Per Knopfdruck vom Modell zum Code mit automatischer Generierung von Seriencode

Model to Code via Push-Button Production Code Generation

Michael Fröstl
Pilot Engineering, Production Code Generation
MathWorks

http://www.mathworks.de/embedded-code-generation/
switch(braindump)
{

case ‘Applications’:
Model GNC Algorithms, Perform Real-Time Simulations, and Generate Production Flight Code
OHB AG

Developing an Algorithm and Making It Fly with MATLAB and Simulink
Technische Universität München

Pendolino tilting train.

The MEDUMAT Transport ventilator. Image © Weinmann Medical Technology.

http://www.mathworks.com/company/user_stories
case ‘Programming’:
MATLAB

ASCII

C

C++

ASM

IEC 61131-3
case 'Hardware':
Analog Devices®

ARM®

Atmel®

Freescale™

Infineon®

Intel®

Microchip®

NXP™

Xilinx®

Texas Instruments™

Renesas®

STMicroelectronics®

MCU / DSP
case ‘Operating Systems’:
Green Hills® Integrity RTOS

Texas Instruments™ DSP/BIOS™

OSEK-OS

Embedded Linux®

QNX® Neutrino® RTOS

Android™

Wind River® VxWorks®

Microsoft® Windows Embedded
case 'Standards':
default:
printf(“Need a break?”);
}
MBD_Overview();
INTEGRATION IMPLEMENTATION

- Environment Models
- Physical Components
- Algorithms

IMPLEMENTATION

- C, C++
- VHDL, Verilog
- Structured Text
- MCU
- DSP
- FPGA
- ASIC
- PLC

RESEARCH REQUIREMENTS

DESIGN

TEST & VERIFICATION

TEST SYSTEM

MATLAB EXPO 2013
INTEGRATION

IMPLEMENTATION

C, C++
VHDL, Verilog
Structured Text
MCU
DSP
FPGA
ASIC
PLC

TEST & VERIFICATION

RESEARCH

REQUIREMENTS

DESIGN

Environment Models
Mechanical
Electrical
Supervisory Logic
Control Algorithms

TEST SYSTEM
switch(topics)
{

case ‘Fixed-Point’:
void diffEq( void )
{
    /* Implements a fixed point 
       first order difference equation */

    int Prod;
    long Accum;
    static short lastVal=0;
    short a=0x7eb8; /* 0.99 in s16,15 */
    short oneminusa=0x0148; /* .01 in s16,15 */
    short temp;

    Prod = gAlg_in1 * gAlg_in1;
    temp = Prod >> 15;
    Accum = a*lastVal + oneminusa*temp;

    gAlg_out1 = (short)(Accum >> 15);
    lastVal = gAlg_out1;
}

**Convert variables to integer types**

**Need lots of comments to understand code**

**Keep track of binary point location**

**No saturation or rounding**
Proof of Concept

- Design and simulate floating-point algorithms
- Iterate on algorithm trade-offs

Model Hardware Constraints

- Convert algorithm to fixed-point and simulate
- Iterate on implementation trade-offs

Verifying Fixed-Point Algorithms

- Verify fixed-point results against floating-point reference
- Verify results against original requirements

Fixed-Point Design Workflow
case ‘Code Generation’:
Algorithm Export (MAAB\(^1\) Model, MISRA C\(^2\))
/* Model step function */

void very_simple_step(void)
{
    /* Sum: '<Root>/Add' incorporates:
     * Inport: '<Root>/a'
     * Inport: '<Root>/b'
    */
    /*
    * c = a + b;
    */
19 /* Model step function */
20 void Run_Cyclic(void)
21 {
22  /* SignalConversion: '<Root>/TmpSignal ConversionAtcInport2' in
23   * Inport: '<Root>/a'
24   * Inport: '<Root>/b'
25   * SignalConversion: '<Root>/TmpSignal ConversionAtaOutport2'
26   * SignalConversion: '<Root>/TmpSignal ConversionAtbOutport2'
27   * Sum: '<Root>/Add'
28 */
29  Rte_IWrite_Run_Cyclic_SPC_c(Rte_IRead_Run_Cyclic_RPa_a() +
30     Rte_IRead_Run_Cyclic_RPb_b());
31 }
Saturation: on
void very_simple_sat_step(void)
{
    int32_T tmp;

    /* Sum: '<Root>/Add' incorporates:
       * Inport: '<Root>/a'
       * Inport: '<Root>/b'
    */
    tmp = (int32_T)a + b;
    if (tmp > 32767L) { 
        tmp = 32767L;
    } else {
        if (tmp < -32768L) {
            tmp = -32768L;
        }
    }

    *c = (int16_T)tmp;

    /* End of Sum: '<Root>/Add' */
Saturation: on
Algorithm Export (MAAB Model, Target Code)

Controller Model

Generated Algorithm Code

Input Drivers

Comm Drivers

Output Drivers

Special Device Drivers

Scheduler/Operating System and Support Utilities

Included Target Optimized Code

Tuning

Communication Interfaces

Actuators

Sensors

Special Interfaces

Input Drivers

Output Drivers

Scheduler/Operating System and Support Utilities

Generated Algorithm Code

Included Target Optimized Code

Tuning

Communication Interfaces

Actuators

Sensors

Special Interfaces
19 /* Model step function */
20 void easy_integrate_step(int16_T a, int16_T b, int16_T *c)
21 {
22    /* Output: '<Root>/c' incorporates:
23       * Input: '<Root>/a'
24       * Input: '<Root>/b'
25       * Sum: '<Root>/Add'
26    */
27    *c = c28x_add_s16_s16_s16_sat(a, b);
28 }
Full Executable (Target Model, Target Code)

Controller Model w/Driver Blocks

Communication Interfaces

Sensors

Input Drivers

Generated Algorithm Code

Output Drivers

Optional Target Optimized Code

Special Device Drivers

Scheduler/Operating System and Support Utilities

Actuators

Special Interfaces

Tuning
/* Model step function */
void very_simple_io_wrapper_step(void)
{
    /* S-Function (c2802xadc): '<Root>/ADC' */
    {
        /* Internal Reference Voltage: Fixed scale 0 to 3.3 V range. */
        /* External Reference Voltage: Allowable ranges of VREFHI(ADCINA0) = 3.3
         * AdcRegs.ADCSOCFRCl.bit.SOC0 = 1;
         * asm(" RPT #22 || NOP");
         * rtDW.a = (AdcResult.ADCRESULT0);
         */
    }

    /* S-Function (c2802xadc): '<Root>/ADC1' */
    {
        /* Internal Reference Voltage: Fixed scale 0 to 3.3 V range. */
        /* External Reference Voltage: Allowable ranges of VREFHI(ADCINA0) = 3.3
         * AdcRegs.ADCSOCFRCl.bit.SOC1 = 1;
         * asm(" RPT #22 || NOP");
         * rtDW.b = (AdcResult.ADCRESULT1);
         */
    }

    /* ModelReference: '<Root>/Algorithm' */
simple_integrate_step(rtDW.a, rtDW.b, &rtDW.c);

    /* S-Function (c28xsci_tx): '<Root>/SCI_Transmit' */
    {
    }
void simple_integrate_step(int16_T a, int16_T b, int16_T *c)
{
    *c = c28x_add_s16_s16_s16_sat(a, b);
}
case 'Targets':
Hardware Support

Connect MATLAB and Simulink to Hardware

Connect MATLAB® and Simulink® to hardware platforms for project-based learning, signal processing, computer vision, communications, data acquisition, instrument control, embedded systems, and more. Use low-cost hardware platforms like Arduino® and Raspberry Pi™, embedded systems like ARM®- and Zynq®-based architectures, or high-end systems for real-time testing. Find the right solution to help you prototype and implement your applications with MATLAB and Simulink.

Search Hardware Support

Explore supported hardware by:

Vendors
Select vendor

Applications
Select application

Product Families & Products
Select product or family

Or enter a keyword:
Trainings Services

Developing Embedded Targets Advisory Service
case ‘Getting ARMed’:
INTEGRATION
IMPLEMENTATION
DESIGN
RESEARCH
REQUIREMENTS

TEST & VERIFICATION
TEST SYSTEM

Environment Models
  Mechanical
  Electrical

Supervisory Logic
Control Algorithms

C, C++, VHDL, Verilog, Structured Text

MCU, DSP, FPGA, ASIC, PLC
Create Model
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

Controller Mode Scheduler

Simulink

Stateflow
Simulate and Test (on Host)
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

Model Description: Field-Oriented Control of Permanent Magnet Synchronous Machine

Demonstrates a Field-Oriented Control algorithm with Space Vector Modulation for a Permanent Magnet Synchronous Machine (PMSM). The test bench can be used to evaluate the system performance. Examples include turning the motor on, observing for a valid rotor position, transitioning to closed-loop operation, and changing speed and torque during closed-loop control. The Embedded Processor subsystem contains the controller algorithm (which supports C code generation) as well as simulation models of peripherals.
Generate ARM Optimized Code
/* Trigonometry: '<S14>/sine_cosine1' */

\[
\text{cos\_coefficient} = \text{arm\_cos\_f32}(\text{sin\_coefficient});
\]
Processor-in-the-Loop Execution Profiling
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

Copyright 2010-2012 The MathWorks, Inc.

System Inputs

Processor

Inverter and Motor

System Analysis

Discrete, Ts = 1e-05 s.

powergui

Host

Cable

Target

STM32F
Add I/O Blocks, Generate Code, and Deploy!
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

STM32F4xx Blockset
default:
    printf("Time for a break!");
}
INTEGRATION
IMPLEMENTATION
DESIGN
RESEARCH
REQUIREMENTS