to 'AUTO', 'Differentiator data storage' set to 'Logic Element', and 'RAM type of differentiator data storage' set to 'AUTO'.

The screenshot shows the 'Input/Output Options' tab of the CIC Filter IP configuration tool. The 'Input Options' section has 'Input data width' set to '16' Bits. The 'Output Options' section includes an unchecked checkbox for 'Full output resolution', 'Output data width' set to '24' Bits, and an 'Output Rounding Options' section with radio buttons for 'Truncation', 'Convergent rounding' (which is selected), 'Rounding up', and 'Saturation'. At the bottom, the checkbox 'Apply Hogener pruning across filter stages' is checked.

- Commercial IP allows for single instantiation for all 32 CIC filters needed
  - Provides standard serial data output which can be directly interfaced to serial FIR filter

# FIR Filter IP



- Provides simple filter design tool or ability to import filter coefficients
- Option to allow modification of coefficients

---

# System Integration Tools

- Streamline system integration and design
    - Connect standard interfaces
      - Internal memory, external memory, configuration devices, etc
    - Connect custom interfaces
    - Easy CPU integration
      - Handles addressing & interrupts
      - Even generates drivers for system components!
  - Use well defined interfaces
    - Generates all logic for system interconnects!
    - Handles all the timing - clock domains, multiplexing, etc
    - Built in error checking at compile time
  - Facilitates implementing re-usable HDL blocks and group design methodologies
-

# System Integration Tool Example

The screenshot displays the System Integration Tool interface for a Cyclone IV E target. The interface is divided into several sections:

- Component Library:** Located on the left, it shows a tree view of components categorized into Project, Interfaces, Memories and Memory Controllers, Peripherals, and Library. The Library section includes components like Avalon Verification Suite, Bridges and Adapters, Debug Components, Digital Signal Processing, Interface Protocols, Legacy Components, Memories and Memory Controllers, On-Chip SDRAM, Merlin Components, Peripherals, PLL, Processor Additions, Processors, SLS, and Video and Image Processing.
- Target:** Set to "Cyclone IV E".
- Clock Settings:** A table listing system clocks:
 

Name	Source	MHz
osc_clk	External	50.0
adc_clk	External	125.0
ddr2_sysclk	ddr2.s...	75.0
ddr2_auxfull	ddr2.a...	150.0
- Design Comps:** A central table listing components and their connections. The "Use" column has checkboxes. The "Connections" column shows a tree diagram of connections. The "Name" column lists components like cpu, cpu\_bridge, epcs\_controller, ram, cpu\_vme\_ram, ddr2, cpu\_io\_bridge, sysid, jtag\_uart, vme, adc, adc\_dma\_4ch, dac, dac\_dma\_4ch, SYNC, Mode, vxs, pie\_cdce62005, spi\_cdce62005, spi\_ads62p49, spi\_dac3283, and sys\_timer. The "Description" column provides details for each component. The "Clock" column lists the clock source for each component. The "Base" column shows memory addresses. The "IRQ" column shows interrupt numbers.

# System Integration Tool Example

The screenshot displays the System Integration Tool (SIT) interface for a Cyclone IV E target. The main window is divided into several sections:

- Component Library:** A tree view on the left showing various components like ADCs, DACs, SPI controllers, and memory controllers.
- Target:** Shows the device family as Cyclone IV E.
- Clock Settings:** A table listing clock sources and frequencies:
 

Name	Source	MHz
osc_clk	External	50.0
adc_clk	External	125.0
ddr2_sysclk	ddr2.s...	75.0
ddr2_aufull	ddr2.a...	150.0
- Component Table:** A detailed table listing components, their connections, names, descriptions, clocks, and base addresses.
 

Use	Connections	Name	Description	Clock	Base	IRQ
<input checked="" type="checkbox"/>		cpu	Nios II Processor	[clk]		
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master	ddr2_sysclk		
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master	[clk]		IRQ 0
<input checked="" type="checkbox"/>		jtag_debug_module	Avalon Memory Mapped Slaves	[clk]	0x00000000	
<input checked="" type="checkbox"/>		cpu_bridge	Avalon-MM Pipeline Bridge	ddr2_sysclk	0x00000000	
<input checked="" type="checkbox"/>		epcs_controller	EPCS Serial Flash Controller	ddr2_sysclk	0x00000800	
<input checked="" type="checkbox"/>		ram	On-Chip Memory (RAM or ROM)	[clk1]		
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slaves	ddr2_sysclk	0x00200000	
<input checked="" type="checkbox"/>		cpu_vme_ram	On-Chip Memory (RAM or ROM)	[clk1]		
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slaves	ddr2_sysclk	0x00408000	
<input checked="" type="checkbox"/>		ddr2	DDR2 SDRAM Controller with ALTMEMPHY	osc_clk		
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slaves	ddr2_sysclk	0x10000000	
<input checked="" type="checkbox"/>		cpu_io_bridge	Avalon-MM Pipeline Bridge	ddr2_sysclk	0x00400000	
<input checked="" type="checkbox"/>		sysid	System ID Peripheral	ddr2_sysclk	0x00000000	
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART	ddr2_sysclk	0x00000040	
<input checked="" type="checkbox"/>		vme	vme_interface			
<input checked="" type="checkbox"/>		av2	Avalon Memory Mapped Slaves	ddr2_sysclk	0x00000080	
<input checked="" type="checkbox"/>		avn	Avalon Memory Mapped Master	[clock]		
<input checked="" type="checkbox"/>		adc	ADC_Interface			
<input checked="" type="checkbox"/>		aso_adc_data	Avalon Streaming Source	adc_clk		
<input checked="" type="checkbox"/>		avo	Avalon Memory Mapped Slaves	ddr2_sysclk	0x000000a0	
<input checked="" type="checkbox"/>		adc_dma_1ch	adc_dms	[sys_cbclk]		
<input checked="" type="checkbox"/>		asi_adc	Avalon Streaming Sink	adc_clk		
<input checked="" type="checkbox"/>		av2	Avalon Memory Mapped Slaves	ddr2_sysclk	0x000000b0	
<input checked="" type="checkbox"/>		avn	Avalon Memory Mapped Master	ddr2_sysclk		
<input checked="" type="checkbox"/>		dac	dac_controller	[asi_clk]		
<input checked="" type="checkbox"/>		asi	Avalon Streaming Sink	adc_clk		
<input checked="" type="checkbox"/>		dac_dma_1ch	dac_dms	[sys_cbclk]		
<input checked="" type="checkbox"/>		aso	Avalon Streaming Source	adc_clk		
<input checked="" type="checkbox"/>		avn	Avalon Memory Mapped Master	ddr2_sysclk	0x00000120	
<input checked="" type="checkbox"/>		av2	Avalon Memory Mapped Slaves	ddr2_sysclk	0x000000c0	
<input checked="" type="checkbox"/>		SYNC	SYNC controller	adc_clk		
<input checked="" type="checkbox"/>		av2	Avalon Memory Mapped Slaves	ddr2_sysclk	0x000000d0	
<input checked="" type="checkbox"/>		Mode	Mode_controller	adc_clk		
<input checked="" type="checkbox"/>		av2	Avalon Memory Mapped Slaves	ddr2_sysclk	0x00000100	
<input checked="" type="checkbox"/>		vxs	VXS_controller	ddr2_sysclk	0x00000110	
<input checked="" type="checkbox"/>		pio_dce62005	PID (Parallel I/O)	ddr2_sysclk	0x00000200	
<input checked="" type="checkbox"/>		spi_dce62005	SPI (3 Wire Serial)	ddr2_sysclk	0x00000300	
<input checked="" type="checkbox"/>		spi_ads62p49	SPI (3 Wire Serial)	ddr2_sysclk	0x00000400	
<input checked="" type="checkbox"/>		spi_dac32x3	SPI (3 Wire Serial)	ddr2_sysclk	0x00000500	
<input checked="" type="checkbox"/>		sys_timer	Interval Timer	ddr2_sysclk	0x00000600	

# System Integration Tool Example

The screenshot displays the System Integration Tool (SIT) interface for configuring an On-Chip Memory (RAM or ROM) component. The interface is divided into several panes:

- Component Library:** Shows a tree view of components under 'Memories and Memory Controllers' and 'External Memory Interfaces'. The 'On-Chip' component is selected.
- Target:** Shows the device family as 'Cyclone IV E'.
- Clock Settings:** A table listing clock sources and frequencies:

Name	Source	MHz
osc_clk	External	50.0
adc_clk	External	125.0
ddr2_sysclk	ddr2.s...	75.0
ddr2_audfull	ddr2.a...	150.0
- Use Connections:** A table showing the connections for the selected component (ram):

Use	Connections	Name
<input checked="" type="checkbox"/>		cpu
<input checked="" type="checkbox"/>		instruction_master
<input checked="" type="checkbox"/>		data_master
<input checked="" type="checkbox"/>		jtag_debug_module
<input checked="" type="checkbox"/>		cpu_bridge
<input checked="" type="checkbox"/>		epcs_controller
<input checked="" type="checkbox"/>		ram
<input checked="" type="checkbox"/>		s1
<input checked="" type="checkbox"/>		cpu_vme_ram
<input checked="" type="checkbox"/>		s1
<input checked="" type="checkbox"/>		ddr2
<input checked="" type="checkbox"/>		s1
<input checked="" type="checkbox"/>		cpu_io_bridge
<input checked="" type="checkbox"/>		sysid
<input checked="" type="checkbox"/>		jtag_uart
<input checked="" type="checkbox"/>		vme
<input checked="" type="checkbox"/>		avs
<input checked="" type="checkbox"/>		avm
<input checked="" type="checkbox"/>		adc
<input checked="" type="checkbox"/>		aso_adc_data
<input checked="" type="checkbox"/>		avs
<input checked="" type="checkbox"/>		adc_dma_4ch
<input checked="" type="checkbox"/>		asi_adc
<input checked="" type="checkbox"/>		avs
<input checked="" type="checkbox"/>		avm
<input checked="" type="checkbox"/>		dac
<input checked="" type="checkbox"/>		asi
<input checked="" type="checkbox"/>		dac_dma_4ch
<input checked="" type="checkbox"/>		aso
<input checked="" type="checkbox"/>		avm
<input checked="" type="checkbox"/>		avs
<input checked="" type="checkbox"/>		SYNC
<input checked="" type="checkbox"/>		avs
<input checked="" type="checkbox"/>		Mode
<input checked="" type="checkbox"/>		avs
<input checked="" type="checkbox"/>		vxa
<input checked="" type="checkbox"/>		pio_cdc62005
<input checked="" type="checkbox"/>		spi_cdc62005
<input checked="" type="checkbox"/>		spi_ads62p49
<input checked="" type="checkbox"/>		spi_dac3283
<input checked="" type="checkbox"/>		sys_timer
- On-Chip Memory (RAM or ROM) Configuration:**
  - Block Diagram:** Shows a block labeled 'ram' with three inputs: 'clock' (clk1), 'avalon' (s1), and 'reset' (reset1).
  - Memory type:** Type is set to 'RAM (Writable)'. Options for 'Dual-port access' and 'Single clock operation' are unchecked. 'Read During Write Mode' is set to 'DONT\_CARE'. 'Block type' is set to 'Auto'.
  - Size:** Data width is 32. Total memory size is 131072 bytes. 'Minimize memory block usage (may impact fmax)' is unchecked.
  - Read latency:** Slave s1 Latency is 2. Slave s2 Latency is 1.
  - Memory initialization:** 'Initialize memory content' is checked. 'Enable non-default initialization file' is unchecked. 'User created initialization file' is 'ram'. 'Enable In-System Memory Content Editor feature' is unchecked. 'Instance ID' is 'NONE'.

---

# Summary - the FPGA Pitch

- Sure things...
  - FPGA are now the acknowledged leader of cutting edge fast DSP applications where speed and flexibility are needed
  - Accelerator Control and Instrumentation is already using FPGAs to implement fast online applications, especially feedback & control
  - The size, speed, and feature sets continue to grow by leaps and bounds
  - Today's mid level chips are offering features only available in high end chips just a few years ago at a fraction of the cost
  - Design tools are getting closer to traditional programming and becoming easier to use
- Looks promising...
  - Use of FPGA's to implement online orbit measurements and optic calculations which could be used for realtime feedback
  - The next step is cluster and mesh architectures using FPGAs to further increase the processing power
- It could happen..
  - FPGA based co-processors for dedicated calculations
  - FPGA based super computers which configure their hardware to optimize the performance for the algorithms being used

# Thanks for Your Attention!



April 15-19, 2012  
Newport News Marriott City Center  
*hosted by: Jefferson Lab*

# BIW 12

The 2012 Beam Instrumentation Workshop (BIW12) will be the 15th biennial workshop dedicated to exploring the physics and engineering challenges of beam diagnostic and measurement techniques for charged particle accelerators. This meeting program will include tutorials on selected topics, invited and contributed talks, as well as poster sessions. This will be the last BIW, after which IBIC (International Beam Instrumentation Conference) will follow the PAC rotation; Asia, Europe, and the US.

**Program Committee:**

Kevin Jordan, (Chair, JLab)	Tom Shea, (SNS)
Daniele Filippetto, (LBNL)	Om Singh, (BNL)
Doug Gilpatrick, (LANL)	Steve Smith, (SLAC)
Ken Jacobs, (SRC)	Hitoshi Tanaka, (SPRING-8)
Rhodri Jones, (CERN)	Jonah Weber, (LBNL)
Bob Lill, (ANL)	Manfred Wendt, (FNAL)
Toshiyuki Mitsubishi, (KEK)	Michelle Wilinski, (BNL)
Guenther Rehm, (DIAMOND)	Kay Wittenburg, (DESY)
Jim Sebek, (SLAC)	Jim Zagel, (FNAL)

**Jefferson Lab Local Organizing Committee:**

Pavel Evtushenko
Cynthia Lockwood
John Musson
Todd Satogata