Design and Hardware Implementation of Embedded Controller Devices

Slim Ben Othman and Slim Ben Saoud
INSAT / LECAP-EPT, Tunisia

1. Introduction

The goal of this work is to implement a PI controller on FPGA device using RTL Processor architecture composed of specific Finite State Machine associated with the custom Data-path (FSMD). In this project, we are interested in the current loop controller with a variable speed DC motor control. However, in order to generalize this study to any other system, control can be done easily using the same steps discussed in the following sections.

We present at the beginning, a brief presentation of the DC control loops, and we describe the digitalization of the control algorithm in order to obtain an easy and efficient implementation. From this description and according to the FPGA design flow, we deduce the dataflow graph (DFG) behavioural description of the current PI controller. Then, we discuss two FSMD architectures with timing and surface optimization.

After that, we present the implementation results of these architectures. Finally, we present the used Co-simulation model for the validation of the FPGA PI current components in the cascade control loop.

2. Command device

The case study is a physical process composed of a loaded DC motor supplied with a DC chopper, an optical incremental encoder (OIE) speed sensor and a Hall effects current sensor. The control device is composed of a PI current loop (fast loop) in a cascade with PI speed loop (slow loop).

The design of such control device was specified and validated in [1, 2] by using Spec C methodology [3, 4]. In the model obtained, the system is composed of a software part including the speed control loop and a hardware part including the current control loop as well as the various interfaces circuits. Subsequently, we have been interested in the hardware FPGA implementation of current PI controller.

3. FSMD architectures [4]

For the given system, there are several FSMD architectures that are different due to their implementation. In our application case, we have developed two FSMD architectures:

- In the first architecture, we aim for the space optimization. The command computing is carried out in a sequential way.
- In the second architecture, we aim timing optimization. Some operations are then, computed in a parallel way.

3.1. First architecture

Fig. 1 shows the data-path unit structure adapted for a PI controller. According to the control signals, this unit allows us to restore external data inputs and output results, to generate command value and to keep it at output.

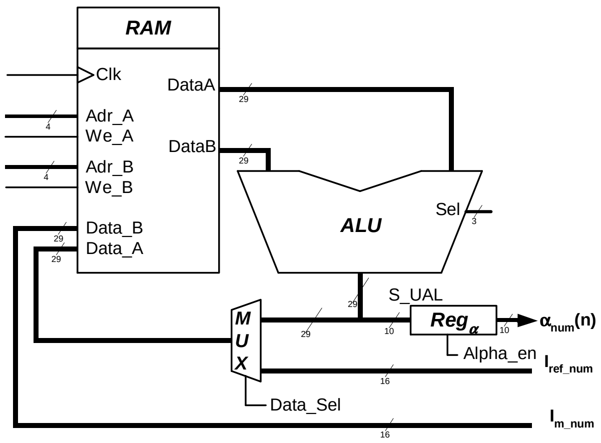

Fig.1: Datapath structure of the first architecture.

The control unit is described by a finite state machine (FSM). It carries out one computing cycle in 27 states. Fig. 2 illustrate this unit structure.

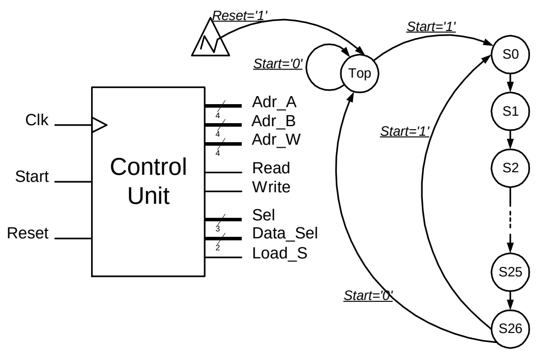

Fig. 2: Control unit diagram of the first architecture.

3.2. Second architecture

Starting from the first architecture data-path structure, we add to the ALU, two multipliers instead of one, so that multiplication is done in parallel

With this architecture, computing cycle in the control unit is reduced to 25 states.

4. Implementation results

We present on the following table, the structural and functional characteristics of the two architectures implemented on Xilinx FPGA type Virtex2 XC2V250.

Table 1: Implementation results.

	1st Arch.	2nd Arch.
Number of External IOBs	45 (22%)	45 (22%)
Number of SLICEs	389 (25%)	543 (35%)
Minimum period (ns)	15.212	16.113
Number of state	27	25
Minimal latency (ns)	410.724	402.825

Although the second architecture is better in execution time, it is clear through this table that the first architecture presents a more interesting surface/latency compromise.

5. Simulation setups

We have integrated these two architectures in the Xilinx System Generator tool [5] installed on Matlab. This tool allows us to test our design in the control loop environment as shown in Fig. 3.

We have simulated these two architectures with different speed targets and load disturbances. Then, we compared these simulations with responses produced by continuous model. We note that errors are relatively very low for both architectures. Fig. 4 shows an example of simulation using the first architecture.

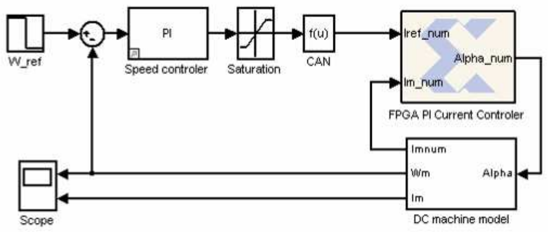

Fig. 3: Co-simulating of the designed system.

6. Conclusion

In this paper, we presented a hardware implementation of PI controller that benefit from the high-performance and high-integrity of programmable logic device. We studied the case of a DC motor cascade command, which can be easily generalized to any other process control.

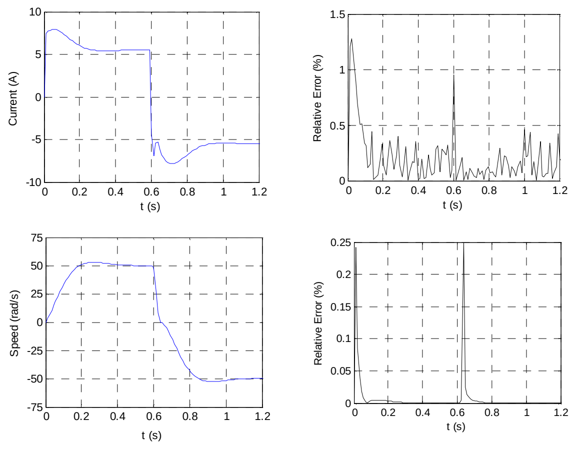

Fig.4: Responses example with speed target of ±50rad/s and load disturbance of 1Nm.

We proposed two RTL Processors architectures to implement integer PI algorithm. These architectures allowing the surface and timing optimization, respectively, are mapped using Xilinx FPGA and are validated successfully.

References

[1] S. B. Saoud, A. Gerstlauer and D. D. Gajski, American Journal of Applied Sciences **2**, 1331 (2005)

[2] S. B. Saoud, and D. D. Gajski, "Specification and validation of new control

algorithms for electric drives using SpecC language", Technical Report ICS-TR-01-44, UC Irvine, 2001.

[3] A. Gerstlauer, R. Dömer, J. Peng, and D. D. Gajski, *System Design: A Practical Guide with SpecC* (Kluwer Academic Publishers, CECS, 2001).

[4] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer and S. Zhao, *SpecC: Specification language and methodology* (Kluwer Academic Publishers, CECS, 2001).

[5] Xilinx, *Xilinx System Generator for DSP version 3.1 Using the Xilinx Software* (Xilinx INC, USA, 2003).