Rapid Prototyping Using HDL Coder
Who Are We?

Esa-Matti Turtinen
R&D Manager, SoC Prototyping, Nokia Oulu
- M.Sc., Electrical Engineering
- 31 years old
- About 6 years of experience working on different roles related to SoC development

Joonas Järviluoma
Prototype Engineer, SoC, Nokia Oulu
- M.Sc., Electrical Engineering
- 26 years old
- Just graduated
- Currently working on FPGA lab testing
Expanding the human possibilities of the connected world
Nokia has been at the forefront of every fundamental change in how we communicate and connect.

Telephony begins

Bell Telephone Laboratories formed in 1925

Analog revolution

Long distance voice communication
- Copper networks
- Circuit switches
- Amplifiers

Digital revolution

Voice, data, and video communication
- Laser
- Satellite communications
- UNIX
- DWDM
- 100Gbps optical transport
- 400G routers

Mobile revolution

Wireless communication
- First ever calls on GSM and LTE
- First car phone
- Commercialization of Small Cells
- MIMO

The new connectivity

Intelligent and seamless connectivity through the Cloud
- 5G
- G.Fast: 1Gbps over copper
- Optical super channels
- Terabit IP routing
- Datacenter infrastructure and applications for the Cloud
- Smart sensors for the Internet of Things
A financially strong leader

<table>
<thead>
<tr>
<th></th>
<th>Revenue*</th>
<th>R&amp;D spend*</th>
<th>Net cash*</th>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>€26.6bn</td>
<td>€4.5bn</td>
<td>€10.0bn</td>
<td>106,000</td>
</tr>
</tbody>
</table>

* Combined Nokia and Alcatel-Lucent 2015 numbers according to Nokia accounting policies, non-IFRS
Would it be possible to left-shift this and trial algorithms in HW earlier?
FPGA Prototyping Flow Timeline
Proportional Estimation in Generic HLS flow
HDL Coder Flow
From Algorithm to FPGA Programmable Model

HLS

Logic Synthesis

Co-simulation

FPGA-in-the-loop
Example Design for HDL Coder Flow
Scaling and Power Limitation Block

- Arithmetic logic (multipliers, adders etc.)
- Loop structures
- State-Machine
- Look-up tables for dB conversions
- Registers for state control and buffering
- Variable indexing
- Configurable parameters
Classic Division of Models
Algorithm and RTL

Algorithm Model

RTL Model

• Hand-written based on algorithm model
• ASIC optimized performance
• Thorough verification required

MATLAB operations optimized for maximized simulation performance

Object-Oriented Programming

Class

Object

Person

Anna

Properties

Methods

Properties

Methods

Eye Color

Speak

Eye Color: Blue

Speak

Height

Walk

Height: 165cm

Nationality

Walk

Weight: 55kg

Nationality: Swedish

Sleep

Sleep

Properties

Methods
Division in HDL Coder Workflow
Algorithm and RTL

Algorithm Model:
• Written in MATLAB function blocks/System Objects and Simulink library components
• Has to be written from HW perspective to generate feasible RTL

RTL Model:
• Rapid generation from Simulink (or MATLAB) model
• Verification focus moves towards algorithm
• Cosimulation verificates RTL against algorithm model
• ”Is as good as the algorithm”
RTL Generation

Example 1: Algorithm without Data Type Definition

```plaintext
if (run)
    mk_tmp = Gk;
% Multiplied branches, format: 1.015 * 0.414 -> 1.429 (output format is set automatically by matlab)
    mul_I = data_I*mk_tmp;
    mul_Q = data_Q*mk_tmp;
```

```plaintext
259 IF run = '1' THEN
260   -- <$S37>1:44'
261   -- Multiplied branches, format: 1.015 * 0.414 -> 1.429 (output format is set automatically by matlab)
262   -- <$S37>1:44'
263   mul_temp := data_I_signed * signed(resize(Gk_unsigned, 19));
264   IF (mul_temp(34) = '0') AND (mul_temp(33) /= '0') THEN
265       mul_I := "011111111111111111111111111111111";
266       ELSIF (mul_temp(34) = '1') AND (mul_temp(33) /= '1') THEN
267       mul_I := "1000000000000000000000000000000000";
268       ELSE
269       mul_I := mul_temp(33 DOWNTO 0);
270       END IF;
271   -- <$S37>1:46'
272   IF (mul_temp_0(34) = '0') AND (mul_temp_0(33) /= '0') THEN
273       mul_Q := "011111111111111111111111111111111";
274       ELSIF (mul_temp_0(34) = '1') AND (mul_temp_0(33) /= '1') THEN
275       mul_Q := "1000000000000000000000000000000000";
276       ELSE
277       mul_Q := mul_temp_0(33 DOWNTO 0);
278       END IF;
```
RTL Generation
Example 2: Algorithm with Data Type Definition

```plaintext
if (run)
    mk_tmp = fi(Gk, 1, 19, 14);
    % Multiplied branches, format: 1.0.15 * 0.4.14 -> 1.4.29 (output format is set automatically by matlab)
    mul_I = data_I*mk_tmp;
    mul_Q = data_Q*mk_tmp;
```

```plaintext
IF run = '1' THEN
  '<<S37>:1:41'
  '<<S37>:1:42'
  mk_tmp := signed(resize(Gk_unsigned, 19));
  % Multiplied branches, format: 1.0.15 * 0.4.14 -> 1.4.29 (output format is set automatically by matlab)
  '<<S37>:1:44'
  mul_I := data_I_signed * mk_tmp;
  '<<S37>:1:45'
  mul_Q := data_Q_signed * mk_tmp;
```

Two multipliers
RTL Resource Utilization Comparison
FPGA Prototype Vs. Original ASIC Targeted Model

• Original hand-written model, targeted for ASIC, had slightly more signals and routing logic compared to generated model!

• Generated model tested successfully in FPGA-in-the-loop configuration
ASIC Optimization
Area and Timing Results

Further timing optimization could have been performed (Work focused on FPGA prototyping)
FPGA Prototyping Flow Timeline
Proportional Estimation in HDL Coder Flow

FPGA Prototyping with HLS Flow
- HDL code (RTL)
- Verification
- Logic Synthesis
- FPGA Verification
- ASIC
- Software development
- Time save in SW development
- SoC ready

FPGA Prototyping with MathWorks HLS Flow
- HDL code (RTL)
- Verification
- Logic Synthesis
- FPGA Verification
- ASIC
- Software development
- Time save in SW development
- Time save in verification
- SoC ready

Time
Conclusion
Benefits and Shortages

Benefits:

• Human readable HDL output
• Design work and verification focus moves on higher level
• Good synthesis results in both FPGA and ASIC cases
• Distinct GUI
• Support for 3rd party tools and FPGA boards

Shortages:

• For feasible HDL generation and FPGA prototyping, algorithms have to be written strictly from HW perspective
• No trivial way to generate generic variables to create scalable IPs (due to Model-Based Design flow)
Future Work

Algorithm design work change towards RTL design style required

- Close co-operation with algorithm and RTL designers is vital
- Algorithm simulation speed might be critical

IP generation with generic interfaces

- Was left out of scope in this study
- Needs to be verified

Projects ongoing
Q & A