DSP DESIGN USING SYSTEM GENERATOR

Jean Marc Koch - ESRF Grenoble France

Abstract

When designing a real time control system, a fast data transfer between the different pieces of hardware must be guaranteed since synchronization and determinism have to be respected. One efficient solution regarding these constraints is to embed the data collection, the signal-processing and the driving of the acting devices in an FPGA even if this solution does not come without difficulties. To overcome one of these difficulties, it is possible to open the development of the signal processing to non HDL (Hardware Description Language) specialists; here, System Generator [1] has been chosen for development purposes, in Simulink / Matlab from Xilinx and The MathWorks. Another challenge with such a system design is the ability to integrate real time models on already pre-configured hardware platforms. Although the hardware can be ready for communication, standard PCI or PCI express bus and dedicated fast data links like Gb Ethernet, the corresponding interfaces must be carefully defined and designed to communicate with HDL System Generator control systems blocks. Therefore, the work in this paper describes with two examples how to take advantage of hardware delivered ready for use from a communication point of view and how to get it ready to integrate a design under System Generator. The advantage of Simulink for the simulation phase of the design is also presented.

INTRODUCTION

Particle accelerators rely on hardware and software platforms with high speed, high-bandwidth acquisition, timing and synchronization, and advanced control to solve some instrumentation and control challenges.

When it comes to real-time data acquisition and processing, one solution is to keep this part at a low level to avoid the overhead imposed by multiple layers and data transfer.

FPGAs are now widely used as signal processors at the ESRF: one of the latest developments is the dynamic computing of tune estimate based on the accurate measurement of the currents in the booster magnets during the accelerating cycle of the electrons.

The part of the code concerning the measurement of the currents will be re-used later for driving new ramping power supplies for the booster magnets.

Another example of processing embedded in FPGA was the beam position correction updated at 10 kHz:

224 electron beam position monitors connected to 8 FPGA stations, themselves connected to 96 horizontal and vertical steerers make up this system which runs continuously during operation of the storage ring to compensate for the beam motion [2].

FPGA code development is based on High Level Language for the different data transfers and on System Generator for the signal processing.

FPGA CODE DEVELOPMENT

There are several possibilities for the development of a code using System Generator; in the following two examples it is used in the same way: System Generator design is part of a bigger project and as such needs to be connected to this.

System Generator Project Integration

The Signal Processing is described with System Generator, simulated with the Simulink tools, either in computer simulation or in hardware in the loop and then the corresponding HDL code is generated. During this process a wrapper is produced describing the gateways which have to be connected with the "hand written" part of the code. The integration of the System Generator Project to ISE Project Navigator is made by simply adding one file with the extension ".SGP", an empty file which identifies the location of the System Generator model (see Fig. 1).

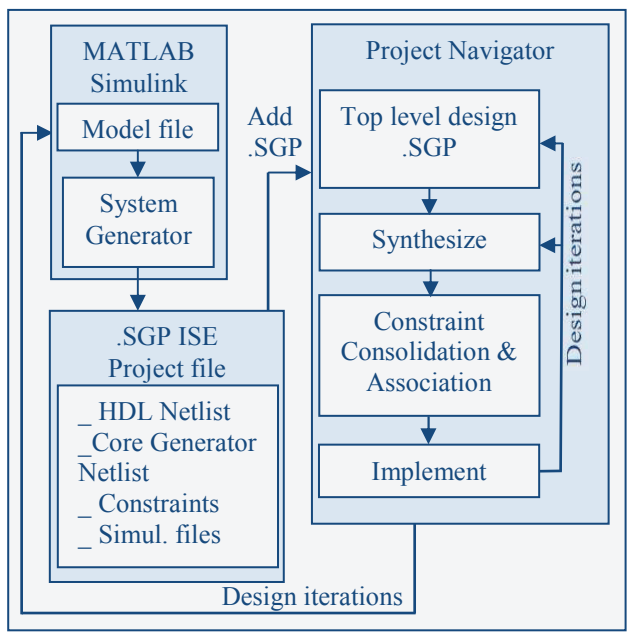

Figure 1: Integration flow between System Generator and Project Navigator.

ISBN 978-3-95450-139-7

APPLICATIONS

The two applications described here follow the flow mentioned above. All the tasks related to data communication external to the FPGA are coded in vhdl.

Booster Tunes Estimation Calculated from the Currents in the Magnets

The booster is a circular machine designed to accelerate the electron beam from ~200Mev to 6 GeV energy.

During the ramping, since the currents in the 3 main magnet families involved are not perfectly proportional all along the accelerating cycle and can slightly vary in an unpredictable way from one cycle to the next, the tunes are varying accordingly. As we need to know the value of the tunes to excite the beam for a cleaning process [3], a measurement and calculation in real time will be performed to drive an amplifier with the correct frequency to perform a blow-up of the unwanted bunches of the beam (see Fig. 2).

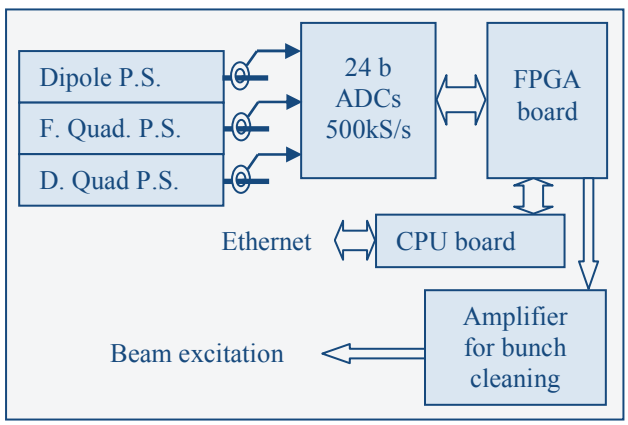

Figure 2: Layout of the booster tunes estimation calculated from the magnets currents.

Required Interfaces in the FPGA

1) Driver for ADCs readout from serial to 24bits word
2) Misc I/Os for the trigger and cleaning device
3) PCIe interface with ComExpress board (see Fig. 3)

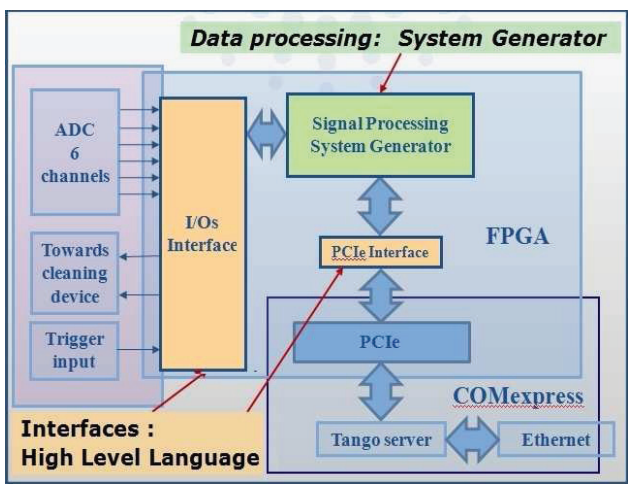

Figure 3: Pieces of code to be connected.

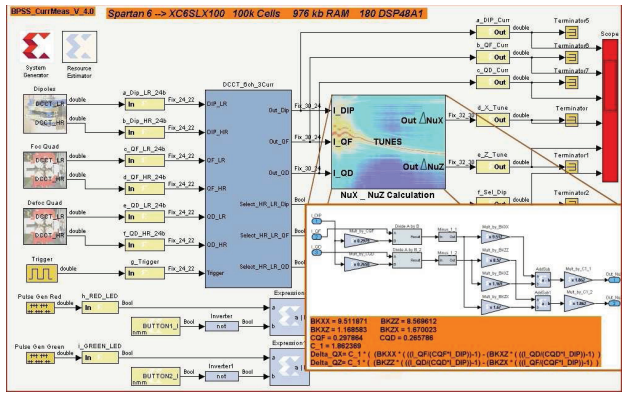

Figure 4: Simulation with Simulink / System Generator.

To achieve a very accurate measurement, the DCCTs have two ranges; an extra output has a full scale of only 10% of the full range. The selection between the two outputs is performed in the first System Generator block. The second block is dedicated to the calculation with fix point data of the variations of the tunes around their nominal values (see Fig. 4).

Once the code has been simulated, it can be tested on a preconfigured hardware platform thanks to the "Hardware in the Loop" (see Fig. 5) of System Generator. After that the code produced is transferred to a target through a JTAG or Ethernet connexion, the FPGA is configured and all the processing described in between the gateways on figure 4 then runs on the FPGA. The gateways will convert floating-point inputs to fixed-point and fixed-point to floating-points outputs for the simulation.

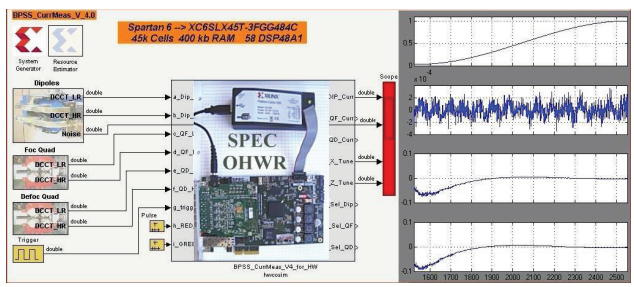

Figure 5: Signal processing with Hardware in the Loop.

The input signals are theoretically pure sine waves with a well defined relation in amplitude and phase between them, but the accuracy required, between 10^{-4} to 10^{-5}, makes that all disturbances must be taken into account. We can see on the right part of figure 5 the current in a dipole during the energy ramping, the noise introduced and the resulting tunes estimation. The current in the quadrupoles are described without noise.

By introducing noise and harmonics on the signals at Simulink level, we can check the reproducibility that can be reached with such measurements, suppress digital noise by choosing the right words size and, finally, generate the code that will be embedded in the real target.

ISBN 978-3-95450-139-7

FPGA Based Fast Orbit Correction System

The architecture chosen is such that from the electrons beam to the power converters, the whole data processing is performed inside FPGAs: the front-ends are digital BPMs "Liberas" and the data are dispatched to 8 stations equipped with Virtex-5 FPGAs through optic fibers thanks to a custom Communication Controller designed and implemented in VHDL at Diamond Light Source [4]

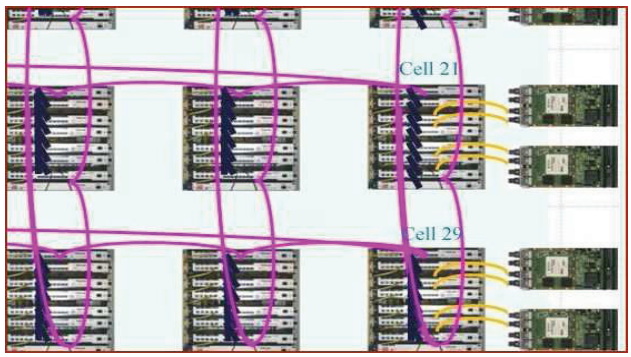

Figure 6: Fast Orbit Correction FPGAs interconnections.

FPGAs Stations for Orbit Correction System

The choice of performing the corrections in the FPGA has been guided by the fact that no real time operating system was supported at the ESRF when we launched this development. The obvious advantage of this solution is the effectiveness of the FPGA and therefore the simplicity of the final equipment but with nevertheless a drawback in the sense that the development process is more complex if compared with a DSP or processor programming.

Required Interfaces in the FPGA

1) Shared memories between Communication Controller and correction process
2) Outputs registers towards the power supplies
3) Shared memories and registers for the communication with PCI for the remote control

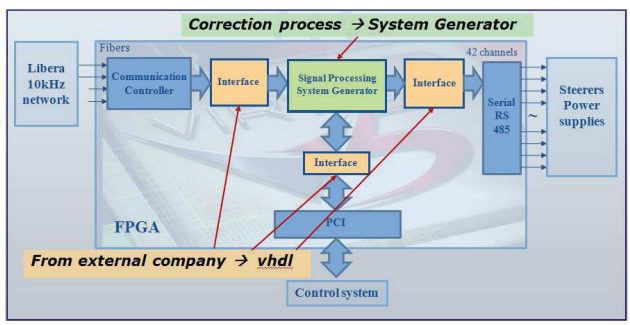

Figure 7: Code development distribution.

Thanks to the libraries delivered ready to use for the communication with the PCI by the supplier and the developments already done for the Communication Controller [4], the development in vhdl is quite limited.

Processing

Functions of the code embedded in the FPGA:

1) Collect the data from the BPMs at 10 kHz with the Communication Controller and transfer to System Generator through shared memories and registers (coded in High Level Language) These interfaces have to conform to the symbolic names, the words length and memories depth defined in the vhdl development.

2) Obtain the parameters from the PCI (coded H.L.L.)

3) Process the corrections (coded in System Generator) (see Fig. 8)

4) Send the set-points to the power supplies (coded H.L.L.)

5) Send the set-points for data recording to the Communication Controller (coded H.L.L.)

6) Synchronous start and stop of the correction on the 8 stations in the same 10 kHz cycle.

FPGA code development with Simulink

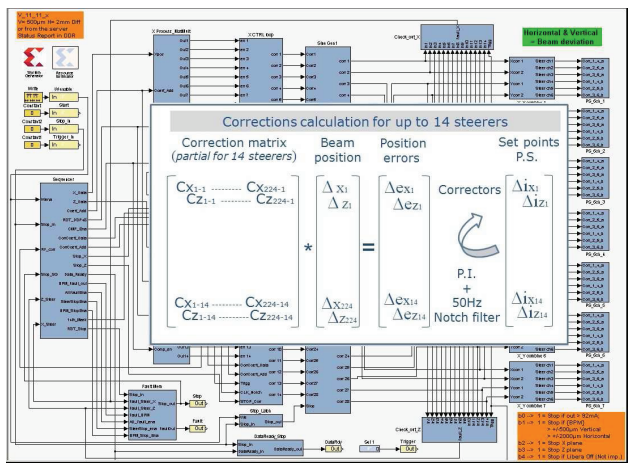

Figure 8: Whole signal processing in System Generator.

The processing can be tested and debugged first with bit true cycles in Simulink before any FPGA code production and configuration, for example the tuning of correctors dedicated to the beam stabilisation (see Fig. 9).

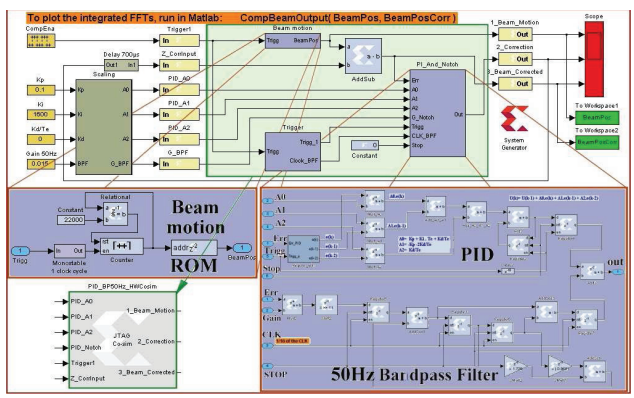

Figure 9: Corrector behaviour test with Simulink.

ISBN 978-3-95450-139-7

To check the behavior of the model, real data are introduced at the initialization of the simulation and processed with the components from Xilinx and then, data obtained from the output can be visualized (see Fig. 10) or saved for analysis under Matlab (see Fig. 11). In this simulation the power supplies, magnets and vacuum chambers are modeled as a simple delay of 700µs between the correction output and the reading of the beam position.

If required, as shown in the previous example, a test with "hardware in the loop" can be performed thanks to the possibility to transfer the code produced to a target through a JTAG or Ethernet connexion.

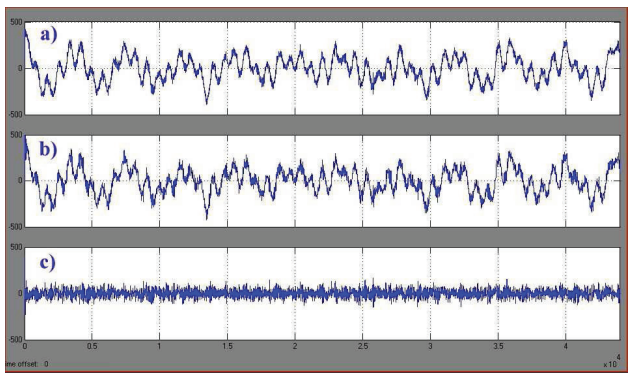

Figure 10: a) Beam position before correction. b) Correction applied. c) Beam position stabilised.

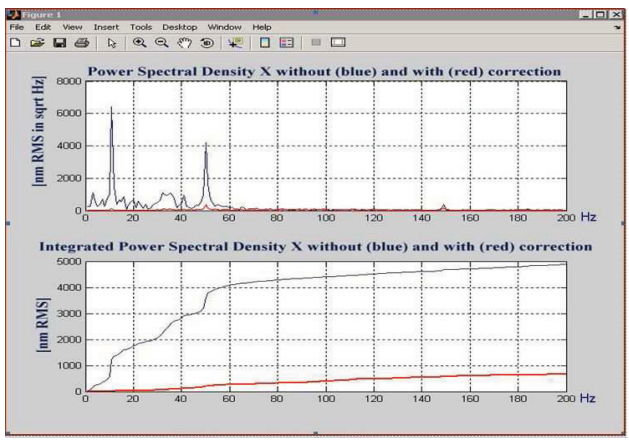

Figure 11: Expected results of the correction computed from the data written in the MATLAB base workspace.

All the functions except the signal processing were either developed or integrated by an external company specialized in vhdl coding. Provided that the interfaces are instantiated to cover the needs of all data transfers envisaged, the data processing under System Generator may evolve without changing the other codes. The synthesis, placement and routing can be launched with a new file from System Generator all the others being frozen (see Fig.12).

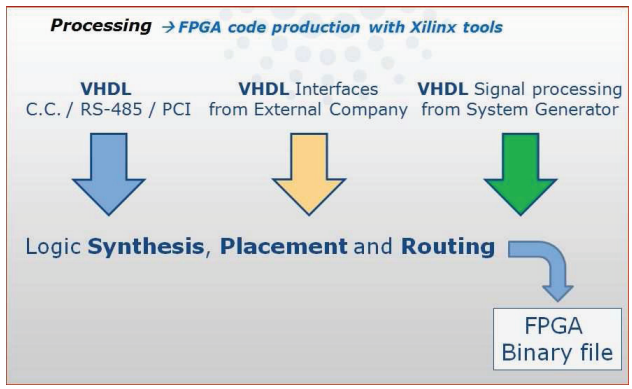

Figure 12: Merging of all sources with Xilinx tools.

Tests of the Hardware Setup

The data reception of the Beam Positions at each cycle was checked as well as the right connection with the power supplies. Indeed, it is easy to design a model in System Generator which copy the data received in a shared memory to the outputs connected to the Power Supplies inputs and check the current of these last ones. To achieve this test, an FPGA board simulating the network of the 224 BPMs delivers known data according to their numbering.

CONCLUSION

Progress in FPGA technology along with availability of high level tools for modelling, simulation and synthesis have made FPGA a key platform. Today, it is a good choice for the hardware development and implementation of high-performance applications requiring rigorous calculations in real-time. Many developments at ESRF have proven that FPGAs can outperform DSPs and embedded processors in signal processing, while minimizing the hardware burden of multiple data paths. The right tools and techniques coupled with innovative features in silicon architecture can produce complex DSP functions in a single FPGA. Xilinx System Generator is a system level modelling tool that facilitates FPGA hardware design by extending Simulink / Matlab in numerous ways to provide a powerful modelling environment. It opened the field of FPGA design to non specialists through its design environment. In addition, the possibilities for testing and verification both at simulation level and combined simulation / hardware in the loop are an essential point to successfully implement a design.

REFERENCES

[1] Xilinx System Generator for DSP Reference Guide.
[2] The New Fast Orbit Correction System of the ESRF Storage Ring E. Plouviez et al. DIPAC 2011.
[3] Cleaning of parasitic bunches in the ESRF booster synchrotron E. Plouviez et al. EPAC 2004.
[4] Diamond Light Source Fast Orbit Feedback Communication Controller Specification and Design Isa Uzun et al. January 2009.

ISBN 978-3-95450-139-7