MATLAB EXPO

Spinning Brushless Motors with Simulink

Nukul Sehgal Vamshi Kumbham



We will spin a brushless motor using Simulink and Model-Based Design







Brushless motors are everywhere



















Developing embedded motor control software has its challenges

ITK Engineering develops IEC 62304– compliant controller for dental drill motor with Model-Based Design

Challenge

Develop and implement field-oriented controller software for sensorless brushless DC motors for use in dental drills

Solution

Use Model-Based Design with Simulink, Stateflow, and Embedded Coder to model the controller and plant, run closed-loop simulations, generate production code, and streamline unit testing

Results

- Development time halved
- Hardware problems discovered early
- Contract won, client confidence established



Dental drills featuring ITK Engineering's sensorless brushless motor control.

"Model-Based Design with Simulink enabled us to design and optimize the controller even before the motor hardware was available for testing and then generate production code for the controller once we had the motor. It would have been impossible to complete this project on schedule if we had written the code by hand."

- Michael Schwarz, ITK Engineering



Developing embedded motor control software has its challenges

- Design work needed to be started before motor hardware was available and needed extensive testing to comply with standards
- Team needed to rapidly implement control software on embedded processor once more hardware became available
- Complex algorithms running at high sample rates were difficult to implement in short amount of time

ITK Engineering develops IEC 62304– compliant controller for dental drill motor with Model-Based Design

Challenge

Develop and implement field-oriented controller software for sensorless brushless DC motors for use in dental drills

Solution

Use Model-Based Design with Simulink, Stateflow, and Embedded Coder to model the controller and plant, run closed-loop simulations, generate production code, and streamline unit testing

Results

- Development time halved
- Hardware problems discovered early
- Contract won, client confidence established



Dental drills featuring ITK Engineering's sensorless brushless motor control.

"Model-Based Design with Simulink enabled us to design and optimize the controller even before the motor hardware was available for testing and then generate production code for the controller once we had the motor. It would have been impossible to complete this project on schedule if we had written the code by hand."

- Michael Schwarz, ITK Engineering



Why Simulink for motor control?

 Verify control algorithm with desktop simulation

Generate compact and fast code from models

 Minimize development time using reference examples Customers routinely report 50% faster time to market





Motor Control Blockset simplifies the workflow

- Control blocks optimized for code generation
- Sensor decoders and observers
- Motor parameter estimation
- Controller autotuning
- Reference examples



Motor Control Blockset[™] provides reference examples and blocks for developing field-oriented control algorithms for brushless motors. The examples show how to configure a controller model to generate compact and fast C code for any target microcontroller (with Embedded Code[#]). You can also use the reference examples to generate algorithmic C code and driver code for specific motor control kits.

The blockset includes Park and Clarke transforms, sliding mode and flux observers, a space-vector generator, and other components for creating speed and torque controllers. You can automatically tune controller gains based on specified bandwidth and phase margins for current and speed loops (with Simulink Control Design™).

The blockset lets you create an accurate motor model by providing tools for collecting data directly from hardware and calculating motor parameters. You can use the parameterized motor model to test your control algorithm in closed-loop simulations.

Get Started:	
Reference Examples	Latest Features
Motor Control Algorithms	Documentation and Resources
Sensor Decoders and Observers	Try or Buy
Controller Autotuning	
Motor Parameter Estimation	
Motor Models	





Brushless motors require complex algorithms – field-oriented control







Workflow for implementing field-oriented control







We will use Texas Instruments motor control kit







Sensor calibration



Calibrate ADC offsets





case 'BoostXL-DRV8305'			
inverter.model	= 'BoostXI	L-DRV8305'	;%]
inverter.sn	= 'INV_XXX	CX';	1
inverter.V_dc	= 24;	\$V	11
inverter.I_max	= 19.3;	%Amps	11
inverter.I_trip	= 10;	%Amps	11
inverter.Rds_on	= 2e-3;	%Ohms	11
inverter.Rshunt	= 0.007;	%Ohms	11
inverter.MaxADCCnt	= 4095;	%Counts	11
inverter.CtSensAOffset	= 2300	%Cou	nts
inverter.CtSensBOffset	= 2303	%Cou	nts
inverter.ADCGain	= 1;	ę	11









Sensor calibration

Calibrate ADC offsets

 Calibrate position sensor offset







Parameter estimation

- Instrumented tests running on the target
- Host model to start and control parameter estimation

	C	Calibrate Sensors Parc	Model otor imeters Motor & Inverter	Design Control Algorithm
a mcp"t	param_est_host_read * - Simulink trial use			- 0 >
SIMUL New Jag	ATION DEBUG MODELIN Open - DEBUG Gold Save - Diracy Iog Print - Browser Signals Rie UBRARY	NG CORMAT APPS Ann Signal Viewer Table PREPARE	Run Step Stop Forward Inspector Analyzer Arte Review Results	- C - C - C - C - C - C - C - C - C - C
Model Brow		Board Selection DRV8305 and F28379D Launchpad Communication Port	Run Stop	Fault Status Over Current Under Voltage Serial communication
		The Cold Pit Analysis and Pite States For F331760 Launchood, soil Bourinas to 566 Required Inputs Nominal Voltage: 24 V Nominal Current: 7,1 A (rms value)	Ks ohm Ld H Lq H Bemf Vee/krpm	
		Nominal Speed: 4000 rpm Pole pairs: 4 Input DC Voltage: 20 v Hall Offset: 0.2039 Per Unit Position	Motor Inertia - kg.m^2 Friction constant - N.m.s Seve Parameters. Open Model	Signal from Target
in fi	6 5	Note: Press CtrI+D to update the workspace Hall Offset: For Hall offset calculation open required model for the hardware mcb_presm_hall_offset (23096m mcb_presm_hall_offset (23379d) Target Models: Click Build load and Run in required model for loading the target mcb_param_est (28056) DRV8312 mcb_param_est (28376D_DRV8305	Signal Conditioning and Scaling	SecondSyra SecondSyra





Bonus: you can use other techniques to parameterize motor models



Two test harnesses that add confidence that a PMSM is correctly parameterized from a datasheet. It also calculates motor efficiency at Import a motor design from ANSYS® Maxwell® into a Simscape™ simulation.

Open Model



Import IPMSM Flux Linkage Data from Motor-CAD

Import a motor design from Motor-CAD into a Simscape™ simulation.

Open Model

Generate Parameters for Flux-Based PMSM Block

Using MathWorks tools, you can create lookup tables for an interior permanent magnet synchronous motor (PMSM) controller that characterizes the *d*-axis and *q*-axis current as a function of *d*-axis and *q*-axis flux.

To generate the flux parameters for the Flux-Based PMSM block, follow these workflow steps. Example script CreatingIdqTable.m calls gridfit to model the current surface using scattered or semi-scattered flux data.

Workflow	Description
Step 1: Load and Preprocess Data	 Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA): <i>d</i>- and <i>q</i>- axis current <i>d</i>- and <i>q</i>- axis flux Electromagnetic motor torque
Step 2: Generate Evenly Spaced Table Data From Scattered Data	Use the gridfit function to generate evenly spaced data. Visualize the flux surface plots.
Step 3: Set Block Parameters	Set workspace variables that you can use for the Flux-Based PM Controller block parameters.

From datasheet

Open Model

MATLAB EXPO

Simscape Electrical

From ANSYS Maxwell, JMAG, Motor-CAD FEA tools

Simscape Electrical

From dyno data

Powertrain Blockset



Modeling motor and inverter

- Use linear lumped-parameter motor model
- Model inverter as an average-value inverter or model switching with Simscape Electrical







Bonus: you can model at needed level of fidelity



17

Control algorithm design

- Model field-oriented control algorithm
- Model sensor decoders or sensorless observers
- Tune loop gains

 Verify in closed-loop simulation





Control algorithm design

- Model field-oriented control algorithm
- Model sensor decoders or sensorless observers
- Tune loop gains

 Verify in closed-loop simulation

MATLAB EXPO





19

Bonus: you can use several techniques to tune loop gains

88 Set PWM Swi	tching frequenc	У			
PWM frequency	= 20e3; %H	z	// cont	verter s/w freq	
T_pwm	= 1/PWM_frequ	ency; %s	// PWM	switching time period	
%% Set Sample	Times				
Ts	= T_pwm;	*sec	11	sample time for controller	
Ts_simulink	= T_pwm/2;	*sec	11	simulation time step for mod	del simulation
Ts_motor	= T_pwm/2;	*Sec	11	simulation sample time	
Ts_inverter	= T_pwm/2;	\$sec	11	simulation time step for ave	erage value inverter
Ts_speed	= 10*Ts;	*Sec	11	sample time for speed contro	oller
%% Set data ty	pe for controll	er & code-g	ren		
<pre>% dataType = f</pre>	ixdt(1,32,17);	% Fixed	point (code-generation	
dataType = 'si	ngle';	% Floati	ng poir	nt code-generation	
%% System Para	meters // Hardw	are paramet	ers		
pmsm = mcb_Set	PMSMMotorParame	ters('BLY17	1D');		
%% Parameters	below are not m	andatory fo	r offse	et computation	
inverter = mcb	_SetInverterPar	ameters('DR	V8312-0	C2-KIT');	
inverter.ADCOf	fsetCalibEnable	= 1; % Ena	ble: 1,	Disable:0	
target = mcb_S	etProcessorDeta	ils('F 28069	M', PWM	frequency);	
%% Derive Char	acteristics				
pmsm.N_base = :	mcb_getBaseSpee	d (pmsm, inve	rter);	<pre>%rpm // Base speed of motor</pre>	at given Vdc
% mcb_getChara	cteristics (pmsm	, inverter);			
th PU System d	etails // Set b	ase values	for pu	conversion	
PU_System = mc	b_SetPUSystem(p	msm,inverte	r);		
%% Controller	design // Get b	allpark val	ues!		
PI_params = mc	b.internal.SetC	ontrollerPa	rameter	rs(pmsm,inverter,PU_System,T_	_pwm,Ts,Ts_speed);

Empirical Computation

Motor Control Blockset

MATLAB EXPO



FOC Autotuner

Motor Control Blockset and Simulink Control Design

Field-Oriented Control Of Motor Velocity rincity Commen ħŞ M ωr -**D** Tune Field-Oriented Controllers Using SYSTUNE Tune a field-oriented controller for an asynchronous machine in one simulation Open Script

Classic Control Theory

Simulink Control Design



Control algorithm design

- Model field-oriented control algorithm
- Model sensor decoders or sensorless observers
- Tune loop gains

 Verify in closed-loop simulation





Deployment



- Target any processor with ANSI C code
- Use provided example to partition the model into algorithmic and hardware-specific parts
- Generate algorithmic code for integration into embedded application





Deployment

- Generate code (floating and fixed-point)
- Use host model to control and debug
- Validate on hardware







MathWorks Training Services: Exploit the full potential of MathWorks products



Simulation Based Testing



Production Code Generation









You can verify and profile code using Processor-In-the-Loop testing

Code Execution Profiling Report for mcb_pmsm_foc_sim_v2/Current Control1

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See <u>Code Execution</u> <u>Profiling</u> for more information.

1. Summary

Total time	50681790
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e- 09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	2e+08
Profiling data created	16-Jan-2020 18:09:48

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Sel Time in n	f Average Self s Time in ns	f Calls		
[+] Current_initialize	2260	2260	1365	1365	1	-	
Current_step [5e-05 0]	5135	5067	5135	5067	10001	-	
Current_terminate	540	540	540	540	1	-	

3. CPU Utilization

Task	Average CPU Utilization	Maximum CPU Utilization
Current_step [5e-05 0]	10.13%	10.27%
Overall CPU Utilization	10.13%	10.27%





Bonus: you can target FPGAs as well

Code Generation Report		_ 0
🗧 😪 😂 Find:	😔 🔮 Match Case	
Contents unimary Cock Summary Code Interface Report	 HDL Code Generation Report Sum Summary 	mary for focZynqHdl
iming And Area Report		
High-level Resource Report	Model	focZynqHdl
Distributed Displining	Model version	1.368
Streaming and Sharing	HDL Coder version	3.10
Delay Balancing	HDL code generated on	2017-04-21 14:19:09
Adaptive Pipelining	HDL code generated for	<u>focZynqHdl</u>
Tradition to the state of	Target Language	VHDL
P Core Generation Report		
P Core Generation Report Traceability Report	Target Directory Non-default model properties	hdl_prj\hdlsrc
P Core Generation Report raceability Report	Target Directory Non-default model properties ClockRatePipelining	hdl_prj\hdlsrc off
Core Generation Report raceability Report	Target Directory Non-default model properties ClockRatePipelining EnablePrefix	hdl_prj\hdlsrc off oversampledClockEnable
Core Generation Report raceability Report Generated Source Files rocZyngHd_ip_src_focZyngHdl_p	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem	hdl_prj\hdisrc off oversampledClockEnable focZynqHdl
Core Generation Report raceability Report Generated Source Files CocZyngHd_ip_src_focZyngHdl_p CocZyngHd_ip_src_ADC_Count_T	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix	hdl_prj\hdisrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_
Core Generation Report raceability Report Generated Source Files focZyngHd_ip_src_focZyngHdl_pi focZyngHd_ip_src_ADC_Count_T focZyngHd_ip_src_Mod_Two_Pi	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport	hdl_prj\hdlsrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_ on
<u> P Core Generation Report</u> <u> Taceability Report</u> <u> Senerated Source Files</u> <u> ocZyngHd_ip_src_focZyngHd_p</u> <u> ocZyngHd_ip_src_ADC_Count_T</u> <u> ocZyngHd_ip_src_Mod_Two_Pi</u> <u> ocZyngHd_ip_src_Encoder_Court</u>	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport Oversampling	hdl_prj\hdisrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_ on 2000
<u>Core Generation Report</u> <u>Taceability Report</u> <u>Senerated Source Files</u> <u>ocZyngHd_ip_src_focZyngHd_p</u> <u>ocZyngHd_ip_src_ADC_Count_T</u> <u>ocZyngHd_ip_src_Mod_Two_Pi</u> <u>ocZyngHd_ip_src_Encoder_Court</u> <u>ocZyngHd_ip_src_Encoder_Stc_Stc_Stc_Stc_Stc_Stc_Stc_Stc_Stc_Stc</u>	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport Oversampling ReferenceDesign	hdl_prj\hdisrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_ on 2000 Motor Control Reference Design
Core Generation Report raceability Report Generated Source Files ocZyngHd_ip_src_focZyngHdl_p ocZyngHd_ip_src_ADC_Count_T ocZyngHd_ip_src_Mod_Two_Pi ocZyngHd_ip_src_Encoder_Cour ocZyngHd_ip_src_Wrap_Neg_Pi	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport Oversampling ReferenceDesign ResetType	hdl_prj\hdisrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_ on 2000 Motor Control Reference Design Synchronous
Core Generation Report raceability Report Generated Source Files ocZyngHd_ip_src_focZyngHdl_p ocZyngHd_ip_src_ADC_Count_T ocZyngHd_ip_src_Mod_Two_Pi ocZyngHd_ip_src_Encoder_Cour ocZyngHd_ip_src_Wrap_Neg_Pi ocZyngHd_ip_src_Rotor_Position	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport Oversampling ReferenceDesign ResetType ResourceReport	hdl_prj\hdisrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_ on 2000 Motor Control Reference Design Synchronous on
P Core Generation Report raceability Report Generated Source Files focZyngHd_ip_src_focZyngHdl_p ocZyngHd_ip_src_ADC_Count_T ocZyngHd_ip_src_Mod_Two_Pi ocZyngHd_ip_src_Encoder_Cour ocZyngHd_ip_src_Encoder_Cour ocZyngHd_ip_src_Rotor_Position focZyngHd_ip_src_Rotor_Position	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport Oversampling ReferenceDesign ResetType ResourceReport ScalarizePorts	hdl_prj\hdisrc off oversampledClockEnable focZynqHd focZynqHd_ip_src_ on 2000 Motor Control Reference Design Synchronous on on
P Core Generation Report Traceability Report Generated Source Files TocZyngHd_ip_src_focZyngHd_p TocZyngHd_ip_src_ADC_Count_T TocZyngHd_ip_src_Mod_Two_Pi TocZyngHd_ip_src_Encoder_Cour TocZyngHd_ip_src_Encoder_Cour TocZyngHd_ip_src_Rotor_Position TocZyngHd_ip_src_Rotor_Position TocZyngHd_ip_src_Rotor_Position TocZyngHd_ip_src_Rotor_Position TocZyngHd_ip_src_Rotor_Position	Target Directory Non-default model properties ClockRatePipelining EnablePrefix HDLSubsystem ModulePrefix OptimizationReport Oversampling ReferenceDesign ResetType ResourceReport ScalarizePorts SynthesisTool	hdl_prj\hdlsrc off oversampledClockEnable focZynqHdl focZynqHd_ip_src_ on 2000 Motor Control Reference Design Synchronous on On Xilinx Vivado

HDL Code Generation





Workflow for implementing field-oriented control







ATB Technologies cuts electric motor controller development time by 50% using code generation for TI's C2000 MCU

Challenge

Develop control software to maximize the efficiency and performance of a permanent magnet synchronous motor

Solution

Use MathWorks tools for Model-Based Design to model, simulate, and implement the control system on a target processor

Results

Development time cut in half

MATLAB EXPO

- Design reviews simplified
- Target verification and deployment accelerated



ATB Technologies permanent magnet synchronous motor.

"MathWorks tools enabled us to verify the quality of our design at multiple stages of development, and to produce a high-quality component within a short time frame."

- Markus Schertler, ATB Technologies



Use Model-Based Design for your next motor control project!

 Verify control algorithm with desktop simulation

Generate compact and fast code from models

 Minimize development time using reference examples, built-in algorithmic blocks, automated parameter estimation, and gaintuning





Learn More

- Visit <u>mathworks.com/products/motor-control</u> and <u>mathworks.com/solutions/power-</u> <u>electronics-control</u>
- Get <u>power electronics control design trial</u> <u>package</u> with necessary tools for desktop modeling, simulation, control design, and production code generation of your next motor control project



START TODAY. Download and install the trial software package.





As a follow up, in which area would you like to talk to us?

- Buy/Try the product
- Speak with our technical expert
- Training (Paid)
- Consulting (Paid)
- I am not ready for any of the above



