

A Model Generator for Simulation and Testing of RDCs

2011-01-2636

Published
10/18/2011

Kai Schories
Hamburg University of Technology

Heinrich Rotgang
Diehl Aerospace GmbH

Frank Thielecke
Hamburg University of Technology

Copyright © 2011 SAE International

doi:10.4271/2011-01-2636

ABSTRACT

Simulation of avionics equipment is essential due to the complex nature of its development and integration process. Throughout the development process, executable component models are used to demonstrate the feasibility and the compliance of the system design with respect to its functional requirements. In later development phases, there is the need for system integration tests where a mix of real and simulated equipment is used to verify the overall system behavior. Since Boeing 777 and Airbus A380 programs, IMA¹ technology has entered several civil aircraft systems. In recent programs like Boeing 787 and Airbus A350 the number of IMA components has significantly increased. In this paper we present a simulation model for a new IMA component - the common Remote Data Concentrator (CRDC)², which is developed by Thales-Diehl for the Airbus A350 XWB. Building simulation models of IMA components is in general a challenging task due to their complexity on both software and hardware level. The particular challenges faced here, have their roots in two areas. The first is due to the configurable functional behavior of the CRDC, whilst the second comes from its versatile and configurable interface. Together, this leads to a potentially infinite number of unique model instances where each instance constitutes a different CRDC configuration. Moreover, since simulating IMA

components always requires stimulation from external models and/or real equipment connected through a set of standardized interfaces, another important aspect is the need for a standardized model integration process which ensures compliance among all system component models and which facilitates an automated integration on a system simulation platform.

Our solution is a software tool called SiToRDC which comprises (1) a model generator and (2) a functional core model of the CRDC. The model generator is used to process a user-supplied set of configuration files which parameterize the interface and functions of the core model. The result of the generation process is a CRDC model instance provided as a set of ANSI C-language modules. Those modules are then distributed to simulation workshops where platform integration will take place. We will describe the principal architecture of the SiToRDC toolchain and show how to build CRDC models for different simulation workshops.

INTRODUCTION

The concept of IMA is centered on standardized, modular computing modules, where each type of module provides a specific set of shared resources to avionics applications [MOR99]. Shared hardware resources can be distinguished into three functional groups: (1) computing resources - core

¹IMA: Integrated Modular Avionics; the term IMA is here meant according to ARINC 651

²The term RDC is here meant according to ARINC Report 655

processor and mass memory; (2) I/O resources - avionics data busses, wired I/Os, gateways; (3) power supply and conversion. These functional groups are housed on dedicated Line Replaceable Modules (LRMs). Several LRMs are then integrated into a standardized cabinet which provides a common backplane bus and mechanical connectors. The term IMA module usually refers to an equipped cabinet which hosts a number of applications [WAT06].³ An ARINC-653 compliant operating system controls the resource access from applications at runtime and provides means for real-time execution, partitioning and fault tolerance. The two latter properties enable the incremental development and certification of avionics software, and allow for modular updates at later times when the aircraft is already in service.

Typical aircraft domains where IMA is employed are Cabin and Cockpit. In these domains the number of complex software functions has rapidly increased over the last 20 years. A classic, federated architecture [OTT07] based on dedicated “black box” computers and ARINC-429 networks cannot scale up well with this trend. Moreover, from the perspective of an airframer it is desirable to have a small set of flexible standard modules for all avionics applications because of the reduced system development costs and the ability to optimize resource allocation and spare. For airliners similar arguments apply with respect to reduced maintenance costs and customization options [PRI92].

In today's civil aircraft avionics architectures all IMA modules are installed in the Avionics Compartment located in the fuselage below the cockpit. A switched AFDX (Airbus A380) or ARINC-629 (Boeing 777) backbone network facilitates the inter-module data exchange. Sensors and peripheral equipment⁴ is either connected via fieldbusses like CAN or ARINC-429, or through wired I/Os⁵. The latter connection variant is usually realized by means of special I/O modules (IOMs) which act as data concentrators towards IMA modules. For that purpose, each IOM provides a configurable routing function⁶, together with necessary signal conditioning and conversion. ARINC standard 655 - Remote Data Concentrator (RDC) - provides a generic description for this type of equipment.

For Airbus' next long-range aircraft A350 XWB, Thales-Diehl is going to develop a new type of RDC which will extend the concept of a pure data concentrator by a set of general combinatorial logic functions [DAS11]. This will allow performing basic closed control loops directly at the connected equipment without the need for intervention on

application software level (in IMA modules far located in Avionics Bay). The common RDC (CRDC) will become a new standard IMA component in future programs replacing several former IOM types and is hence a step forward towards standardized avionics hardware. Introducing such a new standard module will have impact on virtually all systems using IMA. In particular there is a completely new branch of processes and tools for configuration, testing and maintenance. This is a critical point for successful introduction of this new technology. Hence, many efforts have been made so far to define means for early evaluation at platform and system level. With platform level we mean the IMA configuration process which is mainly concerned with the configuration tools and documents, whereas the system level evaluation is more related to simulation and integration test activities.

In the following we present the Simulation Tool for CRDC (SiToRDC) which is used for simulation of a CRDC instance in early integration tests. Section 2 gives an overview of the model functional architecture and the toolchain developed for model generation. Section 3 describes the mapping rules of the model generator and the generation of runtime configuration tables from template files. In section 4 we briefly describe the Airbus simulation process and the integration of CRDC models in simulation platforms. Section 5 points out future directions and applications of the tool. Section 6 gives conclusions.

MODEL FUNCTIONAL ARCHITECTURE AND TOOLCHAIN

Figure 1 illustrates a possible usage scenario of the CRDC in future avionics applications. As can be seen therein, the CRDC acts as a versatile gateway and data concentrator between IMA modules, denoted as CPIOM (Core Processing I/O Module), and remote system electronics which can be any kind of LRU ranging from simple sensors to complex controller modules with embedded computing capabilities.⁷ For this purpose the CRDC provides standard avionics data busses, in particular AFDX, CAN and ARINC-429, as well as the full range of Analogue and Discrete interface types. Since it is a common RDC, the number and types of interfaces which are actually used at a aircraft particular position must be configurable. Also the number of different applications sharing a common resource of the CRDC is not fixed. Therefore, the range of possible functions and combinations of configuration parameters is virtually unlimited.

³The term *application* refers here to one or more related software functions and its (shared) IMA resources.

⁴Here, peripheral equipment refers to all kinds of system specific remote electronics such as control valves, switches etc.

⁵Wired I/O refers to standard Analogue and Discrete interfaces as well as to special sensor interfaces like LVDT.

⁶For instance, routing a Discrete input into an AFDX message, or routing an AFDX float datum to a Analogue output

⁷In Figure 1, system specific parts are denoted as “ATAxx”; for instance, parts marked as ATA-32 would belong to Landing Gear.

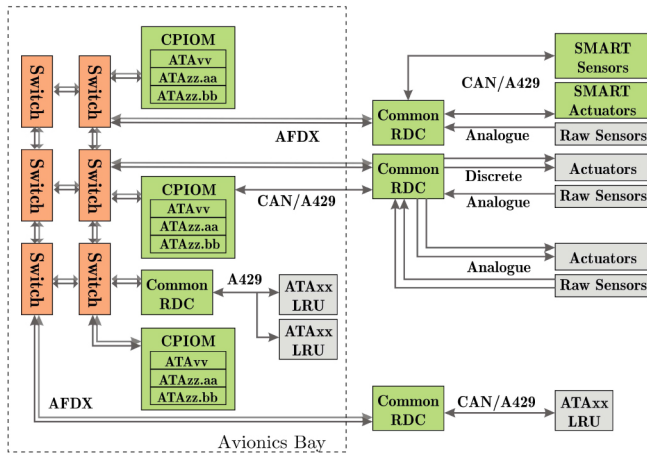


Figure 1. CRDC usage scenario in future avionics applications

Three fundamental parts of the CRDC functional architecture had to be considered during model development. First, there are the hardware characteristics which determine the physical behavior on circuit and signal level. SiToRDC only partially deals with these issues since the main objective was to simulate the logical behavior under real-time constraints. So, we won't discuss this part further. Second, there is the CRDC core software which determines the functional behavior comprising I/O drivers, low-level routing and conversion functions, higher-level combinatorial logic, and the I/O scheduler. This part is implemented in the SiToRDC core model which is discussed at the end of this section. The third part is the CRDC configuration tool chain which comprises the data model, the configuration editor and data loading facilities. SiToRDC adapts to this toolchain by use of a special model generator which is discussed in the following two subsections.

MODEL GENERATION PROCESS

It is convenient to start with a short description of how a model instance for simulating a particular CRDC position will be obtained. Figure 2 illustrates the model generation process. The SiToRDC core model consists of a set of ANSI-C modules that implement the common, position-independent part of the CRDC core software. It contains a lot of placeholders which have to be filled with position dependent parameters for I/O and scheduling. These parameters are obtained from a validated CRDC configuration which is processed by the SiToRDC model generator. The output of the model generator is twofold; first it generates a number of additional ANSI-C modules from a set of templates. These templates define data structures for interface tables and the I/O schedule which will be linked to their respective stubs in the core model during model compilation. The other part of

the generator output is the Model Interface Configuration Document (MICD) which is used on simulation platforms for model integration. The Airbus simulation framework provides model developers a set of tools which can be used for building executable models from source and to perform basic functional testing within a desktop environment.

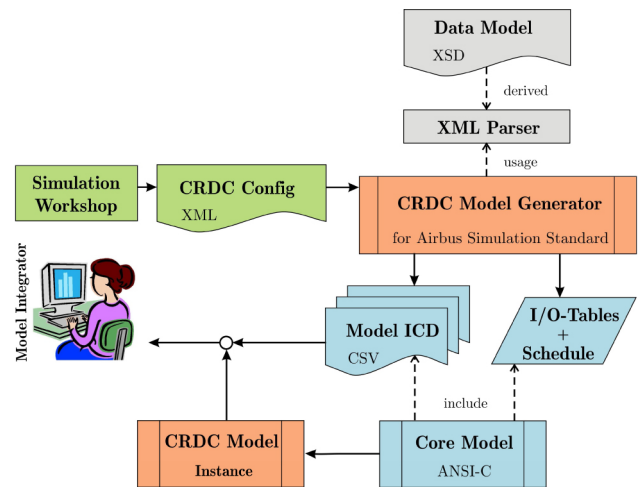


Figure 2. Model generation process

SITORDC GENERATOR

Figure 3 shows the architecture of the developed model generator which consists of three main parts:

1. The CRDC configuration is pre-processed by a set of XQuery scripts which extract all relevant configuration parameters
2. The generator component transforms the parsed parameters into workspace objects of a template engine
3. The template engine fills a number of ANSI-C and CSV templates with parameter from template workspace

The XQuery language [WAL07] has been chosen for preprocessing large and complex XML data because it comes with a flexible and intuitive syntax which essentially provides the same features like modern query languages for relational data bases. Another reason was the need for decoupling the view on the CRDC data model from the template specific mapping rules of the generator component. So, changes in the structure of the data model would only require modifications in the queries as long as the mapping rules remain the same. We use the XQilla library in connection with XSD data binding and Xerces XML parser.⁸ All these tools are freely available and implement the most recent language specifications. The generator component is implemented as a C++ console application which can be either used in batch

⁸XQilla project: <http://xqilla.sourceforge.net>; XSD product page: <http://www.codesynthesis.com/products/xsd> Xerces-C project: <http://xerces.apache.org/xerces-c/>