

Distributed, Modular, Open Control Architecture for Power Conversion Systems

Jinghong Guo, Stephen H. Edwards*, and Dushan Boroyevich

Center for Power Electronics Systems
The Bradley Department of Electrical and Computer
Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA

*The Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA

Abstract—The legacy approaches to control development of power conversion systems is topology or application oriented. Due to close coupling to hardware and lack of software engineering technologies, the control software in digitally controlled power conversion system is difficult to design and maintain. This paper proposes a distributed, modular, open control architecture for power conversion systems is proposed to decompose control design complexity and encapsulate and localize independent varieties to reduce unnecessary redesign effort and improve software quality. An implementation of the proposed control architecture will also be presented and evaluated by comparison to the legacy approach to control design in industry and academic.

I. INTRODUCTION

In medium- to high- power conversion applications, digital control is widely used to implement complicated control algorithms. Because control software is easier to change than other design domains, such as power stage hardware, control host hardware, etc., it is normally the last area to be implemented in the whole system design. The legacy approaches to the control development of power conversion systems, mainly in monolithic assembly or C language, is per-application-based. Due to both the close coupling to hardware and the lack of application of software engineering technologies, the control software in the digitally controlled power conversion system is difficult to design and maintain, which in turn causes the following problems:

- Multi-disciplinary expertise is required for the individual control software designer: A single control designer needs to know every detail of control theory, control host hardware, the controlled device, and even digital communication techniques; the control construction is time consuming and error prone;
- Software development is highly dependent on hardware; this dependency and the lack of well-defined application programming interfaces (APIs) complicate system integration and maintenance;
- Quality achieved in prior software designs does not carry over well;

- New functionality or components are hard to add to existing systems; and
- The low adaptability to changes in the application requirements and runtime environment leads to a tremendous amount of redesign needed for each new system.

To address these problems, this paper does not emphasize the functionality of power converter control algorithms, but concentrates instead on the design methodology and software engineering aspects that could benefit control design in a wide array of power conversion systems. The drawbacks of legacy software design approaches originate from their monolithic software structure, which achieves customization at computation capacity but at the cost of flexibility. This research work explores the feasibility of applying software engineering techniques to shift the design embedded control of power conversion systems from the sole choice of control algorithms and data structures to a well-structured, open software architecture paradigm based on dataflow software style and a scalable lightweight real-time operating system. Since power conversion system control is hard real-time, the open control architecture presented in this work attempts to balance the maximum flexibility with the performance of the control software.

In Section II, a distributed, modular, open control architecture is proposed based on analysis of power conversion system control from control algorithms, real-time characteristics and system design perspectives. Section III gives an implementation of the open control architecture using dataflow software style. In Section IV, to verify and evaluate the open control architecture and the dataflow-based implementation, one control application of a PEBB-based three-phase inverter is implemented to compare the control software performance and construction effort of the proposed approach with legacy approaches.

II. CHARACTERIZATION OF DIGITAL CONTROL OF POWER CONVERSION SYSTEMS

A. Analysis of Control Algorithms in Power Converters

The generic control structure of three-phase PWM converters shown in Fig. 1 illustrates that the control normally has three components – modulator, current control (or inner loop control) and output control (or outer loop

This work was supported primarily by the Office of Naval Research under Award Number N00015-01-1-0954 and secondarily by the ERC Program of the National Science Foundation under Award Number EEC-9731677.

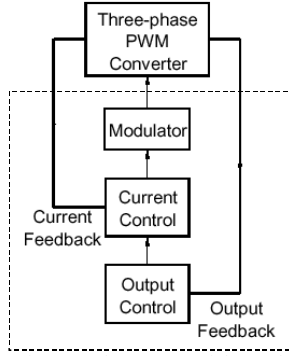


Fig. 1. Generic control structure of three-phase converters.

control). For each of the control components, there are numerous control algorithms that can be categorized in different ways [1, 2, 3, 4, 5].

Fig. 2 shows a current loop control using dq transformation technique. By studying different control scheme applied in power conversion applications, it can be seen that a single control scheme can be composed of a set of commonly used functionalities. For example, current control and outer loop control can be constructed from math functions, such as algebraic functions or filters in time domain, s-domain or z-domain with single or multiple input/output. On the other hand, various modulators are normally implemented as logic functions.

B. Real-Time Computing Characteristics of Power Conversion Control

One routine task of the real-time control of a power converter is to generate turn on/off commands to be sent to the switching network. This type of event is periodic and clock-based. During each switching period, stimuli inputs cause the beginning of real-time computing. In the real-time analysis, from the beginning of a real-time computing brought about by a set of stimuli, inputs to the generation of outputs to the same converter with same updating frequency are defined as an event. An event may be further decomposed into multiple actions.

A system based on a switching clock may contain a single control loop or multiple control loops. And within a system, control events for multiple converters may coexist. To control a converter, no matter what control technique is used, the

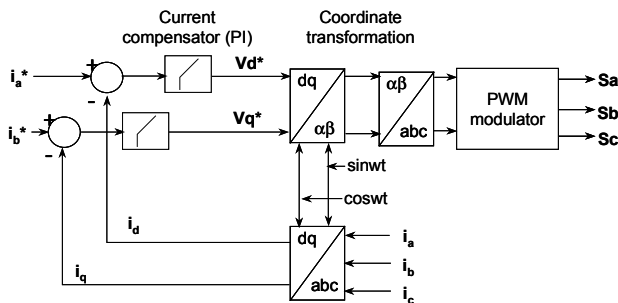


Fig. 2. Current control loop using dq transformation technique.

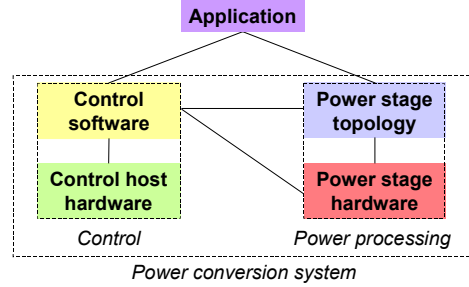


Fig. 3. Power conversion system structure.

switching network control information must be updated every switching period, while the stimuli inputs may arrive synchronously or asynchronously. It is very likely that in one power conversion application, the controller needs to control more than one converter, or more generally speaking, handle two or more events at their respective rates. Under this scenario, the multi-rate events model should be used.

C. Interaction Between Embedded Control Software and Other Design Areas

From the standpoint of system design, a digitally controlled power conversion system involves several design areas that relate to each other, as shown in Fig. 3. The application can be seen as a set of requirements that define the behaviors of a specific power conversion system, such as the input/output relationship, the load range and the transient-response characteristics. The desired system behaviors are implemented by control and power processing subsystems working interactively. The control subsystem can be seen as control software running on the control host hardware, which is the hardware chip set or digital boards. The power processing subsystem has two levels of interpretations. The power stage topology is the logical circuit description of power processing, and the power stage hardware means all parts of the physical implementation of the circuit, such as switches, passive components, sensing circuits and their interconnections.

Normally the selection of control algorithm is dependent on the application, and is decided upon after the power stage topology is chosen. The controlled power stage hardware specifications affect how the control software should communicate with it. And the structure of the power stage hardware brings up the requirements for coordination and communication to control the software. The control host hardware sets limitations on control software size, response speed to external events, and capability of communication with external components.

Since control software has dependencies on application, control host hardware, power stage topology and power stage hardware, if these dependencies are not localized and encapsulated in the control software design, the embedded code will inevitably require significant redesign in order to adapt to even minor changes in a system. The traditional procedural or imperative approach to designing embedded

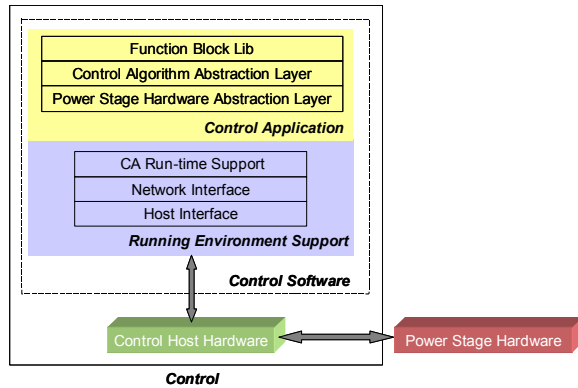


Fig. 4. Distributed, modular, open control architecture for power conversion systems.

control software results in control software that is hard to maintain and modify, is tightly coupled with other areas of the system, and does not encourage software reusability. Under the main-program-and-subroutine style widely used to compose embedded control software, information dependent on different design areas are interwoven. Changes in one related design area may cause overall redesign of the control software very likely.

Academically and commercially, some research has attempted to tackle some of the dependencies. The OPCoDe [6] project from ABB targets software concepts for control resources and power stage hardware independence. IEC 1131-3 [7] provides a generalized software design model for programming industrial control systems in order to build large systems using equipment from different manufacturers. However, none of these approaches completely achieved localization and encapsulation of every type of dependency.

Based on the analysis of how control software relates to other areas in power conversion system design, this paper proposes an open control architecture for power conversion systems, which is capable of mapping and localizing different types of dependencies, as shown in Fig. 4. In this architecture, the control software is constructed as two main layers: control application and runtime environment support. The control application encapsulates all the dependencies related to the system application. The runtime environment support makes all the non-application-related dependencies transparent to the control application.

It should be noticed that although the choice of control algorithm is dependent on the application or power stage topology, it could be implemented from a set of commonly used functionalities, which can be repeated in different applications. Function Block Lib is a library of such independent function modules, which makes it possible to construct a control algorithm from a fairly large piece of reusable code instead of instruction lines. The control algorithm abstraction layer captures all dependencies used to form a control algorithm, such as the collection of function blocks that will be used, the pattern of those function blocks being connected, and computation parameters, etc. The power stage hardware abstraction layer encapsulates power stage

hardware specifications, such as interpretation of digital control information to analog control signals and analog feedback information to digital numbers.

Control applications (CAs) will work smoothly on the host hardware with the runtime environment support layer. The CA run-time support provides API to other layers and necessary real-time support for internal interactions within the control application. Using the network interface, the control software can communicate with other control intelligence in a distributed control environment. And through the host interface, control software can be easily migrated to different host hardware.

III. AN IMPLEMENTATION OF OPEN CONTROL ARCHITECTURE

As revealed by the analysis of real-time embedded control for medium- to high-power conversion systems, several considerations are important in the selection of a suitable software style to implement the proposed distributed, modular, open control architecture.

Firstly, in software design, it is desirable that under the applied software style, the expression and composition of control software can naturally fit the understanding of the problem to be solved. This will save the software designer tremendous time in mapping the problem domain into the software paradigm. The objective of digital control of power conversion systems is to control the power converter behaviors as designed by manipulating the input data through some math functions or logic functions in order to generate output signals. The chosen software style should be able to naturally map control schemes used in the power conversion application.

Secondly, because modularity is a main means to reduce the system construction effort in the proposed open control architecture, the applied software style should be able to support the implementation of certain functionality in the modules. Besides functionality encapsulation, the component style should also be able to separate functionality from its implementation, such that the pure internal modification of one module will not affect other modules in the system. This requires that the component style have the capability of good data abstraction and information hiding.

Thirdly, support for distributed computation is also an important merit, since modular and distributed power processing hardware is the trend. This requires that the control software be easy to decompose and allocate into a distributed computing environment. This also means that the interaction pattern should render components independent of each other so that reallocation of one component will not cause an overall redesign. Since parallel processing will improve the performance of control in a distributed computing environment, it is beneficial if the applied software style support parallel execution between components.

Lastly, because of the real-time nature of power conversion control, facilities must be provided for real-time scheduling

under the applied software style, and the resulting control software must maintain high execution efficiency.

Several candidate software styles are compared using the main considerations discussed above. The comparative results in Table 1 shows that the dataflow style might be a good candidate.

TABLE 1
COMPARISON OF SOFTWARE STYLES.

	Accurate and Easy Map to Control Algorithm	Components of Well-Defined Interface	Easy to Reallocate	Efficient Execution
Dataflow	+	+	+	+/-
Main-Program-and-Subroutine	+	-	-	-
Implicit Invocation	-	+	+	+/-
Layered System	-	+	-	-

Dataflow is a software design technique with a long history in computer science [8, 9, 10]. In the dataflow style, a control application is implemented as a set of concurrently executing processes. Dataflow processes communicate by sending messages through one-way message queues called data flows or channels. Each dataflow process is independent, and knows nothing about the other processes in the application—it merely consumes data from some channels and produces results on other channels.

Fig. 5 shows the dataflow architecture. In the dataflow implementation of embedded control for power conversion systems, a control algorithm is implemented as a set of concurrently executing processes, which we call elementary control objects (ECOs) [11]. ECOs communicate through one-way message queues called data channels. Each ECO is independent, and knows nothing about the other ECOs in the application—it merely consumes data from some channels and produces results on other channels. Dataflow computing is reminiscent of signal filtering and processing, and leads one to design ECOs that are modular and reusable. Constructing control applications then becomes the process of picking ECOs from a library and “plugging them together” into the desired pattern. ECOs protect independent functionalities that can be reused in different control applications. The dependency of the control algorithm is encapsulated in the dataflow graph, and can only affect that dataflow graph. In other words, the changes in the control algorithm, caused by either application requirements or the power stage topology, will be adapted in the dataflow graph. A driver ECO is especially written for the control software to communicate with a specific piece of controlled hardware. All the hardware-related information is hidden inside the driver ECO. Using a new piece of hardware will only cause the corresponding ECO driver to be substituted. The dataflow architecture real-time kernel (DARK) [12] supports lightweight process management, data channel management,

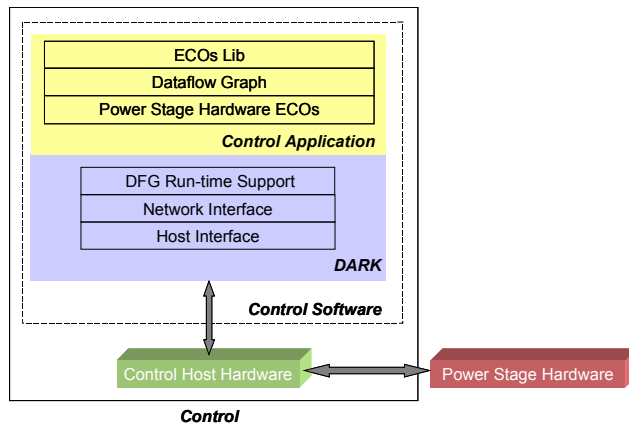


Fig. 5. Dataflow implementation of open control architecture.

system resource allocation, device driver support, and interrupt handling. DARK also provides a unified interface to control host hardware and a communication network to other control software.

Using the dataflow approach allows reusable and reconfigurable designs for control software that could “plug-and-play” in different power conversion applications and systems. Fig. 6 shows the dataflow graph of closed current loop control of a power electronics building blocks (PEBB) based three-phase inverter.

IV. EXPERIMENTAL ASSESSMENT OF DATAFLOW BASED OPEN CONTROL ARCHITECTURE

While the software engineering benefits of using a compositional, component-based architectural design are well known [13, 4, 14], the perceived overhead imposed by dataflow is a serious obstacle. In order to explore the feasibility of applying the dataflow approach to real-time power electronics control tasks, a preliminary study of the performance issues was undertaken. The preliminary evaluation to date has been based on the three-phase inverter control application described in Sections III. The PEBB-based inverter system is shown in The control software runs on an Analog 80MHz SHARC DSP (ADSP) 21160. The control application is designed both in dataflow style and

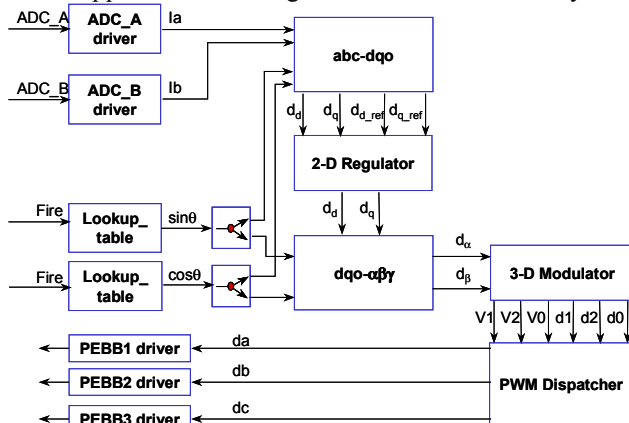


Fig. 6. Dataflow graph of current closed loop control for a three-phase inverter.

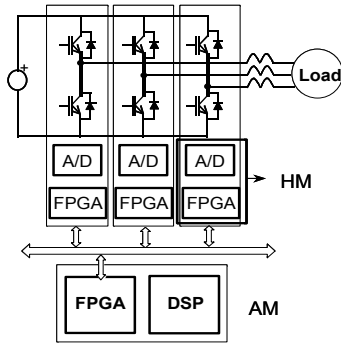


Fig. 7. A PEBB-based three-phase inverter.

main-program-and-subroutine style for the purpose of performance comparisons.

The specifications of the three-phase VSI are:

Input: $V_{dc} = 200$ V;

Outputs: balanced three-phase sinusoidal with line-to-line voltage of 100 V;

Switching frequency: $f_s = 10$ kHz;

Output inductance $L = 300$ μ H at each phase;

Output capacitance $C = 100$ μ F at each phase.

The voltage loop is designed to have a phase margin of 35 degrees and 10dB gain margin. To assess the feasibility of the dataflow approach, the code performances of dataflow software will be compared to those from the legacy custom-designed C code and from the Real-time Workshop Embedded Coder [15]. All implementations were written in C.

Fig. 8 shows the performance comparison between the three versions of the three-phase closed-loop control inverter application. The DARK solution was tested on the actual hardware to verify its correct operation.

From Fig. 8, the additional overhead imposed by the use of concurrent processes is evident. The instruction cycles devoted to “data channel operations” were those executed by the data channel read and write system calls made by ECOs in the dataflow design. The “ECO scheduling” instruction cycles were those devoted to determining which ECO was to be executed next. The “context switching” instruction cycles were those devoted to saving or restoring register contents when switching from the currently active process to the

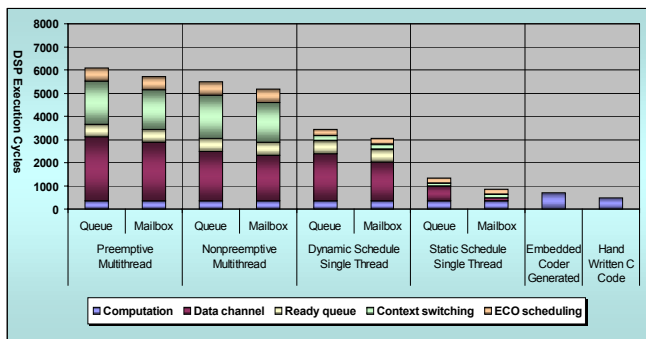


Fig. 8. Code performance comparison.

RTOS, or when switching from one active process to another.

The DARK used for this comparison employed streamlined choices for how to manage ECO processes in order to provide a picture of what may be feasible. It supported ECOs by providing non-preemptive scheduling and by supporting single-unit data channels (instead of multi-item queues). More comprehensive RTOS features, including preemptive task switching, full support for dynamic process priorities, and arbitrarily sized data channels, are all possible. The “mailbox” messaging method provides faster connections between ECOs, but with less protection against data loss due to different update frequencies of source and sink ECOs. The “queue” messaging method, on the other hand, is slower at passing data but provides a buffer between ECOs with different updating frequencies. When the ECO execution sequence can be predicted, static scheduling introduces the least performance overhead for context switching and process scheduling. If there are control events occurring at different frequencies, dynamic scheduling allows for more flexibility in control. The multi-thread allows better concurrent control in the system, and pre-emptive scheduling makes the control software capable of responding to the most critical events in the order of highest priority. These features will each introduce additional run-time overhead, however.

For the closed-loop, three-phase inverter control application, the DARK with the smallest footprint – mailbox and static single thread – is sufficient to provide full control to the power processing system. Fig. 8 shows that with this DARK feature, the performance of the dataflow software can be compared to the C code generated from MatLab® Real-time Workshop® Embedded C Coder, which is used in the ABB OPCoDe project.

From Fig. 8, it is clear that interprocess communication, in the form of data channel actions, dominated the overhead introduced by the ECO approach. Context switching and process scheduling were also important factors in the increased time required by the ECO application. Nevertheless, the application still ran within the limits necessary for proper performance.

Also, it is important to note that this experiment does not provide an accurate picture of the relative fraction of time spent on overhead issues versus the time spent in computation. That is because the computational aspects of this control algorithm are so small as to be negligible. This provides a much better picture of the absolute amount of overhead introduced, but does little to give a true impression of how this compares to a realistic computational burden.

The rewritten code size is also calculated to show to some extent the impact of software architecture and style on the complexity of software development. Fig. 9 shows the rewritten code sizes under main-program-and-subroutine and dataflow style, respectively, for the four applications presented previously in this chapter. Under the main-program-and-subroutine style, because every application is developed nearly from scratch, the rewritten code size is roughly the size of the control code itself. Under the

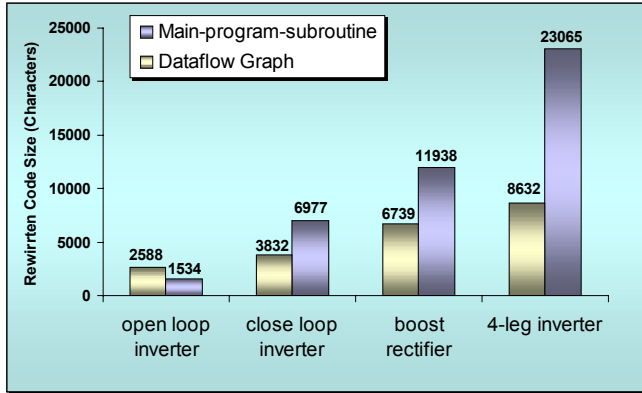


Fig. 9. Comparison of code size.

dataflow-based open architecture, the rewritten work corresponds to the implementation of the dataflow graph. For more complex applications, the dataflow-based open architecture potentially saves more engineering effort.

V. CONCLUSION AND FUTURE WORK

Due to close coupling to hardware and lack of standardization, the legacy proprietary approach to real-time embedded control software design in power conversion systems has a number of problems. After analyzing the real-time embedded control from various aspects – functionality, real time, and interactions with the real world – this research work has proposed an open control architecture to localize and encapsulate control software dependencies from other areas in power conversion systems in order to reduce the overall design time and cost of software development. This solution provides a better mechanism for managing complexity within the problem domain. Based on the exploration of mainstream software styles, the dataflow style is chosen as a full implementation of open control architecture. To assess the feasibility of dataflow-based control software in power conversion systems, four design applications are implemented in this style; these range from a relatively simple open loop, SPWM three-phase inverter to a fairly complex four-leg inverter. The levels of performance for dataflow software are compared to both handwritten main-program-and subroutine code and generated code from Simulink + Real-time Workshop.

Future work includes utilizing graphical control design and auto-generation environment, such as MatLab® SimuLink and Real-time Workshop to facilitate the control algorithm abstraction layer development. Future work also includes exploring the feasibility to design reconfigurable control software to support the “plug-and-play” concept in power conversion systems.

REFERENCES

[1] A. Schonung and H. Stemmler, “Static Frequency Changers with Subharmonic Control in Conjunction with Reversible Variable Speed AC Drives,” *Brown Boveri Rev.* 51, 1964, pp. 555-577.

[2] V. Blasko, “Analysis of a Hybrid PWM Based on Modified Space-Vector and Triangle-Comparison Methods,” *IEEE Transactions Industrial Applications* 33, 1997, pp.756-764.

[3] H. W. van der Broeck, H. C. Skudelny and G. Stanke, “Analysis and Realization of a Pulse Width Modulator Based on Voltage Space Vectors,” *IEEE Transactions on Industrial Applications*,24, 1998, pp. 142-150.

[4] D. M. Brod and D.W. Novotny, “Current Control of VSI-PWM Inverters,” *IEEE Transactions on Industrial Applications*, IA-21, 1985, pp. 562-570.

[5] I. Boldea and S.A. Nasar, “Electric Drives,” *CRC Press*, Boca Raton. FL, 1999.

[6] ABB OPCoDe website, <http://www.abb.com/>.

[7] Lewis, R. W. (Robert W.), “Programming industrial control systems using IEC 1131-3,” *London : Institution of Electrical Engineers*, 1998.

[8] A.L. Davis, and R. M. Keller, “Dataflow program graphs”, *IEEE Computer*, vol. 15, no.2, Feb, 1982, pp.26-41.

[9] D. E. Culler, “Dataflow architectures,” *Annual Review of Computer Science*, vol. 1, Annual Reviews Inc., Palo Alto, CA, 1986.

[10] B. S. Shuvra, P. K. Murthy, and E.A. Lee *Software synthesis from dataflow graphs*, Kluwer Academic Publishers, Boston, 1996.

[11] J. Guo, S. H. Edwards, and D. Boroyevich. “Elementary control objects: Toward a dataflow architecture for power electronics systems.” *IEEE. In Proceedings of the IEEE 33rd Annual Power Electronics Specialists Conference, PESC 02*, 2002, pp. 1705-1710.

[12] K. Singh and S. H. Edwards, ”DARK: Designing A High Performance Micro-kernel for Power Electronics Controllers,” *CPES Seminar*, Virginia Tech, Blacksburg, VA, 2002, pp. 362-367.

[13] A.L. Davis, and R. M. Keller, “Dataflow program graphs”, *IEEE Computer*, vol. 15, no.2, Feb, 1982, pp.26-41.

[14] B. S. Shuvra, P. K. Murthy, and E.A. Lee *Software synthesis from dataflow graphs*, Kluwer Academic Publishers, Boston, 1996.

[15] MathWorks Real-Time Workshop Embedded Coder web site, <http://www.mathworks.com/products/rtwembedded>.