Multi-target Production Code Generation for optimal use of Hardware Resources

Sebastien Dupertuis, EngD, B.Sc.
Senior Applications Engineer
Three Key Takeaways

- Target heterogeneous systems from one model in MATLAB and Simulink
- Automation through automatic code generation and connect to hardware
- Verification at each step of the development process
IMT Developed a highly complex ventilator called Bellavista, partially unknown, MIMO system requiring total robustness and reliability

Challenge
Develop and deliver such a complex and reliable product without spending too much time and money

Solution
Use Model-Based Design to model, implement, test, and deploy the MIMO system onto an embedded hardware

Results
- Improved time to market and used fewer prototypes
- 156k+ lines of source code generated in 15 minutes
- No bugs in the generated code since the project began
- Validation time for ventilation performance reduced by 80%, 2 weeks down to 2 days

“Model-Based Design not only helps to deal with highly complex MIMO systems, but also enables us to identify problems early in the development process. That reduces costs and improves time to market.”

Matthias van der Staay
IMT
Agenda

- Multi-target Production Code Generation
  - Who
  - Capabilities
  - Workflows
  - Standards
  - Summary
Philips Healthcare MRI scanner

AirSonea device, which connects to a patient's smartphone

Sonova's hearing aid and cochlear implant solutions

The HB-SIA aircraft on a test flight over San Francisco Bay

Toyota engine

Alstom Grid's HVDC demonstrator system with power converter modules

AirSonea

Philips Healthcare MRI scanner

MATLAB EXPO 2016

http://ch.mathworks.com/company/user_stories/
Model-Based Co-Design for C and HDL

Motor Control

Application – Stabilised Mirror Motor Controller

Doppler Radar

Application: Range Doppler Processing

PIL: Processor-in-loop
FIL: FPGA-in-loop
HIL: Hardware-in-loop

Agenda

- Multi-target Production Code Generation
  - Who
  - **Capabilities**
  - Workflows
  - Standards
  - Summary
Modeling: Three methods

- Simulink: Block Oriented For Systems Design
- Stateflow: State Machines for Event-based Logic
- MATLAB: Textual for Concise Numerics
Code Generation: **Five languages**

- C/intrinsics
- C++
- VHDL
- Verilog
- Structured Text
Hardware Support: Any device

- Portable code: any device for algorithm code generation supported
- Support packages offer select target-specific system executable generation and workflows
  - ARM ... C2000 ... Zynq
- Hardware vendors offer their own support packages
  - Freescale, Infineon, Microchip, Renesas, TI, STMicroelectronics, ...
Challenge: Selecting the Right Hardware?

**HW Characteristics:**
- Price
- Sampling frequency
- Workflow complexity
- Form-factor
- Availability
- ...

Let’s make an informed decision on HW selection within the development process...
Code generation for PLC / PAC

- **Target-Independent Models:**
  - Enabling design portability across PLC HW

- **Workflows:**
  - IEC 61131-3
  - Rockwell
  - Siemens
  - Beckhoff
  - B&R
  - …
Code Generation for DSP / MCU / SoC

- **Target-Independent Models:**
  - Enabling design portability across DSP/MCU and SoC

- **Workflows:**
  - ANSI C/C++
  - TI, ARM, etc
  - Verification In-the-Loop
  - Rapid prototyping and HIL test
  - Certification and standards

- **Optimized code for production:**
  - Target-aware code (ARM, TI, etc)
Code Generation for FPGA / ASIC / SoC

- **Target-Independent Models:**
  - Enabling design portability across FPGA/ASIC and SoC

- **Workflows:**
  - Verilog/VHDL
  - Xilinx, Altera, SoC
  - HDL co-simulation (Mentor/Cadence)
  - Verification In-the-Loop
  - Rapid prototyping
  - Certification and standards

- **Optimized code for production:**
  - Resource sharing, timing optimization
MathWorks Coders

One model to many targets
Agenda

- Multi-target Production Code Generation
  - Who
  - Capabilities
  - Workflows
  - Standards
  - Summary
Design Elaboration

E.g. Floating-point or Fixed-point?
Sample Times
Discrete or time continuous
Component Testing and Profiling

Algorithm verification via SIL, PIL, FIL

Is the generated code functionally equivalent to the model?

Non-Real-Time functional verification of the algorithm component
Prototyping

Does algorithm perform well on actual device with true latencies?

Exhibitor: Speedgoat

MATLAB EXPO 2016
Production Code Generation

Does the code work on production hardware?

Exhibitor: MathWorks
Results for ARM Cortex-A (FIR Filter)

Run Format: [ANSI or Ne10], [gcc no opt or gcc -02], ARM 1Ghz Cortex A8
Results for Xilinx Zynq-7000 All-Programmable SoC

Baseline: 30 (14%)
Constant Multipliers: 17 (8%)
Resource Sharing: 10 (4%)

Example: Field-Oriented Controller
Agenda

- Multi-target Production Code Generation
  - Who
  - Capabilities
  - Workflows
  - Standards
  - Summary
Coding Standards

MISRA-C

STARC HDL

MATLAB EXPO 2016
Model-Based Design – Certification Examples

**DO-178B (Level A)**
- Honeywell Aerospace USA
- Flight Control Systems

**ISO 26262**
- TRW Germany
- Electronic parking brake control system

**ARP4754 & DO-178**
- Airbus Helicopters
- Certified flight software

**IEC 62304**
- Weinmann Medical Germany
- Transport ventilator

**IEC 61508**
- Alstom Grid UK
- HDVC Power Systems

**EN 50128**
- Alstom France
- Train Control Systems
Agenda

- Multi-target Production Code Generation
  - Who
  - Capabilities
  - Workflows
  - Standards
  - Summary
Get Started with Automatic Code Generation

use 5 proven principles

1. Experiment with a small piece of the project
2. Re-use system-level models for implementation
3. Use automatic code generation to iterate towards maximum performance
4. Pick hardware platform that make sense for you
5. Take advantage of MathWorks resources

Link to MathWorks white paper