Hardware-Software Codesign of EKF-based Motor Control for Domain-Specific Reconfigurable Platform

Yan Lin Aung, Siew-Kei Lam, Thambipillai Srikanthan
CHiPES Research Centre, School of Computer Engineering
Nanyang Technological University
Singapore
{layan, assklam, astsrikan}@ntu.edu.sg

***Abstract*—This paper presents hardware-software codesign of Extended Kalman Filter (EKF) based motor control for a domain-specific reconfigurable platform, which consists of a processor for automotive applications and an FPGA for application-specific customization. Considering the existing MISRA C compliant software harnessing dedicated on-chip peripherals, we employ a codesign methodology aiming to enable product differentiation through modest hardware accelerator implementation in the reconfigurable logic thus meeting the application constraints under tight time-to-market pressure. A key step in the design methodology for reducing the effort of hardware customization lies in platform-aware hardware-software partitioning, which takes into accounts communication overhead between the various computing elements. We show that our approach can effectively identify a suitable candidate for hardware acceleration embracing domain-specific characteristics and existing standard-compliant software.**

Keywords-Extended Kalman Filter, Kalman gain, codesign, domain-specific reconfigurable platform

I. INTRODUCTION

The advent of SoC FPGAs from Xilinx and Altera which incorporates ARM-based processor system and reconfigurable logic is paving the way towards incorporating reconfigurability on all future integrated circuits. Application processors, for example ARM Cortex-A9 in SoC FPGAs, are well-suited for a wide range of computing platforms from low-cost handsets to smartphones, digital TV and set-top boxes to enterprise networking and server solutions. We expect that domain-specific processors such as TriCore™ family of processors from Infineon Technologies [1], which are designed for unique application areas (e.g. automotive, industrial), will eventually ride on the wave of integrating reconfigurability to increase product differentiation. Domain-specific processor-based systems typically operate at lower clock frequency (e.g. 180 MHz for Infineon TC1797 TriCore™ processor) compared to application processor counterparts (e.g. 800 MHz – 2 GHz for ARM Cortex-A9) and feature specialized peripherals (e.g. Timer Arrays providing a set of timer, compare and capture functionalities). Moreover, a sizable amount of software optimized for the platform by leveraging the peripherals exists and typically the software must be compliant to standards set out for specified application domain (e.g. MISRA C standard for automotive industry). The codesign methodology that takes advantage of reconfigurability in domain-specific platforms must take into account the above-mentioned constraints, in particular lower operating clock frequency, specialized peripherals and existing certified standard-compliant software.

We chose the EKF based motor control as it becomes pertinent in emerging electric vehicle applications. While it is possible and tempting to use the existing hardware platform with SoC FPGA on-board (e.g. ZedBoard from [2]), the efforts require to create a prototype motor control platform with necessary hardware and software is enormous. On the other hand, there is no processor-based system with on-chip reconfigurability available for automotive applications. This has led us to design a prototype platform consisting of 32-bit AUDO family TC1797 TriCore™ processor from Infineon Technologies [3], Spartan-6 LX25 FPGA from Xilinx, motor driver, current sensors and brushless DC (BLDC) motor. We then employ a simplified hardware-software co-design methodology for maximizing the productivity of application development on the prototype platform. Our approach places high emphasis on the use of existing standard-compliant software and communication-centric hardware-software partitioning that aids in reducing the hardware implementation efforts for meeting the application constraints.

The rest of this paper is organized as follows. Section II provides related work. Section III describes the hardware prototype for EKF-based motor control. We then present a codesign methodology for domain-specific reconfigurable platform, discuss algorithm development, verification and profiling study for EKF followed by constraint-aware hardware-software partitioning in Section V, hardware implementation of Kalman gain in Section VI.

II. RELATED WORK

Significant amount of research work highlighted the benefits of using FPGAs in a variety of application domains: automotive, industrial, medical, aerospace, consumer, etc. FPGAs are of prime interest for implementing digital control systems in automotive and industrial applications [4, 5, 6, 7]. At the same time, domain-specific processors with specialized peripherals and a large amount of existing standard compliant software continue to dominate embedded products. With emerging research in future electric vehicles, control techniques for brushless/permanent magnet synchronous motors have become increasingly popular.

Among those techniques, the Extended Kalman Filter is well-known for its inherent robustness in random noise environment. As conventional solutions based on DSPs failed to meet performance requirement of sensorless applications, Idkhajine et al. proposed fully hardware FPGA-based sensorless controller for synchronous machine using EKF [8]. However, with significant increase in adoption of FPGAs with microprocessor cores [9] and introduction of SoC FPGAs, a hybrid approach that harnesses flexibility and software programmability of microprocessors, hardware customization and programmability of reconfigurable logic for performance will be a preferred solution in future. Hence, there is a need for high productivity design methodologies for domain-specific reconfigurable platforms that is able to recommend hardware-software solutions to meet the functional and non-functional constraints of application.

III. HARDWARE PROTOTYPE FOR EKF-BASED MOTOR CONTROL

Our hardware prototype consists of 1) Infineon TC1797 TriBoard, 2) motor driver board from Infineon's FOC kit, 3) Xilinx Spartan-6 LX25 FPGA board, 4) BL3056 BLDC motor and 5) power regulation and voltage-level shifting circuitries as shown in Fig. 1. To facilitate the terminal voltage and phase current measurement inputs as required by the Kalman filter algorithm, the prototype platform relies on the motor driver board for V_{as}, V_{bs}, V_{cs} measurements and utilizes the current sensors from LEM for I_{as}, I_{bs}, I_{cs} measurements. The 32-bit interface has been established between the TC1797 TriCore™ processor and FPGA module through the external bus unit. The FPGA is memory-mapped to the processor's address space. Infineon Technologies provide MISRA C compliant software for the HybridPACK™ 2 power module designed for full hybrid electrical vehicle applications [10]. The software runs on TriCore™ processor and leverages specialized on-chip peripherals. For example, it uses 23 Local Timer Cells (LTCs) in General Purpose Timer Array (GPTA™) peripheral to generate three phase glitch-less pulse width modulation (PWM) signals, two independent ADCs for phase voltage and current measurements. The software implements complete field oriented control (FOC) together with an outer proportional-integral (PI) speed control loop. The FOC itself consists of two PI controllers, several transformations (e.g. Park, Clarke and reverse Park transforms) and space vector modulation scheme. The speed measurement input for the outer PI control loop is derived from rotor position-sensing hall sensors placed inside the motor during manufacturing.

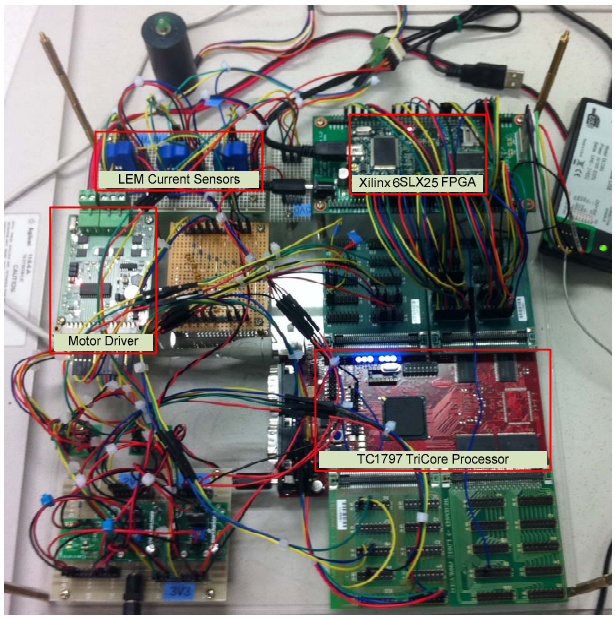

Figure 1. Hardware Platform Prototype

The purpose of Extended Kalman Filter is to estimate position of the rotor and provide the estimated speed to the PI control loop in the absence of hall sensors. It is evident in this case that a large amount of existing MISRA C compliant code prohibits re-implementation of similar functionalities in FPGA as dedicated hardware or processor-accelerator system. Hence, we access the possibility of incorporating Extended Kalman Filter into the current system as merely software only implementation, in-part or complete hardware acceleration of the EKF algorithm till it fits into the existing motor control system using the codesign methodology as described in subsequent sections.

IV. CODESIGN METHODOLOGY FOR DOMAIN-SPECIFIC RECONFIGURABLE PLATFORMS

An overview of the proposed design methodology is shown in Fig. 2. Based on a given specification, algorithm development and verification is first undertaken, possibly in a target independent manner with the aid of suitable high-level design tools (e.g. Matlab). Platform-aware software implementation is performed next in which designers maximize the use of existing software to fulfill functional requirements of the application. Application profiling and runtime analysis is then relied upon to evaluate whether the software only solution meets the application constraints and to identify the application bottlenecks. Hardware-software

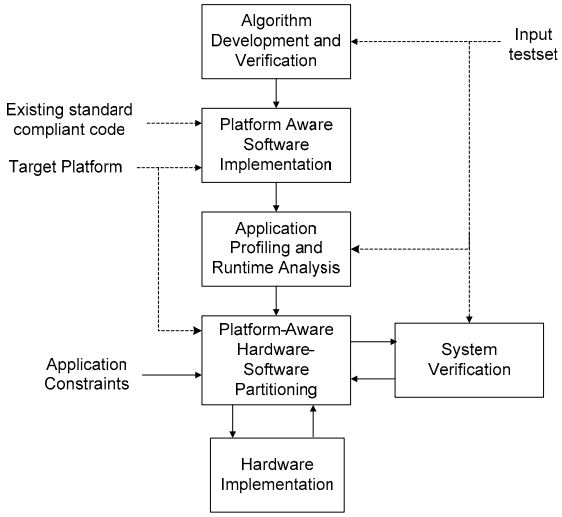

Figure 2. Proposed Codesign Methodology for Domain-Specific Reconfigurable Platforms

partitioning is then carried out iteratively. It aims to determine a set of minimal hardware candidates so as to meet the application constraints. The target platform characteristics such as communication latencies must be taken into account during the partitioning process. The partitioning process also requires area-time measures of the hardware implementation, which is based on either hand-crafted designs or auto-generated using high-level synthesis tools. We employ the above-mentioned methodology during the codesign of EKF-based motor control algorithm and details of the steps are discussed in the following sections.

A. Algorithm Development and Verification

Firstly, we carried out modeling and simulation of the EKF algorithm for motor control in Matlab environment. The Extended Kalman Filter estimates rotor position (θ_r) and speed (ω_r) of the BLDC motor using terminal voltages (V_{as}, V_{bs}, V_{cs}) across three phases and current measurements (I_{as}, I_{bs}, I_{cs}) of the motor. The EKF algorithm relies on state space representation of three-phase BLDC motor model in [11] and consists of two major steps: 1) time update and 2) measurement update. In time update step, the prediction of state vector at sampling time k from the input vector $u(k)$ and the state vector at previous sampling time $x(k-1)$ is first obtained by performing:

$$x_k^- = (I + A \cdot T) \cdot x_{k-1} + B \cdot T \cdot u_k = x_{k-1} + \left(A \cdot x_{k-1} + B \cdot u_k\right) \cdot T \quad (1)$$

where T is the sampling interval, A and B are the state and input matrices respectively. Then, the error covariance is estimated as shown in Eq. (2), where Q is the system noise co-variance and f is the gradient matrix obtained by (3).

$$P_k^- = f_k \cdot P_{k-1} \cdot f_k^T + Q \quad (2)$$

$$f_k = \frac{d\left[x_k^-\right]}{dx} = \frac{d\left[x_{k-1} + \left(A \cdot x_{k-1} + B \cdot u_k\right) \cdot T\right]}{dx} \quad (3)$$

In measurement update step, the Kalman filter gain is then computed as in Eq. (4), where C is the output matrix and R is the measurement noise covariance.

$$K_k = P_k^- \cdot C^T \cdot \left(C \cdot P_k^- \cdot C^T + R\right)^{-1} \quad (4)$$

The state vector estimation is performed in Eq. (5) and Eq. (6).

$$x_k = x_k^- + K_k \cdot \left(y_k - C \cdot x_k^-\right) \quad (5)$$

$$x_k = x_{k-1} + \left(A \cdot x_{k-1} + B \cdot u_k\right) \cdot T + K_k \cdot \left(y_k - C \cdot \left(x_{k-1} + \left(A \cdot x_{k-1} + B \cdot u_k\right) \cdot T\right)\right) \quad (6)$$

Finally, the error covariance matrix can be calculated as shown in Eq. (7).

$$P_k = \left(I - K_k \cdot C\right) \cdot P_k^- \quad (7)$$

Simulation is in Matlab Simulink environment to ensure the correctness of EKF algorithm (simulation results are not provided here for space reasons).Once verified, the C code, which is functionally equivalent to the EKF Matlab code, is developed.

B. Application Profiling and Runtime Analysis

Application profiling of the EKF C code is then carried out for TC1797 TriCore™ processor from two different perspectives of the algorithm.

1) Profiling the Computation Steps of EKF Algorithm

The EKF algorithm can further be broken down into four distinct computation steps, namely 1) error covariance estimation, 2) Kalman gain computation, 3) measurement update, and 4) error covariance update. The computation steps of EKF algorithm is profiled for TC1797 TriCore™ processor running at 180 MHz, which is the maximum operating frequency. Each iteration of EKF algorithm requires 68 µs. The breakdown of computation time requirements for the EKF computation steps are shown in Fig. 3. It can be observed that the Kalman gain computation takes up 41% of the total computation time.

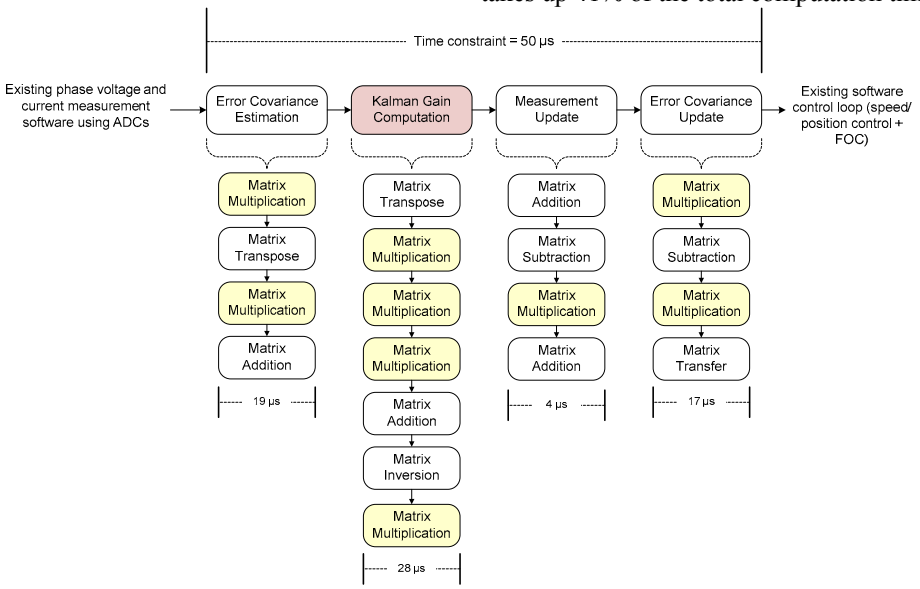

Figure 3. Dependence Flow Graph of EKF Algorithm

2) Profiling the Matrix Operations of EKF Algorithm

Several matrix operations are involved in the EKF computation. Hence, profiling study is undertaken for matrix multiplication, inversion, addition, transpose, subtraction and transfer operation. The profiling study reveals that matrix multiplication needs to be executed 9 times in each iteration of EKF algorithm and occupies 74% of the total computation time.

3) Run-Time Analysis

The HybridPACK™ 2 software allows users to specify the PWM frequency between 8-20 kHz at compile-time. The PWM frequency of 10 kHz is commonly used for control of permanent magnet motors. The control algorithm is synchronized with the generated PWM. An interrupt event is raised at the rising edge and falling edge of PWM period. The former event initiates conversion of terminal voltages across three phases of the motor. User customized code can be implemented during this rising edge interrupt. The falling-edge interrupt starts conversion of the motor phase currents and DC link voltage. This interrupt handles the motor control step which comprises of reading the motor electrical position and phase currents, field oriented control (FOC) and PWM duty cycle calculation.

The EKF algorithm can be incorporated into the HybridPACK™ 2 software during the rising-edge interrupt since the falling-edge counterpart is occupied by the motor control code. The EKF algorithm must complete computation in 50 µs for the 10 kHz PWM frequency. Since profiling study mentioned earlier indicated that the C implementation of EKF algorithm requires 68 µs, computational capability of FPGA has to be exploited in this case to speed up the Kalman filter computation.

V. Platform-aware Hardware-software Partitioning

The purpose of hardware-software partitioning is to identify a minimal set of hardware candidates that can meet the performance constraints. From the analysis on profiling results, we identified two possibilities to accelerate the EKF algorithm: a) floating point matrix multiplication or b) Kalman gain computation in FPGA.

A. Hardware Acceleration of Matrix Multiplication in FPGA

Since matrix multiplication contributes 74% of the total computation time, the possibility of hardware acceleration of matrix multiplication in FPGA is first analyzed. A 5×5 floating-point matrix multiplication can be realized in FPGA to cater for 9 such operations involved in the EKF algorithm. In this approach, the C code on the TriCore™ processor writes 50 single precision floating-point data to FPGA. The matrix multiplication module in FPGA performs the computation and the TriCore™ processor reads back the 25 results. 50 32-bit data transfer from the TriCore™ processor to FPGA requires 5.55 µs (i.e. 50×10×11.11ns) as the external bus unit (EBU) operates at 90 MHz and 10 clock cycles are required for one 32-bit write operation. To read back the 25 data from FPGA, the TriCore™ processor requires 5 µs (i.e. 25×18×11.11ns). One 32-bit read transaction requires 18 clock cycles in this case. If the computation time for matrix multiplication can be overlapped with the data transfer time from the TriCore™ processor to FPGA during data write and read operations, a total of 10.55 µs is required to complete 5×5 matrix multiplication. Therefore, the EKF algorithm requires 94.95 µs for nine matrix multiplications assuming that a single 5×5 matrix multiplier is utilized for the multiplication of matrices with smaller dimension. As the EKF algorithm must complete computation in 50 µs, an FPGA based hardware accelerator for floating-point multiplication will lead to timing constraint violation.

B. Hardware Acceleration of Kalman Gain Computation in FPGA

Next, we consider mapping of the Kalman gain computation in FPGA. In this approach, the TriCore™ processor first sends 15 single precision floating-point data to FPGA. The Kalman gain computation does not require all the 15 input data to be available before it begins processing in order to maximize overlapping of the computation time with the data transfer time. This results in only 8 clock cycles for the Kalman gain computation which does not overlap with the data transfer. The Kalman Gain computation produces 15 single precision floating-point data which are then read by the TriCore™ processor. Hence, the total time required for Kalman gain computation on FPGA is 4.90 µs. Since the C code Kalman gain computation requires 28.02 µs, the FPGA hardware accelerator can accelerate the Kalman gain computation execution time by 5.72×. The EKF algorithm requires 44.89 µs with FPGA hardware acceleration of the Kalman gain computation, which meets the 50 µs timing constraint.

VI. Hardware Implementation of Kalman Gain Computation

We have identified the Kalman gain computation as a suitable candidate for hardware acceleration in FPGA. Hence, we propose an optimized FPGA implementation for Kalman gain computation that takes advantage of the restricted data transfer bandwidth between the TriCore™ processor and FPGA by overlapping the computation time with the data transfer time. This results in hardware acceleration of Kalman gain computation that will incur minimal computation overhead. The proposed implementation for Kalman gain computation, which is seven-stage pipelined architecture, is shown in Fig. 4. The implementation requires 15 single precision floating-point inputs and produces 15 single precision floating-point outputs. Each of the inputs is supplied to the pipelined architecture one at a time and is processed immediately in the pipeline stages. It can be observed that the architecture requires a floating-point adder, a floating-point multiply-accumulator, a floating-point matrix multiplier and a floating-point multiplier. The inverse matrix operation in

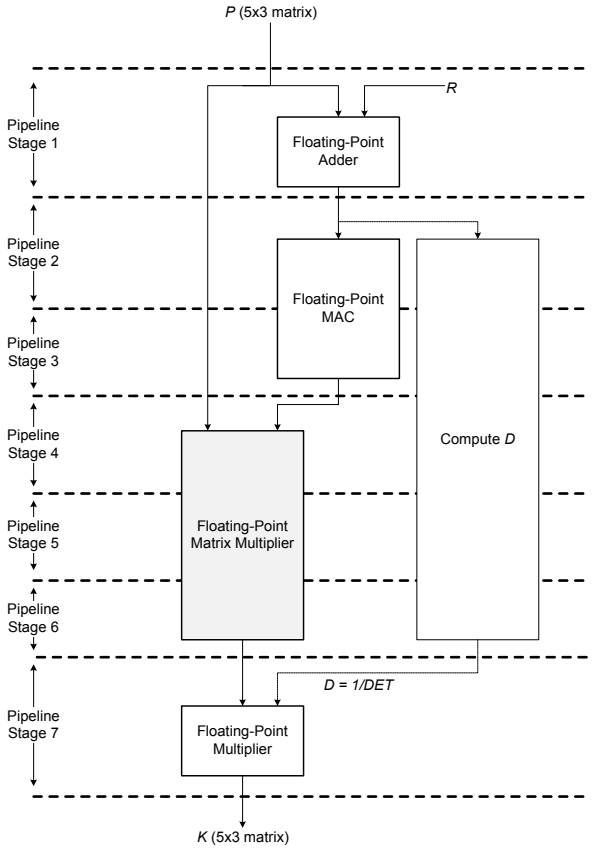

Figure 4. Seven-Stage Pipeline Implementation of Kalman Gain Computation

(4) is further decomposed as computing determinant and taking the reciprocal. Our analysis on the input dataset from the motor model in Matlab shows that D can be set to a constant value. Therefore, the current implementation does not require the module to compute D.

The Kalman gain computation does not require all the 15 input data to be available before it begins processing. This results in only 8 clock cycles for the Kalman gain computation, which does not overlap with the data transfer. The Kalman Gain computation produces 15 single precision floating-point data which are then read by the TriCore™ processor. Hence, the total number of clock cycles for Kalman gain computation on FPGA is 38.

A. Implementation Results

Table I shows the FPGA implementation results of the various modules in the proposed Kalman gain computation. The target FPGA device is Spartan-6 XC6SLX25 from Xilinx. It can be observed that the Kalman computation gain implementation utilizes less than 50% of the FPGA resources. In addition, the Kalman computation gain implementation can be clocked at about 20 MHz. Hence, this implementation enables the write cycle of TC1797 EBU to execute up to 20 MHz. Higher clock frequency can be achieved by further pipelining the modules.

TABLE I. IMPLEMENTATION RESULTS

Module	LUTs		Registers		freq. (MHz)
	Total	Usage	Total	Usage	
Floating-Point Adder	526	3%	0	0%	20
Single Pipeline Stage of Floating-Point Multiply-Accumulator	687	4%	32	0%	20
Single Pipeline Stage of Floating-Point Matrix Multiplier	5186	34%	1505	5%	33
Floating-Point Multiplier	162	1%	0	0%	34

VII. CONCLUSION

We present hardware-software codesign of Extended Kalman Filter based motor control for a domain specific reconfigurable platform. Although it is possible to implement the entire motor control loop in reconfigurable hardware, we show that the codesign methodology must take into account a significant amount of existing and certified motor control software that leverages on-chip features dedicated for automotive applications in order to meet the ever-shrinking time-to-market window. In addition to application analysis and profiling, which often guide the codesign strategies, our study also highlights that the communication latencies between the processor and hardware accelerator is crucial due to its impact on performance and must be given high emphasis during the hardware-software partitioning process.

REFERENCES

[1] "32-bit TriCore™ Microcontrollers," 2013. Available: http://www.infineon.com/TriCore.

[2] (2013). ZedBoard - A low-cost development board for the Xilinx Zynq-7000 All Programmable SoC. [Online]. Available: http://www.zedboard.org/.

[3] TC1797 32-bit Single-Chip Microcontroller Data Sheet V1.2 2009-09. [Online].

[4] E. Monmasson, I. Bahri, L. Idkhajine, A. Maalouf, and W. M. Naouar, "Recent advancements in FPGA-based controllers for AC drives applications," in Optimization of Electrical and Electronic Equipment (OPTIM), 2012 13th International Conference on, 2012, pp. 8-15.

[5] E. Monmasson and M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems — A Review," Industrial Electronics, IEEE Transactions on, vol. 54, pp. 1824-1842, 2007.

[6] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in Industrial Control Applications," Industrial Informatics, IEEE Transactions on, vol. 7, pp. 224-243, 2011.

[7] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, Design Tools, and Application Domains of FPGAs," Industrial Electronics, IEEE Transactions on, vol. 54, pp. 1810-1823, 2007.

[8] L. Idkhajine, E. Monmasson, and A. Maalouf, "FPGA-based Sensorless controller for Synchronous Machine using an Extended Kalman Filter," in Power Electronics and Applications, 2009. EPE '09. 13th European Conference on, 2009, pp. 1-10.

[9] M. Santarini, "Xilinx Customer Innovation: 85,000 to 2.5 Billion Transistors and Beyond," Xcell, pp. 8-15, 2010.

[10] (2013). Evaluation Kit for Applications with HybridPACK™ 2 Module. [Online].

[11] R. Krishnan, Permanent magnet synchronous and brushless DC motor drives. Boca Raton: CRC Press/Taylor & Francis, 2010.