Autonomous SoC for Fuzzy Robot Path Tracking

Kyriakos M. Deliparaschos, George P. Moustris and Spyros G. Tzafestas

Abstract—**In this paper a System-on-a-Chip (SoC) for the path following task of autonomous non-holonomic mobile robots is presented. The SoC consists of a digital fuzzy logic processor and a flow control program that runs under the Xilinx Microblaze™ soft processor core. The fuzzy processor implements a fuzzy path tracking algorithm introduced by the authors. The system was tied to an actual P3-DX8 robot and field experiments have been performed in order to assess the overall performance. Quantization problems and limitations imposed by the system configuration are also discussed.**

Index Terms—**Non-Holonomic Mobile Robots, Path Tracking, Digital Fuzzy Logic Processor (DFLP), System-on-a-Chip (SoC), Intellectual Property (IP) core.**

I. INTRODUCTION

This paper presents a System-on-a-Chip (SoC) implementation for robot path tracking. The SoC mainly consists of a parameterized digital fuzzy logic processor (DFLP) [1] Intellectual Property (IP) core implementing the fuzzy tracking algorithm and a Xilinx Microblaze™ soft processor core as the top level flow controller. The SoC was tied to an actual differential-drive Pioneer 3-DX8 robot that has been used in experiments of the tracking scheme. Similar use of hardware designs on field programmable gate array (FPGA) chips in robotic applications have been considered by several other researchers [16-19] since FPGAs provide several advantages over single processor hardware, on the one hand, and custom dedicated hardware (ASIC) on the other. FPGAs give a faster time-to-market, have simpler and more predictable design cycle, and offer field re-programmability, to name a few. A review of the application of FPGAs in robotic systems is provided be Leong and Tsoi in [16]. A notable case study is the use of FPGA's in the Mars Pathfinder, Mars Surveyor '98 and Mars Surveyor '01 lander crafts, analyzed in [19].

In our application the DFLP facilitates scaling and can be configured for different number of inputs and outputs, number of triangular or trapezoidal fuzzy sets per input, number of singletons per output, antecedent method (t-norm. s-norm), divider type, and number of pipeline registers for the various components in the model. This parameterization enabled the creation of a generic DFLP IP core that was used to produce a fuzzy processor of different specifications without the need of redesigning the IP core from the beginning. The fuzzy logic processor architecture assumes overlap of two fuzzy sets between adjacent fuzzy sets and requires 2^n clock cycles (input data processing rate), where n is the number of inputs, since it processes one active rule per clock cycle. At the present paper the SoC design achieves a core frequency speed of 71 MHz. The input data to the DFLP IP core can be sampled at a clock rate equal to 1/4n of the core frequency speed and processing is performed accounting for only the active rules. To achieve this timing result the latency of the chip architecture involves 9 pipeline stages each one requiring 14.085 ns. The featured DFLP IP core is based on a simple algorithm similar to the zero-order Takagi-Sugeno inference scheme and the weighted average defuzzification method, and using the chosen parameters (see Table II (a)), employs two 12-bit inputs and one 12-bit output, with up to 9 trapezoidal or triangular shape membership functions per input with a degree of truth resolution of 8 bits and a rule base of up to 81 rules.

The fuzzy tracking algorithm used, is based on a previous fuzzy path tracker developed by the authors [2]. The fuzzy logic (FL) tracker has undergone some alterations due to the hardware restrictions posed by the DFLP IP core. While the original fuzzy logic controller (FLC) was Mamdani-based with Gaussian membership functions, the one deployed here is a Takagi-Sugeno zero-order type FLC with triangular membership functions and an overlap of two between adjacent membership functions. Besides the FLC, the "spatial window" technique that was also introduced in the previous paper has been incorporated in the tracking scheme.

II. OVERVIEW OF THE SYSTEM

The system consists of four parts that have been tied together. An overview can be seen in Fig. 1, and an actual depiction in Fig. 11.

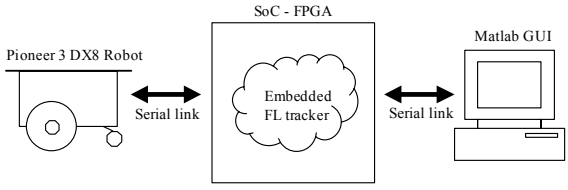

Fig. 1. Overview of the system.

The FPGA SoC implements the autonomous control logic of the P3 robot. It receives odometry information from the robot and issues steering commands outputted by the FL tracker. The SoC realizes several other tasks besides the steering control; it decodes the information packets sent by

Manuscript received October 16, 2006.

G. Moustris, K.M. Deliparaschos and S. G. Tzafestas are with the School of Electrical and Computer Engineering, National Technical University of Athens, Intelligent Robotics and Automation Laboratory, Zographou Campus, Athens, GR 157 73 (phone: +30-210-7721527, fax:+30-210-7722489, e-mails: gmoustri@central.ntua.gr, kdelip@mail.ntua.gr, tzafesta@softlab.ntua.gr)

the robot which include the pose estimation done by the robot, the status of the motors, sonar readings etc, and encodes the steering commands in a data frame that is accepted by the robot. In other words, the SoC implements a codec for the I/O communication with the P3 robot. Furthermore, it also relays some critical information to a Matlab monitoring program that has been developed. The top-level program that attends to all these tasks is written in C and executed by the Microblaze™ soft processor core. This top-level program is also treating synchronization and timing requirements.

The Matlab application displays information about the robot's pose and speed, as estimated by the robot's odometry, as well as some other data used for the path tracking control. It also calculates the robot's position relative to the world and the local coordinate systems. Another important function of the Matlab application is to provide a path for the robot to track. The current paper deals only with the path tracking task and not path planning. To compensate for this, the path is drawn in Matlab, encoded properly and downloaded to the SoC. Then, the SoC begins the tracking control.

The test platform on which the SoC is tested on is the ActivMedia P3-DX8 robot [4]. The robot uses a 1 mm resolution for the position estimation and 1°angle resolution for the heading. The kinematics of the robot are emulated to that of a bounded curvature vehicle and not of a differential drive one i.e., there is an imposed constraint on the maximum curvature it can turn with. This has been introduced because the fuzzy tracking algorithm is intended for the Dubin's Car model [3] where there is a minimum turning radius constraint on the robot and only forward motion. As it will be explained in a later section, the curvature constraint along with the one degree resolution of the P3 robot presents a quantization problem on the curvature.

The robot is connected to the FPGA through a serial cable through which it sends and receives framed data. ActivMedia has produced its own data frame that is encoded into the robot's microcontroller. The data that are sent from the robot are named Server Information Packets (SIP packets) while the data that are received are named Command Packets. Both data frames can be found in the robot's manual [5]. The experiments clearly showed that even though the FL tracker performs well, its actual performance is severely degraded by the odometry's accumulation of errors over time. Several calibration tests have been carried out in order improve odometry localization but, as it was expected, position estimation through odometry proved inefficient.

I. HARDWARE HIGH LEVEL SYSTEM VIEW

A high level view of the proposed SoC architecture is shown in Fig. 2. The Microblaze™ soft processor core [6] is licensed as part of the Xilinx Embedded Development Kit (EDK) [7]. The processor is a soft core, meaning that it is implemented using general logic primitives rather than a hard dedicated block on the FPGA.

The Microblaze™ processor is based on a RISC architecture which is very similar to the DLX architecture described in [8]. It features a 3-stage pipeline with most instruction completing in a single cycle. Both the instruction and data words are 32-bits. The core alone can obtain a speed of up to 100MHz on the Spartan 3 FPGA family. The processor can connect to the OPB bus [9] for access to a wide range of different modules, it can communicate via the LMB bus [11] for a fast access to local memory, normally block RAM (BRAM) inside the FPGA.

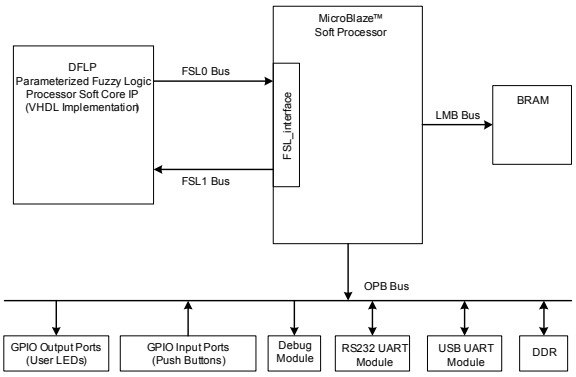

Fig. 2. SoC high-level hardware system view.

Moreover, the Fast Simplex Link (FSL) [10] offers the ability to connect user IP cores acting as co-processors to accelerate time critical algorithms. The FSL channels are dedicated unidirectional point-to-point data streaming interfaces. Each FSL channel provides a low latency interface to the processor pipeline allowing extending the processor's execution unit with custom soft core co-processors. In this paper the DFLP IP core is playing the role of such a co-processor and is connected to the Microblaze™ via the FSL bus [13].

The architecture of the present SoC consists mainly of the DFLP IP core that communicates with the Microblaze™ Processor through the Fast Simplex Bus (FSL), the utilized block RAMs (BRAM) through the LMB bus, and other peripherals such as the general purpose input/output ports (GPIO), and UART modules via the OPB bus. Here, the DFLP incorporates the fuzzy tracking algorithm, whereas the Microblaze™ processor mainly executes the C code for the flow control.

II. FUZZY LOGIC PATH TRACKING ALGORITHM

The FL tracking algorithm is based on [2], although some modifications where appropriately made in order to tailor it to the FPGA hardware platform available. The original algorithm consists of a 9x9 Mamdani FL tracker with Gaussian input and output membership functions. In this work, the tracker was converted to a zero-order Takagi-Sugeno FLC with triangular membership input functions and an overlap of two. It is worth noting here that the two controllers where tested in simulation against each other and the Takagi-Sugeno FLC outperformed the Mamdani controller.

The FL tracker uses two angles as inputs, and outputs the curvature κ with which the robot has to turn. It is assumed

that the path is provided as a sequence of points in $\mathbb{R}^2$ following a fixed sampling spacing Δs. In each control loop the closest path point is picked up and the two input angles are calculated. These angles are the angle φ_1 of the closest point with respect to the current robot heading and the direction φ_2 of the tangent of the path at that point, as depicted in Fig. 3.

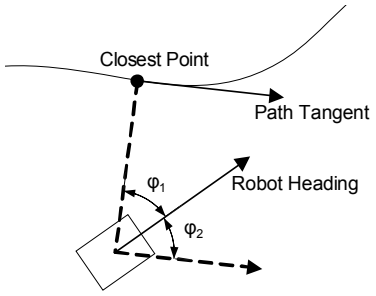

Fig. 3. Illustration of the controller inputs.

The authors have also introduced in [2] a technique called "spatial window" that enhances the path tracking control. This technique is based on the idea of having a *spatial window* on the path rather just a single point. In this arrangement a window of the path is used in order to calculate the steering command, thus introducing a "perspective" to the controller. The "spatial window" is defined by three parameters; the *window order*, which is the number of points used in the window, the *window step*, which is the number of points skipped for each active point, and the *window offset* which is the number of points, counting from the closest, that moves the window forward on the path. These parameters are depicted in Fig. 4.

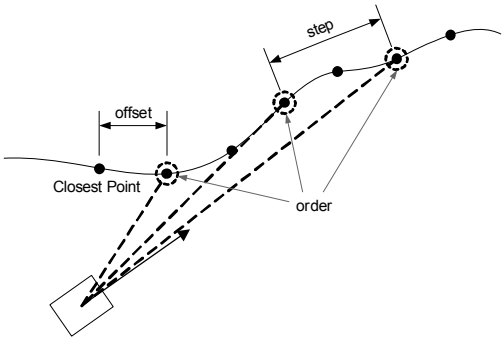

Fig. 4. Illustration of the *spatial window*.

Every point of the window is presented to the controller and an appropriate curvature is calculated for each one. Consequently, there are n computed curvatures for each control loop, where n is the *window order*. From these curvatures a final output curvature must be calculated. The simplest method, which was exclusively used in this paper, is the mean of the curvatures i.e.,

$$\kappa = \frac{\kappa_1 + \kappa_2 + \ldots + \kappa_n}{n} \quad (1)$$

The spatial window technique provides a smoother path tracking control and an overall better performance. It also introduces some other interesting attributes that are analyzed in [2] mainly high robustness and path "denoising".

III. TOP-LEVEL CONTROL PROGRAM

The top-level program that coordinates all actions is written in C and runs in the FPGA under the Microblaze™ soft processor core. For the communication with the outside world it uses two I/O channels, one serial and one USB, having 16-byte input and output buffers. The flow chart of the main program is illustrated in Fig. 5.

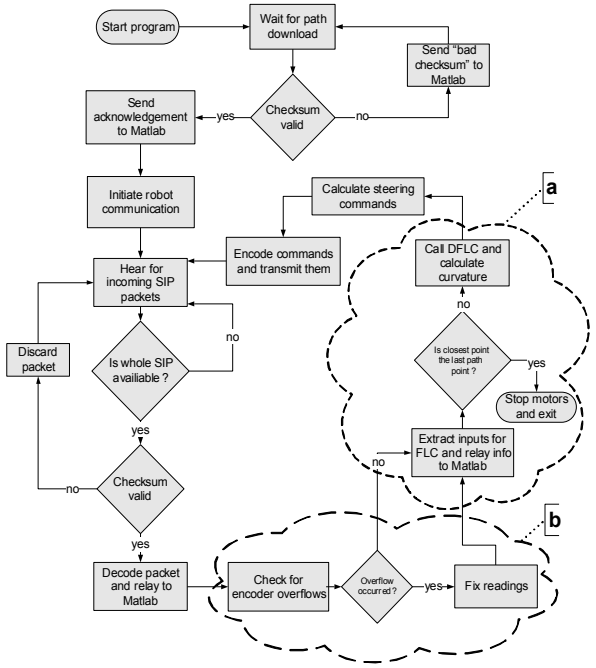

Fig. 5. Main program flow-chart.

At the beginning, the program waits for the reference path to download. If the checksum is valid it continues execution, opens the connection to the robot and performs synchronization by sending specially formed packets. At this point the robot starts sending the SIP packets to the FPGA.

Next, the program enters into the main control loop, which can be seen in the flowchart and, it hears for incoming SIP packets by flushing the input buffer to an array variable. As soon as a whole valid SIP packet is present, is decoded and odometry information is extracted (x, y, θ). At the same time, the packet is relayed to the Matlab environment. Due to the small word length that is used in the SIP packet for data encoding along with the high odometry resolution (millimeter), the encoder readings cannot be readily used for localization. All data in the SIP packet are encoded in signed 16-bit integers. The data in the odometry packets are multiplied by a scaling factor 0.485, which yields an effective range of -15892 to 15892 mm i.e., -16 to 16 meters, which is rather limited. To overcome this obstacle, the program uses two internal coordinate variables of type double, which maximize the range of the localization. Essentially the program calculates the difference in the coordinates' change at each step and adds it to an absolute coordinate variable. This also incorporates a "fix" algorithm since the 16-bit integer coordinates overflow when their range is exceeded (group (b) of Fig. 5).

After the calculation of odometry data, the program execution passes to the algorithm responsible for the path tracking control (group (a) of Fig. 5). This algorithm implements the spatial window technique of the path tracker. It picks up the closest path point to the robot, calculates the two controller inputs ϕ_1^j, ϕ_2^j of point j of the window and calls the DFLP for each set of inputs. Upon completion it outputs the mean of all computed curvatures κ_j and returns to the main loop.

In order to find the closest path point, the square of the Euclidean L2 norm is being used due to the fact that the square root calculation of the standard Euclidean distance is computationally costly hence is being avoided in this way. Moreover the variables used to compute this norm are of type long long int (64-bit integer) instead of type double because code execution is much faster this way when executed in the Microblaze™ processor without the use of the FPU unit. This is of crucial importance since the main algorithm bottleneck takes place in this operation. If one considers the fact that the downloaded path can consist of hundred of sample points, the algorithm must cycle through all of them in order to find the closest one. The latter can delay the execution long enough so that the packets sent by the robot are not flushed fast enough to the variable that contains the data, resulting to fragmented SIP packets. This leads to the loss of the synchronization between the FPGA and the robot because no whole SIP packet can be reconstructed from the FPGA.

Following the calculation of the steering curvature from the previous routine, the steering commands must be computed. Since the Pioneer robot does not have an explicit command regarding the curvature a turn of predefined curvature must be implicitly issued through the combination of two other commands; one regarding the linear velocity and one regarding the angular velocity. Curvature κ is defined by:

$$\kappa=\frac{d\theta}{ds}=\frac{d\theta/dt}{ds/dt}=\frac{\omega}{\upsilon} \tag{2}$$

where θ is the steering angle of the robot, s the traveled distance, ω the angular velocity (rad/sec) and υ the linear velocity (m/sec). Robot command packets use linear velocity in mm/sec and angular velocity in deg/sec. Suppose that the actual curvature to be followed is κ in *rad/m*. If the linear velocity υ' (*mm/sec*) is given, then the angular velocity ω' (*deg/sec*) that must be issued as a command is calculated by:

$$\kappa=\frac{\omega}{\upsilon}\left(\frac{rad}{m}\right)=\frac{\omega'(\mathrm{deg/sec})}{\upsilon'(mm/\mathrm{sec})}\frac{\pi/180}{/1000}\Rightarrow\omega'=\kappa\cdot\upsilon'\cdot\frac{180}{1000\cdot\pi} \tag{3}$$

However, the actual curvature κ outputted by the DFLP is an integer ranging from -2048 to 2047 (12-bit resolution). This curvature is normalized to [-1, 1] and multiplied by a user-defined maximum curvature κ_{max} since the differential drive does not have bounds on the turning radius. In this way, one can control the minimum turning radius that can be steered by the P3 robot. Thus, (3) is transformed to:

$$\omega'=\frac{\kappa}{2048}\kappa_{\max}\cdot\upsilon'\cdot\frac{180}{1000\cdot\pi} \tag{4}$$

As soon as the robot's velocity[1] υ' is extracted from the SIP packet and the angular velocity ω' is calculated from (4), the two motion commands are transmitted back to the robot resulting in an augmented curvature-ruled motion. One can also note that the velocity is decoupled from the tracking control since the control input is the curvature. The velocity can be controlled from a speed controller independently. This path tracking control is actually a *geometric* tracking control since the curvature completely defines the robot's route. In other applications, such as kinodynamic tracking where the dynamics of the robot are taken into account, one must also incorporate the velocity into the control loop. However in this work the velocity control is not necessary for the tracking problem under the assumption that the speed of the robot is small enough such that the dynamics do not affect the kinematic behaviour.

In an actual car-like robot the curvature constraints and the actual motion are mechanically imposed, not emulated in the manner described here. The way that the curvature is being emulated in this work, presents us with another problem that is analyzed in the following section.

IV. QUANTIZATION ISSUES

The P3 robot makes use of 16-bit integer to encode SIP packet data. Accordingly, it also uses 16-bit integers to encode command arguments in a command packet. However, the range of the angular velocity command is [0, 300] deg/sec with one deg/sec/bit. Since ω' can take only integer values with one deg/sec resolution, this imposes a quantization on the curvature as well. Solving. (3) for κ (or R=1/κ, the turning radius in meters), one gets (5):

$$R=\frac{\upsilon'}{\omega'}\cdot\frac{180}{1000\cdot\pi} \tag{5}$$

Plotting the radius R versus the quantized angular velocity ω' for $\upsilon'=100\ mm/\mathrm{sec}$ (see Fig. 7) one can see that the resolution towards small curvatures (big radii) is poor going from $R=5.73$ ($\omega'=1$) to $R=2.86$ ($\omega'=2$) to $R=1.91$ ($\omega'=3$) while the resolution towards big curvatures (small radii) is high. This has a severe impact on the way the controller can follow the path. When the robot is on the path i.e., $\phi_1, \phi_2 \approx 0$ the FLC issues commands of small curvature (big radius) to avoid "nervous" steering, e.g. oscillations. If the resolution in the range of small curvatures is poor, as described above, the control is degraded since the curvature commands are clipped to the available resolution levels. Furthermore, since there is a constraint on the minimum turning radius, all values of Fig. 6 below the lower bound R_{min}, which was set to 1m in our case studies, are unattainable by the robot. For $\upsilon'=100\ mm/\mathrm{sec}$, there are only six radii over 1m i.e. only six curvatures the robot can

[1] The P3 robot uses its own controller for velocity control. When the user issues a velocity command, the robot uses a velocity profile algorithm to acquire the prescribed speed. Thus, the speed used in the calculation of (4) is not the set-point speed but the actual robot speed as reported back from the robot in the latest SIP packet.

turn with. Since R is monotonically decreasing with respect to ω' and the latter starts from 1deg/sec, one can find the number of available quantization levels by substituting R=1 in (5) and solving for ω', i.e.,

$$L_{num}=\left\lfloor \upsilon' \cdot \frac{180}{1000 \cdot \pi} \right\rfloor \quad (6)$$

, where $\lfloor\ \rfloor$ is the floor function.

By inspection of (5) and (6) one can see that by increasing the velocity υ', the maximum turning radius increases linearly along with the resolution in that range i.e., the L_{num}.

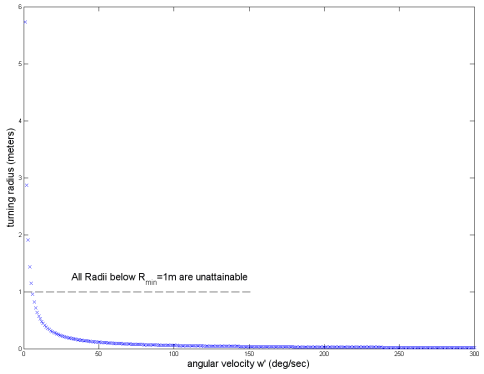

Fig. 6. Illustration of the quantization of the actual turning radius for υ'=100 mm/sec.

The obvious solution for the increase of the curvature resolution is to increase the speed. Of course, this raises the problem of finding an appropriate robot speed since low speed reduces the curvature resolution but a high speed might result to an unresponsive system. To estimate an acceptable error level between the curvature computed by the FLC and the actual curvature the robot follows, we draw the maximum relative error versus all available speeds over all available inputs i.e.,

$$100 \cdot \max(\kappa_{FLC} - \kappa_{ACTUAL}) / \kappa_{ACTUAL}, \forall \varphi_1, \varphi_2 \quad (7)$$

In this way one can see the maximum possible relative error for each speed between the actual and the desired curvature. This error is illustrated in Fig. 7. The minimum turning radius was set to one meter (κ_{max}=10^{-3}).

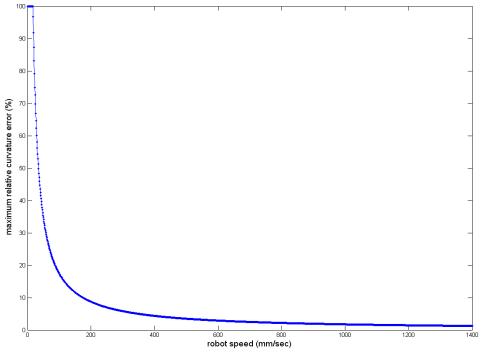

Fig. 7. Maximum relative error between actual and desired turning curvature versus speed, over all possible inputs of the controller.

As one can see, as the speed increases the error decreases. An acceptable trade-off speed seems to be 1000 mm/sec (or 1 m/sec) where the error drops below 1.745%. For this speed, the available quantization levels, as derived from (6) are L_{num}=57. As a result, the robot's speed was set to 1000 mm/sec in all field experiments.

V. SoC HARDWARE ARCHITECTURE

The SoC design presented in this work was implemented on the Spartan-3 MB development kit (DS-KIT-3SMB1500) by Memec Design [14].

TABLE I
FPGA PLATFORM KEY FEATURES

Xilinx XC3S1500-4FG676C Spartan-3 FPGA
16 M x 16 DDR memory, 2 M x 16 flash memory
Platform Flash ISP PROMs
10/100 Ethernet PHY, USB 2.0 and RS232
2 7-segment LED displays
4 User LEDs, 2 Push Buttons, 8 Pos. Dip Switches
On-board clock oscillator
JTAG configuration port
75 MHz Clock Oscillator
2 x 16 Character LCD
Two P160 expansion slots
System ACE/User I/O Header
LVDS tx/rx interface

The Spartan-3 MB system board utilizes the 1.5million-gate Xilinx Spartan-3 device (XC3S1500-4FG676) in the 676-pin fine-grid array package. The key features of the selected FPGA platform are synopsized in Table I.

The component pipeline registers (CPR) blocks on Fig. 10 indicate the number of pipeline stages for each component; the "PSR" blocks indicate the path synchronization registers, while the "U" blocks represent the different components of the DFLP IP core. The U_fpga_fc component is embedded in flc_ip top structural entity wrapper, which basically provides all the necessary peripheral logic to the DFLP IP core in order to be able to receive/send data to the FSL bus and thus be compliant with the FSL bus standard. The flc_ip wrapper architecture is also shown in Fig. 10. The chosen (generic) parameters (generics definition VHDL package) for the parameterized DFLP IP core (U_fpga_fc) and its characteristics are summarized in Table II (a), (b) respectively. The U_fpga_fc alone was synthesized using Synplify Pro synthesizer tool, while the rest of the design components were synthesized by Xilinx Synthesis Tool (XST) through the EDK Platform Studio. The produced .edf file for the U_fpga_fc is been seeing by the flc_ip wrapper as a blackbox during the XST flow.

The placement and routing of the SoC design into the FPGA was done through the EDK by calling the Xilinx ISE tool. Based on the total device utilization (see Table IV), the DFLP IP itself occupies 1600 (6%) LUTs, 4 Block Multipliers (MULT18X18s), 12 64x1 ROMs (ROM64X1), and 56 256x1 ROMs (ROM256X1). The implemented design uses two Digital Clock Manager (DCM) Modules (DCM_0 and DCM_1) for the different clocks production in the FPGA. The SoC achieves a system clock's (DCM_0) operating frequency of 14.140ns or ~71 MHz, while DCM_1 is mainly used for clocking the external DDR RAM. The FPGA device utilization summary after the design has been placed and routed is shown in Table III.

TABLE II (A)
DFLP SOFT CORE IP CHOSEN PARAMETERS

Parameters (VHDL generics)	Value	Generic Description
ip_no	2	Number of inputs
ip_sz	12	Input bus width (bits)
op_no	1	Number of outputs
op_sz	12	Output bus width (bits)
FS_no	9	Number of membership functions (same for all inputs)
dy	8	Degree of Truth width
sel_op	0	Antecedent method connection: 0 : min, 1: prod, 2: max, 3: probor
div_type (Divider Model)	1	0 : restoring array 1 : LUT reciprocal approximation
PSR		**Signal Path Route**
psr1_no	1	ip_set→psr1_no→trap_gen_p
psr2_no	4	s_rom→psr2_no→mult
psr3_no	1	s_rom→psr3_no→rul_sel_p
psr4_no	1	cpr5→psr→int_uns
CPR		**Component (Entity) Name**
cpr1_no	1	addr_gen_p
cpr2_no	1	cons_map_p
cpr3_no	3	trap_gen_p
cpr4_no	0	rule_sel_p
cpr5_no	2	minmax_p
cpr6_no	1	mult
cpr7_no	0	int_uns
cpr8_no	0	int_sig
cpr9_no	2	div_array

TABLE II (B)
DFLP CHARACTERISTICS

Fuzzy Inference System (FIS) type	Takagi-Sugeno zero-order type
Inputs	**2**
Input resolution	**12 bit**
Outputs	**1**
Output resolution	**12 bit**
Antecedent Membership Functions (MF's)	**9 Triangular or Trapezoidal shaped per fuzzy set**
Antecedent MF Degree of Truth (α value) resolution width	**8 bit**
Consequent MF's	**81 Singleton type**
Consequent MF resolution	**8 bit**
Max. no. of fuzzy inference rules	**81 (no. of fuzzy sets $^{\text{no. of inputs}}$)**
AND method	**MIN (T-norm operator implemented by minimum)**
Implication method	**PROD (product operator)**
MF overlapping degree	**2**
Defuzzification method	**Weighted average**

TABLE III
FPGA DEVICE UTILIZATION SUMMARY

Resource	Used	Available	Utilization
BSCANs	1	1	100%
BUFGMUXs	6	8	75%
DCMs	2	4	50%
External IOBs	121	487	24%
LOCed IOBs	120	121	99%
MULT18X18s	11	32	34%
RAMB16s	16	32	50%
Slices	4021	13312	30%
SLICEMs	668	6656	10%
Total LUTs:	5,956	26,624	22%

VI. THE MATLAB INTERFACE

A Matlab program was developed for monitoring and initialization purposes. Matlab is connected to the FPGA through a bridged USB connection. It receives and analyzes data relayed by the SoC, mainly the SIP packets that the robot sends. The program decodes the SIP packets and extracts odometry information. It also incorporates the same routine used in the SoC for catching and fixing encoder overflows.

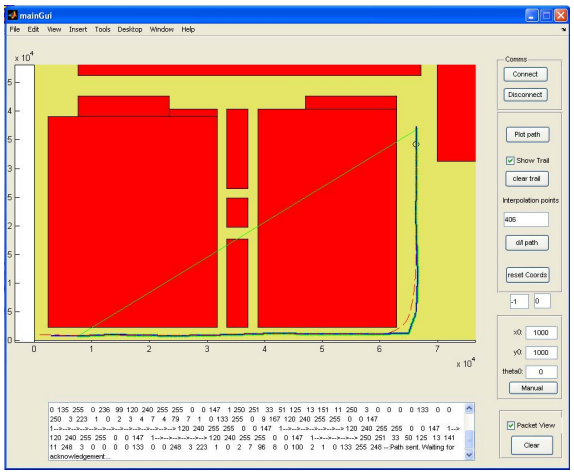

Fig. 8. Snapshot of the GUI after an experiment. The solid line represents the desired path while the dashed line the actual path. The map illustrates part of the 2nd floor of the Electrical & Computer Engineering faculty of NTUA. All units are in millimeters.

The main world frame is also incorporated in this program. Transformations from local to global coordinates are being carried out in Matlab. The GUI depicts the world map in global coordinates, as illustrated in Fig. 8. Since there is no path planning routine implemented in this work, the path is drawn in the GUI by hand as a sequence of points. Consequently the program uses a linear interpolation scheme to produce all the data samples of the path under a fixed sampling spacing, i.e., the distance between two sample points on the path is constant. The user can define the number of interpolation points. This interpolation routine was chosen after field observations on different interpolation schemes such as polynomial, cubic and linear. The interpolation chosen produced the best results. The Matlab GUI depicts the pose of the robot in real time along with other information sent by the FPGA. In particular, when the spatial window is of order one, i.e., when only the closest point is considered, the SoC sends the two calculated controller inputs.

VII. EXPERIMENTS

In this section the results of two experiments of the system are presented. The experiments took place inside the NTUA campus. The goal was to assess the overall efficiency of the system and particularly the fuzzy tracker. The experiments consist of tracking two prescribed paths. The first is a straight line path and the second is an S-shaped path. In order to log the actual position of the robot during the runs, a DGPS antenna and receiver was mounted onto it. The DGPS system used is the Trimble 4700 GPS receiver. The GPS was set to Kinematic Survey mode where the path is solved in post-processing. In this mode the horizontal precision is ±1cm+1ppm for a baseline under 10Km. The occupation is 1 second i.e. a positional sample is calculated at each second. An actual picture of the system can be seen in Fig. 11. The results of the two experiments can be seen in Fig. 9(a, b).

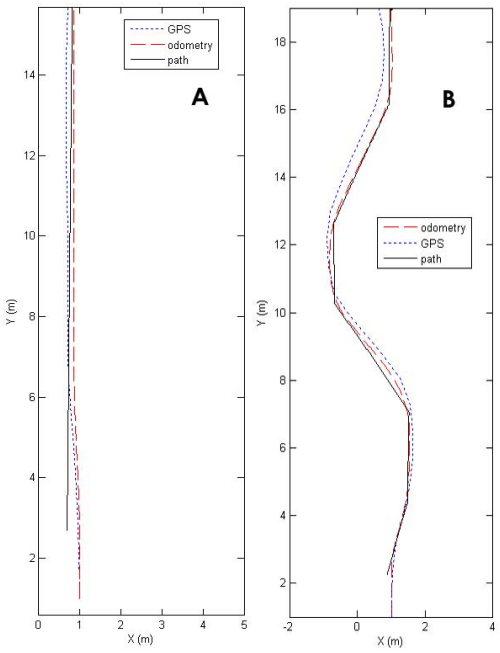

Fig. 9. The straight run (a) and the S-shaped (b) experiments with the reference path (solid), the odometry position estimation (dashed) and the DGPS estimation (dotted).

In the straight run experiment the robot was set to follow a 25m straight path. The robot's initial position was not on the path. The depiction in Fig. 9 does not present the entire run, but rather the segment where the GPS solution is of the highest quality (quality factor Q=1) since in order to assess the path tracker's performance we need a high precision position estimation. This must not be confused with the position estimation module that the tracker uses, which in this case is derived from odometry data. The GPS is used in order to see the actual position of the robot. Thus a degraded GPS solution is useless and positional data of a Q factor greater than 1 (with 1 being the best and 6 the worst) have been discarded.

The second experiment presents the tracking of an S-shaped path. The same conditions regarding the GPS data also apply to this run. The S-shaped path has a length of approximately 25m. All GPS data with Q>1 have been discarded. It is evident from both experiments that the path tracker performs well. It should also be noted that the S-shaped path is not actually a feasible reference path since the curvature derivative is discontinuous at the polygon vertices. However if the discontinuity is small, the robot is expected to provide an accurate tracking. Furthermore the odometry position estimation is very close to the path. This means that if a higher precision position estimation is used with the path tracker, such as a Real-Time Kinematic DGPS data feed that provides positional data to the path tracker in real-time, the tracker will perform even better. This is part of the future work of the authors. Moreover, the FPGA can easily incorporate data from other sensors and provide additional output. The inherent design of FPGAs allows for great scalability. By writing codecs similar to that of the SIP deframer in the Microblaze environment, data from more external sources can be easily manipulated.

VIII. Conclusion

This paper presents a novel SoC for the path following task of autonomous non-holonomic mobile robots. The latency of the control is very small although it is bounded by the response of the controlled system i.e. the robot. Simulations and field experiments showed that the fuzzy tracking algorithm, introduced by the authors, and the overall system performance is satisfactory even under the limitations presented by the real system, namely the quantization of available steering commands and the existence of a dead zone. This is due to the high robustness exhibited by the fuzzy tracking algorithm along with the "smoothing" behavior of the spatial window technique inserted in the control loop.

IX. Appendix

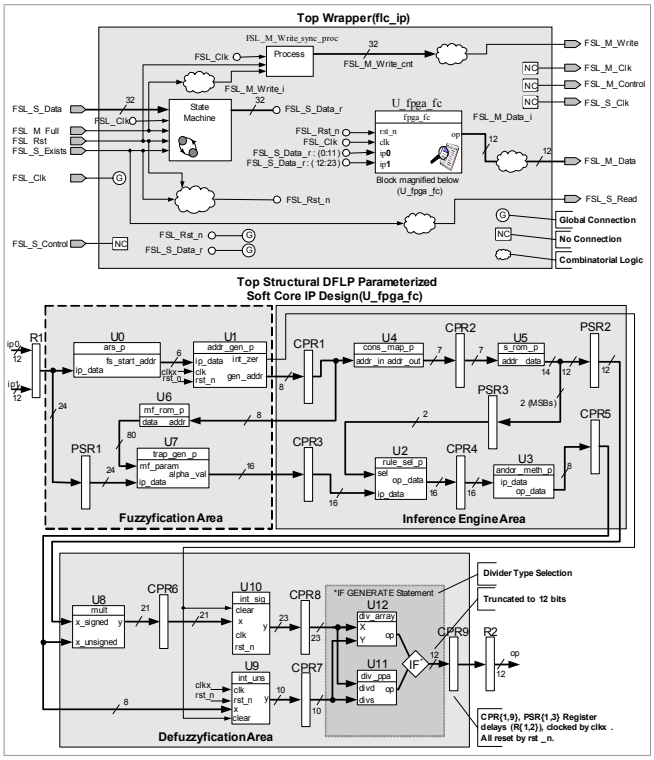

Fig. 10. DFLP Soft Core IP Architecture.

Fig. 11. Actual depiction of the system during an experiment. The FPGA, laptop and GPS antenna are clearly visible while the GPS receiver is under the laptop.

Acknowledgements

The authors would gratefully like to thank Prof. Demetris Paradissis, Mr. Athanassios Zissopoulos and Mr. Vangelis Zacharis from the Dionyssos Satellite Observatory at NTUA,

for their invaluable help in providing the GPS equipment and their expertise. Their expert contribution in performing the field experiments is what made them possible.

REFERENCES

[1] K. M. Deliparaschos, S.G. Tzafestas, "A Parameterized T-S Digital Fuzzy Logic Processor: Soft Core VLSI Design and FPGA Implementation" in International Journal of Factory Automation, Robotics and Soft Computing, Vol. 3, July 2006, pp. 7–15.

[2] G. P. Moustris, S. G. Tzafestas, "A Robust Fuzzy Logic Path Tracker for non Holonomic Mobile Robots" in International Journal of Artificial Intelligence Tools, World Scientific, Vol. 14, No. 6, Dec. 2005, pp. 935–965.

[3] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents" in American Journal of Mathematics, Vol. 79, 1957, pp. 497–517.

[4] ActivMedia Robotics, "P3-DX, World's most popular intelligent wheeled robot", Available from:
http://www.activrobots.com/ROBOTS/p2dx.html

[5] ActivMedia Robotics, "Pioneer 3TM & Pioneer 2TM H8-Series Operations Manual", ActivMedia Robotics, version 3, August 2003, pp. 33–36.

[6] Xilinx, Inc., Microblaze Processor Reference Guide, UG081 (v6.0) June 1, 2006, Available:
http://www.xilinx.com/ise/embedded/mb_ref_guide.pdf

[7] Xilinx, Inc., Embedded System Tools Reference Manual, UG111 (v5.0) October 24, 2005, Available:
http://www.xilinx.com/ise/embedded/edk82i_docs/est_rm.pdf

[8] John L. Hennessy and David A. Patterson, "Computer Architecture: A Quantitative Approach 2nd Ed.", Morgan Kaufmann Publishers, San Mateo, California (1996).

[9] Xilinx, Inc., On-Chip Peripheral Bus V2.0 with OPB Arbiter (v1.10c), DS401 December 2, 2005, Available:
http://www.xilinx.com/bvdocs/ipcenter/data_sheet/opb_v20.pdf

[10] Xilinx, Inc., Fast Simplex Link (FSL) Bus (v2.00a), DS449 December 1, 2005, Available:
http://www.xilinx.com/bvdocs/ipcenter/data_sheet/FSL_V20.pdf

[11] Xilinx, Inc., Local Memory Bus (LMB) v1.0 (v1.00a), DS445 April 4, 2005, Available:
http://www.xilinx.com/bvdocs/ipcenter/data_sheet/lmb.pdf

[12] Xilinx, Inc., Processor Local Bus (PLB) v3.4, DS400 (v1.6) July 7, 2003, Available:
http://www.xilinx.com/ipcenter/catalog/logicore/docs/plb_v34.pdf

[13] Xilinx, Inc., Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel, XAPP529 (v1.3) May 12, 2004, Available:
http://www.xilinx.com/bvdocs/appnotes/xapp529.pdf

[14] Memec Design Spartan-3™ MB Development Kit, Product Brief, Available:
http://avnet.co.jp/products/kits/docs/spartan3mb-1.pdf

[15] Synplicity Synplify Pro, Available:
http://www.synplicity.com/products/synplifypro/index.html

[16] P.H.W. Leong and K.H. Tsoi, "Field Programmable Array Technology for Robotics Applications," IEEE International Conference on Robotics and Bioimetics (ROBIO), Hong Kong 2005.

[17] A. Kongmunvattana and P. Chongstitvatana, "A FPGA-based Behavioral Control System for a Mobile Robot", IEEE Asia-Pacific Conference on Circuits and Systems (IEEE APCCAS 98), Chiangmai, Thailand, 1998.

[18] T.-H. S. Li, S.-J. Chang, and Y.-X. Chen, "Implementation of humanlike driving skills by autonomous fuzzy behavior control on an FPGA based car-like mobile robot," IEEE Transactions on Industrial Electronics, Vol. 50, no. 5, pp. 869–880, Oct 2003.

[19] R. Reynolds, P. Smith, L. Bell, and H. Keller, "The design of mars lander cameras for mars pathfinder, mars surveyor '98 and mars surveyor '01," IEEE Transactions on Instrumentation and Measurement, vol. 50, no. 1 Feb 2001, pp. 63–71.

Kyriakos M. Deliparaschos received the B.Eng. Hons degree in Electronics Engineering from De Montfort University, Leicester, U.K., and the M.Sc. degree in Mechatronics from De Montfort University with the collaboration of National Technical University of Athens (NTUA), Athens, Greece. He is currently completing the Ph.D. degree at the NTUA. His research focuses on digital system design for high-performance FPGA architectures, VLSI systems, System-on-a-Chip (SoC), hardware/software co-designs, fuzzy systems, genetic algorithms and mobile robotics.

George P. Moustris received his M.Eng. in Electrical & Computer Engineering from the Aristotle University, Thessaloniki, Greece and is currently pursuing his Ph.D. in computational intelligence and robotics at the National Technical University of Athens (NTUA), Athens, Greece. His fields of interest include mobile robotics, computational and artificial intelligence, non-holonomic systems, non-linear control theory and the philosophy of information.

Spyros G. Tzafestas professor emeritus, Director of the Institute of Communication and Computer Systems (ICCS), Senior Research Associate of the Signals, Control and Robotics Division and the Intelligent Robotics and Automation Laboratory (IRAL) of the National Technical University of Athens (NTUA). Holder of Ph.D. and D.Sc. in Control and Automation. Recipient of Honorary Doctorates of the International University (D.Sc. (Hon.)), the Technical University of Munich (Dr.-Ing. E.h.) and the Ecole Centrale de Lille (Docteur Honoris Causa). Fellow of IEEE (N.Y.) and IEE (London); Member of ASME (N.Y.) , New York Academy of Sciences , IMACS (Rutgers, N.J.) and SIRES (Brussels). Member of IFAC SECOM and MIM TCs. Project evaluator of national european and international projects (USA, Canada, Italy, Hong Kong, Japan). Project coordinator of national and EU projects in the fields of robotics, CIM and IT (ESPRIT, BRITE-EURAM , TIDE, INTAS , SOCRATES, EUREKA, GROWTH etc.). Publications: 30 research books, 60 book chapters, over 700 journal and conference technical papers. Editor-in-Chief of the Journal of Intelligent and Robotic Systems (1988-2005) and the book series "Microprocessor-Based and Intelligent Systems Engineering" (K1uwer). Organizer of several international conferences (IEEE, IFAC, IMACS, IASTED, SIRES etc.). Listed in several international biographical volumes. Current interests include: control, robotics and CIM