Routes to Production Code Generation

Mark Walker
MathWorks
Aims

- Overview of production code generation in R2013b
- Highlight features for modern microprocessor architectures
  - Hardware / software co-design
  - Multi-processor systems
What is involved – prototype to production

- Working prototype
  - Right answer
  - Right closed loop performance

- What’s missing?
  - Evidence of standards compliance
  - Robustness, reliability, reuse
  - Runs on production hardware
  - Interface to support functions, e.g. diagnostics, reprogramming
Standards Compliance

- Support for a wide range of standards:
  - MISRA, MAAB, IEC 61508, ISO 26262, EN 50128, DO-178, DO-254, ...
  - Up-to-date in every release, including R2013b
  - Supported by automated checks where possible
Mapping Model-Based Design to Standards

SIL 4

DAL A
Support Package Downloads

![](Support Package Installer.png)
Embedded Systems Architectures, 1990s

- Dedicated ASICs for high speed processing
- Interconnected on PCB
- Microprocessor
Embedded Systems Architectures, Today

Single IC: 2 ARM cores, FPGA fabric
Embedded Systems Architectures: Xilinx Zynq

Zynq-7000 All Programmable SoC

Two ARM A9 processors

Fast interconnect with I/O and each other (AXI bus)

FPGA fabric

Target the Whole System with Model-Based Design

Processor-in-the-Loop (PIL)

HDL Coder

Concurrent Execution

External Mode
Offload Processing to Hardware

- Key question: will my design fit on the processor?
- `<sobel_static_metrics.slx>`

2. Global Variables [hide]

Global variables defined in the generated code.

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>filter_static_metrics_B</code></td>
<td>2,073,600</td>
</tr>
<tr>
<td><code>filter_static_metrics_U</code></td>
<td>259,200</td>
</tr>
<tr>
<td><code>filter_static_metrics_Y</code></td>
<td>86,400</td>
</tr>
<tr>
<td><code>filter_static_metrics_M_</code></td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2,419,204</strong></td>
</tr>
</tbody>
</table>
Execution Profiling and Unit Testing with PIL

- Key question: is it fast enough?
- Run your candidate design on hardware = unit test
  - Measure execution time
  - Check for correct answer
  - Measure coverage
- `<sobel_pil.slx>`
Target the FPGA: HDL Coder

Implementation Model

Generated HDL

HDL Workflow Advisor

Generated interface model
Behavioural and Implementation Models

Behavioural

```matlab
% Sobel edge detection. Given a normalized image (with double values).
% return an image where the edges are detected w.r.t. threshold value.

function edgeImage = sobel(im, threshold) %foodegen

% Convert to grayscale
originalImage = (0.2989 * double(im(:,:,1)) + 0.5870 * double(im(:,:,2)) - 0.1140 * double(im(:,:,3)));

assert(all(size(originalImage) <= [1024 1024]));
assert(isa(originalImage, 'double'));
assert(isa(threshold, 'double'));

k = [1 2 1; 0 0 0; -1 -2 -1];
I = conv2(double(originalImage), k, 'same');
V = conv2(double(originalImage), k', 'same');
E = sqrt(I.*I + V.*V);
edgeImage = uint8((E > threshold) * 255);
```

Implementation
Hardware Software Co-Design

HDMI input → Convert to AXI → Sobel Edge → Convert From AXI → HDMI output

AXI Lite

AXI Stream
Hardware Software Co-Design

- **AXI4-Lite Write**
- **Processor**
- **HDMI input**
- **Convert to AXI**
- **Sobel Edge**
- **Convert From AXI**
- **HDMI output**
- **Embedded Coder**
- **HDL Coder**

Diagram with labeled components and flow.
Interface with External Mode

- `<gm_hdlcoder_sobel_video_interface.slx>`
Multi-Tasking in Simulink

- Multi-tasking code generation has been available for a long time
Multi-Tasking in Simulink

- Multi-tasking execution is still sequential
- Parallel threads must not:
  - Share internal data
  - Execute with equal priority (can pre-empt though)
  - Share data immediately at their interfaces
- Now, concurrent execution now allows you to target parallel threads
Concurrent Execution
Concurrent Execution

- Model elements can run in parallel threads
- Immediate, blocking and delayed data transfers
- VxWorks, Linux (+ Embedded) and Windows
- Open API to extend to any parallel architecture

<concurrent_exec.slx>
 Concurrent Execution
Summary

- We have seen:
  - Standards compliance in R2013b
  - Production targets available in R2013b
  - Automating unit testing and profiling with PIL
  - Modern architecture support in R2013b targets
    - Hardware / software co-design
    - Targeting multi-processor systems

- Questions?