Robotics and Autonomous Systems 58 (2010) 1017–1027

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of System on a Chip

S.G. Tzafestas, K.M. Deliparaschos, G.P. Moustris *

Intelligent Automation Systems Research Group, School of Electrical and Computer Engineering, National Technical University of Athens, Zographou Campus, Athens, GR 157 73, Greece

ARTICLE INFO

Article history:
Received 29 March 2009
Received in revised form
26 March 2010
Accepted 31 March 2010
Available online 22 April 2010

Keywords:
Path tracking
Fuzzy logic
Mobile robots
Digital Fuzzy Logic Controller (DFLC)
System-on-a-Chip (SoC)

ABSTRACT

This paper presents a System on Chip (SoC) for the path following task of autonomous non-holonomic mobile robots. The SoC consists of a parameterized Digital Fuzzy Logic Controller (DFLC) core and a flow control algorithm that runs under the Xilinx Microblaze soft processor core. The fuzzy controller supports a fuzzy path tracking algorithm introduced by the authors. The FPGA board hosting the SoC was attached to an actual differential-drive Pioneer 3-DX8 robot, which was used in field experiments in order to assess the overall performance of the tracking scheme. Moreover, quantization problems and limitations imposed by the system configuration are also discussed.

1. Introduction

This work presents a System on Chip (SoC) implementation for the robot path tracking task using fuzzy logic. The major components the SoC is composed of, are a parameterized Digital Fuzzy Logic Controller (DFLC) soft IP core [1,2], implementing the fuzzy tracking algorithm, and a Xilinx Microblaze soft processor core as the top level flow controller. The FPGA board hosting the SoC was mounted to an actual differential-drive Pioneer 3-DX8 robot, which was used in experiments of the tracking scheme.

FPGAs provide several advantages over single processor hardware, on the one hand, and Application Specific Integrated Circuits (ASIC) on the other. FPGA chips are field-upgradable and do not require the time and expense involved with ASIC redesign. Being reconfigurable, FPGA chips are able to keep up with future modifications that might be necessary. They offer a simpler design cycle, reprogrammability, and have a faster time-to-market, since no fabrication (layout, masks, or other manufacturing steps) time is required, when compared to ASICs.

A design on an FPGA could be thought as a "hard" implementation of program execution. The Processor based systems often involve several layers of abstraction to help schedule tasks and share resources among multiple processes. The driver layer controls hardware resources and the operating system manages memory and processor bandwidth. Any given processor core can execute only one instruction at a time, and processor based systems are continually at risk of time critical tasks preempting one another. FPGAs on the other hand do not use operating systems, and minimize reliability concerns with true parallel execution and deterministic hardware dedicated to every task. Today's FPGAs contain hundreds of powerful DSP slices reaching frequencies up to 500 MHz outperforming DSP and RISC processors by a factor of 100 to 1000. Taking advantage of hardware parallelism, FPGAs exceed the computing power of digital signal processors (DSPs) by breaking the paradigm of sequential execution and accomplishing more per clock cycle.

The use of field programmable gate arrays in robotic applications is noted in [3–6] by several other researchers. A review of the application of FPGAs in robotic systems is provided be Leong and Tsoi in [3]. A notable case study is the use of FPGAs in the Mars Pathfinder, Mars Surveyor '98, and Mars Surveyor '01 Lander crafts, analyzed in [6].

Due to an increased number of calculations necessary for the path tracking control, a high performance processing system to efficiently handle the task is required. By using a SoC realized on an FPGA device, we utilize the hardware/software re-configurability of the FPGA to satisfy the needs of fuzzy logic path tracking for autonomous robots for high-performance onboard processing and flexible hardware for different tasks. Software Fuzzy Logic Controller (FLC) implementations suffer from speed limitations

* Corresponding author.
E-mail addresses: tzafesta@softlab.ntua.gr (S.G. Tzafestas), kdelip@mail.ntua.gr (K.M. Deliparaschos), gmoustri@central.ntua.gr (G.P. Moustris).
URL: http://www.ece.ntua.gr/images/pages/ias (S.G. Tzafestas, K.M. Deliparaschos, G.P. Moustris).

0921-8890/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2010.03.014

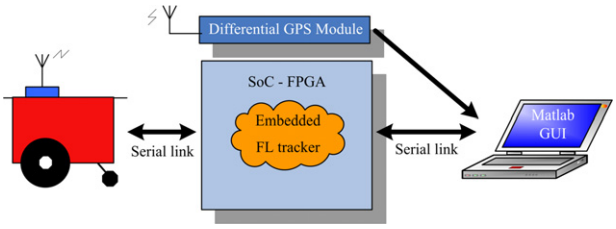

Fig. 1. Overview of the System.

due to the sequential program execution and the fact that standard processors do not directly support many fuzzy operations (i.e., minimum or maximum). In an effort to reduce the lack of fuzzy operations, several modified architectures of standard processors to support fuzzy computation exist [7–9]. Software solutions running on these devices speeds up fuzzy computation by at least one order of magnitude over standard processors, but are still not fast enough for some real-time applications; a dedicated hardware implementation [10] must be used.

In our application the DFLC facilitates scaling and can be configured for different numbers of inputs and outputs, numbers of triangular or trapezoidal fuzzy sets per input, numbers of singletons per output, antecedent method (t-norm, s-norm), divider type, and numbers of pipeline registers for the various components in the model. This parameterization enabled the creation of a generic DFLC soft IP core that was used to produce a fuzzy controller of different specifications without the need of redesigning the IP from the beginning. The fuzzy logic controller architecture assumes overlap of two fuzzy sets among adjoining fuzzy sets, and requires 2^n (n is the number of inputs) clock cycles at the core frequency speed in order to sample the input data (input sample rate of 56.34 ns), since it processes one active rule per clock cycle. In its present form the SoC design achieves a core frequency speed of 71 MHz. To achieve this timing result, the latency of the chip architecture involves 9 pipeline stages each one requiring 14.085 ns. The featured DFLC IP is based on a simple algorithm similar to the zero-order Takagi–Sugeno inference scheme and the weighted average defuzzification method. By using the chosen parameters of Table 2, it employs two 12-bit inputs and one 12-bit output, 9 triangular membership functions (MFs) per input and 5 singleton MFs at the output with 8-bit and 12-bit degree of truth resolution respectively. The rule base consists of 81 rules.

The fuzzy tracking algorithm used, is based on a previous fuzzy path tracker developed by the authors [11]. The fuzzy logic (FL) tracker has undergone some alterations due to the hardware restrictions posed by the DFLC soft IP core. While the original fuzzy logic controller (FLC) was of the Mamdani-type with Gaussian membership functions, the one deployed here is of the 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 used in the previous paper [11] has also been incorporated in the tracking scheme.

2. Overview of the system

Four cooperative functional units properly tied together integrate the system. An overview can be seen in Fig. 1, and an actual picture of the system in Fig. 13.

The 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; namely it decodes the information packets sent by 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 application 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 also treats synchronization and timing requirements.

A MATLAB application was developed for monitoring and initialization purposes. It communicates with the FPGA board through a bridged USB connection and 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 application is to provide a path for the robot to track. Given that 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 application allows choosing the number of interpolation points. The aforementioned interpolation routine was chosen after field observations on different interpolation schemes such as polynomial, cubic and linear and produced the best results.

The test platform on which the SoC was tested is the ActivMedia P3-DX8 robot [12]. The robot uses 1 mm resolution for the position estimation and 1° angle resolution for the heading. The kinematics of the robot are emulated to a bounded curvature steering vehicle and not that 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 was intended for the Dubin's Car model [13] where there is a minimum turning radius constraint on the robot and only for forward motion. As will be explained in a later section, the curvature constraint along with the one degree resolution of the P3 robot presents a quantization problem to the curvature. The robot connects to the FPGA board through a serial cable to send and receive framed data. ActivMedia uses its own data framing protocol handled by the robot's microcontroller. The data sent from the robot are called Server Information Packets (SIP packets) while the received data are called Command Packets. More information on the data framing protocol can be found in the robot's manual [12, pp.33-36]. The experiments showed clearly that even though the FL tracker performs well, its actual performance is severely degraded by the accumulation of odometry errors over time. Several calibration tests have been carried out in order to improve odometry localization but, as it was expected, position estimation through odometry proved inefficient.

3. Fuzzy logic path tracking algorithm

The tracking problem can be formulated in the following way; let Σ be a system described by Eq. (1),

$$\begin{aligned} \dot{x} &= f(t, x, u) \\ y &= h(t, x, u) \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the input vector and $y \in \mathbb{R}^k$ is the output vector. Let $x_r(t)$ be a feasible reference trajectory in the state space that satisfies Eq(1). This solution corresponds to a reference input u_r i.e., $\dot{x}_r = f(t, x_r, u_r)$. Find a feedback law $u = u(t, x, x_r, u_r)$ such that $\lim_{t\to\infty}(x(t) - x_r(t)) = 0$. The path tracking problem can be formulated in the same manner except that the goal is to track the *image* of the reference trajectory $x_r(t)$. In such a setting, the image admits a reparametrization according to some