C and HDL Code Generation from MATLAB

Puneet Kumar
Principal Applications Engineer
Agenda

- Algorithm Design Process and Design Challenges

- MATLAB Coder - MATLAB to C Flow
  - Why translate MATLAB to C?
  - Challenges of manual translation
  - MATLAB Coder Workflow for generating code

- Accelerating Algorithm Execution

- HDL Coder - MATLAB to HDL Flow
  - Why translate MATLAB to HDL?
  - Challenges of manual translation
  - HDL Coder Workflow for generating code

- Q&A
Agenda

- Algorithm Design Process and Design Challenges
  - MATLAB Coder - MATLAB to C Flow
    - Why translate MATLAB to C?
    - Challenges of manual translation
    - MATLAB Coder Workflow for generating code

- Accelerating Algorithm Execution

- HDL Coder - MATLAB to HDL Flow
  - Why translate MATLAB to HDL?
  - Challenges of manual translation
  - HDL Coder Workflow for generating code

- Q&A
Algorithm Development Process

- Explore and discover
- Gain insight into problem
- Evaluate options, trade-offs

Requirements

Research & Design

- Design
- Test
- Elaborate

Implementation

**Desktop**
- .dll
- .exe
- .c, .cpp

**Embedded**
- C
- VHDL / Verilog
- Structured Text
The Algorithm Design Challenge

- How can we:
  - Implement designs faster?
  - Reuse designs on a variety of hardware?
Solution: C and HDL Code Generation

- Design, execute, and verify algorithms in MATLAB
- Automatically generate C or HDL code
- Deploy generated code on hardware
Code Generation Products for C/C++

- **Embedded Coder™**: Automatically generate C and C++ optimized for embedded systems.

- **Simulink® Coder™**: Automatically generate C and C++ from Simulink models and Stateflow charts.

- **MATLAB® Coder™**: Automatically generate C and C++ from MATLAB code.
Code Generation Products for VHDL/Verilog

**HDL Coder™**
Automatically generate VHDL or Verilog from **MATLAB** code and **Simulink** Model

**MATLAB® Coder™**
Automatically generate C and C++ from **MATLAB** code

**Fixed-Point Toolbox™**
provides fixed-point data types and arithmetic in **MATLAB®**
MATLAB Coder Capabilities
Agenda

- Algorithm Design Process and Design Challenges

- MATLAB Coder - MATLAB to C Flow
  - Why translate MATLAB to C?
  - Challenges of manual translation
  - MATLAB Coder Workflow for generating code

- Accelerating Algorithm Execution

- HDL Coder - MATLAB to HDL Flow
  - Why translate MATLAB to HDL?
  - Challenges of manual translation
  - HDL Coder Workflow for generating code

- Q&A
Why Engineers translate MATLAB to C today?

Integrate MATLAB algorithms w/ existing C environment using source code and static/dynamic libraries

Prototype MATLAB algorithms on desktops as standalone executables

Accelerate user-written MATLAB algorithms

Implement C code on processors or hand-off to software engineers
Challenges with Manual Translation from MATLAB to C

- Separate functional and implementation specification
  - Leads to multiple implementations that are inconsistent
  - Hard to modify requirements during development
  - Difficult to keep reference MATLAB code and C code in-sync

- Manual coding errors

- Time consuming and expensive
Algorithm Design and Code Generation in MATLAB

With MATLAB Coder, design engineers can

• Maintain one design in MATLAB
• Design faster and get to C quickly
• Test more systematically and frequently
• Spend more time improving algorithms in MATLAB
Using MATLAB Coder: 3-Step Workflow

Prepare your MATLAB algorithm for code generation
- Make implementation choices
- Use supported language features

Test if your MATLAB code is compliant
- Validate that MATLAB program generates code
- Accelerate execution of user-written algorithm

Generate source code or MEX for final use
- Iterate your MATLAB code to optimize
- Implement as source, executable or library
Example

c = a*b

- Coder UI  R2012a
- Testbench  R2012a
- Code Generation options
- Generate code
- Browse through report

>> Demo
Demo: Newton/Rhapson Algorithm

Workflow example

Preparing your MATLAB code

- Pre-allocate
- Identify more efficient constructs
- Building options

Test & Generate

\[ x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \]

```matlab
function [x,h] = newtonSearchAlgorithm(b,n,tol)
% Given, "a", this function finds the nth root of a
% number by finding where: x^n-a=0.
coder.inline('never');

notDone = 1;
aNew = 0; %Refined Guess Initialization
a = 1; %Initial Guess
cnt = 0;
h=zeros(50,1);
h(1)=a;

while notDone
    cnt = cnt+1;
    [curVal,slope] = f_and_df(a,b,n); %square
    yint = curVal-slope*a;
    aNew = -yint/slope; %The new guess
    h(cnt)=aNew;
    if (abs(aNew-a) < tol) %Break if it's converged
        notDone = 0;
    elseif cnt>49 %after 50 iterations, stop
        notDone = 0;
    end
    a = aNew;
end
```

>> Demo
Embedded Coder for Optimized Code

Embedded Coder extends MATLAB Coder with:

- Processor-specific code generation
  - Built-in support for select processors
  - Open APIs for use with any processor
- Speed, memory, and code appearance advanced features
MATLAB Coder Key Features

- **Code generation for MATLAB**
  - Portable ANSI/ISO C code
  - Floating-point or Fixed-point (requires Fixed-Point Toolbox)
  - Processors specific optimizations (requires Embedded Coder)

- **Verification**
  - Generates MEX test-benches

- **Design and Build**
  - Automated compile and build for desktop execution
  - Ability to cross-compile and build for execution on other platforms
Agenda

- Algorithm Design Process and Design Challenges

- MATLAB Coder - MATLAB to C Flow
  - Why translate MATLAB to C?
  - Challenges of manual translation
  - MATLAB Coder Workflow for generating code

- Accelerating Algorithm Execution

- HDL Coder - MATLAB to HDL Flow
  - Why translate MATLAB to HDL?
  - Challenges of manual translation
  - HDL Coder Workflow for generating code

- Q&A
Why simulation acceleration?

- From algorithm exploration to system design
  - Size and complexity of models increases
  - Time needed for a single simulation increases
  - Number of test cases increases
  - Test cases become larger

- Need to reduce simulation time during design

- Need to reduce time for large scale testing during prototyping and verification
Accelerating Algorithm Execution

User’s Code

- \texttt{for } \texttt{k=1:max}
- \texttt{x = \texttt{fft(dat)}}
- \texttt{y = 20*\texttt{log1}}

Optimize MATLAB Code

Parallel Computing

System Objects

Custom Code using MEX

MATLAB to C

>> Demo
commacceleration
HDL Coder Capabilities
Agenda

- Algorithm Design Process and Design Challenges
  - MATLAB Coder - MATLAB to C Flow
    - Why translate MATLAB to C?
    - Challenges of manual translation
    - MATLAB Coder Workflow for generating code
  - Accelerating Algorithm Execution
    - HDL Coder - MATLAB to HDL Flow
      - Why translate MATLAB to HDL?
      - Challenges of manual translation
      - HDL Coder Workflow for generating code
- Q&A
HDL Coder
Generate VHDL and Verilog Code for FPGA and ASIC designs

MATLAB Simulink HDL Coder

Verilog and VHDL

New: MATLAB to HDL

- Automatic floating-point to fixed-point conversion
- HDL resource optimizations and reports
- Algorithm-to-HDL traceability
- Integration with simulation & synthesis tools
Which benefits Engineers get from MATLAB to HDL?

**Automate for implementing HDL** without direct RTL hand-coding

**Less LOC** by using MATLAB TestBench and **Reusing** it through Entire Design Process

**Easy conversion** from Floating-Point to Fixed-Point

**Fast to simulate** compare with using HDL DUT/TB in the HDL simulator environment
Using HDL Coder: 5-Step Workflow

**Prepare** your MATLAB algorithm for code generation
- Use supported language features
- Make implementation choices

**Fixed-Point** MATLAB code generation from your floating-point design using your MATLAB TestBench
- Accelerate TestBench for fast simulation
- Automatically propose Fixed-Point type
- Iterate data-type customization to optimize
- Verify Fixed-Point code against original Floating-Point code.

**Generate** synthesizable RTL & TestBench code from Fixed-Point MATLAB code for final use
- Iterate your MATLAB code to optimize
- Implement as source, executable or library

**Simulate** the generated HDL code with test vectors from the test bench using the specified simulation tool

**Synthesis, Place and Route** the generated RTL code by creating project with ISE/Quartus II
- Check timing analysis report to optimize
Example: Symmetric FIR Filter

```matlab
function [y_out, delayed_xout] = mlhdlc_sfir(x_in,
% Symmetric FIR Filter

% declare and initialize the delay registers
persistent ud1 ud2 ud3 ud4 ud5 ud6 ud7 ud8;
if isempty(ud1)
    ud1 = 0; ud2 = 0; ud3 = 0; ud4 = 0; ud5 = 0;
end

% access the previous value of states/registers
a1 = ud1 + ud8; a2 = ud2 + ud7;
a3 = ud3 + ud6; a4 = ud4 + ud5;

% multiplier chain
m1 = h_in1 * a1; m2 = h_in2 * a2;
m3 = h_in3 * a3; m4 = h_in4 * a4;

always @(posedge clk or posedge reset)
    begin : ud2_reg_process
        if (reset == 1'b1) begin
            ud2_1 <= 0;
            end
        else begin
            if (enb) begin
                ud2_1 <= ud2;
            end
            end
    end

assign tmp_4 = ud2_1;
```

>> Demo
Workflow Advisor

- Automatically convert floating point to fixed-point
- Automatic HDL Verification
- Integration with FPGA implementation workflows
- Automatic HDL code generation with built-in optimizations
- Generates synthesizable HDL code for your fixed-point MATLAB code.

### HDL Code Generation

- MATLAB HDL Coder Workflow
  - Float-to-Fixed Workflow
  - Verify Floating-Point Design
  - Propose Fixed-Point Types
  - Generate Fixed-Point Code
  - Verify Fixed-Point Design
- MATLAB to HDL Workflow
  - Code Generation
  - Simulation and Verification
  - Synthesis and Analysis
  - Create Project
  - Run Logic Synthesis
  - Run Place and Route

### Automatic HDL Code Generation

- Target:
  - Code Generation Style
  - Clocks & Ports
  - Test Bench
  - Optimizations
- RAM Mapping:
  - Map persistent array variables to RAMs
  - RAM mapping threshold:
- Pipelining:
  - Input pipelining:
  - Output pipelining:
  - Distribute pipeline registers
- Area Optimizations:
  - Resource sharing factor:
  - Constant multiplier optimization:
  - CSD
  - None
- Loop Optimizations:
  - None
  - CSD
  - PCD
  - Auto

### Code Generation

```plaintext
### Begin VHDL Code Generation
### Working on sobel_edge_detection_fixPt/u_d_ram/dualPortRAM_128x8b as
codegen\sobel_edge_detection\hlslib\dualPortRAM_128x8b.v vhdl
### Working on sobel_edge_detection_fixPt/u_d_ram as
codegen\sobel_edge_detection\hlslib\u_d_ram.v vhdl
### Working on sobel_edge_detection_fixPt/u_d_ram/dualPortRAM_128x8b as
codegen\sobel_edge_detection\hlslib\dualPortRAM_128x8b_block.v vhdl
### Working on sobel_edge_detection_fixPt/u_d_ram as
```
HDL Coder Key Features

- Code Generation for MATLAB and Simulink
  - Target-independent RTL level HDL Code
  - IEEE 1376 compliant VHDL
  - IEEE 1364-2001 compliant Verilog
  - Automate Fixed-Point conversion

- Verification
  - Generate HDL test-bench
  - Generate Scripts for Synthesis tool & HDL simulator

- Design automation
  - Synthesize and P&R using integrated Xilinx and Altera synthesis tool interface
  - Optimize for area or speed
Model-Based Design flow using Simulink
from Algorithm to FPGA Implementation

MATLAB® and Simulink®
Algorithm and System Design

HDL Coder
RTL Creation

HDL Verifier
HDL Co-Simulation

RTL

Back Annotation

Implement Design
Synthesis
Map
Place & Route

Verification
Functional Simulation
Static Timing Analysis
Timing Simulation

HDL Verifier
FPGA in the Loop

FPGA in the Loop

MATLAB®
Simulink®
Stateflow
Customer Successes
Ono Sokki Reduces Development Time for Speed Measurement Device Using MATLAB Generated Code

Challenge
Develop a high-precision speedometer for prototype vehicles within a tight schedule

Solution
Use MathWorks tools for Model-Based Design to develop algorithms, generate production code for an embedded processor, and perform processor-in-the-loop verification

Results
- Development time cut significantly
- Code base reduced dramatically
- System model reused as a test bench

“With MathWorks tools we have a seamless environment for development, simulation, code generation, and processor-in-the-loop verification. The advantages of Model-Based Design over hand-coding in C can’t be overestimated.”

Kazuhiro Ichikawa
Ono Sokki

LC-8100 GPS Speedometer System.
FLIR Accelerates Development of Thermal Imaging FPGA

Challenge
Accelerate the implementation of advanced thermal imaging filters and algorithms on FPGA hardware

Solution
Use MATLAB to develop, simulate, and evaluate algorithms, and use HDL Coder to implement the best algorithms on FPGAs

Results
- Time from concept to field-testable prototype reduced by 60%
- Enhancements completed in hours, not weeks
- Code reuse increased from zero to 30%

“With MATLAB and HDL Coder we are much more responsive to marketplace needs. We now embrace change, because we can take a new idea to a real-time-capable hardware prototype in just a few weeks. There is more joy in engineering, so we’ve increased job satisfaction as well as customer satisfaction.”

Nicholas Hogasten
FLIR Systems
## Public Trainings in the next Few Months

<table>
<thead>
<tr>
<th>Course</th>
<th>Dates</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>20 Aug 2012 – 21 Aug 2012</td>
<td>Bangalore</td>
</tr>
<tr>
<td>Embedded Coder for Production Code Generation</td>
<td>22 Aug 2012 – 24 Aug 2012</td>
<td>Bangalore</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>03 Sep 2012 – 05 Sep 2012</td>
<td>Bangalore</td>
</tr>
<tr>
<td>MATLAB Programming Techniques</td>
<td>06 Sep 2012 – 07 Sep 2012</td>
<td>Bangalore</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>24 Sep 2012 – 26 Sep 2012</td>
<td>Pune</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>27 Sep 2012 – 28 Sep 2012</td>
<td>Pune</td>
</tr>
<tr>
<td>MATLAB Based Optimization Techniques</td>
<td>17 Oct 2012</td>
<td>Bangalore</td>
</tr>
</tbody>
</table>

Email: [training@mathworks.in](mailto:training@mathworks.in)  
URL: [http://www.mathworks.in/services/training](http://www.mathworks.in/services/training)  
Phone: 080-6632-6000
Summary

- Design using MATLAB
- Generate C and HDL
- Deploy code to Processors and Hardware
- Accelerate simulation speed
- Easy conversion from floating-point to fixed-point
- Iterate your designs faster
Thank You