Rapid Prototyping of Communication Waveforms from a High-Level Design Language

Mark Beardslee, Ph.D.

Coherent Logix™

Enabling Low Power Software Defined Systems

Photo Courtesy NASA

Agenda - The Rapid Prototyping System

- **Design Goals**
- **Implementation**
  - High-level design
  - Hardware platform
  - Tool flow
- **Results**

The Goals of the Rapid Prototyping System

Goal: Design Productivity

- **High-level model-based design input**
- **Automated translation to hardware**
- **Real-time execution**
- **High-level debug and verification**

Goal: a Path to Performance

- **Performance Evaluation**
  - Automated characterization
  - Tools for real-time analysis

- **Performance Improvement**
  - Change mapping to hardware
  - Re-implement blocks
  - Re-factor the architecture

Overview: The Rapid Prototyping System

- **Based on a high-level design system: Simulink™ from The Mathworks™**
  - Graphical, dataflow style design representation
  - Coarse parallelism provided by the designer
- **Real-time hardware implementation system**
  - Based on the "C" programming language
  - Efficiently implements dataflow style designs
- **Simulink-to-hardware tool flow**
  - Automatically maps the Simulink design to hardware
- **Debugging and analysis infrastructure**

High-Level Model-Based Design Input

Simulink for Design Implementation

- **Employ only built-in Simulink blocks and libraries**
  - No vendor specific block sets
- **Generate "C" models for each block**
  - Simulink's "Real-time Workshop" tool
  - Uni-processor "C" model
- **Automated test vector capture from simulation**
  - Used to verify translated blocks
  - Verify entire translated design

Real-Time Hardware System

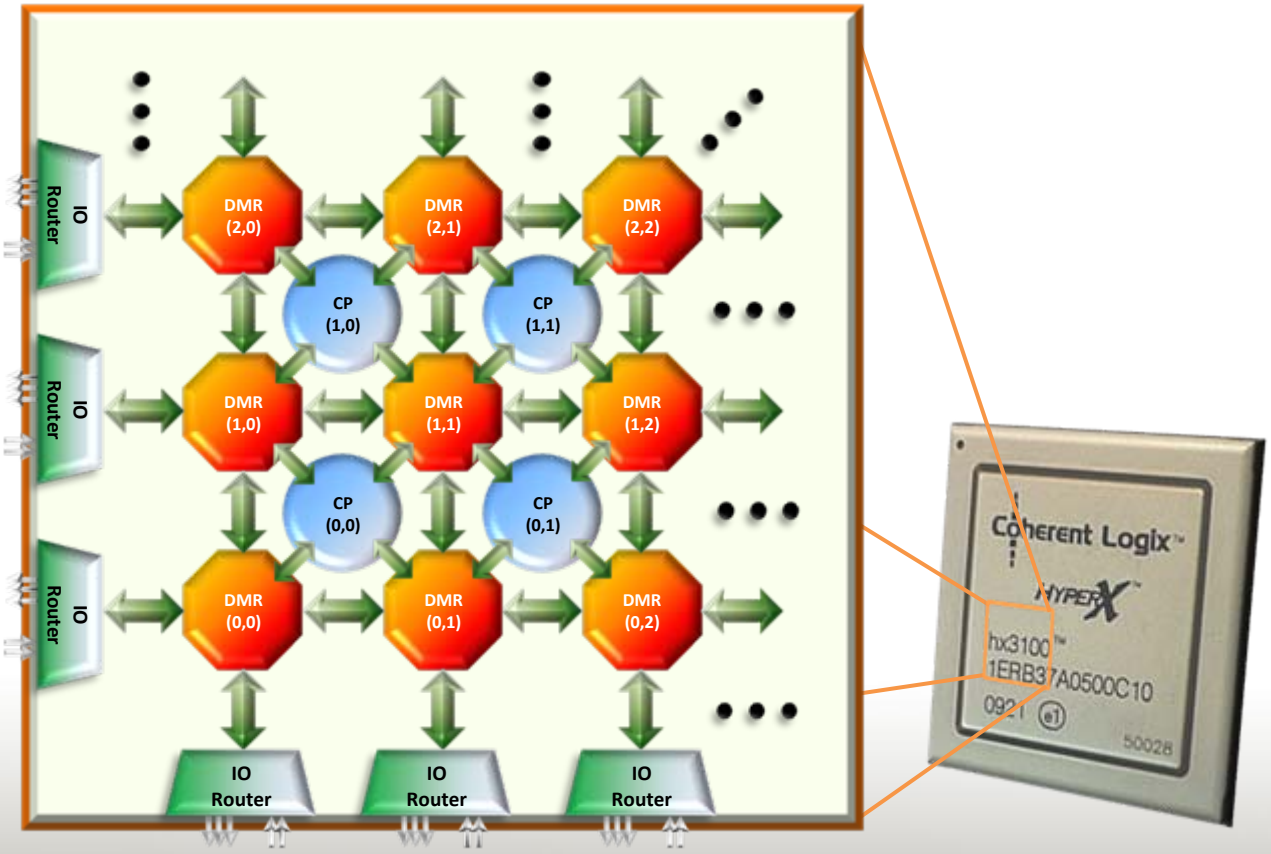
Real-Time Hardware System

- **The hardware platform is inherently well suited to implementing dataflow designs**

- **Compute engine – the HyperX™ Processor**
  - Compute Nodes:
    - 10 x 10 Grid of Core Processors
    - Each executes a "C" language program (MIMD)
    - Integer and floating point math supported
  - Data Communication:
    - 11x11 fabric of Data Memory and Router (DMR) blocks
    - DMA engines for processor-to-processor communication

- **Multiple HyperX processors form larger grids**

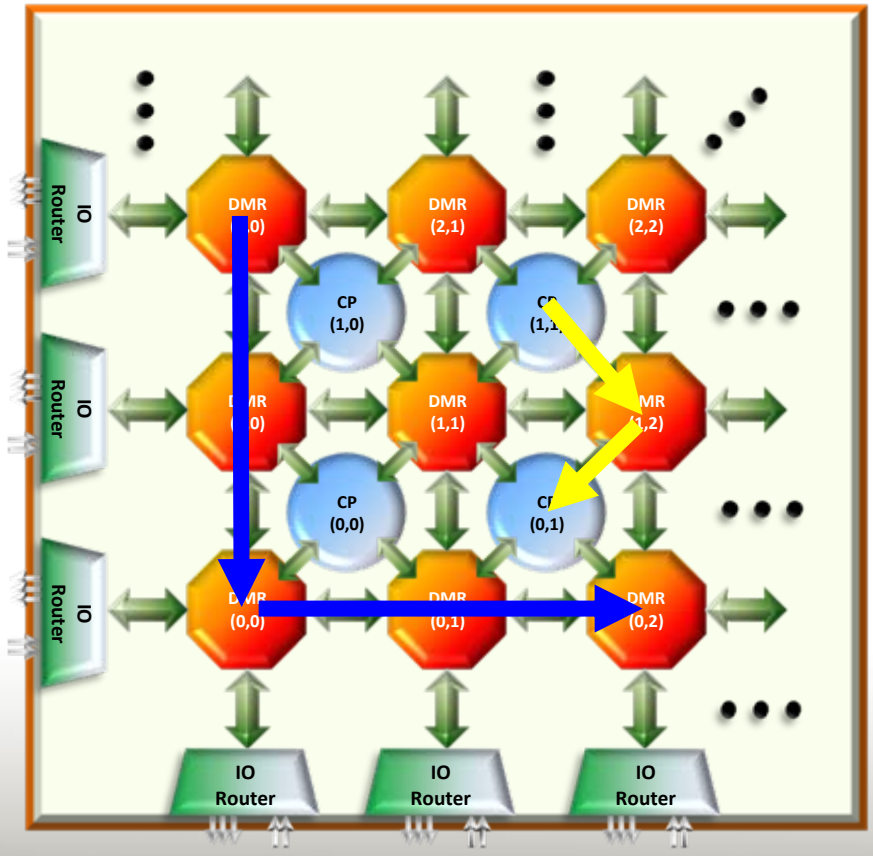
The HyperX™ Processor: hx3100™



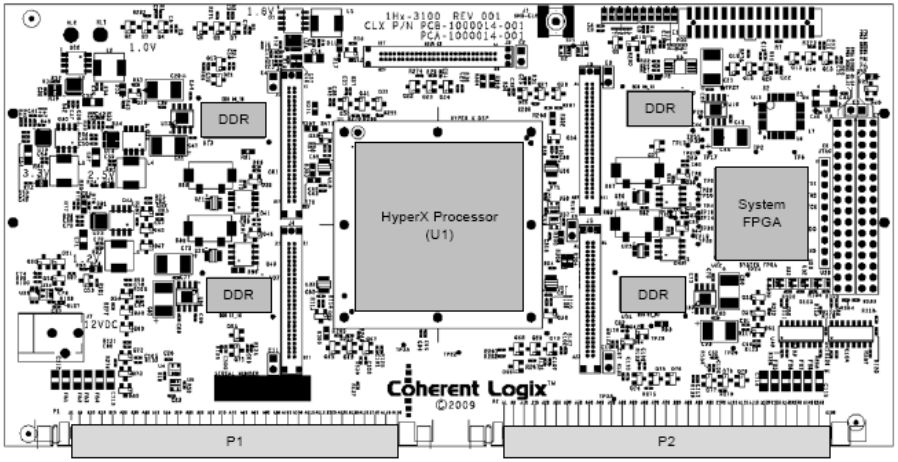
Processor to Processor Communication

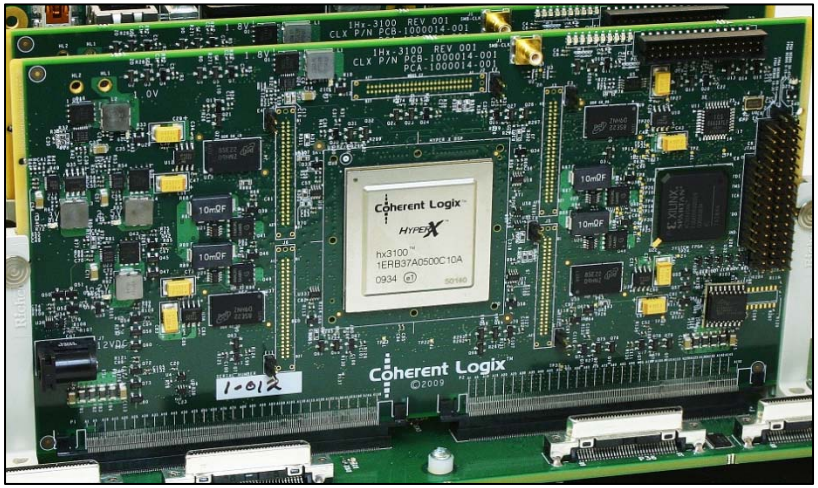
DMA/router – “message passing” communication

Shared variable communication



hxHADS Development System

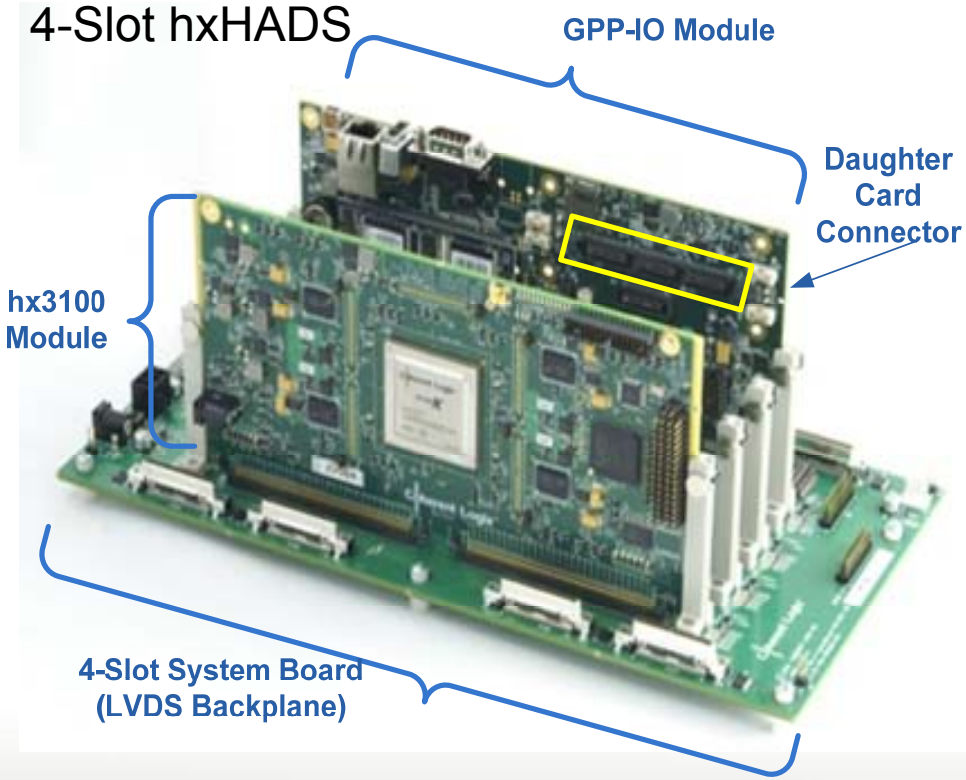




- HyperX chip(s)
- DRAM
- FLASH RAM
- System connections

hxHADS: Hardware Application Development System

4-Slot hxHADS

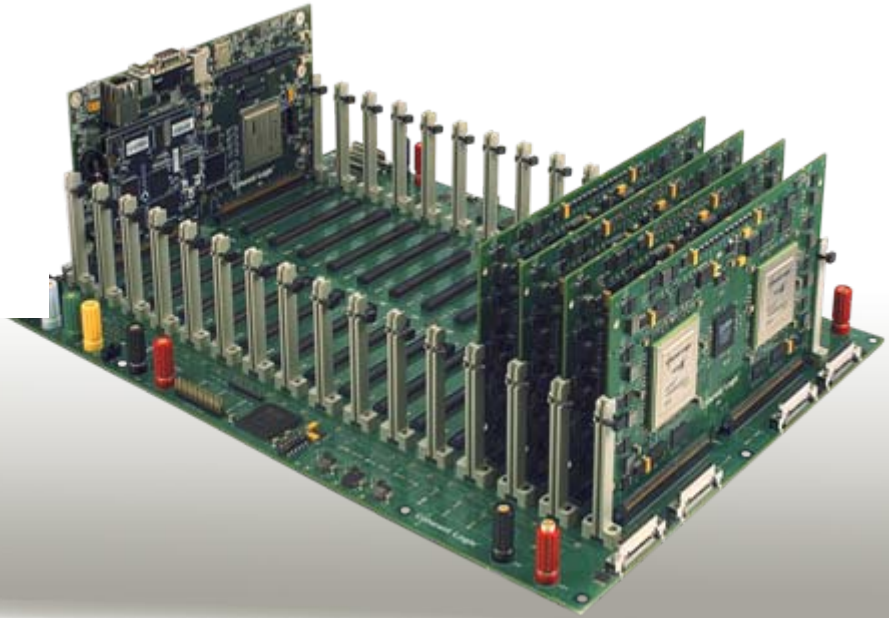


Extendable processor fabric

External I/O support

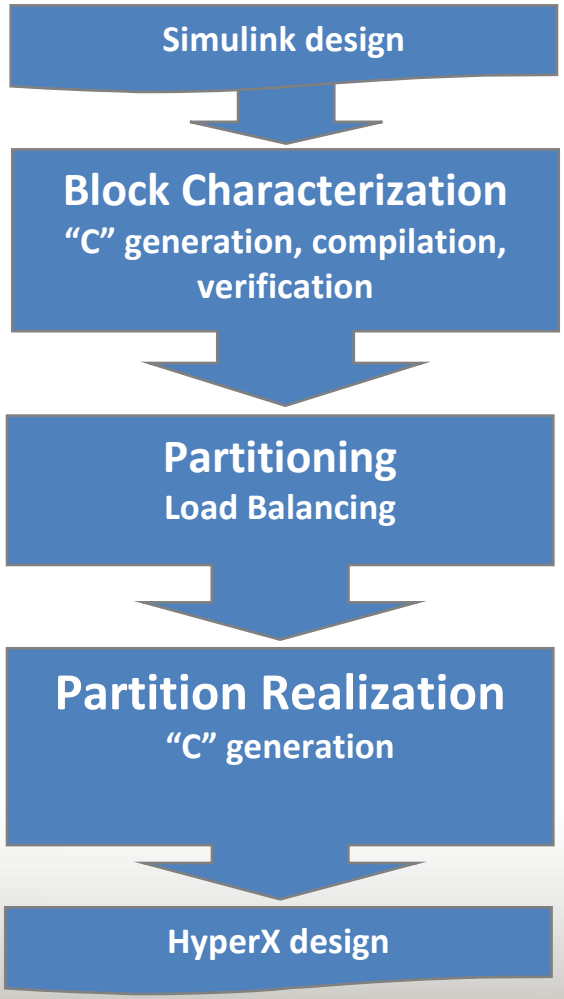
Data converters

16-Slot hxHADS



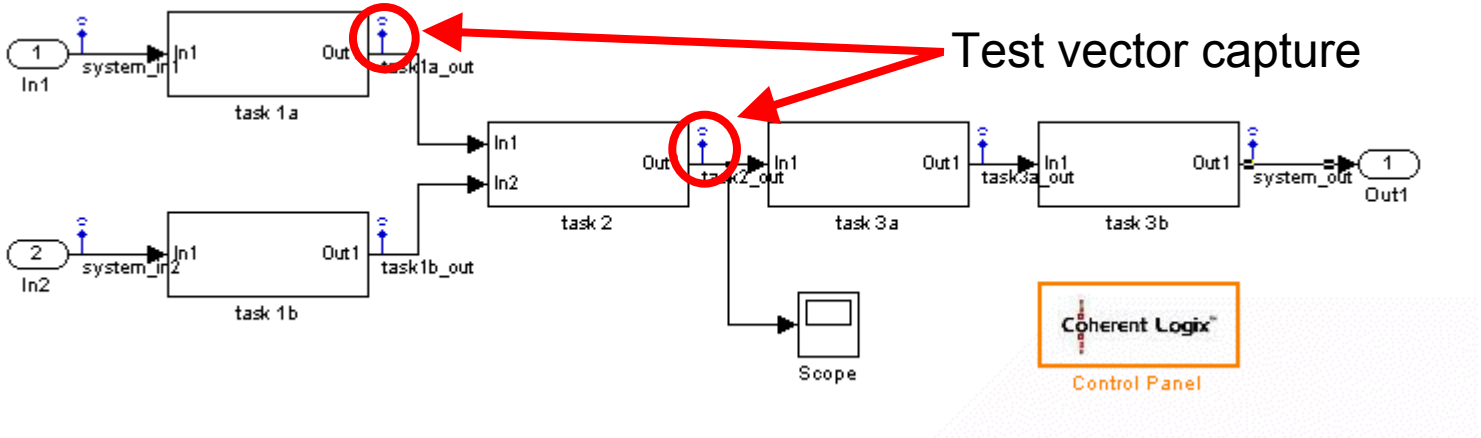
Tool Flow for Translation to Hardware

Model Based Design Flow (MBDF) Overview



MBDF – Generation Step

- Generate “C” code using “Real-time Workshop”
- Wrap the code for each block in a HyperX communication layer
- Automatically select signals for vector capture and run Simulink simulation



- Export a test bench for each block

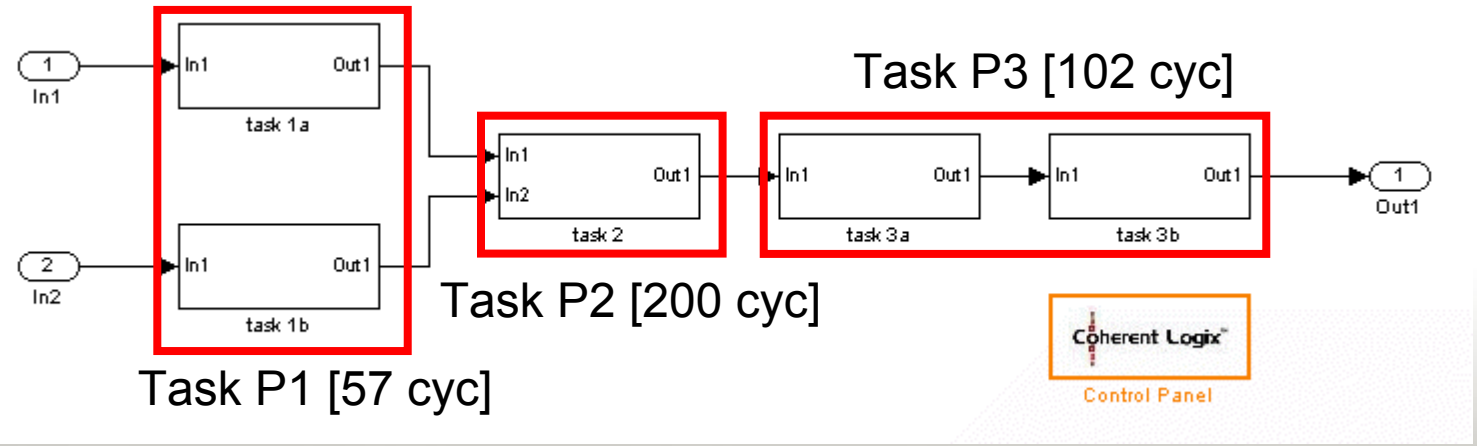
17

MBDF – Block Characterization

- **Compile and simulate using HxISDE tools**
- **Record block performance**
  - HyperX cycles of latency
  - Instruction memory usage
  - Data memory usage
- **Avoid redundant characterization – the characterization database**
  - Start with a pre-populated DB

MBDF – Automatic Partitioning

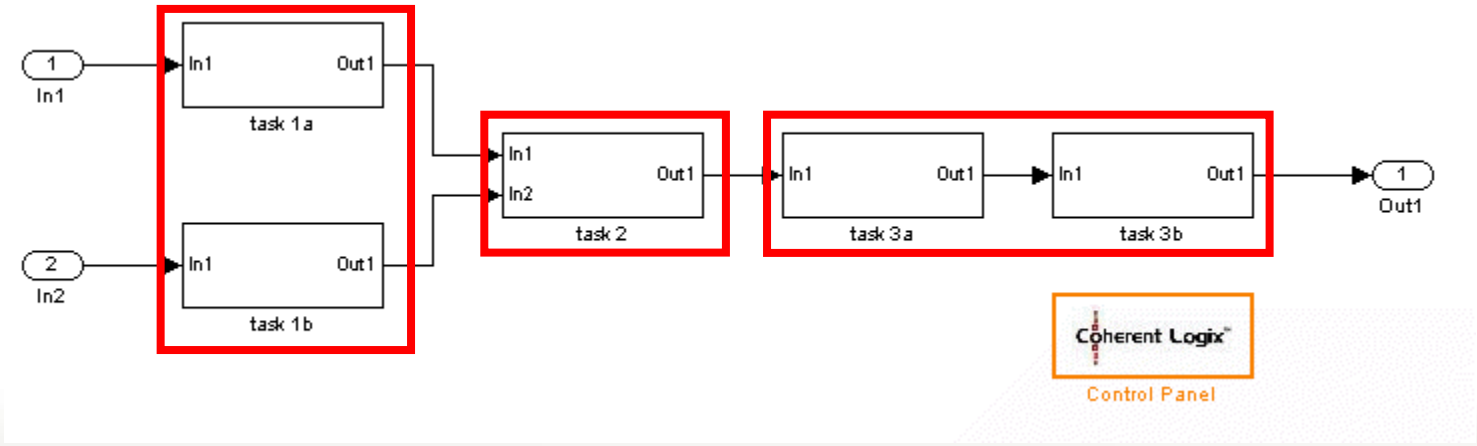
- **Partitioning is guided by**
  - User given performance constraints (throughput and latency)
  - User given hardware resource constraints (processors and memory)
  - Inherent platform limits (instruction and data memory)
- **Manual partitions can be assigned within Simulink GUI**
- **Performance model: Block characteristics and comm cost model**



19

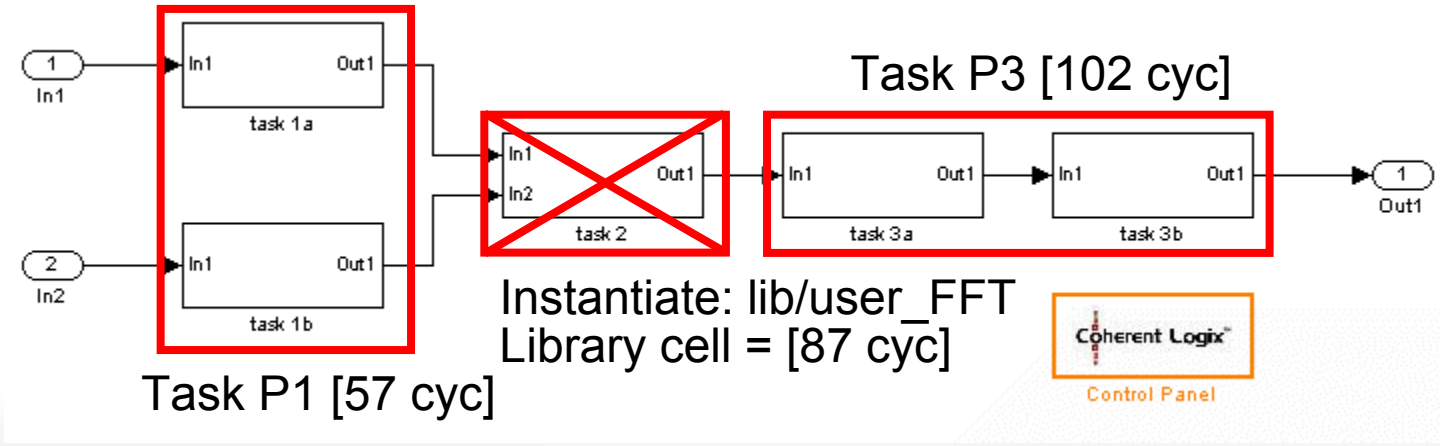
MBDF – Partition Realization

- **Two options:**
  - Generate “C” code for each partition in the design:
    - Opportunity for “Compiler Like” optimizations
  - Platform supported multi-tasking



MBDF – Block Speedup

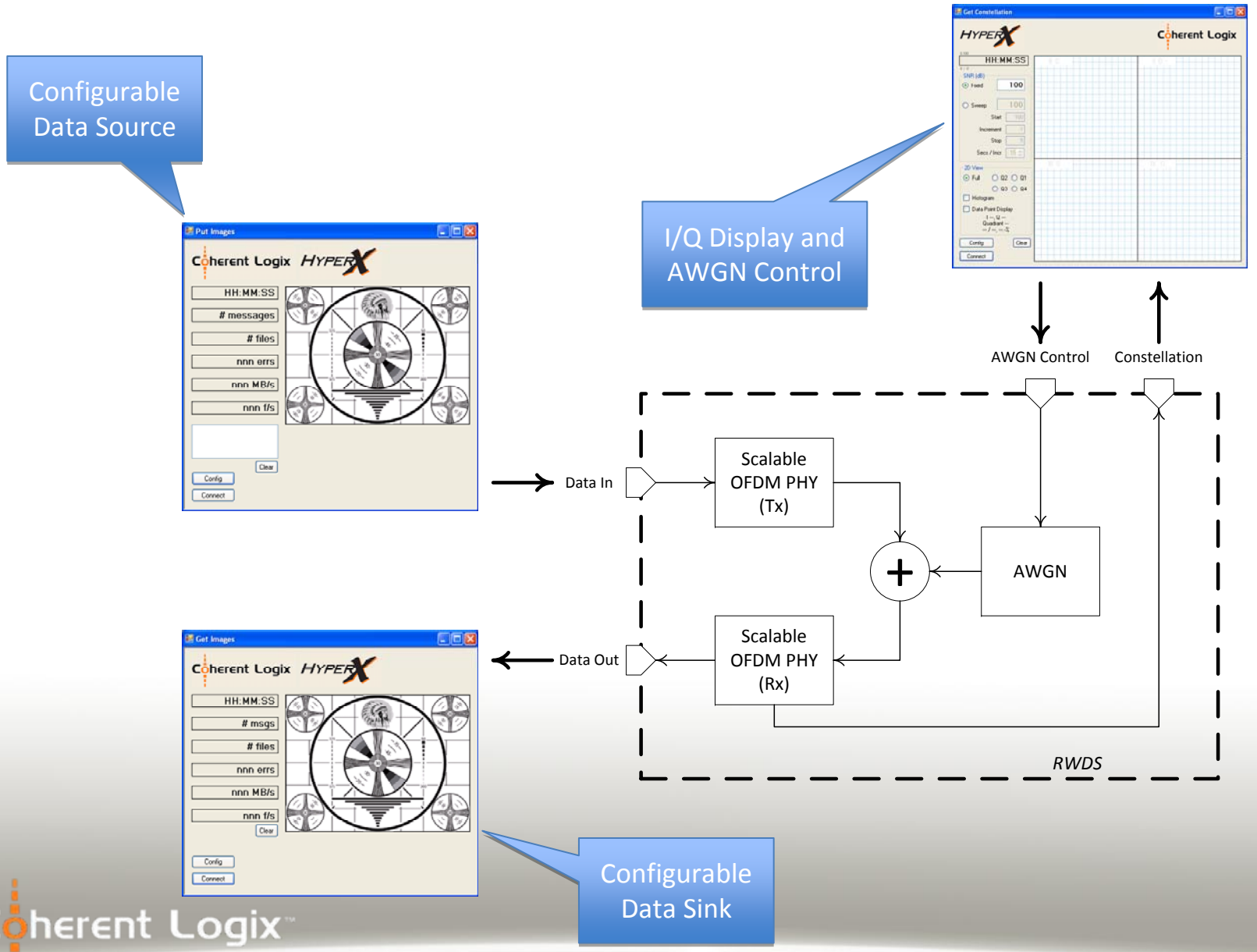
- **Block restructuring**
- **Block replacement**
  - Generated code can limit design performance
  - Hand-written C blocks replace cells within the generated design



Real-time Debug and Analysis: Overview

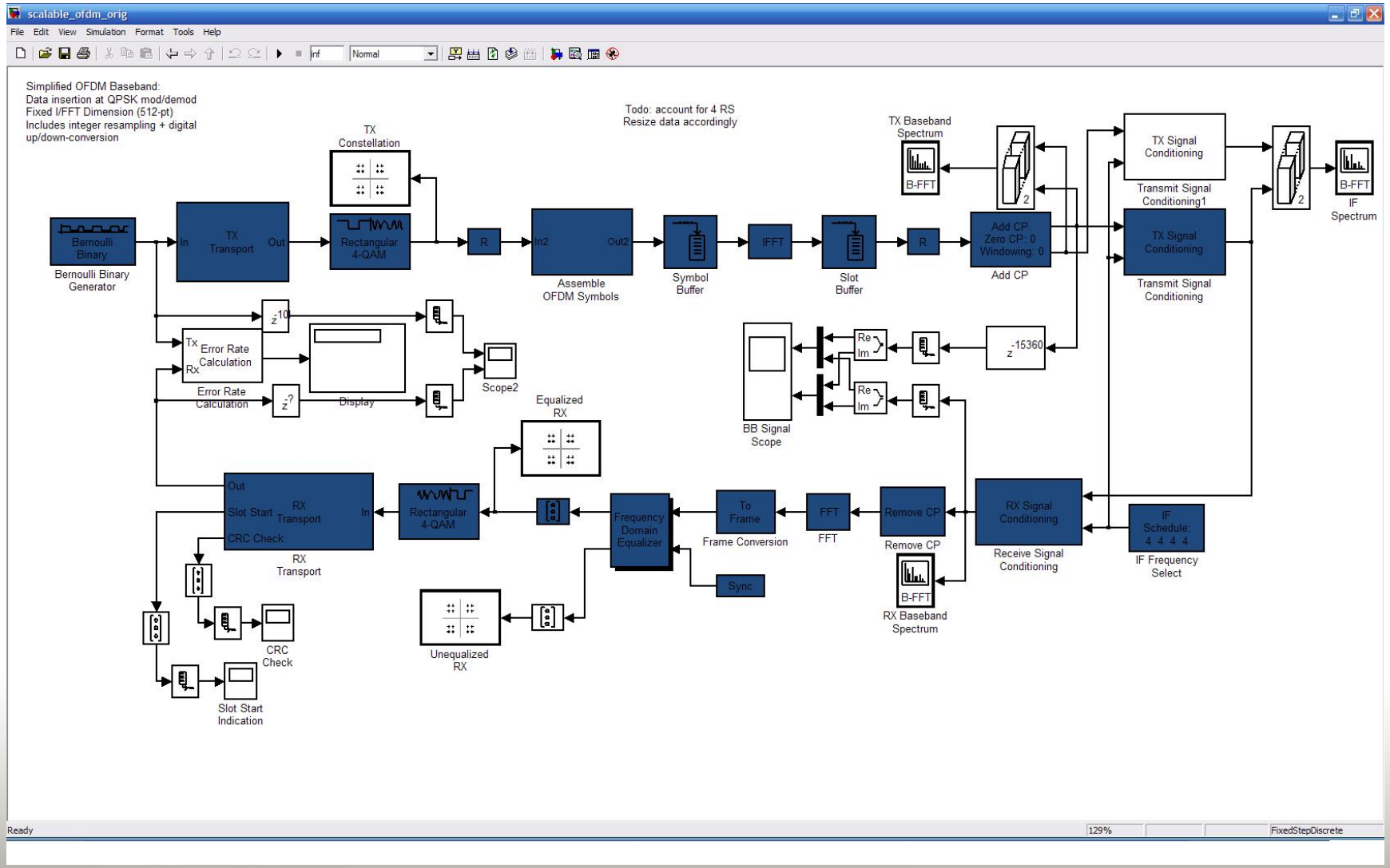
- **Real-time design instrumentation**
  - "Scope" and "Display" blocks
  - Error calculation
- **Configurable sources/sinks**
- **Performance characterization**
  - Final waveform validation: BER, PER, EVM performance assessment
- **Real-time control of design parameters**

Real-Time Analysis Framework



Results

Scalable OFDM Transmitter/Receiver



Results – SC-OFDM 1.44Mhz Transmitter

- Target throughput
  - 83.3 µs
- Blocks: 54
- CP usage
  - 12 CP's

Encoder	7.8 µs
Puncture	2.0 µs
matrixInterleaver	4.3 µs
blockInterleaver	4.3 µs
QAM	9.6 µs
Mod	2.2 µs
Group	12.5 µs
Pilots	0.3 µs
Assemble	15.0 µs
Pad	3.3 µs
Transform	60.7 µs
Cyclic_Multiplex	6.2 µs

26

Results – SC-OFDM 1.44Mhz Receiver

- Target throughput
  - 83.3 µs
- Blocks: 55
- CP usage
  - 12 CP's

Remove_Demulti	3.4 µs
Transform	56.8 µs
Select	5.1 µs
Equalizer_Sub1	6.7 µs
Equalizer_Sub2	31.5 µs
Dissasseble_Sub1	29.7 µs
Dissassemble_Sub2	9.4 µs
QAM_demod	19.8 µs
blockDeinterleaver	4.3 µs
matrixDeinterleaver	8.2 µs
Viterbi	1224.3 µs
Demod_1	27.1 µs

27

Results – LTE 1.44Mhz Receiver [with Viterbi Library cell]

- Target throughput
  - 83.3 µs
- Blocks: 55
- CP usage
  - 12 CP's

Remove_Demulti	3.4 µs
Transform	56.8 µs
Select	5.1 µs
Equalizer_Sub1	6.7 µs
Equalizer_Sub2	31.5 µs
Dissassemble_Sub1	29.7 µs
Dissassemble_Sub2	9.4 µs
QAM_demod	19.8 µs
blockDeinterleaver	4.3 µs
matrixDeinterleaver	8.2 µs
Viterbi	45.0 µs
Demod_1	27.1 µs

28

Thank you for your time and attention!

Mark Beardslee, Ph.D., Software Architect

Coherent Logix, Incorporated

+1-408-242-4801, Fax: +1-512-382-8941

beards@coherentlogix.com

Photo Courtesy NASA