Reprint

A Multiprocessor Architecture for Intelligent Control of Standby Electrical Power Generating Systems

N. Papandreou, A. Leventis, D. Anastasiadou and T. Antonakopoulos

The International Conference on Power and Energy Systems - Euro-PES 2001

RHODES, JUNE 2001

A MULTIPROCESSOR ARCHITECTURE FOR INTELLIGENT CONTROL OF STANDBY ELECTRICAL POWER GENERATING SYSTEMS

N. PAPANDREOU, A. LEVENTIS, D. ANASTASIADOU AND T. ANTONAKOPOULOS
Industrial Systems Institute - ISI
Department of Electrical Engineering and Computers Technology,
University of Patras, 26500 Patras, Greece
Tel: +30 (61) 997346, Fax: +30 (61) 997342 Email: antonako@ee.upatras.gr

ABSTRACT

This paper presents a multiprocessor control system for standby power supply applications. The system is responsible to initiate a standby generator in order to provide power supply to a target load in case of mains supply failure. Typical applications are standby generators at bank offices, hospital buildings, airport premises and mobile telephony base stations. The system possesses a great concurrency of individual processes and handles a large number of input and output signals. The application requirements introduce specific constraints upon system operation. The proposed design is based on a multiprocessor architecture that incorporates two DSPs, two microcontrollers and programmable logic. The paper focuses on the functional decomposition of the overall system into individual modules and their efficient mapping into a system-level architecture that guarantees the critical requirements of the application.

KEY WORDS: Power System Planning and Control, Real-Time Control Systems, Real-Time Design Methodology, Multiprocessor Design Architecture.

1. INTRODUCTION

In many industrial control applications, system functionality is primarily determined by its interactions with the working environment. Real-time systems have to incorporate run-time mechanisms that guarantee upper bounds for the maximum execution time of tasks and for maximum duration of inter-process communication protocols [1]. Control of a standby generator that provides power supply to a target load imposes accurate measuring and fast response, otherwise a catastrophic situation may result.

A Generator Control System (GCS) provides three-phase voltage metering and supervision for the mains power supply. Exact voltage and frequency limits determine a main grid failure and initiate the generator according to an automatic process. The GCS controls and supervises the generator performance and takes critical actions in case of malfunction or emergency.

The system functionality consists of independent concurrent processes that control a great number of input and output signals. Due to system complexity, its overall functionality had to be decomposed into independent modules. In the presented system, a multiprocessor architecture was developed for the integration of the functional decomposition. Each module was partitioned among various system components and communication protocols providing accurate inter-module message delivery. This paper focuses on the architecture design aspects of such a multiprocessor GCS system.

In Section 2 we describe the general working environment and the interfaces of the GCS unit and introduce the critical features of the target application. In Section 3 we present our design methodology approach and give emphasis to the decomposition of the system functionality into independent modules. In Section 4 we give a description of the multiprocessor system architecture. Based on the decomposition model we describe the mapping of the functional specification to the component-level architecture. Finally we summarize our design analysis in Section 5.

2. GCS ENVIRONMENT INTERFACE

The GCS environment interface is shown in Figure 1. Analog conditioning circuitries transform the power system voltages and currents to biased signals and provide galvanic isolation between the power system and the digital subsystem. The later performs all metering and processing computations and forwards the results to an LCD display. Upper and lower voltage and frequency limits determine the condition of the main grid. When the main grid provides power that lies in the desired range, the GCS connects the target load to the main power grid. In case of 'out of range' conditions or total power failure, the GCS disconnects the load and initiates the generator to provide power supply. Generator performance is constantly supervised by the GCS. Indications of

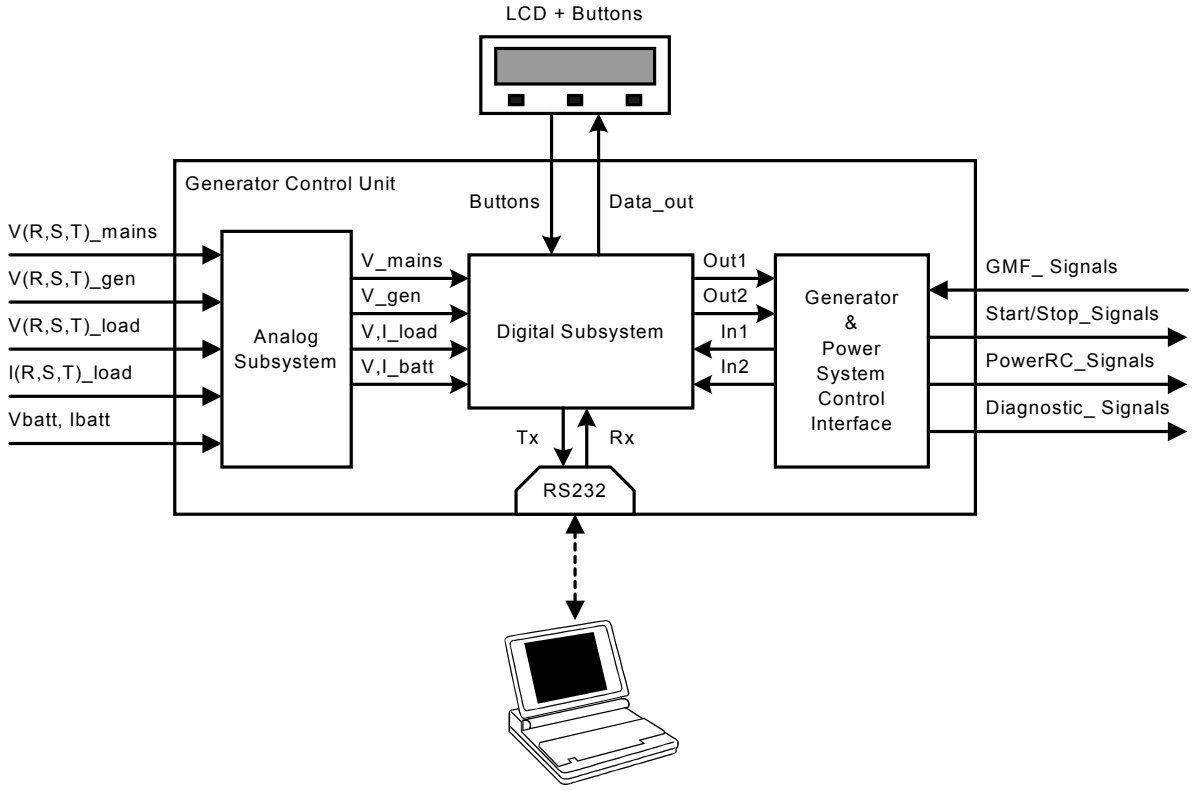

Figure 1. The GCS Environment Interface.

malfunction (GMF) or emergency are rapidly recognized and resolved by the system with simultaneous activation of diagnostic signals. One of the most attractive features of the system is its ability to maintain a real-time database where critical information is recorded. A host PC can be connected to the GCS and monitor the system performance including metering results, mains and generator status, history malfunction indications. A security mechanism enables an authorized user to monitor and modify operational parameters, such as upper and lower voltage and frequency limits.

3. SYSTEM DESIGN APPROACH

The design and implementation of a real-time embedded-system involves a thorough and consistent analysis procedure. The target application introduces specific constraints upon system response. Several design methodologies have been proposed in literature, each suited to the characteristics of the described system and application.

The presented GCS system is characterized by a large number of individual functions that require reliable and transparent implementation, a great deal of input and output signals of various voltage characteristics that require efficient handling and stringent implementation constraints. In order to minimize design inadequacy and avoid unnecessary iterations, we utilized a hierarchical approach [2] that involved the following tasks:

Specification capture: This first task offered a comprehensive understanding of the target application as well as the system functionality. In order to deal with the system complexity we decomposed it into smaller modules (units) that could be treated independently. Each module was assigned only a part of the general functionality, enabling a detailed investigation of the system from the early beginning of the design process.

Exploration: Given the functional system specifications, we were ready to explore design alternatives that would best satisfy our constraints. In order to map the conceptual model into a system-level architecture, we partitioned the functional specification among different system components and estimated each design alternative according to the following criteria: Low cost and rapid prototyping, both of which restricted our choice of component solutions, expansion capability for prospective addition of new services, design simplicity regarding development, implementation, testing and debugging.

Specification Refinement: After concluding into a system-level architecture we refined the initial

specification and generated a description that reflects the implementation decision made during exploration. This task involved the estimation of memory requirements, analysis of interface protocols between components, design of arbiters to support concurrent accesses to a single resource and time scheduling of the embedded program logical functions. The result of the specification refinement was a system-level description detailing the system processors, memories, buses and programmable logic.

Software and Hardware design: This step involved the implementation of each component using well-known software and hardware techniques. Processor programming was developed through software synthesis satisfying resource and performance constraints. We used mixed C and assembly code to compensate for the inability of C to produce compact machine code for real-time signal processing functions. Programmable logic was developed through standard high-level (behavioral) synthesis that converts the behavioral description to RT components.

Physical design: This task involved the actual programming and debugging procedure for all system components. We followed a hierarchical process in order to minimize debugging effort. We first implemented each module component individually and then merged all modules into a complete system. During the debugging procedure we had to make software and hardware rearrangements, which led to optimized assembly code and RT netlist.

3.1. FUNCTIONAL DECOMPOSITION

Figure 2 depicts the system model decomposed into five independent modules:

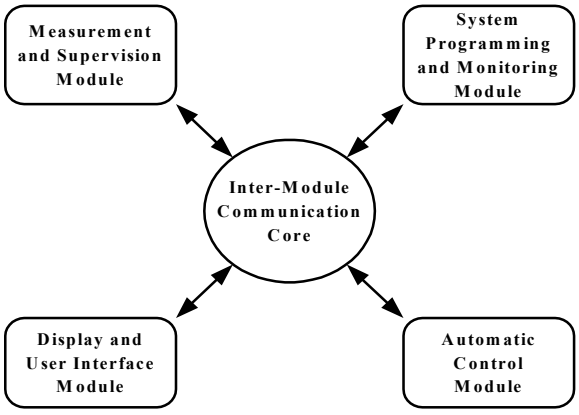

Figure 2. Decomposition of the system functional specification

Measurement and Supervision Module (MSU): This module provides three-phase metering for the power system. It meters the three phases voltage, currents and frequency. Synchronization to the power system is maintained by a zero-crossing detection approach [3], [4]. In addition to reporting the rms value of each of the three voltages and the three currents monitored, the module also meters the three line-to-line voltages, the total active power, as well as the total power factor. All measurements are performed independently for the main power grid, the generator and the load. MSU module also provides DC voltage and current metering for the generator battery supply. The performance of all power modules is continuously supervised according to programmable voltage and frequency limits. The results are directly reported to the automatic control module in order to take the proper actions

Automatic Control Module (ACO): This module is responsible for running an automation process that controls the generator standby according to the power grid state. The main power system always has priority in supplying the load against generator. The automation mechanism was implemented as collaborating state machines that determine the generator's state of operation. Data from the supervision unit define the state transitions. The module controls the generator start/stop signals, input signals indicating mechanical malfunction, output diagnostic signals and the system power-relays.

System Programming and Monitoring Module (SPM): The automatic control process includes several mechanisms that have specific timing features. In addition, the supervision module (MSU) indications depend upon specific upper and lower voltage and frequency limits, as well as timing constraints for determining that a signal is 'out of range'. These operational parameters are often subject to change according to specific application needs or the applications characteristics, e.g. the main power grid behavior differs from area to area. The SPM module provides the user with a simple and transparent mechanism for real-time parameter programming and updating. The module also maintains a real time database for keeping track of the system behavior. All indications of malfunction, as well as critical system changes are recorded along with the exact occurrence time. Thus, the system functionality can be monitored by examining the system history information contained in this database.

Display and User Interface Module (DUI): This module is responsible for providing a friendly interface, enabling the user to examine all metering units and operational parameters. We define two modes of DUI operation, the *metering display mode* and the *programming mode*. In metering display mode, all measurements are continuously displayed in an LCD unit. Entering the programming mode the authorized user is allowed to examine the system current operational parameters and also modify them using a small keypad. The module was also developed to facilitate measured

data monitoring and parameter programming using a host computer. DUI was also designed to support remote monitoring through a modem connection, thus enabling dependable evaluation and control of systems located at remote areas.

Inter-Module Communication Core (IMC): The IMC core provides communication between the previously described independent modules. IMC incorporates several communication mechanisms that are realized by well-known microprocessor protocols. Due to the critical nature of inter-module information, IMC guarantees safe and accurate message delivery. The module also incorporates fast resolved arbitration mechanisms to support concurrent access to single resources by independent modules

4. HARDWARE ARCHITECTURE

The GCS was implemented using all-digital technologies as shown in Figure 3. The functional modules of Section 3.1 were allocated to a number of DSP processors and microcontrollers, supported by peripheral devices such as memories, a real time clock and programmable logic. The specifications and limitations of the design obligated the fragmentation of some modules (MSU, ACO) and their mapping to more than one physical device. Since these modules implement hard real time tasks, the use of a hard real-time interface protocol for the interconnection of the respective devices was mandatory. In addition, as more than one processor may access each peripheral device, access control mechanisms incorporating arbitration protocols were implemented in programmable logic.

The GCS was built around two DSP processors *(processor 1, 2)* that realize both real time and non-real time tasks. The real time tasks implement the measurement module and the automation-control process, while the non-real time ones perform basic functions such as measurements handling and system parameters update. The management of these tasks is performed by two microcontrollers *(processor 3, 4)* that act supplementary to the DSPs. Processor 3 implements the local and remote user interface, manages the programming procedure and supports the serial interface, while Processor 4 maintains the system operation mode and performs elementary signal handling. The automation mechanism is user programmable, thus an EEPROM chip *(Memory 1)* was used for the storage of user-defined values. A Flash EPROM *(Memory 2)* was used to record any abnormal situation detected, together with a timestamp provided by the Real Time Clock chip *(RTC)*. Since more than one processor may need access to each memory, an arbiter was implemented in programmable logic *(PrLogic 2)* to share the memory resources among the various processors. Finally, another programmable logic *(PrLogic 1)* was used for preprocessing the I/O signals.

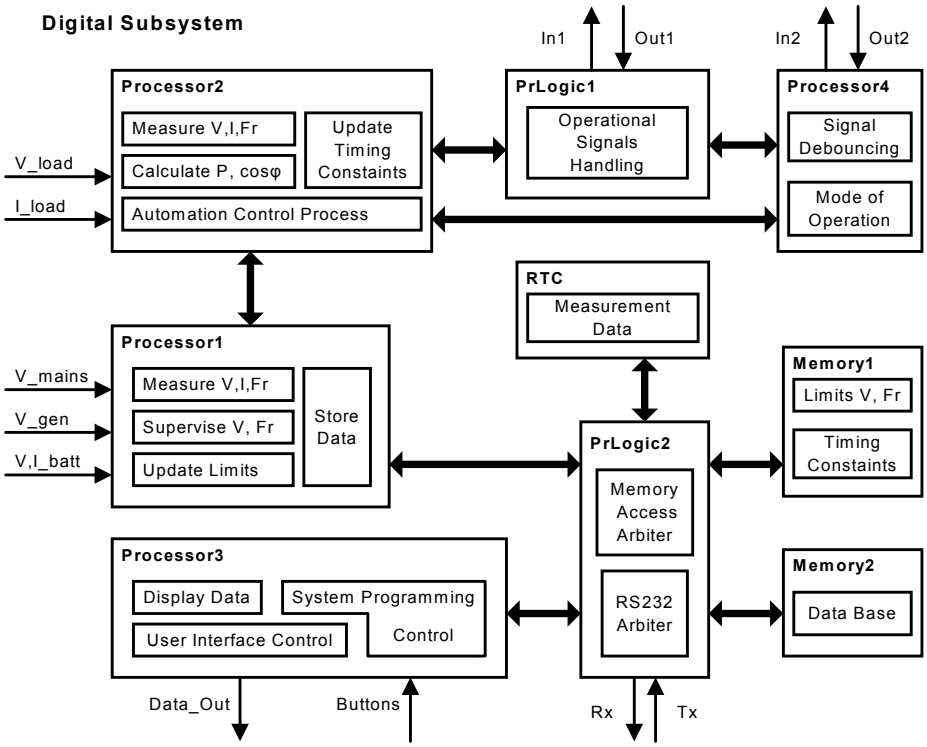

Figure 3. GCS system-level design.

A critical aspect of the system described herein is the functionality of the IMC core. Since many processors were used and real-time modules were allocated to more than one processor, the need for reliable inter-processor communication protocols was evident. CAN bus was used for message delivery between the two DSPs, while the standard SCI and SPI protocols were employed for the rest of the inter-processor communication. All mechanisms are controlled by communication firmware enhanced with real time features and form the software part of the IMC core. The hardware part consists of Memory Access Arbiter that allows intercommunication between system modules.

The design of the Memory Access Arbiter was focused on creating a fast resource-sharing module to meet the real time constraints of the system. Due to the low cost requirements only the absolutely necessary hardware was used. Therefore error control mechanisms were not implemented inside the Arbiter, thus leaving that function to the processors integrated peripherals.

5. CONCLUSION

A multiprocessor design architecture was described for the implementation of a power supply control system. The real-time critical nature of the application introduced great concurrency and complexity to the overall system. The functional specification was decomposed into a number of individual modules that were allocated in different system-level components. Each module consists of several real-time and non real-time tasks. The described architecture incorporates inter-processor communication and arbitration mechanisms that enable the reliable message delivery between individual tasks. Furthermore, provision was taken for prospective services expansion by reducing the processing power used and gate-level resources through efficient task scheduling and resources utilization.

The prototype used for system evaluation before the industrial production is shown in Fig. 4.

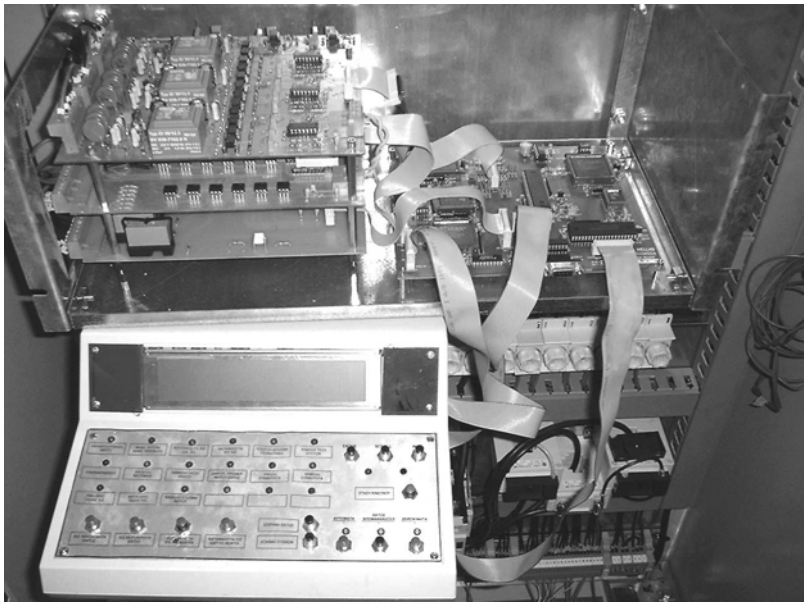

Figure 4. The evaluation prototype.

6. ACKNOWLEDGEMENT

This work was financially supported by ALSTOM Hellas SA under the framework of project “EPIMITHEAS”.

REFERENCES

[1] D.M. Auslander, C.H. Tham, *Real Time Software for Control* (Englewood Cliffs, NJ: Prentice-Hall, 1990).

[2] D. Gajski, F. Vahid, Specification and Design of Embedded Hardware-Software Synthesis, *IEEE Design & Test of Computers*, Spring 1995, 53-67.

[3] V. Backmutsky, V. Zmudikov, Accurate Frequency Estimation in Power Systems by DSP, *Proc. 18th IEEE Conf. of Israel*, Tel - Aviv, March 1995, 5.2.4, 5pp.

[4] H.C. Wood, M.S. Sachdev, Application of Digital Filters in Power Systems, *Proc. 29th IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, 1989.