

Improved Architecture of PEBB Plug and Play Power Electronics Systems: Elementary Control Object (ECO) and Dataflow

Jinghong Guo, Stephen Edwards, and Dushan Borojevic

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

Abstract—This paper presents a reconfigurable and reusable decentralized architecture, ECO—Dataflow architecture, for control of power electronics systems. Under this architecture, the control of a power electronics system is composed of elementary control objects (ECOs), and the connections between ECOs is implemented by Dataflow. Software implementations of ECOs and Dataflow will be discussed. A 3-phase inverter will be constructed under ECO—Dataflow architecture, and the system under the proposed architecture will be compared to the system designed in the legacy architecture.

I. INTRODUCTION

The traditional central approach of developing power electronics control systems has been significantly challenged to face rapidly changing requirements and environments [1]. As digital control introduced into power electronics systems, some researches have been done in modularization and standardization of power electronics systems. Power electronics building blocks (PEBB) is one of the early efforts to modularize power stage hardware [2]. Shortly after some converter systems were built on PEBBs successfully, how to modularize and standardize control of large-scale power electronics systems came into concern. General speaking, the control of a power electronics system has two main tasks. One is to conduct the converter control algorithm, which is application specific. The other is to interface with system hardware. Application manager (AM) and hardware manager (HM) concept reflects some initial considerations to develop decentralized power electronics systems [3]. At the same time, power electronics systems oriented high-speed communication network was investigated, such as PESNet, which can support distributed control in power electronics systems [4].

The goal of these researches is to provide a building block based plug and play capability for developing highly decentralized power electronics systems, built from standardized, co-operative and intelligent modules. One big issue comes out is how to develop the system architecture to reach this goal.

This paper presents a component-based system architecture, ECO—Dataflow architecture, to compose

modularized, standardized, reusable and automatic configurable power electronics systems. Section II of this paper shows how to divide the power electronics system control into elementary control objects (ECO) and how to connect ECOs by Dataflow to a complete control. The Software implementation of ECOs and Dataflow will be discussed in section III. In section IV, a 3-phase PEBB based inverter will be used as an example to show how a power electronics system is constructed under ECO—Dataflow architecture. And the ECO—Dataflow architecture will be compared with legacy approach of power electronics system design.

II. ECO—DATAFLOW ARCHITECTURE

By investigating control of different types of power electronics systems, it should be noticed that control of a power electronics system could be seen as composed of many basic functional blocks, such as regulator, modulator, etc. Thus we propose elementary control object (ECO) approach to modularize the control of power electronics systems. Dataflow, a type of software architecture, is introduced to organize ECOs into a run time controller.

A. Elementary Control Object (ECO)

The division of the power electronics system control into elementary objects is functional based. An ECO is defined as:

- Functional self-contained;
- Having standard interface;
- Independent;
- Concurrently executing;
- Implemented by multiple methods.

These objects should be commonly used and abstract. By analyzing control of different power electronics system applications, elementary control functions can be identified. Fig. 1 shows some candidate ECOs in power electronics control.

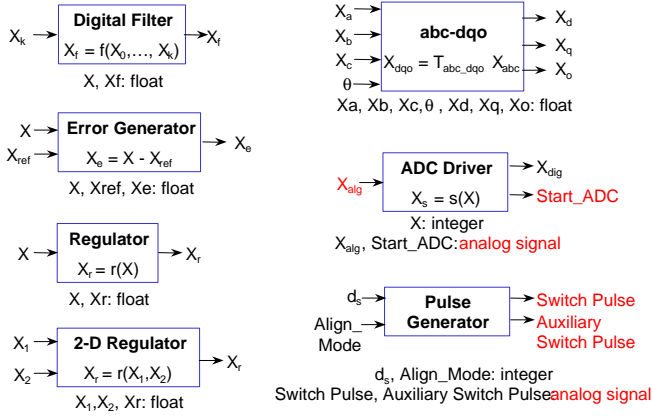


Fig. 1. Some ECO examples of power electronics systems.

Each ECO is designed to perform some dedicated function, such as digital filter, regulator, etc. If the function of an ECO is identified, the input and output to this ECO can be well defined. For example, in Fig. 1., what the ECO candidate Error Generator does is to generate difference between data X and its X_{ref} . The inputs of Error Generator is X and X_{ref} , and the output is X_e . From this simple example, it can be seen that ECOs are functionally abstract. For example, the Regulator ECO can be used as a current regulator, if the input is current information; it can also be used as a voltage regulator, if the input is voltage information. The ECO concept itself provides reusability at the component (or module) level. That means an ECO can be used in different applications without changing its interface and implementation, as long as the function is the same. By generalizing the data format of input and output, the interface of ECOs can be standardized.

What an ECO does is manipulating input and generating output according to its functionality. It is not necessary for one ECO to know where the inputs come from and where the outputs go. In another words, an ECO does not need to know what other ECOs sit on the up stream and the down stream, what the interfaces of other ECOs, and where the other ECOs exist. There is no direct relation between ECOs. They are naturally independent, so they are naturally able to execute concurrently. Thus distributed control of power electronics system is easy to build with ECOs.

The condition for an ECO to execute is called ECO firing rule. The general ECO firing rule is the ECO is ready to be activated once all its inputs are available. Inputs to an ECO may have different update rates, which may lead to different firing rules of an ECO in different applications. For example, for a Regulator ECO, the reference X_{ref} input may have lower update rate than the data input X . Once X is updated, the Regulator should be fired. So the firing rule under the above circumstance is modified to when X is available, the Regulator is able to be activated. But in some other application, the firing rule may be changed to check the availability of both X_{ref} and X . Practically it is possible that several ECOs are satisfied with their firing rules. Since ECOs are independent, they can naturally be executed concurrently.

However, there an important issue associated with concurrent executing of ECOs. For example, within a control of a power electronics system, a set ECOs serving for one control purpose needs synchronization. If more than ECOs in this set are under firing rules, the sequence of which ECO to activate is relevant. For another example, if only one ECO can be activate at a time, and several ECOs are under firing condition, which one should be activated. This is an ECO scheduling mechanism issue. Some of the scheduling mechanism will be discussed in the next subsection.

A merit of ECO is separating its implementation from its interface definition. Again take the Regulator ECO as an example. The Regulator can be a P regulator, or can be a PI regulator. But whatever method is used will not change the interface of the Regulator ECO. This also means, changing the implementation of function of an ECO has no impact of other ECOs, thus has not impact of the construction of the control system.

Next the data format of ECO interface is concerned. It is defined that data transferred between ECOs should digital, and data communicated between ECOs and environments can either be digital or analog. For example, in Fig. 1, Pulse Generator is defined as an ECO that translate duty cycle information into switching pulses to gate drives. The duty cycle information d_s and modulation method indicator $Align_Mode$ are digital data, while the switching plus to gate drives are analog. ECOs can be implemented in software, hardware or mix. An ECO implemented by software can work with an ECO implemented by hardware since the interface format has been standardized. This provides more flexibility of system development.

The purpose of ECO approach is to provide standardized and reusable building blocks to build configurable and reusable power electronics control system. However, only the ECOs are inadequate, unless there are flexible and efficient interconnection between ECOs and interaction between ECOs and environments. In the following subsection a system architecture will be proposed to organize, connect and manage ECOs to implement a complete control system.

B. Dataflow structure

Software engineering has the similar experience in design reusable software as that we attempt to build reusable power electronics systems. For small system, central and hierarchical approach is good enough. As software size and complexity dramatically increasing, object-oriented approach was proposed to modularize software. But later it is realized that object itself does not necessarily yield software architecture that can be easily adapt to changing requirements. The component-based system architecture tries to address these issues by clearly separating the stable parts of the system, i.e. the components, from the specification of their composition [5].

Dataflow [6] is such a software architecture, in which a system is composed of nodes and arcs, as shown in Fig. 2.

Nodes represent processes, which implement dedicated functions respectively. Arcs represent data connections between nodes. Once inputs to a node are all available, a node will be activated and generate outputs to push the data moving through the system.

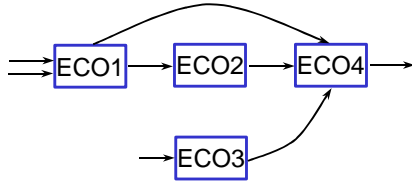


Fig. 2. Dataflow software architecture.

From the brief description of Dataflow above, it can be seen that the definition of ECO fits the Dataflow architecture. In this paper, Dataflow concept is extended beyond software architecture to the system architecture of power electronics systems. Combined, we call the ECO based Dataflow architecture ECO—Dataflow architecture.

Under the ECO—Dataflow architecture, the control algorithm of a power electronics system is defined as a combination of ECOs with data connections. Fig. 3 shows some examples of control algorithms, in which arrows represent data connections between ECOs. To simplify implementation of and synchronization between ECOs, each data channel is limited to have only one source ECO and one sink ECO.

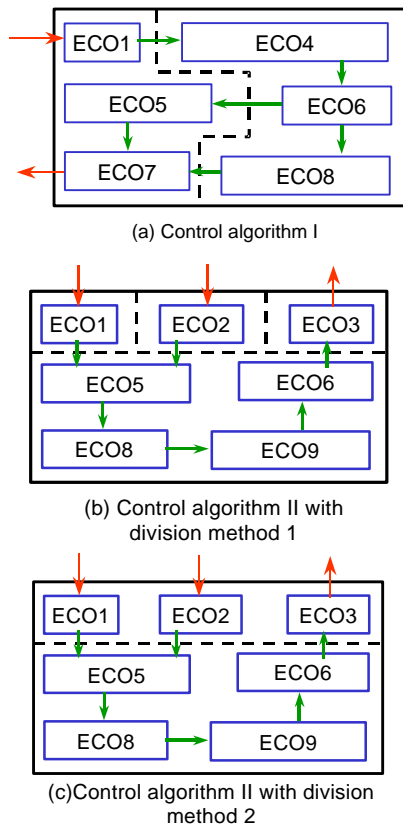


Fig. 3. Control algorithm examples under ECO—Dataflow architecture.

Different combinations of ECOs will implement control of different power electronics system applications. As shown in Fig. 3., there are two control algorithms composed of two set of ECOs. The control represented in Fig. 3. (a) is composed of ECO set {ECO1, ECO4, ECO5, ECO6, ECO7, ECO8}. ECO1 reads some input information from the system hardware, for example data from A/D converters, while ECO7 output some information to the system hardware, for example switching pulse. The control represented in Fig. 3. (b) is composed of ECO set { ECO1, ECO4, ECO5, ECO6, ECO7, ECO8}.

For a same control algorithm, different ways of control distribution can be achieved. In Fig. 3., dotted lines represent physical distribution of control. For example, the control represented by Fig. 3. (b) is distributed to several physical boards. Say ECO5, ECO6, ECO8 and ECO9 implement the control algorithm of a power electronics system, and they may sit together on a master controller board. ECO1, ECO2 and ECO3 interface with power stage, and each one of them is responsible for a PEBB phase leg, sitting on a local controller board respectively. Fig. 3. (c) shows a same control as of Fig. 3. (b) but different distribution. In this case, ECO1, ECO2 and ECO3, sitting on one board, are together responsible for the interface with power stage. Though in the two cases, the physical construction changes for ECO1, ECO2 and ECO3, the rest of the control can stay the same. This means the mater controller board is able to be adept to different system constructions.

In the AM—HM approach [3], how to divide the control into AM and HM and how to implement the interface between AM and HM are big issues. Under the ECO—Dataflow architecture, there is more flexibility to construct the system, as shown in the case analysis above. Also the interface will be naturally defined by the interfaces of ECOs. For example, in the construction shown in Fig. 3. (c), the interface between the two distributed parts are defined by interfaces of ECO1, ECO2, ECO3, ECO5 and ECO6.

Fig. 4. gives a piece of ECO—Dataflow model representing a DQ current regulator, which contains 3 ECOs, abc_dqo, Error Generator and 2-D Regulator. The Error Generator has two instantiations in this example, one is to generator error for i_d , and the other is to generate error for i_q . ECOs are interconnected by data channel.

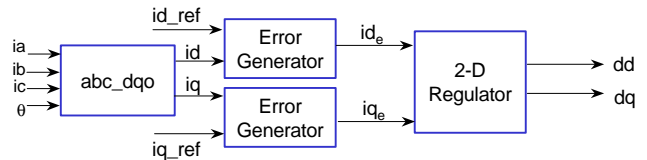


Fig. 4. ECO—Dataflow model of a DQ current regulator.

Once i_a , i_b , i_c and θ are all available, the abc-dqo is activated and generates i_d and i_q . The firing rule of i_d Error Generator is once i_d is available it will be activated. The same as of i_q Error Generator. Thus the two Error Generator ECOs are then activated and generates i_{de} and i_{qe} respectively, which

consequently causes 2-D Regulator to be activated. Finally d_d and d_q are generated. This simple example shows how the data is pushed through the ECO—Dataflow architecture.

ECOs are plug-compatible components in the ECO—Dataflow architecture. Dataflow provides a component framework so that ECOs can run independently and concurrently.

III. SOFTWARE IMPLEMENTATION OF ECO—DATAFLOW ARCHITECTURE

In section II, it has been mentioned that ECOs and Dataflow can be implemented in software, hardware or mix. In this section, the software implementation of ECO—Dataflow architecture is discussed, because in most case of digital control of power electronics systems, the main part of the control is implemented by software.

In software, ECOs can be implemented as light-weighted processes, which consume input data and produce output data. Each arc in the Dataflow can be implemented as a software data channel, which interconnects a pair of ECOs. To make the ECO—Dataflow working properly, there must be some management software to schedule and monitor execution of all the ECO process.

ECOs are reusable components in the ECO—Dataflow architecture, which take the computation part of the control. The data channels are interconnections between ECOs. The ECO management software is the compositional part of the architecture. Then how can a configurable and reusable power electronics system can be developed from these pieces of components?

Fig. 5 shows how to build ECO—Dataflow model of a control algorithm. The ECO management software at system start up time is a control generator, and at the system running time is a run-time supervisor. The roles of these two management software will be introduced next.

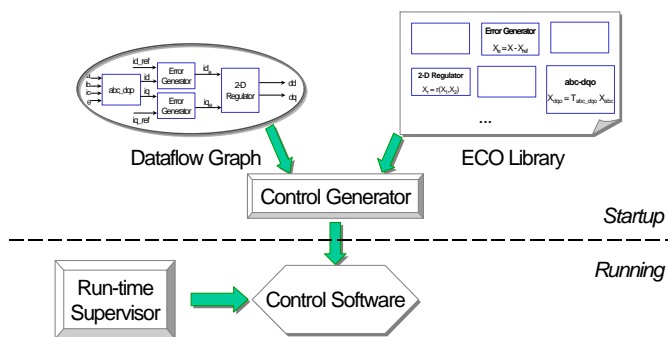


Fig. 5. ECO—Dataflow programming.

The control algorithm is described in Dataflow graph, represented in some tables for example. The Dataflow graph describes ECOs that will be used to compose the control algorithm and data channels connecting ECOs. In the Dataflow graph, the interface features, such as input, output

number, firing rule, etc, of each ECO used are described. Initialization information of each ECO is also described. The description of Dataflow graph is served as a system script that contains all the application specific information and requirements.

The ECO library stores the operation code for each ECO that may be used.

At the system startup time, the Dataflow graph will be fed into the control generator. According to the script, the control generator knows that which ECOs should be chosen to compose the control software. From the ECO library, it can get the operation code for each ECO. Then from the Dataflow graph, the control generator builds up data channels, which will connect ECOs. Finally, each ECO will be built into a process with its own process data, which contains information of the firing rule, input and output data channel pointers, and parameters. After an ECO process is created, the process will be initialized once immediately. Thus the run-time control software is generated.

And during the system running time, a run-time supervisor manages and monitors executing of ECO processes. It checks firing condition of each ECO. If the firing rule of some ECO is satisfied, the ECO process will be added to a queue, which contains all the ready-to-run ECO processes. According to some scheduling mechanism, one of the ECOs in the ready queue will be activated. After execution, the ECO process will update the data channels connected to it outputs, and then will be removed from the ready-to-run process queue.

If the hardware construction of the system does not change, new control can be set up simply by feed new Dataflow graph to the control generator. That means the computation part of the architecture is changed according to the changing requirements, but the compositional part keeps the same. From the user point of view, the effort of developing a power electronics system is reduced to create a Dataflow graph of the application.

IV. ECO—DATAFLOW DESIGN EXAMPLE: A THREE-PHASE PEBB BASED INVERTER

In this section, a three-phase PEBB based inverter will be used as an example to show how to design a converter system under ECO—Dataflow architecture. Fig. 6 shows the system architecture of the 3-phase PEBB based inverter, with the control implemented by ECO—Dataflow model. Fig. 7 shows the hardware setup of the inverter.

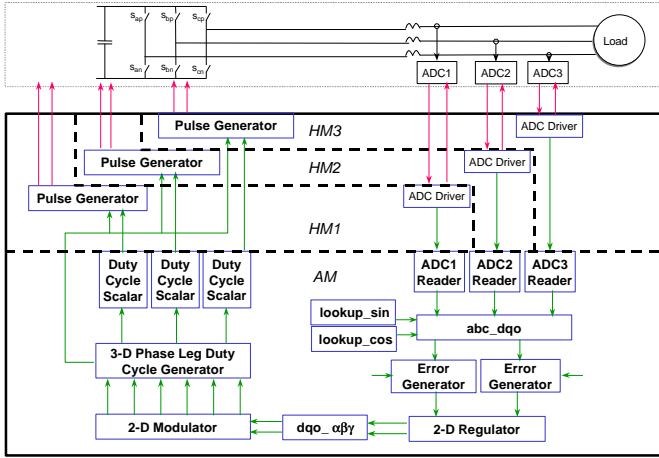


Fig. 6. ECO-Dataflow architecture of a 3-phase inverter system.

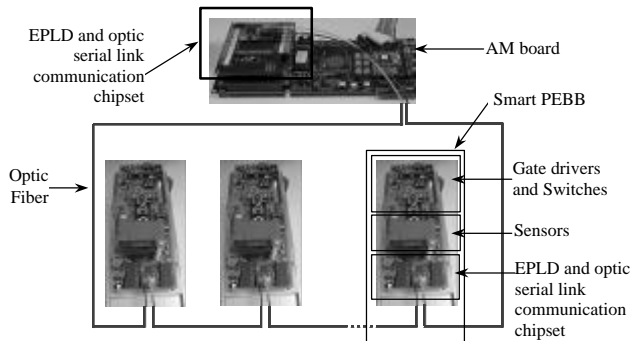


Fig. 7. PEBB based 3-phase inverter.

The control of the system is divided and distributed. The control algorithm is executed on a controller board. ECOs composing the control algorithm are all implemented by software. The ADC drivers and gate drive switching pulse generator for each phase leg are implemented by mixed software and hardware and integrated with a PEBB.

Compared to the traditional design method of power electronics systems, the ECO—Dataflow architecture has many advantages, such as modularization and standardization, automatic system configuration, reusability and flexibility. Instead of scratching everything from the begging in the legacy design of power electronics systems, the design effort is reduced to choose ECOs and set up connection between ECOs in the proposed ECO—Dataflow architecture.

The side effect of this architecture is overhead, including more system resource usage and running time, because of the interconnections between ECOs and ECO scheduling. But as the on design next generation power electronics universal controller [7] is coming to the stage, which has powerful computation capability and multi-type and fast communication interfaces, the overhead problem of the ECO—Dataflow could be overcome.

V. CONCLUSION

This paper proposes ECO—Dataflow architecture, which is aim to design modularized, standardized, automatic configurable and reusable control for power electronics systems. How to divide a control into ECOs and how to connect ECOs by Dataflow are discussed. Software implementation of ECO—Dataflow is presented. And a 3-phase PEBB based inverter system is design under the proposed architecture.

There are more researches need to do further. Power electronics systems are real-time systems. It is important to design timing control mechanism under the ECO—Dataflow architecture. How to design an development environment of ECO—Dataflow architecture is also need to be considered.

REFERENCES

- [1] I. Celanovic, "A distributed digital control architecture for power electronics systems," *Thesis, Virginia Polytechnic Institute and State University*, Blacksburg, VA; 2000.
- [2] T. Ericson and A. Tucker, "Power electronics building blocks and potential power modulator applications," *IEEE Conference Record of the Twenty-Third International Power Modulator Symposium*, New York, NY, pp. p.12-15; 1998.
- [3] J. Guo, D. Borojevic and I. Celanovic, "Software structure of the PEBB-based Plug and Play power electronics systems," *Sixteenth IEEE Applied Power Electronics Conference and Exposition*, Anaheim, Ca, in publishing; 2000.
- [4] I. Milosavljevic, D. Borojevic and I. Celanovic, "Modularized Communication and Control Structure for Power Converters," *8th European Conference on Power Electronics and Applications, EPE*, September 1999.
- [5] L.Barroca, J. Hall and P. Hall (eds.), "Software architecture: advances and applications," *Springer*; 2000.
- [6] D Garlan and M. Shaw, "An introduction to software architecture," *Advancs In Software Engineering and Knowledge Engineering*, vol. I, World Scientific Publishing Company, New Jersey; 1993.
- [7] J. Francis and D. Borojevic, "Design of a Universal Controller for Distributed Control and Power Electronics Applications," *CPES Seminar*, 2001.