

Developing Heterogeneous Embedded Systems, **Simplified!**

Tuesday, August 30, 2011

© 2011 MathWorks GmbH

Agenda (Four things to takeaway!)

- Graphical embedded applications development
- Production code generation
- Achieving a reduction in verification effort
- Processor specific code generation

Mixed Hardware/Software Systems

- Many digital systems contain both hardware and software
- Combining hardware and software design tasks has several advantages
- One is that may accelerate the design process. Another is that may enable hardware/software trade-offs to be **made dynamically**, as the design progresses

Several Examples of Hardware/Software Co-Designs Exist

- Embedded microprocessor systems
- Heterogeneous multiprocessing systems
- Application-specific instruction set processors
- Special-purpose functional units
- Application-specific co-processor
- +more

Heterogeneous Applications

- Heterogeneous (or asymmetric) embedded systems present opportunities for
 - improving application throughput
 - reducing power
- On-board heterogeneity expects application to better match execution resources to
 - address a wider spectrum of system loads
 - capitalize on parallelism with high efficiency

Why Hardware/Software Co-Synthesis?

- Performance requirements
- Implementation cost
- Modifiability
- Nature of computation
- Concurrency
- Communication
- **TTM!**

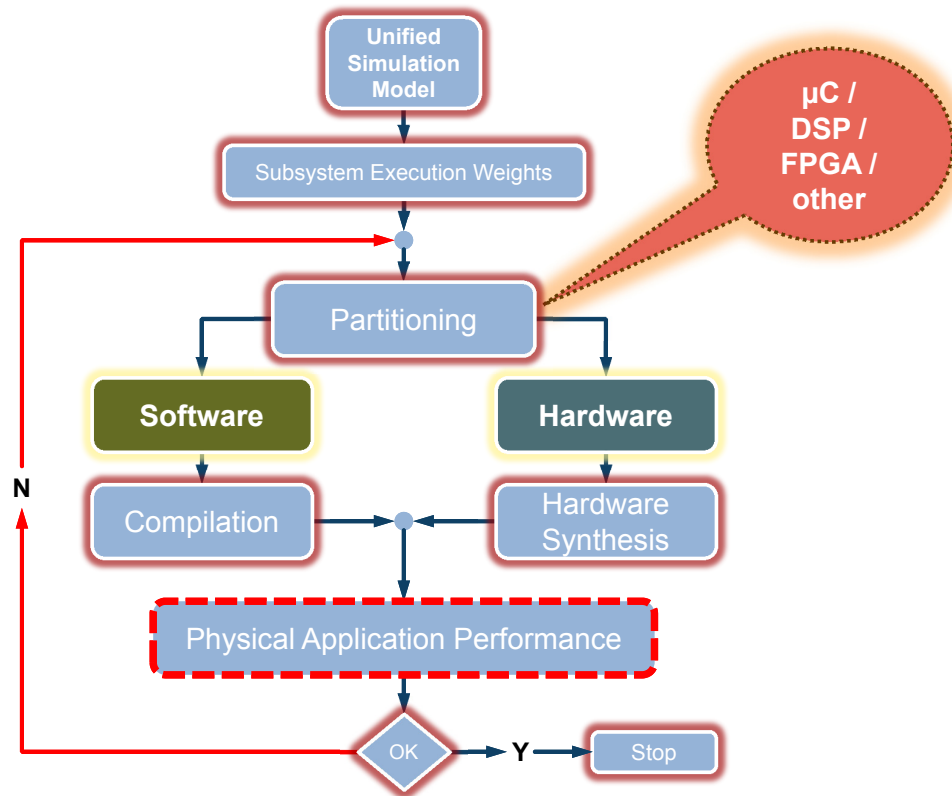
What is Graphical Development?

- Not a separate programming language!
- Essentially a library of functional blocks
- Gives a notion of timing and event driven simulations
- Sequential and concurrent in nature
 - Allows description and integration of complex hardware and software
- Inherent support for mixed data types
- Provides with capabilities for Modelling, **Simulation**, Analysis and Implementation

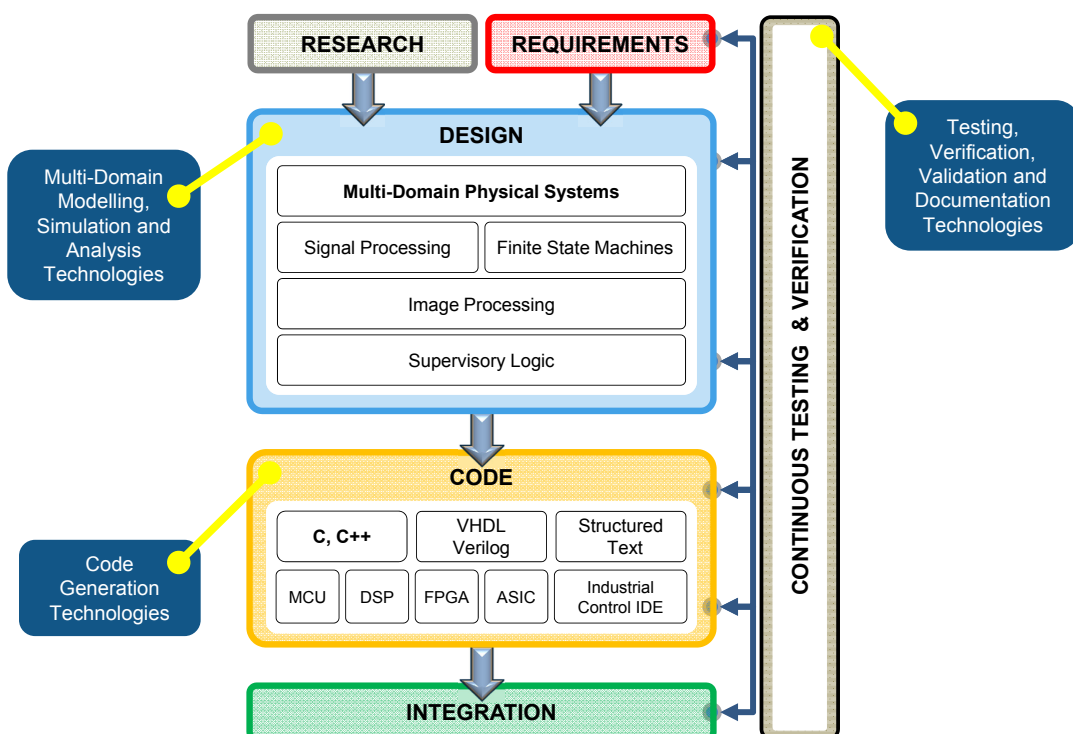
Simulation of Hardware/Software Systems

- Presents the problem of modelling the behaviour of a system based on the behaviour of the hardware and software components
- Requires a simulation environment that can understand the semantics of both the software and the hardware components

The Partitioning Decision



Model-Based Design

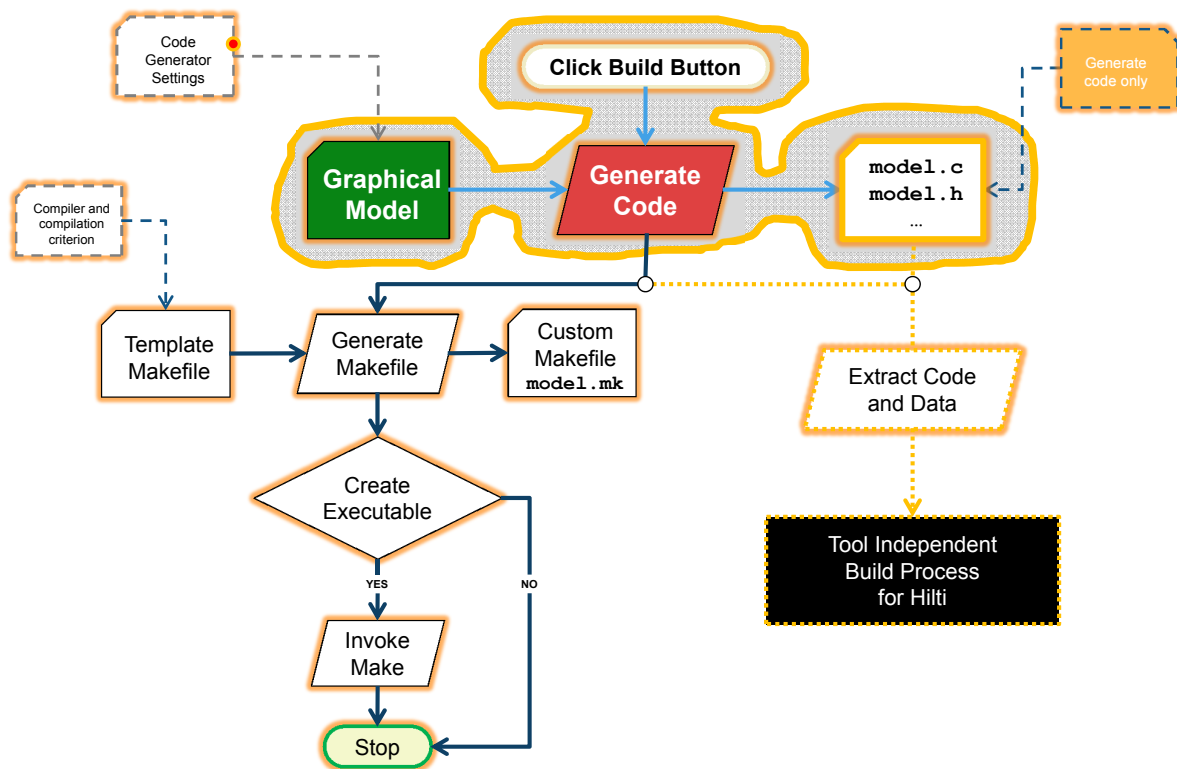


>>Demo

Important Elements of Code Generation

Customer Requirements	Our Tools
Reuse and Integration of Handwritten Code	✓
Data Management	✓
Simulation, Analysis and Verification	✓
Efficient Code Generation	✓
Coverage Analysis	✓
Model to Code Traceability and vice versa	✓
Automated Documentation Generation	✓
Source Code Verification	✓
Object Code Verification	✓
Requirements Traceability	✓

Model to Target Binary



13

Target Specific Optimizations

- Modern embedded processors and associated compilers support “**intrinsic**”
- Such **processor-specific** instructions execute much faster than their C and C++ equivalents
- These intrinsics can **optimize code performance** significantly
- Selecting your processor’s **Target Function Library** (TFL) enables you to generate processor-specific code that takes advantage the processor dependent intrinsics

14

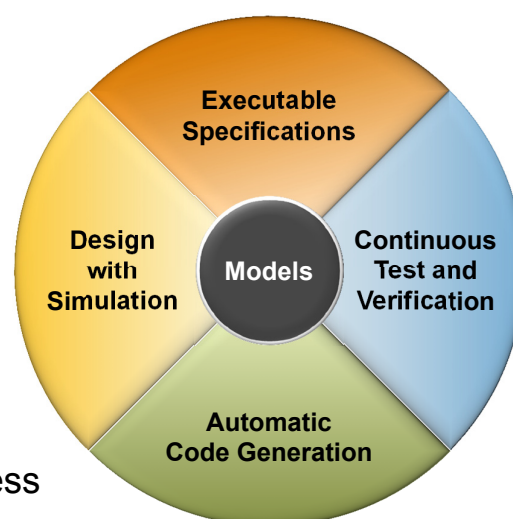
Summary and Conclusions

- Graphical modelling enables **unified design** of hardware and software systems
- All design based off of logical model
 - no HW/SW partition, unified approach
 - Maintained **throughout design process**
- **Concurrent** Design
- Hardware/Software **optimized for peak performance**

15

Summary and Conclusions

- **Provides a CAE solution** for multidomain system-level development
- **Uses simulation models** to mathematically describe the system
- **Automatically generates code** for real-time testing and implementation
- **Lets you test continuously** throughout the development process



16