Analyzing Code Generated from a Simulink Model

Jim Freudenberg
Embedded Control Systems
ETH Zurich

Matlab 2011b
Introduction

- This set of slides will show you how generate C code from a simple Simulink model, and analyze the generated code using Embedded Coder utilities.

- In particular, we will use the **traceability** feature to move back and forth from the Simulink blocks to the code generated from them.

- We will use 32-bit Matlab release 2011b, the same version of Matlab as currently used in the Embedded Control Lab at ETH.
Create a Simulink model of the system depicted in Figure 3 of the handout “Simulink Models for Autocode Generation”. This model takes one 32-bit integer as input and breaks it into four 8-bit integers.
Make a Subsystem

From the Edit menu:
Edit/Select All
Edit/Create Subsystem
Input Data Type

- Double-click Input Port
- Set data type to \texttt{uint32} in the Signal Attributes Menu
Display Data Types

- Under the Format Menu, select

Format/Port/Signal Displays/Port Data Types
Format/Port/Signal Displays/Signal Dimensions
Format/Port/Signal Displays/Wide Nonscalar Lines
In the Simulation/Configuration Parameters/Solver Menu, select “Fixed Step” solver and “discrete (no continuous states)”
Options for Code Generation

From the Tools Menu, select:
Tools/Code Generation/Options
Options Window

The default target option is “Generic Real-Time Target”, grt.tlc
For more efficient code and additional features, Browse to select “Embedded Coder “, ert.tlc
Embedded Coder Options

- Choose “Set Objectives” from the Code Generation Advisor
- Select “Traceability” from the list of Available objectives
Select Code Generation/Report and check all available options.
Generate C Code

- To generate C code, we must “Build” the model
- Under the Tools Menu, select Tools/Code Generation/Build Model
Code Generation Report

• If the Build is successful, a Code Generation Report window will open.

• Note the many menus - we will take a look at some of these.
Code Interface Report

Describes the generated functions, their arguments, and how often they are executed

Q: why does the function `thirtytwo_bit_foureightbits_step` execute every 0.2 seconds?
## Code Interface Report: Parameters

- **Input and Output Ports**
- **Numerical Parameters and the Simulink blocks in which they are found**
  - 4 bit masks
  - 4 bit shifts

<table>
<thead>
<tr>
<th>Inports</th>
<th>Block Name</th>
<th>Code Identifier</th>
<th>Data Type</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;Root&gt;/In1</td>
<td>thirtytwobitto8bit_U.In1</td>
<td>uint32_T</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outports</th>
<th>Block Name</th>
<th>Code Identifier</th>
<th>Data Type</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;Root&gt;/Out1</td>
<td>thirtytwobitto8bit_Y.Out1</td>
<td>uint8_T</td>
<td>[4]</td>
</tr>
</tbody>
</table>

### Interface Parameters

<table>
<thead>
<tr>
<th>Parameter Source</th>
<th>Code Identifier</th>
<th>Data Type</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;S1&gt;/Least significant 8 bits</td>
<td>thirtytwobitto8bit_P.Leastsignificant8bits_BitMask</td>
<td>uint32_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Bitwise Operator1</td>
<td>thirtytwobitto8bit_P.BitwiseOperator1_BitMask</td>
<td>uint32_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Bitwise Operator2</td>
<td>thirtytwobitto8bit_P.BitwiseOperator2_BitMask</td>
<td>uint32_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Most significant 8 bits</td>
<td>thirtytwobitto8bit_P.Mostsignificant8bits_BitMask</td>
<td>uint32_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Shift Arithmetic</td>
<td>thirtytwobitto8bit_P.ShiftArithmetic_Power2Exponent</td>
<td>int16_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Shift Right 8 Bits</td>
<td>thirtytwobitto8bit_P.ShiftRight8Bits_Power2Exponent</td>
<td>int16_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Shift Right 16 Bits</td>
<td>thirtytwobitto8bit_P.ShiftRight16Bits_Power2Exponent</td>
<td>int16_T</td>
<td>1</td>
</tr>
<tr>
<td>&lt;S1&gt;/Shift Right 24 Bits</td>
<td>thirtytwobitto8bit_P.ShiftRight24Bits_Power2Exponent</td>
<td>int16_T</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Simulink block where parameter appears**
- **Name of Parameter**
- **Data type**
Static Code Metrics Report

- Number of lines of code in the generated `.c` and `.h` files
- Total lines of code, including comments
Traceability Report: Eliminated Blocks

Certain Simulink blocks are used only to improve model readability, and do not affect the generated code. These are eliminated (in this case, the subsystem block, Mux block, and redundant input and output port block(s))

<Root>: Top Level Simulink diagram
<S1>: Subsystem that contains the rest of the Simulink blocks
Traceability Report: Blocks to Code

<table>
<thead>
<tr>
<th>Simulink block name</th>
<th>Lines of code in associated .c and .h files</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;S1&gt;</code>/Shift Arithmetic</td>
<td>thirtytwobittofourbit.c</td>
</tr>
</tbody>
</table>

Q: why are there no lines of code for the block `<S1>`/Shift Arithmetic in the file thirtytwobittofourbit.c?
Traceability

Click on block name $< S1>/Shift Right 16 Bits$
Corresponding Simulink block is highlighted
Traceability: `.c` File

- Click on `.c` filename name next to Simulink block name
- Location of block for that block is highlighted
- Can also trace `.h` and `data.c` files
Function Definitions

- Function definitions are found in `thirtytwobitto8bit.c`
- The function `thirtytwobitto8bit_step` performs the desired bit manipulations
- C code for bit masking and shifting
- Simulink blocks implemented in C code
- Note the block `<S1>/Shift Arithmetic` is not on the list
Data Definitions

Numerical values of the masking and bitshifting parameters in the Simulink blocks are found in the data file `thirtytwobitto8bit_data.c`.

Q: what are the hex equivalents of the decimal numbers 255, 65280, 16711680, and 4278190080? 
Note: sign convention is opposite that in Simulink block (right shifts in Simulink block are positive)