A Streaming FFT on 3GSPS ADC Data using Core Libraries and DIME-C

Malachy Devlin, CTO, Nallatech
Robin Bruce, EngD Research Engineer, ISLI
Chee Kin Tang, MSc Student, ISLI
www.nallatech.com

Nallatech
Our Expertise. Your Success.

Introduction

Background

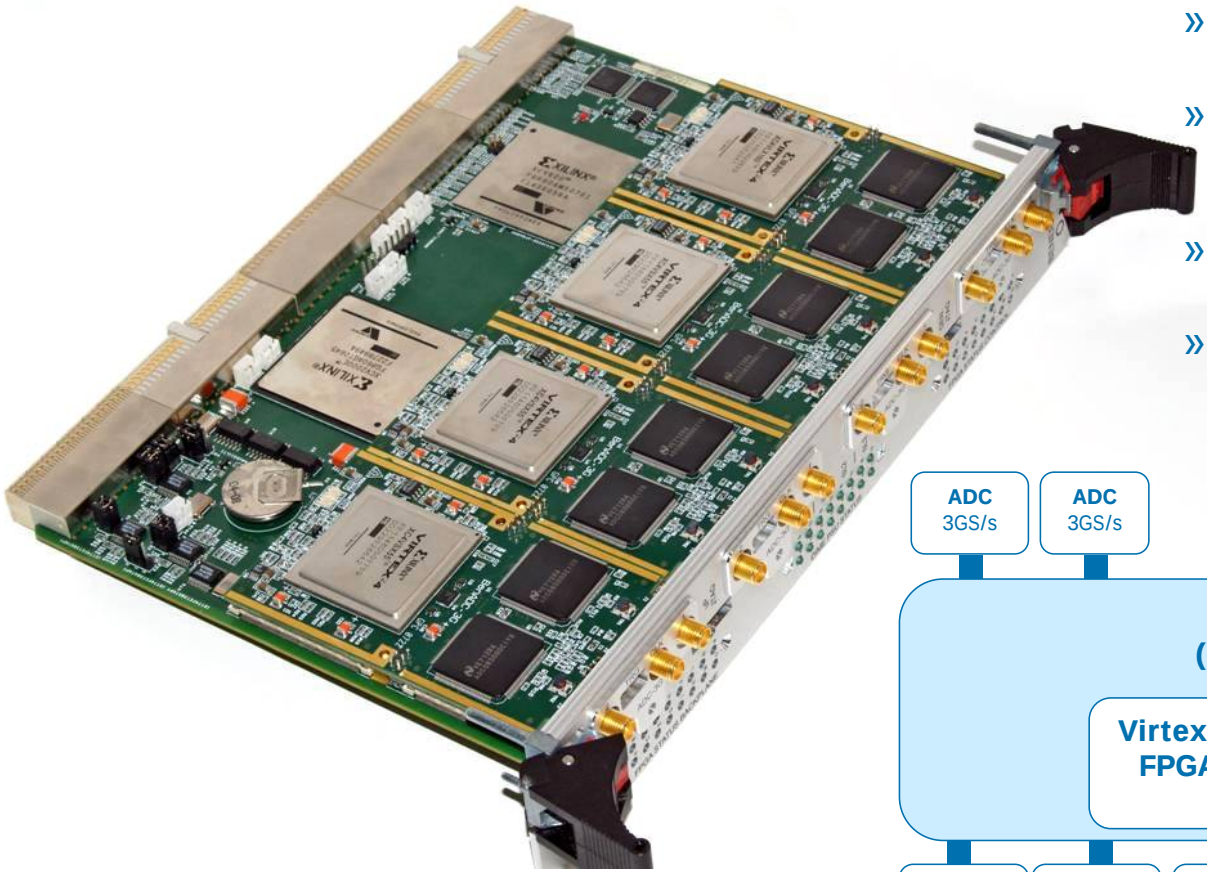
- **Problem: Increasing logic density of FPGAs**
  - Design challenge is increasing
  - High-End FPGAs now have > 100k LUTs and Registers
  - Industry-wide recognition for design tools that simplify the problem
- **Potential Solutions to Problem**
  - Block-based visual tools
    - Simulink blocksets from Xilinx, Altera, Synplicity, etc.
    - Good for DSP
  - C-to-FPGA compilers
    - Many on the market
      - DIME-C, Mitrion-C, ImpulseC, Carte, etc.
      - Principally Aimed at HPC FPGA Computing
    - How are they for more HPEC/DSP type operations?
  - Libraries of IP Cores
    - Standard HDL Core Interface
    - XML Descriptor File
    - Automatic, Error Free Instantiation by C-to-FPGA Compiler from Software Syntax Source Code
    - Benefits to user of high-level design environment, with resource consumption and performance of HDL
- **FPGAs Have Increasingly Powerful Input/Output Capabilities**
  - How can this fit into a high-level design environment?
  - How can core libraries work with external I/O?
  - How does this fit into high-level design environments?
- **Objective: Implement an example application that combines**
  - High-Level Design Tools
  - Core Libraries for FPGA high-level languages
  - High-Performance Input/Output
  - Widely understood DSP operation or operations

Objective

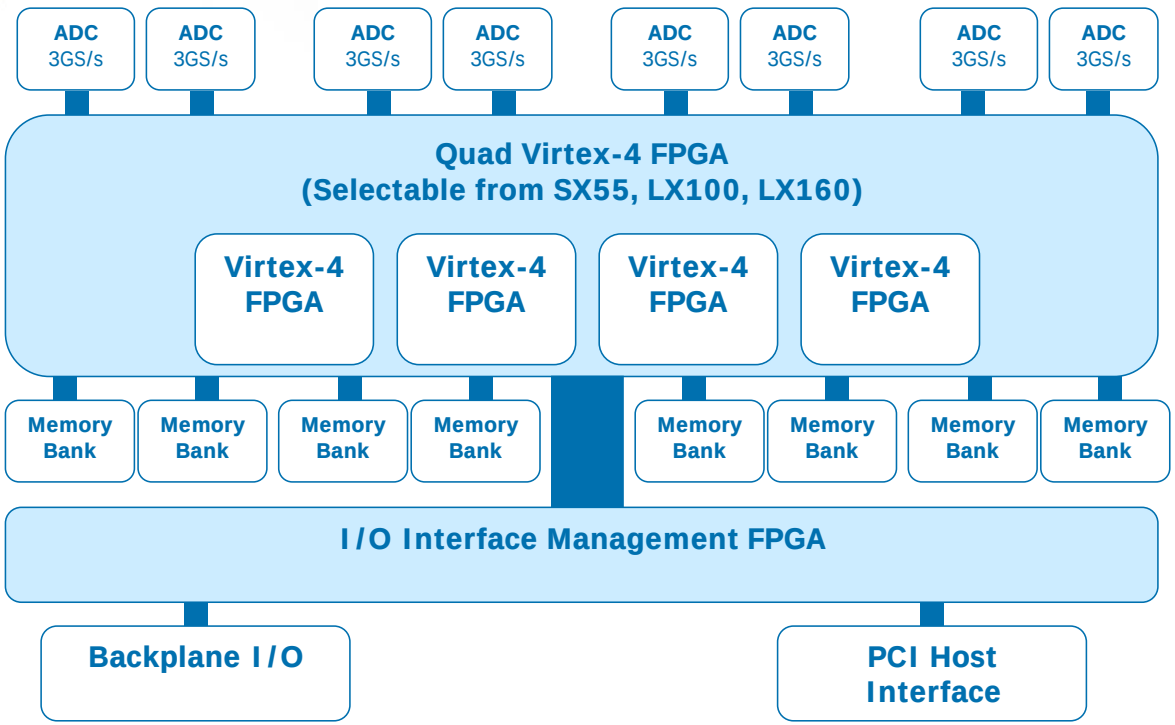
- **Algorithm:**
  - To implement A 1024-point fully-pipelined FFT on a single FPGA capable of processing 3 Giga-samples per second.
- **Hardware:**
  - Nallatech BenADC-3G Module
    - Xilinx Virtex-4 LX160 FPGA
    - Dual-Channel 3GSPS ADCs
    - Main Compute Element in Design
  - Nallatech BenERA Motherboard
    - Used to host the BenADC-3G Modules
  - Test Environment
    - ROHDE & SCHWARZ Signal Generator SML 01
  - Intel based Computer, Windows XP
    - No heavy lifting, acts as test harness to FPGAa
- **Software:**
  - Matlab
    - Generates test environment for Implementation
    - Manages the interactions with the hardware via FUSE toolbox for Matlab
  - Xilinx Coregen
    - IP Core Creation used in DIME-C Cores
  - Aldec Active-HDL
    - Creates and verifies DIME-C & DIMETalk IP Cores
  - Nallatech DIME-C
    - C-to-FPGA Compiler
    - Integrates IP cores with compiler-generated logic to produce pipelined FFT architecture
  - Nallatech DIMETalk
    - Maps component created by DIME-C to hardware
    - Used to instantiate control components for ADCs
    - Provides control plane for Matlab host program

Nallatech
Our Expertise. Your Success.

Reconfigurable Computing Platform



- Multi Channel Acquisition System
  - Scalable from 2 – 8 channels
- Dual Channel 3GSPS 8-bit ADC Modules
  - Uses 2 National Semiconductor ADC083000
  - Dual Channels for I/Q Applications
- Xilinx Virtex-4 FPGA
  - LX100, LX160, SX55
- 64Mbytes DDR-II SRAM



Nallatech
Our Expertise. Your Success.

DIME-C & Core Libraries

- DIME-C presents user with a software-like development environment
  - Uses a Subset of ANSI C Syntax
  - Handles ANSI C Datatypes
  - Plus boolean and FIFO types
- Compiles Input C code to VHDL and pre-synthesized logical netlists
- Leverages pre-existing hand-coded IP cores, either:
  - Natively, as in the case of basic arithmetic operations, or
  - Via the inclusion of libraries of IP cores
    - Analogous to software function calls
- Core Libraries consist of three elements:
  - .h header file, for function call prototypes
  - .lib file, XML file that carries metadata regarding the cores, e.g. datatypes of ports, latency, clocking, port naming, support files etc…
  - Support Files
    - Raw HDL, EDIF, NGC, etc.
- OpenFPGA CORELIB Working Group
  - Industry and Academia working towards defining a single standard for Core Libraries in FPGA high-level languages
  - Aim is to promote the migration of IP to and between FPGA High-Level Compilers

```
#include <benadc3g.h>
void eight_combine2_1_1(unsigned short pipein raw1,
                        unsigned int real_out1_mem[8192], unsigned int real_out2_mem[8192],
                        unsigned int imag_out1_mem[8192], unsigned int imag_out2_mem[8192],
                        int stop, unsigned int mode, int range)
{
  int i=0;
  unsigned short raw_in1;
  char  fft1_real_in_1, fft1_imag_in_1, fft1_real_in_2, fft1_imag_in_2;,
  /* output registers from the FFT wrappers - used as input to the separators*/
  int fft1_real_out_reg1, fft1_imag_out_reg1, fft1_real_out_reg2, fft1_imag_out_reg2;
  /* outputs from the separators */
  int fft1_out1_r1, fft1_out2_r1, fft1_out1_r2, fft1_out2_r2,
      fft1_out1_i1, fft1_out2_i1, fft1_out1_i2, fft1_out2_i2;
  bool local_mode;
  unsigned int fft1_word_in;
  local_mode = (bool) mode;
  while( GetPipeCount(raw1) != 1);
  for(i=0; i<range; i++) {
    raw_in1 = readFIFO(raw1);

    fft1_real_in_1 = (char) (raw_in1 >> 8);
    fft1_real_in_2 = (char) (raw_in1);
    fft1_imag_in_1 = (char) (raw_in2 >> 8);
    fft1_imag_in_2 = (char) (raw_in2);
    fft_v1_1(fft1_real_in_1, fft1_imag_in_1, fft1_real_in_2, fft1_imag_in_2,
             fft1_real_out_reg1, fft1_imag_out_reg1, fft1_real_out_reg2, fft1_imag_out_reg2,
             local_mode);
    /*Separators for the result of FFTs*/
    #pragma DIMEC instance FFT_SEPARATOR_1
    fft_separate(fft1_real_out_reg1,
                 fft1_real_out_reg2,      //real inputs
                 fft1_imag_out_reg1,
                 fft1_imag_out_reg2,      //imag inputs
                 fft1_out1_r1,
                 fft1_out1_r2,
                 fft1_out1_i1,
                 fft1_out1_i2,            //ouputs stream 1
                 fft1_out2_r1,
                 fft1_out2_r2,
                 fft1_out2_i1,
                 fft1_out2_i2);            //ouputs stream 2
    real_out1_mem[i] = fft1_out1_r1 + fft1_out2_r1;
    real_out2_mem[i] = fft1_out1_r2 + fft1_out2_r2;
    imag_out1_mem[i] = fft1_out1_i1 + fft1_out2_i1;
    imag_out2_mem[i] = fft1_out1_i2 + fft1_out2_i2;
  }
}
```

Example DIME-C Code for Streaming FFT

Nallatech
Our Expertise. Your Success.

FFT Algorithm

- **Hardware Efficient FFT Algorithm**
  - FFT Core has Real and Imaginary Inputs
  - Only transforming real data, hence compute two real streams using single FFT core
  - Theory as follows:
- Assume that F(n) is the fourier transform of f(n), where n ranges 0 to N-1 with N the transform size.

$$f(n) = a(n) + j \times b(n)$$

$$F(n) = X(n) + j \times Y(n)$$

- if we set a(n) = $f_1(n)$ and b(n) = $f_2(n)$ where f_1 and f_2 are two independent real functions in the time domain, we can obtain their transforms as follows:

$$F_1(n) = X_{even}(n) + j \times Y_{odd}(n)$$

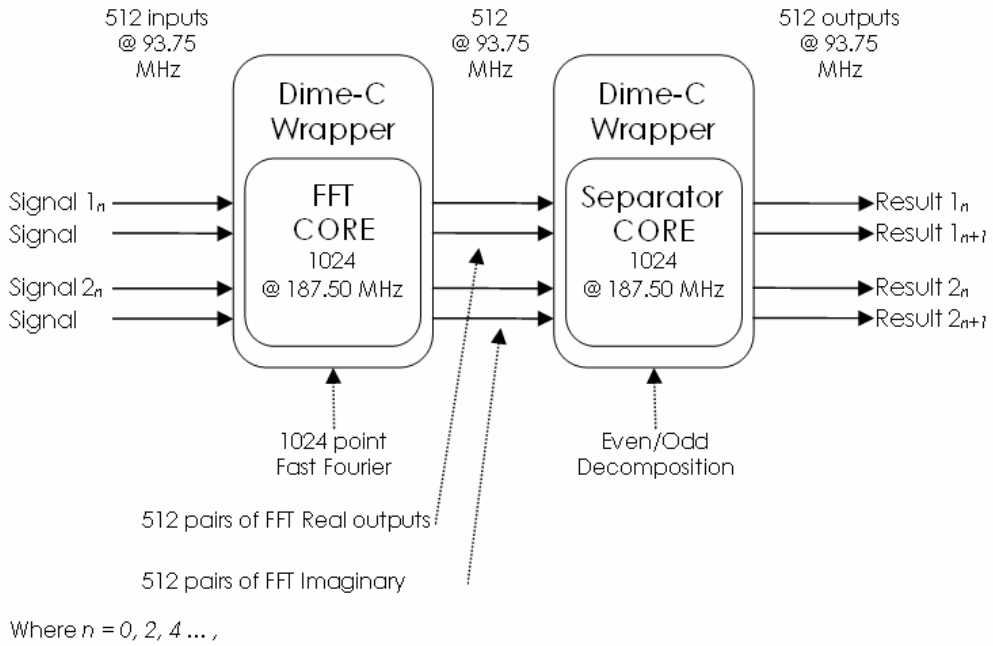
$$F_2(n) = Y_{even}(n) + j \times X_{odd}(n)$$

- where

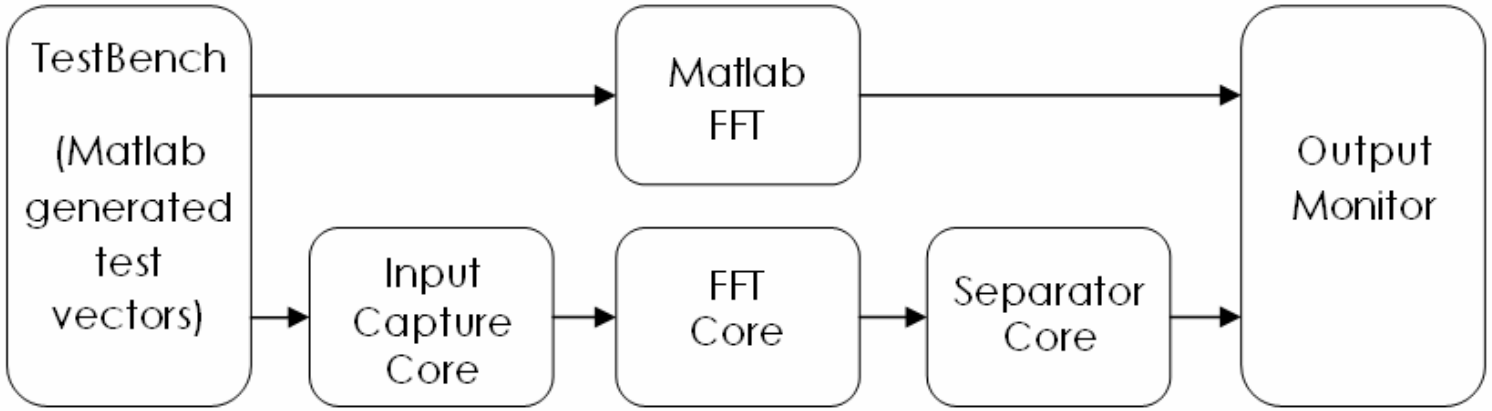
$$X_{even} = (X(n) + X(N-1-n))/2$$

$$X_{odd} = -1(X(n) - X(N-1-n))/2$$

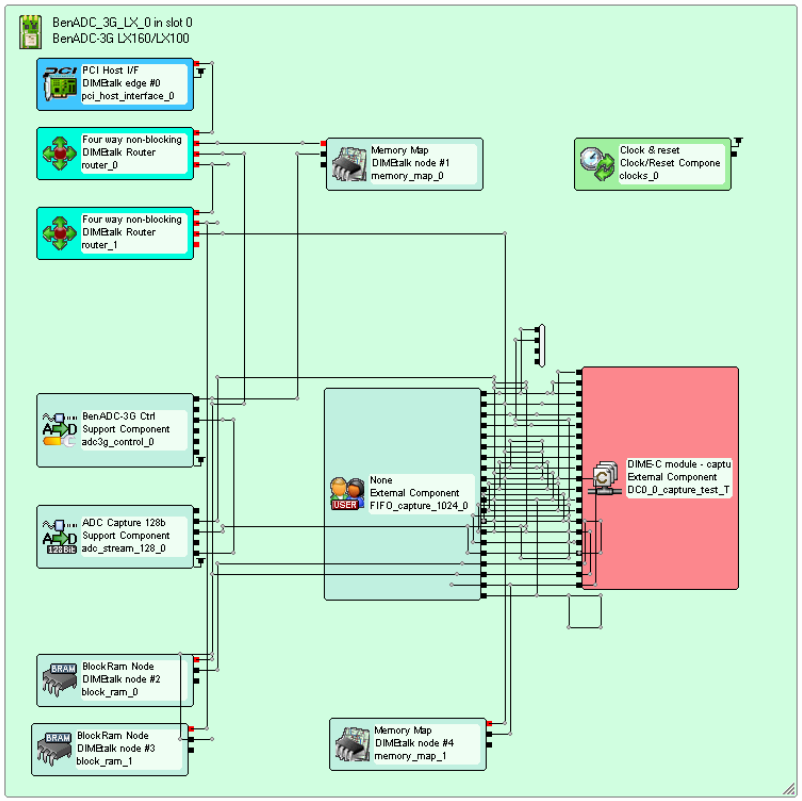
- Requires the Design of Two Custom VHDL Cores
  - Fully-Pipelined FFT Core
  - Fully-Pipelined Result Separator
- Cores are integrated into DIME-C
  - Wrapper Packages Data and Control Signals in a manner understood by DIME-C
  - XML Descriptor Created
  - Cores are clocked at twice the rate of DIME-C generated logic



System-Level View

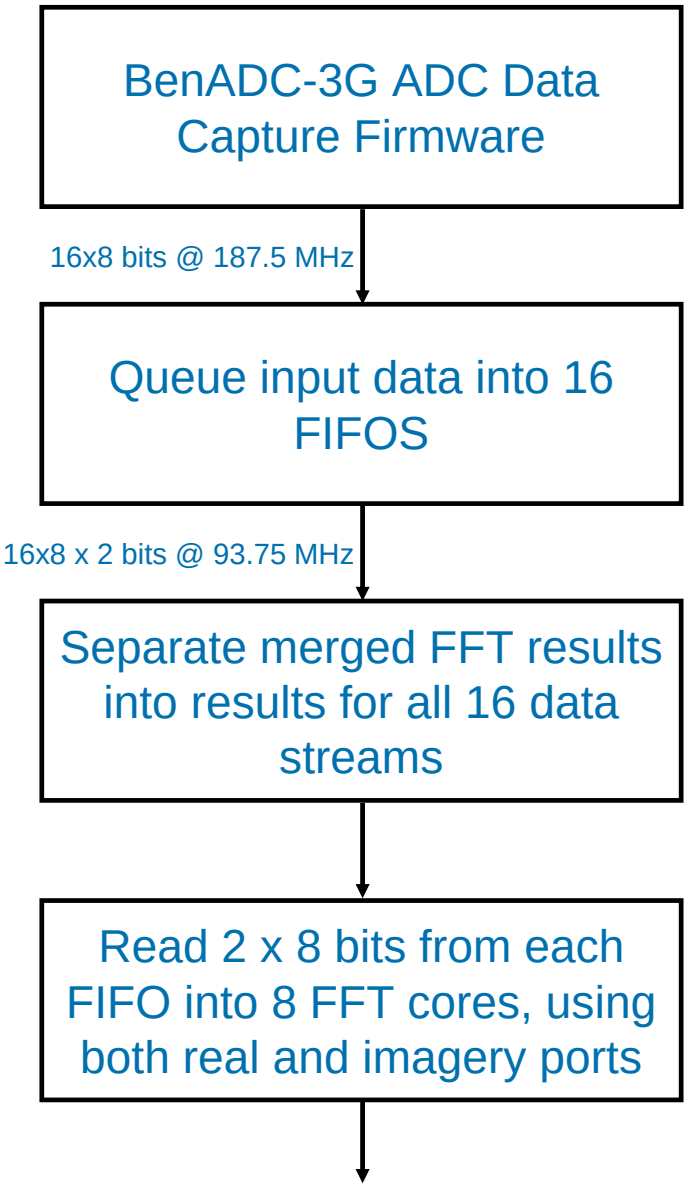


- DIMETalk:
  - Integrates DIME-C Streaming FFT Architecture with The Input Capture Core
  - Manages Memory Allocation
  - Maps Everything to BenADC-3G-LX160
  - Launches Build Scripts (ISE called behind the scenes)
  - Sets up FPGA-Host Communications Network



Nallatech
Our Expertise. Your Success.

System Design for 3GSPS Pipelined FFTs



- Single channel of BenADC-3G brings in 8-bit data in at a rate of 3 GHz
  - No single FFT transform could handle this rate of data ingress
  - Multiple parallel transforms required
  - Target clock rate - 187.5 MHz (3 GHz / 16)
- At 187.5 MHz, 16 parallel FFT computations are required in order to match the data ingest rate
- DIME-C generated VHDL
  - Targets 93.75MHz (3 GHz / 32)
  - Functional cores are dual-ported, and double-clocked by DIME-C logic
- Queue up 16 sets of 1024 points, the raw real data for 16 independent transforms
- FFT core
  - Accepts 4 inputs per clock cycle:
    - 2 Imaginary 8-bit inputs, 2 Real 8-bits outputs
  - Two independent transforms using one core
    - Real input stream 1 goes in real input
    - Real input stream 2 goes in imaginary input
    - Requires a hardware component to untangle the results
  - Results are 19-bit 2's bit complement integers
    - Mapped to 32-bit integers in DIME-C

Nallatech
Our Expertise. Your Success.

Performance and Results

- Streaming FFT Architecture Implemented on BenADC-3G
  - Naïve Computation Using FPUs would require 150 *sustained* GFLOPS
  - Using optimal integer arithmetic and memory structure, entire operation takes place on a single FPGA

Resource Consumption (LX160)		
Slices	DSP48s	BRAMB16s
46,779 (69%)	96 (100%)	236 (82%)

- Two People Worked on the Project for 2 months
  - Majority of project time was spent creating and verifying the 3 firmware cores
  - With IP cores in place, creating and modifying the entire system is relatively trivial
  - Occasionally, time spent waiting on place and route cannot be usefully used for concurrent project work

Nallatech
Our Expertise. Your Success.

Conclusions & Future Work

- Core/Library Development
  - Creation, Verification, Generation, Meta-data description
  - Typically proprietary for FPGA Computers
  - Standardization is key
    - Efforts in this area include OpenFPGA CORELIB, IP-XACT and OCP-IP
  - Library available accelerates application developments
- Integration of External Data to Processing Pipe
  - Typically HDL knowledge required
  - Improve ability to handle memory staging/corner turning
  - VITA 57 investigating I/O standardization to FPGAs
- FPGA for Sensor Data Formatting and Processing
  - Can be intimate with I/O interface
  - Need to orientate high-level tools to FPGA strengths
  - Lowest latency and high bandwidth
  - Reduce power and size
  - High Speed design need user intervention of device tools
    - Barrier to pure abstraction models

Nallatech
Our Expertise. Your Success.