HARDWARE-SOFTWARE SIMULATION FOR LLRF CONTROL SYSTEM DEVELOPMENT

A. Vaccaro, L. Doolittle, A. Ratti, C. Serrano, LBNL, Berkeley, CA 94710, USA

Abstract

Field Programmable Gate Arrays (FPGA) have been used in accelerator controls for a long time. Stricter performance requirements in new accelerator designs force LLRF control systems to continuously improve, and the increasing density of available FPGAs enables such progress. The increased complexity in FPGA design is not always followed by new Radio Frequency (RF) systems availability for development and testing. Therefore, a hardware-software simulation tool has been developed to model RF systems by a software simulator. It simulates the interaction of Hardware Description Language (HDL) code that is to be synthesized with both RF systems and communication ports to external controls software, reproducing realistic working conditions of the FPGA. The hardware-software interaction for Low-level RF (LLRF) control system design is discussed here.

INTRODUCTION

The structure of digital LLRF systems can be divided into two main blocks: the signal conditioning logic to interface with the outside world, and the Digital Signal Processing (DSP) performing control of the sampled RF fields.

From a purely digital prospective, the RF fields from A/D converter(s) are available in a digital word in an FPGA, and updated every sampling period. Then they are conditioned digitally (filtered, IQ and down converted), DSP is performed, and after some delay the FPGA conditions and delivers a control signal to a D/A converter.

Modern FPGAs communicate directly or indirectly with modern computers using standard hardware such as USB or Ethernet. The communications are used for monitoring or communication with external control systems. This bi-directional exchange can also be considered as an all-digital input-processing-output process. Digital LLRF control designers can then take advantage of the increasing complexity of FPGAs, and the high flexibility of the digital world, by using software models of RF systems.

RF MODEL

The hardware-software simulation tool described here reproduces normal operation of each element in the system, and their interactions in a virtualized-time environment, where simulation steps match the FPGA clock period. For efficiency, the response of analog components needs then to be characterized in discrete-time domain, and optimized for computational speed.

Discrete-time transfer function

The entire system is modeled at baseband, thus all signals are represented in vector form. The RF transfer function of filters and cavities can be expressed as a combination of single-pole low-pass filters. The adopted discrete-time approximation of a single-pole low-pass filter differential equation, is expressed as follows:

$$v'_o = av_o + \frac{1}{2}b(v_i + v'_i) \tag{1}$$

$$\text{where} \quad a = \frac{1 + \frac{1}{2}\Delta t \cdot p}{1 - \frac{1}{2}\Delta t \cdot p}, \quad \text{and} \quad b = \frac{\Delta t}{1 - \frac{1}{2}\Delta t \cdot p}$$

$v'_{i/o}$ and $v_{i/o}$ are the current and previous inputs/outputs, respectively. Δt is the simulation step duration, and p is the pole location (a complex number). Cavity detuning is represented by a slight pole shifting into the imaginary direction. Waveform propagation through waveguides, measuring noise, and other effects are characterized using simple operations of complex signals. Finally, beam current and external RF sources (including non-linear effects) are also represented.

Software implementation

All RF components are modeled in C programming language. Fig. 1 shows the development process followed to obtain the response of the RF system after one simulation step. First we determine the characteristic parameters defining each element (e.g. quality factor, couplings, etc. in the case of a cavity), and its corresponding discrete-time transfer function. Then, we instantiate those parameters with numeric values (e.g. Q=2100, β_{in}=0.98, etc.), and finally we concatenate the inputs and outputs to obtain the response of a given RF system after one simulation step.

LLRF CONTROLS

Modern LLRF control systems are implemented in an FPGA, whose complexity is continuously increasing, and is capable of communicating with software devices. Complex and sophisticated algorithms are easily implemented in software. This possibility is becoming widely used, and the complexity of LLRF systems is being progressively transferred from hardware to software.

The role of the FPGA is then to perform the traditional digital signal processing in the feedback loop, and to communicate with software control devices (at lower rates due to the limited speed of data computation and transport).

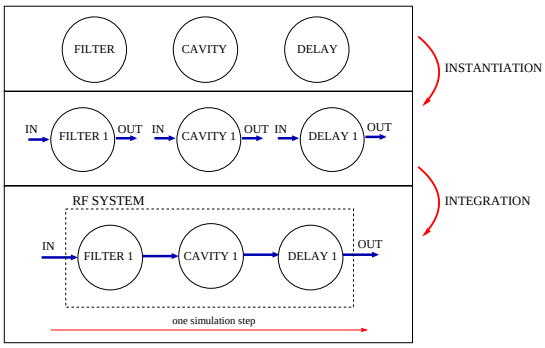

Figure 1: Development process to obtain the response of an RF system after one simulation step (illustrative example).

Digital Signal Processing

The FPGA receives the sampled RF fields from A/D converters. If sampled at $90°$, their interleaved I (in-phase) and Q (quadrature) components are available in the FPGA every sampling period, contained in a digital word. These signals are then propagated through the FPGA logic, and after some time the control signal is available in another digital word to the D/A converter, updated at a system clock period.

The LLRF control designer is then searching for the optimal performance of the logic between those two FPGA registers: the one containing the sampled RF fields, and the other one containing the control signal.

FPGA communication ports

In order to communicate with other devices, the FPGA needs to follow a number of rules defined by a communication protocol. A part of the logic in the FPGA fabric is devoted to condition signals prior to transfer, and then a communication controller will take those signals, adapt their binary format to a given protocol (coding), and transfer them through a digital communication channel. The inverse process (decoding and conditioning) happens to signals received from a communication link.

Several factors such as the capacity of the link between the FPGA and the other communicating entity, their computation speed, etc. determine the transfer data rate. It is generally much lower than the system clock used in the feedback control loop. Signals need then to cross from one clock domain to the other.

Clock domain crossing is performed using data strobes [1], which are asserted by the emmiting logic block when the data is ready, and kept available for a suitable amount of time. Decimation and interpolation can be performed using computationally optimized low-pass filter implementations, such as CIC (Cascaded Integrator-Comb) filters [2].

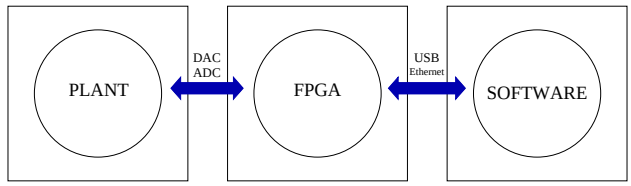

Figure 2: Independent simulation entities used for LLRF controls testing.

SOFTWARE INTERACTION

Even when a feedback control loop is implemented in an FPGA additional complex signal processing is performed in software. Since LLRF system configuration is evolving towards increasing amount of software, testing techniques should be adapted accordingly.

Simulation entities

A flexible hardware-software simulation environment for LLRF system design is composed of three independent entities: the LLRF controls in the FPGA, the RF model (plant), and software devices (see Fig. 2). The functionality of the FPGA and the software include the logic dealing with communication tasks.

The HDL code to be synthesized in the FPGA is simulated in a hardware simulator. The DSP part in the hardware design interacts with an RF system via ADCs and DACs. In parallel, the FPGA also interacts with software devices via USB or Ethernet communication channels.

The integrated simulation runs in virtualized-time. This means that the response of every element in the system has been characterized according to this simulation time step variable, corresponding to the FPGA clock period. Then the interactions among simulation entities are also simulated considering the time variable.

Abstraction of communication channels

The communication channels in our system are ADCs and DACs for the feedback loop, and USB or Ethernet for the FPGA-software link. Both the HDL code in the FPGA and software can be entirely characterized using simulation tools, obtaining their exact response to a given excitation. The RF system has been modeled in software so we can obtain its response every simulation step.

- FPGA - RF System

As the analog components, the physical communication links in the system are not present in simulation and need to be characterized. The RF model includes noise sources due to A/D conversion, as well as quantization error. Thus, in simulation, the RF model reads the FPGA register containing the control signal, calculates the RF system

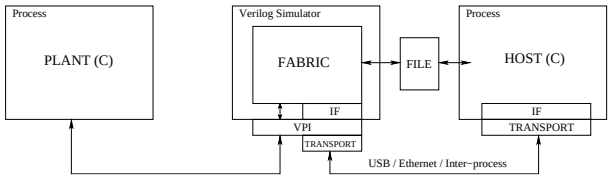

Figure 3: Communication layers in the simulation environment.

response to that excitation, and places the signal coming from the ADCs into the corresponding FPGA register.

The RF model is coded in C, and the HDL used is Verilog. For their interface, Verilog Procedural Interface (VPI) is used. It allows behavioral Verilog code to invoke C functions, and C functions to invoke standard Verilog system tasks.

In practice, a top-level Verilog test-bench instantiates the FPGA fabric as it will be synthesized in the chip. Then, in a hardware simulator, it runs the system for a number of cycles. Every FPGA clock cycle (corresponding to a simulation step), a routine in C is called to obtain the RF system response. That routine gets the control signal from the FPGA as an input argument, and delivers the signal(s) coming from the ADC(s) as output(s), whose values are connected to the FPGA inputs to close the loop.

- FPGA - Software

FPGAs are capable of communicating with software devices via USB or Ethernet. These protocols use very precise data formats, and have a limited bandwidth. In simulation, and for our purposes, there is no interest in modeling the complexity of the physical layer. Therefore the logic in the FPGA, and the software layers dealing with communication purposes are tested abstracting the physical data transfer.

A different system clock is used for communication with computers, and clock domain crossing is performed using data strobes. The FPGA will assert a data strobe when data is ready to be sent, and will check the status of a different data strobe for the availability of the incoming data.

Fig. 3 shows the communication layers in the simulation environment. A communication controller in the FPGA, and a driver in the computer will exchange data with the USB or Ethernet communication link. A transport layer will exchange data with them at both ends, being independent of the physical layer. The interaction of the FPGA and the computer with the communication layer is identical in simulation and in a real system.

Driver-level testing

In simulation, the hardware simulator is running on one process, and the software entity is running on a different one. Then, an inter-process communication link is established to replace the physical link, being the transport layer independent of the communication channel.

At the FPGA level, the data transfers are simulated in the virtualized-time environment. For this purpose, an interface layer (IF in Fig. 3) performs the correspondence between the data transfer using inter-process communication (simulation), and that using USB or Ethernet (real system). The data will be available at a given register, and the data strobe asserted at exactly the same pace as if it were transferred via a physical channel.

VPI is used to make the link between the inter-process communication entities and the Verilog simulator. It allows to read and write FPGA registers with zero simulation latency. Then, the IF module will synchronize that data to emulate real conditions of the communication channel. Additional data can be exchanged using files combined with the inter-process communication for transfer synchronization.

CONCLUSIONS

The hardware-software simulation environment reproduces working conditions of the FPGA, including communication with software devices. LLRF configuration is evolving towards increasing amounts of software. Testing techniques used here allow validation of the FPGA design, software algorithms used for complex signal processing, and the communication between the FPGA and computers via standard hardware such as USB or Ethernet.

A modular RF model has been implemented in software for optimal computational speed, and reliability, allowing a high level of flexibility in its configuration. Finally the hardware design that is to be synthesized in the FPGA, software running in computers, the RF model and their interaction are simulated in an integrated, virtualized-time environment using all free-software tools. An application of the hardware-software simulation is described in [3].

REFERENCES

[1] Mark Stein, “Crossing the abyss: asynchronous signal in a synchronous world,” EDN Magazine, July 24, 59-69 2003

[2] Matthew P. Donadio, “CIC Filter Introduction,” July 18, 2000

[3] C. Serrano, L. Doolittle, A. Ratti, A. Vaccaro, “Ensemble cavity control system simulation using pulse-to-pulse calibration,” EPAC08, June 2008, Genoa, http://www.JACoW.org