FREEDM Software Controller Architecture for a Solid State Transformer *

Balasubramanya Bhat
North Carolina State University
bbhat@ncsu.edu

Frank Mueller
North Carolina State University
mueller@cs.ncsu.edu

Abstract

The Future Renewable Electric Energy Delivery and Management (FREEDM) project aims at providing an efficient electric power grid integrating alternative generating sources and storage with existing power systems to facilitate a green energy in a highly distributed and scalable manner. One of the central aspects of the Reliable and Secure Communication (RSC) subthrust within the FREEDM system is the framework that controls the Distributed Renewable Energy Resource / Distributed Energy Storage Device (DRER/DESD) loads at 120 V and the 12 KV distribution bus. The hardware component that governs power along this bus is the Solid State Transformer (SST) controller. The objective of this paper is to introduce the computational design and describe the software architecture of the SST controller board. This paper details the status of the current software development and discusses future work in this realm.

1. Introduction

In the FREEDM system, the SST Controller enables Intelligent Energy Management (IEM) and Intelligent Fault Management (IFM) functionality with the ability to control DRER/DESD loads (at 120 V) and the 12 KV distribution bus. The inputs to this system are 60 Hz analog power signals originating from power inverters, which are first converted into digital form before being processed further. The goal of our work is to develop a scalable and secure software architecture that runs on the SST controller board under real-time signal processing requirements with regard to its power signal control. The software design shall be portable to different hardware platforms with wide software reuse.

2. System Architecture

Figure 1 depicts the principle components of the FREEDM system and their interaction with each other. Our focus is on the task of closed loop digital control within the SST subsystem. The SST controller is realized on a TI compute platform (the TI TMS320C6713), a relatively high-end embedded digital signal processor (DSP) processor with a floating point unit. This DSP comprises the heart of the control system. The controller board is capable of communicating with the Distributed Grid Intelligence (DGI) system of FREEDM using either USB 2.0 high speed or 10/100 Mbps Ethernet connections. The Valve Based Electronic (VBE) board provides an interface that converts the signals from the electrical domain to a reliable fiber optic channel. This optical connection also isolates low voltage from high voltage control components. The solid state relay board (SSRB) provides interfaces to the solid state relays that form an integral part of power electronic systems. The DGI system will ultimately host distributed control algorithms and provide interfaces to the outside world.

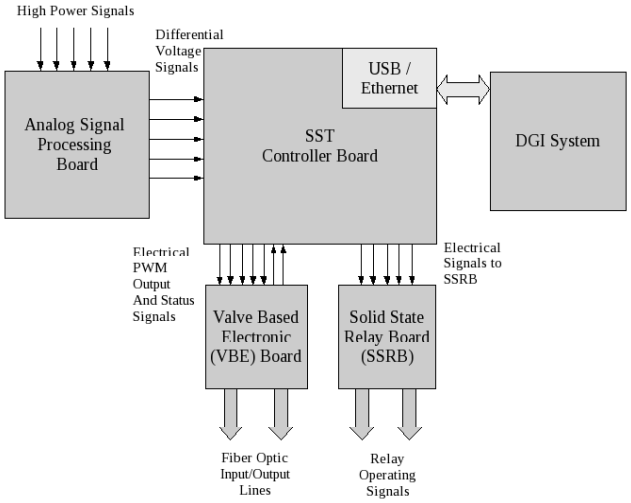

Figure 1. System Architecture

3. Hardware Architecture of the SST Controller

Figure 2 shows how different components are interconnected in with the SST controller board. The TI TMS320C6713B DSP runs at 150 MHz and constitutes the core platform for the power controller software. Most other SST system components are connected as peripherals to this DSP.

The customized board further hosts an Altera EP2C35 Cyclone II FPGA device with 33 K logical elements. This FPGA chip is deployed with a synthesized NIOS II soft core [1] and an extension that provides the USB/Ethernet communication software and PWM (Pulse Width Modulation) functionality.

4. Software Architecture

One of the main goals of the software architecture is to be able to satisfy tight real-time requirements of power signal processing. Also, it shall be portable to different platforms with minimal amount of redesign. Figure 3 shows the overall software architecture on the SST controller board.

4.1 DSP Software Architecture

The software components running on the DSP are:

* This work was supported in part by NSF grant EEC-0812121 and U.S. Army Research Office (ARO) grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI).

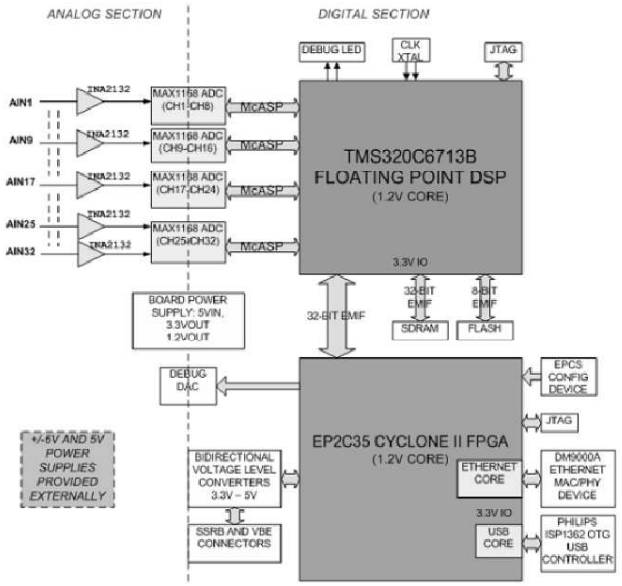

Figure 2. Hardware Architecture of the SST Controller

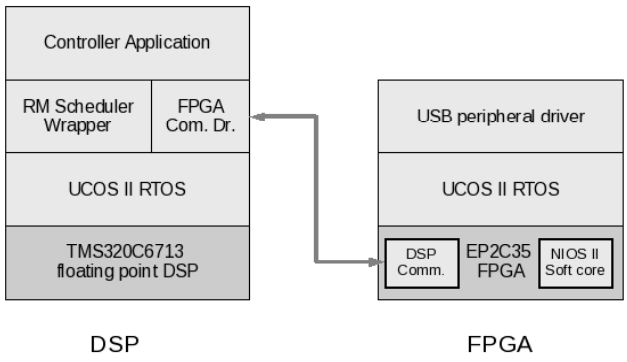

Figure 3. Software Architecture of the SST Controller

- DSP Boot Software,
- Real-time Operating System,
- Periodic Task Scheduler,
- FPGA Communication Driver, and
- SST Controller Application.

DSP Boot Software: We developed a boot software that resides at the reset vector in flash memory. When the SST controller board is powered up, the TMS320C6713 processor starts running the boot software from the flash. This software mainly performs the following functions:

- initialize the processor,
- initialize the memory controller,
- perform basic memory test (SDRAM),
- validate the SST controller application binary in the flash using CRC,
- copy the SST controller application from flash to SDRAM, and
- transfer the control to the SST controller application in the SDRAM.

One of the possible future changes is to completely bypass the boot software and directly start running the SST controller application from flash upon reset. The application can perform all the necessary initializations, copy itself from flash to SDRAM and continue running from the SDRAM. This avoids the need for a separate boot application.

Real-time Operating System: The software architecture for the main DSP software consists of a μC/OS II [2] real-time operating system (RTOS), a publicly available kernel, at the lowest layer. We ported this RTOS to run natively on the TI TMS320C6713 processor of the SST Controller board. This kernel provides basic task switching and synchronization primitives. The μC/OS II RTOS provides simple priority-based preemptive scheduling.

Periodic Task Scheduler: Most of the processing requirements on the SST Controller board involve dealing with periodic tasks of different periods. The μC/OS II kernel supports only an aperiodic task model. In order to efficiently support aperiodic tasks, we developed a rate monotone (RM) scheduler [3] on top on the μC/OS II kernel. This layer is easily portable to any other RTOS or hardware platform with support for aperiodic tasks. This RM scheduler supports:

- periodic tasks, where all parameters of the periodic task model (period / phase / execution time) can be specified;
- 1 μsec resolution for all timing parameters;
- strict execution time / deadline control for every periodic task;
- tracking of task CPU utilization and total system utilization;
- tracking of missed deadlines and execution times;
- providing a sleep() method with a 1 μsec resolution to suspend the task for arbitrary amount of time; and
- additional support for real-time security mechanisms.

The applications should use only the APIs provided by an RM scheduler and shall not directly call the underlying μC/OS II APIs.

FPGA Communication Driver: The FPGA communication driver is responsible for exchanging information with the NIOS II soft processor (configured within the FPGA) using the 16 bit interface provided on the SST controller board. This is mainly used for exchanging the information with the outside world over USB connected to the FPGA chip. Communication with other parts of the FPGA (for example, PWMs) bypasses this driver.

SST Controller Application: The SST controller application is the main application software deployed on the SST controller board that is responsible for controlling the solid state transformers. This software will be deployed in future work.

4.2 FPGA Configuration

The EP2C35 Cyclone II FPGA is configured with an embedded 32-bit processor, the NIOS II soft processor provided by Altera [1]. The SOPC (System On a Programmable Chip) builder tool that depends on the Quartus II compiler provided by Altera for code synthesis on its FPGA devices, is used to generate the system that is deployed on this FPGA. A set of modules provides the DSP interface, PWM generation, custom input/output (I/O) logic and the digital-analog converter (DAC) interface, all of that are coded in the Verilog hardware description language (VHDL). The block shown as SOPC Builder Block is created by the SOPC builder tool that hosts the NIOS II soft processor. Figure 4 depicts how the FPGA is configured on the SST controller board.

One module implements the address decoder for the DSP EMIF (External Memory Interface) lines. It accepts inputs from the EMIF

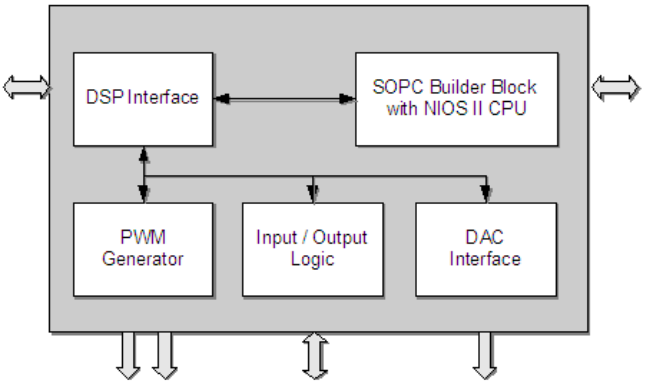

Figure 4. FPGA Configuration

address and control lines. Depending on the FPGA blocks being addressed, corresponding internal blocks (such as the PWM module) are enabled for further functionality. At the same time, this module also provides and interface with custom I/O logic so that the DSP can read the digital information from the FPGA if required. The PWM generator module accepts commands from the DSP interface block and generates PWM signal waveforms on the FPGA I/O pins leading to the VBE. The I/O logic block implements fast acting custom digital I/O logic required for fault protection and diagnosis. The DAC interface is implemented in a customized fashion. As this interface uses the SPI (Serial Peripheral Interface) protocol for communication, it can be moved into the SOPC builder section. The SOPC builder block is generated by the SOPC builder tool. Verilog code can thus be generated with a DSP interface block connected to an instance of this module.

SOPC Builder Block: Figure 5 shows the SOPC builder block configuration in more detail. This configuration has a NIOS II soft processor at the core running at 50MHz connected to the Altera Avalon shared bus. It also has a 40KB on-chip memory connected to the Avalon bus where the entire software code and data will be located. The ISP1362 interface logic provides connectivity to the NXP ISP1362 USB peripheral controller chip. The JTAG UART provides connectivity with the NIOS II IDE for debugging. There are two on-chip PLLs both with input from a 25MHz clock connected to one of the input pins of the FPGA. One of the PLLs produces a 50MHz clock signal that drives the NIOS II processor. The other produces a 12MHz clock signal that drives the ISP1362 USB chip.

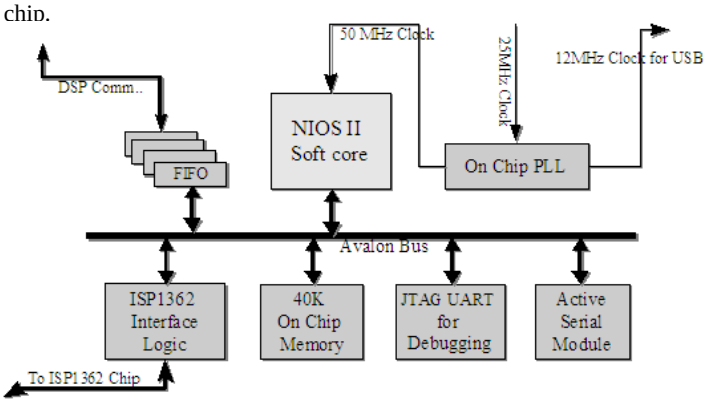

Figure 5. SOPC Builder Block

The FPGA is not directly connected to any non-volatile memory except for the serial configuration device EPCS16. The EPCS16 chip is a 16Mbit serial device used to store the FPGA configuration loaded onto the FPGA upon startup. We use a small portion of the unused space within this chip to store our NIOS II software as well. The Active Serial programming unit in the SOPC builder block automatically loads this software from the EPCS16 chip onto the FPGA on-chip memory. Once the program gets loaded in the memory, it is ready to run.

DSP/FPGA communication: The FPGA is connected to the DSP using the 16 bit EMIF interface. The block shown as FIFO memory in the SOPC builder block (Figure 5) is used for communicating with the DSP. There will be two 128 byte FIFO memory blocks, one used for input and one for output. The NIOS II processor puts the data that it wants to send in the output buffer that is read in FIFO manner. Similarly the DSP puts its output data in the input buffer of FPGA that is read by the NIOS II. Both DSP and NIOS II periodically process read/write data from/to these input/output buffers.

In the future, we plan to enhance the communication between DSP and FPGA via DMA.

4.3 FPGA Software Architecture

As shown in Figure 4, a NIOS II soft core is synthesized on the FPGA. Figure 3 depicts software running on this processor. The μC/OS II real-time kernel is ported to run on the NIOS II soft core. A peripheral controller driver for ISP1362 USB 2.0 device will be running on this processor which provides access to the outside world. This driver reads / writes data to be transmitted / received from / to FIFO buffers, all of which are accessible by the DSP as well. If required, we can host a light-weight TCP/IP stack capable of communicating over 10/100 Mbps Ethernet using the DM9000A chip. Altera provides a NIOS II IDE for developing the software on the NIOS processor.

5. Current and Future Work

We have completed the following software development tasks for the SST controller board:

- We developed the DSP boot software that loads the application program from flash to SDRAM.
- The μC/OS II real-time kernel is ported onto both DSP and NIOS II soft processors.
- We implemented a rate monotone (RM) scheduler on top of the μC/OS II real-time kernel to schedule periodic tasks.
- We implemented a FGPA communication driver on the DSP as one periodic task.
- We configured the FPGA for deployment of the NIOS II soft core.
- We implemented a peripheral controller driver for ISP1362 USB 2.0 device on the FPGA NIOS II Processor that is configurable to support the required bandwidth. This has less than a 20KB footprint without μC/OS II as the total memory available is 40KB.
- We also developed a Windows driver for communicating with the SST controller board supporting duplex communication between the SST controller and a PC.
- We have further developed a socket server on a PC that provides network connectivity to the SST controller board over USB 2.0.

Future work:

- We need to provide SDRAM connectivity with FPGA as the current 40KB limit of the on-chip memory is a constraint for software development on the NIOS II processor.

- We have developed an EDF kernel that can reduce the context switch time for periodic tasks by removing unnecessary context stores and restores. We may consider replacing the RM scheduler with this one.
- We plan to deploy a SST controller application for power control.

6. Conclusion

This paper introduces the SST controller and explains its role within the FREEDM system. It details the hardware architecture of this controller board and describes the overall design of the controller software deployed on this board. The DSP software consists of a custom rate monotone scheduler built on top of μC/OS II RTOS. The FPGA configuration and the design of the software (soft core and extensions) developed on this FPGA are further motivated and detailed. The SST controller has a USB 2.0 connection that is utilized for communicating with the external DGI system. Finally, the paper also gives an overview of the progress of the current software development and forthcoming future work on this board.

References

[1] Altera. Nios ii processor: The world's most versatile embedded processor.

[2] J. Labrosse. *Microc/OS-II*. R & D Books, 1998.

[3] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. of the Association for Computing Machinery*, 20(1):46–61, Jan. 1973.