Implementing MATLAB Algorithms in FPGAs and ASICs

By Alexander Schreiber
Senior Application Engineer
MathWorks
Traditional Implementation Workflow: Challenges

- Long development cycles
- Prevents short iteration cycles
- Difficult to optimize the algorithm at a system level

**Algorithm Development**
- MATLAB
- Simulink
- Stateflow

**Design**
- Manual Fixed Point Conversion
- Text I/O or HW based FPGA Verification
- Manual HDL Code Creation
- Manual HDL Refinement
- Text I/O based HDL Verification
- Text I/O based HDL Verification
Solution: Model-Based Design using MATLAB

- Design, simulate, and validate algorithms and system models in MATLAB and Simulink

- Automatically generate and optimize HDL code

- Verify the HDL/hardware implementation against the system model
From Algorithm to Implementation
From Algorithm to Implementation

MATLAB® Algorithm and System Design
Model Refinement for Hardware

Conversion to Fixed Point and Fixed-Point Verification

HDL Code Generation

HDL Simulation

Implement Design

Synthesis
Map
Place & Route
MATLAB to HDL: The Big Challenges

MATLAB ➔ HDL

Floating point ➔ Fixed-Point
Procedural ➔ Concurrent + optimized
Matrices ➔ Block RAMs
Untimed ➔ Timed with rates
Loops ➔ Streaming, Unrolling
Functions ➔ Hardware-efficient implementations
System objects

Algorithm land

Architecture land
Authoring MATLAB for Hardware

- Leveraging strength of MATLAB
  - Easily creating, accessing, modifying and manipulating vectors and matrices
Authoring MATLAB for Hardware

- Leveraging strength of MATLAB
  - Easily creating, accessing, modifying and manipulating vectors and matrices
- Modeling persistence
  - RAM, ROM, registers, tap delays
  - Finite State Machines

```matlab
persistent tap_delay;

% Clear tap delay line at beginning
if isempty(tap_delay)
    tap_delay = zeros(1,length(coeff));
end
```

```matlab
persistent tap_delay;

% Clear tap delay line at beginning
if isempty(tap_delay)
    Tap_Delay = zeros(1,length(coeff));
end

% Perform sum of products
outdbuf = tap_delay * coeff(end:-1:1);

% Shift tap delay line
tap_delay = [tap_delay(2:length(coeff)) indbuf];
```
Authoring MATLAB for Hardware

- Leveraging strength of MATLAB
  - Easily creating, accessing, modifying and manipulating vectors and matrices

- Modeling persistence
  - RAM, ROM, registers, tap delays
  - Finite State Machines

- Using System Objects
  - Modular and reusable
  - Implicit state handling
  - Library of predefined System Objects
  - Support of User-Defined System Objects

```matlab
% Definition and initialization of System Object
persistent hFIR;
if isempty(hFIR)
    hFIR = dsp.FIRFilter('Numerator',coeff);
end

% Applying System Object to input data
outdatabuf = step(hFIR,indatabuf);
```
From Algorithm to Implementation

MATLAB® Algorithm and System Design
Model Refinement for Hardware

Conversion to Fixed Point
and Fixed-Point Verification

HDL Code Generation

HDL Simulation

Implement Design
- Synthesis
- Map
- Place & Route
Fixed-Point Conversion and Verification

- Dynamic Range Analysis
  - Based on stimuli or derived ranges
Fixed-Point Conversion and Verification

- **Dynamic Range Analysis**
  - Based on stimuli or derived ranges
- **Autoscaling of Fixed-Point Data Type**
  - Either word length or precision based
  - Selectable behaviour
    - Integer: saturation, wrapping
    - Fraction: rounding method
Fixed-Point Conversion and Verification

- Dynamic Range Analysis
  - Based on stimuli or derived ranges

- Autoscaling of Fixed-Point Data Type
  - Either word length or precision based
  - Selectable behaviour
    - Integer: saturation, wrapping
    - Fraction: rounding method

- Automatic Generation of Bit-true Fixed-Point MATLAB Function
Fixed-Point Conversion and Verification

- **Dynamic Range Analysis**
  - Based on stimuli or derived ranges

- **Autoscaling of Fixed-Point Data Type**
  - Either word length or precision based
  - Selectable behaviour
    - Integer: saturation, wrapping
    - Fraction: rounding method

- **Automatic Generation of Bit-true Fixed-Point MATLAB Function**

- **Verification of Fixed-Point Scaling**
  - Comparison vs. floating-point result
  - Iteration in Workflow Advisor if needed
From Algorithm to Implementation

MATLAB® Algorithm and System Design
Model Refinement for Hardware

Conversion to Fixed Point
and Fixed-Point Verification

HDL Code Generation

HDL Simulation

Implement Design

Synthesis
Map
Place & Route
Automatic HDL Code Generation

- Based on native MATLAB
  - Generic, i.e. technology independent algorithm description
  - Leveraging MATLAB strengths, e.g. vector and matrix arithmetic
- Generic RTL HDL
  - FPGA vendor independent
  - VHDL or Verilog
- Bit-true and Cycle-accurate wrt. to Fixed-Point MATLAB Function
Automatic HDL Code Generation

- Based on native MATLAB
  - Generic, i.e. technology independent algorithm description
  - Leveraging MATLAB strengths, e.g. vector and matrix arithmetic

- Generic RTL HDL
  - FPGA vendor independent
  - VHDL or Verilog

- Bit-true and Cycle-accurate wrt. to Fixed-Point MATLAB Function

- Traceability between HDL Code and MATLAB Fixed-Point Function
Optimization: Speed

- Automatic pipelining
- Helps you meet speed objectives

Pipelining:
- Register inputs
- Register outputs
- Distribute pipeline registers

Input pipelining: 0
Output pipelining: 0

Smaller critical path
Optimization: Area
From Algorithm to Implementation

MATLAB® Algorithm and System Design
Model Refinement for Hardware

Conversion to Fixed Point
and Fixed-Point Verification

HDL Code Generation

HDL Simulation

Implement Design

Synthesis
Map
Place & Route
HDL Verification
HDL Coder

- Automatic Recording of input and output streams of MATLAB simulation
- Automatic Generation of stand-alone HDL Test Bench
  - Generic, i.e. pure HDL
  - Can be simulated in any HDL simulator
  - Self-testing, i.e. uses recorded input and output streams
- Drawback
  - MATLAB visualization and analysis capabilities cannot be leveraged
HDL Co-Simulation

- Re-use MATLAB test bench
- Leverage visualization capabilities in time and spectrum domain
- Combined analysis in HDL Simulator and MATLAB/Simulink
FPGA-in-the-Loop Verification
HDL Verifier

- Re-use the MATLAB test bench
- Leverage visualization capabilities in time and spectrum domain
- Accelerate verification with FPGA hardware

MATLAB Functions

HDL Verifier connects FPGA HW with the MATLAB environment!
From Algorithm to Implementation

MATLAB® Algorithm and System Design
Model Refinement for Hardware

Conversion to Fixed Point
and Fixed-Point Verification

HDL Code Generation

HDL Simulation

Implement Design

Synthesis
Map
Place & Route
Automated Workflow for FPGA Implementation

- Integration with Xilinx ISE and Altera Quartus II
  - Project creation
  - Synthesis
  - Place and Route
  - Reporting
    - Resource utilization
    - Timing analysis
- Rapid exploration of implementation options through quick iteration
Model-Based Design for FPGA Implementation

- Shorter development cycles
- Short iteration cycles
- Optimization of algorithm at a system level

automated/assisted Fixed Point Conversion

integrated FPGA Verification

System-level HDL Refinement

integrated HDL Verification

automatic HDL Code Generation

Algorithm Development
MATLAB Simulink Stateflow

DESIGN
Solution: Model-Based Design using MATLAB

- Design, simulate, and validate algorithms and system models in MATLAB and Simulink
- Automatically generate and optimize HDL code
- Verify the HDL/hardware implementation against the system model
Next Steps …

1. Visit our web site:
   www.mathworks.de/fpga-design/
   www.mathworks.de/products/hdl-coder
   www.mathworks.de/products/hdl-verifier

2. Contact us for more information:
   contact@MathWorks.de
   Your local Sales Representative

Questions?