MATLAB EXPO 2018
Real-Time Testing in a Modern, Agile Development Workflow
Simon Eriksson – Application Engineer
Demo – Going from Desktop Testing to Real-Time Testing
Key Take-Aways From This Presentation

- Agile = iterative and short design cycles, high degree of re-use of digital assets is essential

- Technology landscape demands faster control loops, prototyping with SW is not enough anymore
Overview on Real-Time Testing

- Rapid Control Prototyping
  - Run your algorithm against real HW without any manual coding
  - Tune your algorithm live directly from Simulink
Overview on Real-Time Testing

- Hardware-In-the-Loop (HIL)
  - Test your controls against a virtual plant model
  - Test with real I/O, without breaking anything!

![Diagram of HIL setup with Development computer, HIL simulator, and Controller(s)]
Real-Time Simulation and Testing
Simulink Real-Time for RCP and HIL

System Requirements

System Design

Rapid Control Prototyping

Coding

Production Code Generation

Software Design

Software Integration

On-Target Rapid Prototyping

Software-in-the-Loop

Hardware/Software Integration

Processor-in-the-Loop

Hardware-in-the-Loop

System Integration & Calibration
Agile and Real-Time Testing

- **Rapid Control Prototyping**
  - Get ideas in front of stakeholders
  - Changes in SW
  - Changes in HW
  - Deployable prototypes

- **HIL**
  - CI to include HIL as well
  - New/changing requirements
  - “back-to-back” MIL, SIL, PIL and HIL testing
How To Stay Fast/Effective i.e. Agile?

Agile = iterative and with “short” design cycles

http://www.yodiz.com/
http://modernagile.org/
http://alphatechglobal.com/
How To Stay Fast/Effective i.e. Agile?

- MBD $\rightarrow$ **Re-use** of digital assets!

- Digital Assets:
  - Models
  - Test Vectors
  - Field Data
  - Requirements
  - Legacy code
  - …
Rapid Controls Prototyping
Rapid Controls Prototyping - Challenges and Solutions
“Push button” workflow for both SW and HW

- **Challenges**
  - Faster control loops. Before kHz range now MHz
  - Many teams/competencies (SW and HW) required

- **Solutions**
  - Improved HDL code generation allows for prototyping on FPGA
  - Integration with Simulink Real-Time enables connectivity to wide range of I/O combined with FPGAs without manual coding
Rapid Controls Prototyping
On HW
Create FPGA I/O and Algorithmic Subsystems
Accelerate parts of your Simulink model using automated HDL code generation

- Achieve closed-loop sample rates up to several MHz
- Quick reconfiguration of FPGA I/O promotes a flexible real-time testing environment and very fast design iterations

Requirements: HDL Coder, Vivado Studio
Simulink Programmable FPGA I/O modules
CPU to FPGA communication – DMA

Usage:
- High data throughput between FPGA and x86 CPU
- Low latency link between FPGA and x86 CPU
Simulink Programmable FPGA I/O modules
CPU to FPGA communication – DMA

Use Cases:
- Direct Streaming
  - High Speed Data logging (FPGA to CPU)
  - Data playback (CPU to FPGA)
- Onboard RAM
  - Failure debugging with Ring Buffer
- Co-processor modes
  - Interrupt based
  - Polling mode based
FPGA-based I/O Modules
Multiple Interconnected FPGA-based I/O Modules

- Closed-loop rates of 500 kHz and faster
- > 100 analog and digital I/O lines
- Low latency FPGA interconnects (SPI and Aurora protocols)
- Algorithmic execution over multiple FPGAs

Electric motors (HIL)
Solar inverters for MW grids
100kW DC-DC converter
Native Floating-Point
Native floating-point support in HDL Coder for Real-Time Testing

Ok cool, but what does this mean for me?

- **Rapid Control Prototyping**
  - Less effort going to FPGA. \(\leftrightarrow\) i.e. more people can do it and be faster too!
  - Create golden reference. \(\leftrightarrow\) “this is how good it can be”, prove design decisions.

- **HIL Testing**
  - Easier to take models of physical systems, which by nature can be numerically sensitive, to run on FPGA \(\leftrightarrow\) Less risk and time spent creating plant models for real-time
Before: Floating-Point Design Needed to be Converted to Fixed-Point to Generate HDL Code
Sometimes you need to start and stay in floating point to ...

- Implement algorithms with large or unknown dynamic ranges (i.e. integrators in feedback loops)
- Implement operations that are difficult to design in fixed-point (i.e. atan2)

Requires extra integrator saturation to guarantee mathematics will not overflow
Native floating-point support in HDL Coder

- Vendor-independent RTL for FPGA and/or ASIC design
- Full range of IEEE-754 features
  - Optional support for Denormals, INF, NAN, Rounding, …
- Extensive Math and Trigonometry Block support

Operators

- Add, Subtract
- Mul, Gain, Abs
- Min, Max, Relops
- Divide, Mod, Rem
- Reciprocal, Sin, Cos
- Atan, Atan2
- Log, Exp, Power
- Sqrt & Inverse Sqrt
- Data type conversions

Native single-precision support for over 130 Simulink blocks
Hardware-In-the-Loop
Hardware-In-the-Loop and it’s *Agile* Benefits

- HIL enables more efficient/agile testing by
  - Being Repeatable
  - Being Scalable
  - “Failing” without breaking anything
  - Automation
HIL - Challenges and Solutions
Automation and re-use going from testing on desktop and in real-time

- Challenges
  - Sharing test data (test vectors, test results)
  - HW installations can become large or a highly booked resource
  - Numerical sensitivity when creating plant models

- Solutions
  - Integrated desktop and real-time testing framework
  - New high performance formfactors allow for “desktop HIL”
  - Support for native floating-point allow to stay in single precision when running on FPGA
HIL Systems
Typical HW setups

- Intel Core i7 4.2 GHz quad core, or two Xeon CPUs with 20 cores
- Over 100 I/O modules installable leveraging additional expansion chassis

- Challenges:
  - Usually located in large labs, that require timeslots to be booked etc.
  - Not very mobile for use outside the lab
“Desktop HIL”
Power in a Small Form Factor

- **Solution**
  - Ideal for mobile use/“desktop HIL” of for Rapid Control Prototyping
  - Intel Core i7 2.5 GHz dual core CPU and FPGAs
  - Stackable: Up to 14 I/O modules supported
  - **Multi-node HIL-Simulator** for automated testing of large scale plants (e.g. 64 Profibus / 32 Profinet / Analog I/O / Digital I/O)
Product Development Process

- System Requirements
- System Design
- Software Design
- Simulation
- Rapid Control Prototyping
- On-Target Rapid Prototyping
- Coding
- Production Code Generation
- Software Integration
- Software-in-the-Loop
- Hardware Integration
- Hardware-in-the-Loop
- Processor-in-the-Loop
- System Integration & Calibration
- Simulink Test
- Simulink Real-Time
Fully Tested Algorithm in Simulink Test
Fully Tested Algorithm in Simulink Test
Automated Testing with Simulink Test
Real-Time Test Automation, ideal for Hardware-in-the-Loop

- Integrates with MATLAB Unit Test
- Supports the TAP protocol to run from CI systems like Jenkins, VSTS, etc.
Key Take-Aways From This Presentation

- Agile = iterative and short design cycles, high degree of re-use of digital assets is essential

- Technology landscape demands faster control loops, prototyping with SW is not enough anymore
Questions?
Thank You!