Hardware-Software Implementation With Model-Based Design

Sudhir Sharma
Product Manager,
HDL Code Generation And Verification
The MathWorks
Agenda

- What is the System Design Challenge
- Solutions for Embedded Software Development
  - Automatic Code Generation
  - Verification
- Solutions for Hardware Development
  - Automatic Code Generation
  - Verification
System design to implementation gap
Integrated Design Flow for Embedded Software and Hardware

- Design, simulate, and validate system models and algorithms in MATLAB and Simulink
- Automatically generate production software for embedded processors
- Verify the software implementation against the system model
- Verify the hardware implementation against the system model

MATLAB® and Simulink®
Algorithm and System Design

Real-Time Workshop
Embedded Coder, Targets, Links

Simulink HDL Coder
Co-simulation links

Generate
Verify

C

HDL

MCU
DSP

FPGA
ASIC
Case Study: Sobel Edge Detection Algorithm
Floating-Point System Specification

- Start by developing a golden specification
Fixed-Point Modeling

Floating-point model

Fixed-point model

MathWorks
Aerospace and Defense Conference ’07
Fixed-Point Modeling

Floating-point model

Fixed-point model

Executable Specification

Elaborated Design

Difference
Implementation on DSP, GPP, or an FPGA?

Fixed-point model
Agenda

- What is the System Design Challenge
- Solutions for Embedded Software Development
  - Automatic Code Generation
  - Verification
- Solutions for Hardware Development
  - Automatic Code Generation
  - Verification
Implementation on DSP and GPP

Fixed-point model

Code generation options and preferences

Select target or flavor of generated code
Code Execution on Target and Profiling

- Build and execute
  - Auto-generate ‘C’ and ASM
  - Integrate RTOS and scheduler
  - Create full CCS project
  - Invoke compiler, linker, and download code
- Run on target
- Profile code performance

System profiling includes entire DSP application code

Subsystem profiling

MathWorks Aerospace and Defense Conference '07
Design Verification and Visualization:
Simulink as verification test bench

Processor and hardware-in-the-loop testing, simulation, visualization, and verification of embedded software with Simulink

Simulink test bench

Device or design under test (DUT)

Simulink system design embedded on DSP
Review: Code Generation for Embedded Software

- Code Generation
  - Real Time Workshop – ANSI/ISO C code for rapid prototyping, acceleration
  - Real Time Workshop Embedded Coder – Embedded deployment

- Links
  - Link for Altium TASKING
  - Link for Analog Devices VisualDSP++
  - Link for TI Code Composer Studio

- Targets
  - Target for TI C6000 DSP
  - Target for TI C2000 DSP
  - Target for Infineon C166 Microcontrollers
  - Target for Freescale MPC5xx Microcontrollers

MathWorks
Aerospace and Defense Conference ’07
Agenda

- What is the System Design Challenge
- Solutions for Embedded Software Development
  - Automatic Code Generation
  - Verification
- Solutions for Hardware Development
  - Automatic Code Generation
  - Verification
Code Generation for Hardware

Simulink HDL Coder
Correct-by-construction VHDL and Verilog code

Generated Verilog code

Lane Departure Warning System

Stateflow control logic

MathWorks Aerospace and Defense Conference '07
Fixed-Point Implementation on an FPGA
Design Space Exploration

- **Speed**
  How fast can this design run?

- **Area**
  Can I use a smaller chip?

- **Power**
  Can I target a mobile device?

- **Implementation Alternatives**
  - Sum & Product: Linear, Cascade, and Tree
  - Gain: Multiplier, CSD, Factored-CSD
  - Minimum/Maximum: Tree and Cascade
  - Lookup Table: Inline or hierarchical
Code Generation Options

Select subsystem, target language, directory

Select output options

Check model for errors

Generate HDL Code
More Code Generation Options

- Select reset and clock options
- Set language-specific options: input/output datatypes, timescale directives, …
Self-checking HDL test bench compares Simulink results to HDL results.
Automatic HDL Code Generation

- ‘Correct-by-construction’
  - Matches Fixed-Point System Model
  - Faster design implementation
  - Reduces verification burden
- Benefits Include:

1. Rapid FPGA implementation

2. Reference code for HDL engineers
Verification with system specification

```matlab
y_filter = y_directional_out1 -- ufix11
s <= signed(filter_input);
Delay2_process : PROCESS (clk, reset)
BEGIN
  IF reset = '1' THEN
    Delay2_out1 <= (OTHERS => '0')
  ELSEIF clk'event AND clk = '1' THEN
    IF enb = '1' THEN
      Delay2_out1 <= s;
      END IF;
    END IF;
  END PROCESS Delay2_process;
s_1 <= std_logic_vector(Delay2_out1);
```
Making full use of the system model

- Promotes parallelism in design and verification tasks
- Improves focus on critical areas
Making full use of the system model

- Promotes parallelism in design and verification tasks
- Improves focus on critical areas
- Accelerates verification at all levels
Making full use of the system model

- Promotes parallelism in design and verification tasks
- Improves focus on critical areas
- Accelerates verification at all levels
- Supports re-use and “what-if” scenarios
Implementation on an FPGA

Device utilization summary:

Selected Device: 4vsx25ff668-12

- Number of Slices: 1105 out of 10240 (10%)
- Number of Slice Flip Flops: 1903 out of 20480 (9%)
- Number of 4 input LUTs: 155 out of 20480 (0%)
- Number of IOs: 14
- Number of bonded IOBs: 14 out of 320 (4%)
- Number of GCLKs: 1 out of 32 (3%)

Design statistics:
- Minimum period: 4.106ns
- Maximum frequency: 243.546MHz
- Minimum input required time before clock: 3.683ns
- Maximum output delay after clock: 7.323ns
Review: Code Generation for Hardware

- Code Generation
  - Simulink® HDL Coder – FPGA and ASIC deployment using VHDL and Verilog
  - Filter Design HDL Coder – Filter implementation from MATLAB

- Links
  - Link for Mentor ModelSim
  - Link for Cadence® Incisive®
Summary

- Design and verify software and hardware from MATLAB and Simulink

- Accelerate product development using Model-Based Design