MATEC Web of Conferences **41**, 02004 (2016)
DOI: 10.1051/matecconf/20164102004

Automated synthesis of an FPGA-based controller for vehicle lateral control

Christoforos Economakos[1,a], George Economakos[2], Michael Skarpetis[1] and Maria Tzamtzi[1]

[1] *Technol. Educ. Inst. of Central Greece, Department of Automation Engineering, GR34400 Psahna Evias, Greece*
[2] *Nat. Techn. Univ. of Athens, School of Electr. and Comp. Eng., Microprocessors Lab, GR17580 Athens, Greece*

Abstract. The purpose of this paper is to demonstrate the use of an integrated methodology, which was introduced by the authors in previous publications. The purpose of the methodology is to facilitate the automated implementation of FPGA-based industrial controllers using modern high-level synthesis tools. In previous papers the methodology was used for rather simple applications. Here it is applied in a more demanding problem, namely the on-line optimization of a PID controller for the automatic lateral control of a highway vehicle.

1 Introduction

In recent years, a number of researchers have investigated the application of FPGA-based techniques in industrial control problems [1]-[6]. One particular approach [2], [5]-[6] is the use of modern High-Level Synthesis (HLS) tools which facilitate experimental implementation and comparison of alternative designs and which can help in reducing time to market by avoiding time-consuming and error-prone manual hardware design procedures. In [6] the authors have presented a scalable, multicore, FPGA-based System-on-Chip (SoC) architecture for digital controllers. The experimental results of that paper involved rather simple PID and fuzzy controllers. In the present paper the proposed architecture and the associated HLS-based design flow are used for on-line optimization of the parameters of a robust PID controller for vehicle lateral control based on dynamically updated estimates of the unknown system parameters.

2 The control problem

The problem of lane keeping of vehicle in highways is a major lateral vehicle control problem and has been treated using various control approaches (e.g. [7-10] and the references there in). Several of the proposed control methods use look–down systems for measuring the lateral displacement of a vehicle along the reference path.

The vehicle parameter variations due to operation conditions (mass, speed, road-tire contact) and disturbances due to the external forces acting on the vehicle body, indicate the use of robust control techniques in order to keep the vehicle in the reference path. In this paper we start from a robust controller and try to improve it on-line using dynamically updated estimates of the unknown parameters.

[a] Corresponding author: economakos@teihal.gr

The single track, bicycle model illustrated in Fig.1 [13] is used to obtain a mathematical description of the steering dynamics of a vehicle described in state space form as follows:

$$\begin{aligned} \dot{x}(t) &= A(q)x(t) + bu_f(t) + d\rho_{ref}(t) + E(q)f_w(t) \\ y(t) &= cx(t) \end{aligned} \quad (1)$$

where $x(t) = \begin{bmatrix} \beta & r & \Delta\psi & y & \delta_f \end{bmatrix}^T$, u_f is the front steering angle command, $\rho_{ref} = 1 / R_{ref}$ is the curvature of the guideline, u is the speed at the center of gravity (c.g.) of the vehicle, β is the sideslip angle, r is the sideslip angle, $\Delta\psi$ is the angle between the centerline of vehicle and the tangent to the guideline, δ_f is the front wheel steering angle, y (output) is the deviation from the guideline at the sensor location $l_s = 6.12$ m from the c.g. and f_w is an external disturbance acting at a distance $l_w = 0.1$ m from the c.g. (see Figure 1),

$$A(q) = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & b_{11} \\ a_{21} & a_{22} & 0 & 0 & b_{21} \\ 0 & 1 & 0 & 0 & 0 \\ u & l_s & u & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$b = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, \; d = \begin{bmatrix} 0 & 0 & -u & 0 & 0 \end{bmatrix}^T,$$

$$E(q) = \begin{bmatrix} \dfrac{1}{\bar{m}u} & \dfrac{l_w}{J} & 0 & 0 & 0 \end{bmatrix}^T, \; c = \begin{bmatrix} 0\,0\,0\,1\,0 \end{bmatrix},$$

$$a_{11} = \frac{-(c_f + c_r)}{\bar{m}u}, a_{12} = -1 + \frac{(c_r l_r - c_f l_f)}{\bar{m}u^2}, b_{11} = \frac{c_f}{\bar{m}u},$$

$a_{21} = \dfrac{(c_r l_r - c_f l_f)}{\bar{J}}, a_{22} = \dfrac{-(c_r l_r^2 + c_f l_f^2)}{\bar{J}u}, b_{21} = \dfrac{c_f l_f}{\bar{J}}$, $c_f = 50400$ N/rad and $c_r = 33600$ N/rad are the cornering stiffness values of the front and rear wheels, respectively, $l_f = 1.034$ m and $l_r = 1.491$ m are the distances of the wheels from the c.g. and $J = 2783$ Kg m^2 is the moment of inertia.

Also the parameters $q_1 = u \in [u_{\min}, u_{\max}]$ (vehicle speed) and $q_2 = \bar{m} \in [\bar{m}_{\min}, \bar{m}_{\max}]$ ($\bar{m} = m / \mu$ where m is the mass of the vehicle and μ is the road adhesion) are physical uncertain parameters and $q = [q_1, q_2]$ is the uncertain vector. We assume $u_{\min} = 15$ m/sec, $u_{\max} = 40$ m/sec, $\bar{m}_{\min} = 1330$ kg and $\bar{m}_{\max} = 1773$ kg.

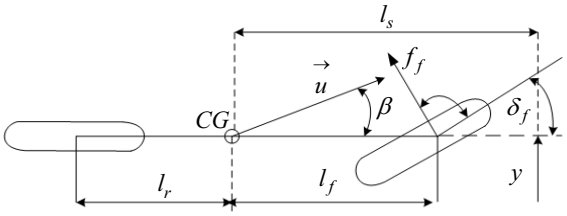

Figure 1. Single-track model for vehicle steering

The closed-loop system is formed by applying the following PID controller to the open loop system (1).

$$U_f(s) = \left(K_p + \frac{K_i}{s} + sK_d \right) \left[R_{ref}(s) - Y(s) \right] \qquad (2)$$

In order to obtain the values of the controller parameters K_p, K_i and K_d we employed a technique based on the algorithms in [11] and [12] and designed a robustly stable controller with optimized worst-case performance over all uncertain parameter variations. The details of the derivation of this controller will be presented in a subsequent paper.

3 On-line improvement of the controller performance

3.1. General Description

During normal vehicle operation, it is generally possible to calculate on-line estimates of at least some unknown parameters by using measurements. In this way the uncertainty ranges can be reduced or even eliminated for as long as the estimates are valid. During that period it makes sense to try to replace the robust controller with another controller which performs better for the particular parameter values.

In this section we present some first results concerning FPGA implementation of an embedded System-on-Chip (SoC) for finding such a better controller based on the general SoC platform architecture presented in [6].

The SoC platform consists of a general purpose microcontroller and a number of special-purpose hardware accelerators, all connected to the same bus. The microcontroller, which may be implemented as a soft processor on the FPGA fabric, is the bus master and performs mostly general-purpose tasks such as communication and coordination of the special-purpose units. The hardware accelerators are connected as slave units to the microcontroller bus and are responsible for application-specific functionality. They are designed as independent hardware components by using modern HLS tools. The primary input of these tools is an algorithmic specification of the desired functionality in the form of standard C code. In this way the functionality of both the microcontroller and the accelerators is coded in the same language, thus shortening development times and making it fairly easy to move tasks between units in order to compare alternative configurations with respect to performance, resource usage, power consumption, etc.

For the particular application that is described here, we have so far designed one type of hardware accelerator, which calculates the 2-norm of the error signal $e(t) = y_{ref}(t) - y(t)$ for given values of K_p, K_i, K_d and q. Overall, the online module operates in the following way:

1. The microcontroller acquires some estimate of the unknown parameter vector q.

2. If this estimate is stable enough then the microcontroller starts searching the controller parameter space (which is precomputed off-line by using simulation) for some better values of K_p, K_i and K_d.

3. For each candidate (K_p, K_i, K_d) the microcontroller invokes the hardware accelerator in order to assess the performance of the corresponding controller.

4. At the end of the search procedure, if a controller that is better than the robust one has been found then this controller becomes the controller of the vehicle.

5. If the parameter estimates change then the robust controller takes over again and the whole procedure is repeated.

The estimate of the vehicle speed u can be a direct measurement read by the FPGA via an appropriate interface (A/D converter). The estimates of m and μ should be calculated by separate processors or dedicated hardware accelerators. As explained in [6], the use of accelerators is preferable because it avoids unnecessary duplication of hardware. As will be demonstrated below, many modern FPGAs offer enough resources to support all the required functionality within the same FPGA unit. Alternatively, the estimators can be implemented on a separate platform, which will communicate with the main unit via a standard interface.

3.2 Controller Optimization

Calculation of $\|e(\cdot)\|_2$ for particular values of K_p, K_i, K_d and q requires the solution of the closed-loop state equation, which is generally of the form

$$\begin{aligned} \dot{x}_{cl}(t) &= A_{cl}x_{cl}(t) + b_{cl}y_{ref} + D_{cl}\gamma(t) \\ y(t) &= c_{cl}^T x_{cl}(t) \end{aligned} \quad (3)$$

where y_{ref} is the reference value of $y(t)$ (assumed constant), $\gamma = \begin{bmatrix} f_w & \rho_{ref} \end{bmatrix}$ and A_{cl}, b_{cl}, c_{cl} and D_{cl} are matrices of appropriate dimensions. The calculations can be greatly simplified if γ can be considered constant for the duration of the transients. Indeed, in this case

$$y(t) = c_{cl}^T e^{A_{cl}t}\left(x_{cl,0} - x_{cl,\infty}\right) + y_\infty$$

where $x_{cl,\infty} = -A_{cl}^{-1}\left(b_{cl}y_{ref} + D_{cl}\gamma\right)$ and $y_\infty = c_{cl}^T x_{cl,\infty}$ are the final values of $x_{cl}(t)$ and $y(t)$, respectively, and we have taken into account the fact that the closed-loop system is asymptotically stable. Furthermore, since the PID controller achieves zero steady-state error, $y_\infty = y_{ref}$ and so $e(t) = c_{cl}^T e^{A_{cl}t}\left(x_{cl,\infty} - x_{cl,0}\right)$. Hence,

$\left\|e(\cdot)\right\|_2^2 = \int_0^\infty e(t)^2\,dt = \left(x_{cl,\infty} - x_{cl,0}\right)^T X \left(x_{cl,\infty} - x_{cl,0}\right)$ where $X = \int_0^\infty e^{A_{cl}^T t} c_{cl} c_{cl}^T e^{A_{cl}t} dt$ can be obtained by solving the Lyapunov equation $A_{cl}^T X + X A_{cl} + c_{cl}c_{cl}^T = 0$. Now if we choose $y_{ref} = 0$ and $x_{cl,0} = 0$ then $\left\|e(\cdot)\right\|_2^2 = \gamma^T \Lambda \gamma$ where $\Lambda = D_{cl}^T A_{cl}^{-T} X A_{cl}^{-1} D_{cl}$ is a 2x2 matrix. Explicit formulas for the entries of $A_{cl}^{-1}D_{cl}$ can be derived by setting $\dot{x}_{cl} = 0$ in (3) and solving for the steady-state values $x_{cl,\infty}$. So once we have X the calculation of $\left\|e(\cdot)\right\|_2$ for any given γ is a trivial matter. Alternatively, one can use the largest eigenvalue of Λ as a performance measure because $\left\|e(\cdot)\right\|_2^2 \le \lambda_{\max}(\Lambda)\gamma^T\gamma$ by the Rayleigh-Ritz theorem.

In our implementation we used the state vector $x_{cl}^T = \begin{bmatrix} x^T & \dot{\delta}_f \end{bmatrix}^T$ and exploited the sparse structure of the matrices in order to solve the problem efficiently. The Lyapunov equation consists of 21 scalar equations. Two of them have only one variable and thus they can be directly solved. By a rearrangement of rows and columns, the remaining 19 can be put in the form (4) where the sizes of ξ_i, $i = 0\ldots5$ are 6, 4, 3, 1, 3, 2, respectively and D_i, $i = 0\ldots5$ are invertible diagonal matrices.

$$\begin{bmatrix} K_{00} & K_{01} & K_{02} & K_{03} & K_{04} & K_{05} \\ K_{10} & D_1 & 0 & 0 & 0 & 0 \\ K_{20} & K_{21} & D_2 & 0 & 0 & 0 \\ K_{30} & K_{31} & K_{32} & D_3 & 0 & 0 \\ K_{40} & K_{41} & K_{42} & K_{43} & D_4 & 0 \\ K_{50} & K_{51} & K_{52} & K_{53} & K_{54} & D_5 \end{bmatrix} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \\ \xi_5 \end{bmatrix} = \begin{bmatrix} \kappa_0 \\ \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \kappa_4 \\ \kappa_5 \end{bmatrix} \quad (4)$$

Equation (4) can be efficiently solved as follows:

1. For $i = 1\ldots5$ compute $\sigma_i = D_i^{-1}\left(\kappa_i - \sum_{j=1}^{i-1} K_{ij}\sigma_j\right)$ and $\Sigma_i = -D_i^{-1}\left(K_{i0} + \sum_{j=1}^{i-1} K_{ij}\Sigma_j\right)$.

2. Solve the 6x6 system $\Sigma_0\xi_0 = \sigma_0$ where $\Sigma_0 = K_{00} + \sum_{j=1}^5 K_{0j}\Sigma_j$ and $\sigma_0 = \kappa_0 - \sum_{j=1}^5 K_{0j}\sigma_j$

3. For $i = 1\ldots5$, compute $\xi_i = K_i\xi_0 + \kappa_i$

The calculations can be further simplified by noticing that K_{ij} and κ_i contain many zeros whereas most diagonal entries of D_i are 1. In order to avoid trivial arithmetic in the hardware, we wrote a Matlab script which examined each entry of the matrices and generated C statements for the necessary computations only.

3.3 Hardware Design

The presented solution was implemented as a hardware design for a number of different FPGAs by using the tool flow by Xilinx (Vivado HLS, Embedded Development Kit, Xilinx Platform Studio, and Software Development Kit). In most cases the microcontroller was a MicroBlaze written in HDL (vendor provided) with floating point unit except for the Zync7 boards, which are equipped with an ARM dual-core microprocessor. The accelerator was created by Vivado HLS from the C code produced by our Matlab script and packed as an independent hardware component (a Pcore in Xilinx terminology). All communication between the microcontroller and the Pcore was done over a standard Axi bus. The calculations used dual precision floating point arithmetic.

The microcontroller generated candidate PID controllers using a random search strategy. The use of some more organized method is currently under investigation. For each candidate controller, the accelerator computed a performance score as described in Section 3.2. The search stopped after a prespecified number of trials.

Another point worth mentioning is that $A_{cl}^{-1}D_{cl}$ has only 6 nonzero entries and they do not depend on the PID parameters. Thus they are computed only once, by the microcontroller and transferred to the accelerator. Furthermore, the sparse structure of the matrix was properly taken into account for the computation of Λ.

Table 1 shows the performance of the accelerator on different FPGA families. For each family, the largest device has been selected. For each device we provide estimated min and max values of latency and throughput for two clock frequencies and for two different designs: one with pipeline optimizations (indicated by the letter p after the clock frequency) and one without. As we can see, all the implementations of the accelerator can complete the required calculations in a few μsec. Therefore even if we use only one accelerator, it is possible to complete the search in much less than a sec. Of course one can put multiple accelerators working in parallel for even faster results. Table 2 shows the corresponding resource utilization for each design. For each resource type (lookup tables, flip flops, DSP units, block ram) the figures in the table indicate percentage of the available amount rounded to the nearest integer. In most cases the accelerator uses only a small part of the available resources. This means that there is enough space in the FPGA for implementing several accelerators,

either of the same type for increased performance or of different types for more functionality (e.g. parameter estimators). The only exception is the pipelined 200 MHz design, which cannot fit into the device because it needs more than the available DSP blocks. In all other cases the pipelined versions require more memory elements (FFs and BRAMs) and more logic (LUTs) but fewer DSPs. This happens because folding the algorithm (pipelining) results in more sharing of expensive FPGA blocks.

Table 1. Accelerator performance

FPGA	Latency (μsec)		Throughput (μsec)	
	Min	max	min	max
Virtex7-xc7v2000 100 MHz	5.96	9.02	5.97	9.03
Virtex7-xc7v2000 100 MHz p	6.81	6.81	2.76	2.76
Virtex7-xc7v2000 200 MHz	5.61	7.92	5.61	7.92
Virtex7-xc7v2000 200 MHz p	6.34	2.38	6.34	2.38
Kintex7-xc7k480 100 Mhz	5.78	8.94	5.79	8.85
Kintex7-xc7k480 100 Mhz p	6.4	6.4	2.62	2.62
Kintex7-xc7k480 200 Mhz	5.23	7.39	5.23	7.39
Kintex7-xc7k480 200 Mhz p	5.74	5.74	2.18	2.18
Artix7-xc7a200 100 Mhz	6.74	10.1	6.75	10.11
Artix7-xc7a200 100 Mhz p	7.9	7.9	3.07	3.07
Artix7-xc7a200 200 Mhz	6.66	9.72	6.66	9.72
Artix7-xc7a200 200 Mhz p	8.035	8.035	3	3
Zynq-xc7z100 100 Mhz	7.41	11.07	7.42	11.08
Zynq-xc7z100 100 Mhz p	9.2	9.2	3.6	3.6
Zynq-xc7z100 200 Mhz	6.88	9.74	6.88	9.74
Zynq-xc7z100 200 Mhz p	8.36	3.14	8.36	3.14
Virtex6-xc6vlx760 100 Mhz	6.35	9.41	6.36	9.42
Virtex6-xc6vlx760 100 Mhz p	7.52	7.52	2.96	2.96
Virtex6-xc6vlx760 200 Mhz	6.48	8.94	6.48	8.94
Virtex6-xc6vlx760 200 Mhz p	7.83	7.83	2.91	2.91
Spartan6xc6slx150 100 Mhz	9.43	13.09	9.44	13.1
Spartan6xc6slx150 100 Mhz p	12.55	12.55	4.95	4.95
Spartan6xc6slx150 200 Mhz	8.85	12.81	8.86	12.82
Spartan6xc6slx150 200 Mhz p	11.63	11.63	4.5	4.5

Table 2. Resource usage

FPGA	LUT	FF	DSP	BRAM
Virtex7-xc7v2000 100 MHz	2	0	7	0
Virtex7-xc7v2000 100 MHz p	3	1	5	0
Virtex7-xc7v2000 200 MHz	2	1	7	0
Virtex7-xc7v2000 200 MHz p	2	1	3	0
Kintex7-xc7k480 100 Mhz	6	3	8	0
Kintex7-xc7k480 100 Mhz p	7	5	6	0
Kintex7-xc7k480 200 Mhz	6	4	6	0
Kintex7-xc7k480 200 Mhz p	6	5	2	0
Artix7-xc7a200 100 Mhz	25	7	21	0
Artix7-xc7a200 100 Mhz p	27	10	9	2
Artix7-xc7a200 200 Mhz	23	10	14	0
Artix7-xc7a200 200 Mhz p	27	14	7	2
Zynq-xc7z100 100 Mhz	15	4	8	0
Zynq-xc7z100 100 Mhz p	15	6	4	0
Zynq-xc7z100 200 Mhz	14	5	7	0
Zynq-xc7z100 200 Mhz p	16	7	7	0
Virtex6-xc6vlx760 100 Mhz	6	2	17	0
Virtex6-xc6vlx760 100 Mhz p	7	3	12	1
Virtex6-xc6vlx760 200 Mhz	6	3	18	0
Virtex6-xc6vlx760 200 Mhz p	7	4	6	1
Spartan6xc6slx150 100 Mhz	53	18	93	1
Spartan6xc6slx150 100 Mhz p	60	26	55	5
Spartan6xc6slx150 200 Mhz	53	29	102	1
Spartan6xc6slx150 200 Mhz p	56	33	46	5

Acknowledgement

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: ARCHIMEDES III. Investing in knowledge society through the European Social Fund. (ARCHIMEDES III-STRENGHTENING RESEARCH GROUPS IN TECHNOLOGICAL EDUCATION, NSRF 2007-2013).

References

1. S. Gosh, et al. “An FPGA based implementation of a flexible digital PID controller for a motion control system”, *IEEE International Conference on Computer Communication and Informatics* (2013).
2. D. Navarro, et al ,“High-level synthesis for accelerating the FPGA implementation of computationally-demanding control algorithms for power converters”, IEEE Trans. Ind. Inf., **9**, pp. 1371–1379 (2013).
3. E. Monmasson, M. N. Cirstea, “FPGA design methodology for industrial control systems - a review,” IEEE Trans. on Ind. Electr., **54**, pp. 1824–1842, (2007).
4. S. B. Othman, A. K. B. Salem, H. Abdelkrim, S. B. Saoud, “MPSoC design approach of FPGA-based controller for induction motor drive”, *IEEE Internat. Conf. on Indust. Technology*, pp. 134–139 (2012).
5. C. Economakos, H. Sidiropoulos, G. Economakos, " Rapid Prototyping of Digital Controllers using FPGAs and ESL/HLS Design Methodologies", *19th IEEE Int. Conf. Autom. and Computing.* (2013).
6. C. Economakos, M. Tzamtzi, M. Skarpetis, G. Economakos, "Performance Improvements in a Modern Hardware Design Environment for Control Applications", *IEEE Int. Conf, Ind. Technol.*, (2015).
7. J. Ackermann and W. Darenberg, “Automatic track control of a city bus”, *IFAC Theory Rep. Benchmark Problems Contr. Syst. Design*, (1990).
8. J. Ackermann, J. Guldner, W. Sienel, R. Steinhauser, and V. I. Utkin, “Linear and nonlinear controller design for robust automatic steering” IEEE Trans. Contr. Syst. Technol., **3**, 1, pp. 132–143 (1995).
9. J. Ackermann, J. Guldner, W. Sienel, R. Steinhauser, and V. I. Utkin, “Linear and nonlinear controller design for robust automatic steering” IEEE Trans. Contr. Syst. Technol., **3**, 1, pp. 132–143 (1995).
10. F. N. Koumboulis, M. G. Skarpetis, and G. Panagiotakis, “Robust Disturbance Rejection via P-D State Feedback – Example for Lane Keeping of Highway Vehicles", Trans. Inst. of Measurement & Control (2005).
11. K Wei, B. and R. Barmish, “Making a polynomial Hurwitz invariant by choice of feedback gain”, Int. J. Contr. **50**, pp 1025-1038,(1989).
12. F. N. Koumboulis and M. G. Skarpetis, “Robust Control of Pneumatic Clutch Actuators using Simulated Annealing Techniques”, *IEEE MED* (2013).
13. M.G. Skarpetis, F.N. Koumboulis, and A.S. Ntellis, "Robust Tracking and Disturbance Attenuation Controllers for Automatic Steering", *IEEE MED* (2009).