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
- Manual HDL Code Creation
- Text I/O based HDL Verification
- Manual HDL Refinement
- Text I/O based HDL Verification

**HDL Code Creation**

- Text I/O or HW based FPGA 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

- 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

Refinement and Design Exploration

Iterative Refinement using Fixed-Point Advisor
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

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

Algorithm land ➔ Architecture land
Authoring MATLAB for Hardware

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

```matlab
% Perform sum of products
outdatabuf = tap_delay * coeff(end:-1:1)';

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

- Leveraging strenght 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);
```

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

% Perform cum of products
outdatabuf = tap_delay * coeff(1:end-1);

% Shift tap delay line
tap_delay = [tap_delay(2:length(coeff)) 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

### Variable | Function Replacements | Type Validation Output
---|---|---
Input | | |
\text{indBuf} | double | -5.27 | 5.35

### Output
\text{outBuf} | double | -4.36 | 4.04

### Persistent
\text{tapDelay} | 1x16 double | -5.07 | 5.35

### Local
\text{coeff} | 1x16 double | 0 | 0.14
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
Optimization: Area

- **RAM Mapping**
  - Map persistent array variables to RAMs
  - RAM mapping threshold: 256

- **Pipelining**
  - Register inputs
  - Register outputs
  - Distribute pipeline registers
  - Input pipelining: 0
  - Output pipelining: 0

- **Area Optimizations**
  - Resource sharing factor: 0
  - Constant multiplier optimization: None

- **Loop Optimizations**
  - None
  - Unroll loops
  - Stream loops
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

HDL Verifier connects FPGA hardware 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

DESIGN

Algorithm Development
MATLAB Simulink Stateflow

- automatic HDL Code Generation
- integrated HDL Verification
- System-level HDL Refinement
- automated/assisted Fixed Point Conversion
- integrated FPGA 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
Next Steps …

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

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

Questions?