Autonomous SoC for Fuzzy Robot Path Tracking

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

School of Electrical and Computer Engineering
National Technical University of Athens
Intelligent Robotics and Automation Laboratory

e-mails: kdelip@mail.ntua.gr
        gmoustri@central.ntua.gr
tzafesta@softlab.ntua.gr
Overview of the System

- **Robot platform:** ActivMedia P3-DX8
  - 1 mm resolution for the position estimation and 1° angle resolution for the heading
  - Kinematics emulated to a bounded curvature vehicle

- **SoC FPGA:**
  - Xilinx XC3S1500-4FG676C Spartan-3 FPGA
    - Parameterized Digital Fuzzy Logic processor (DFLP) Intellectual Property (IP) core implementing the Fuzzy Tracking algorithm
    - Xilinx Microblaze™ soft processor core as the top-level flow controller

- **Matlab GUI:**
  - Visualization/Monitoring running on Laptop
Let $F$ be a nonlinear system of the form

$$
\dot{p} = f(t, p, u)
$$

where $p$ is the state vector and $u$ the input. If $p_{\text{ref}}$ is a feasible reference path in the state space which corresponds to a feasible reference input $u_{\text{ref}}$, then find an appropriate state feedback law $u(p, p_{\text{ref}}, u_{\text{ref}}, t)$ such that

$$
\lim_{t \to \infty} (p - p_{\text{ref}}) = 0
$$
Path Tracking

- Former problem known as trajectory tracking because reference trajectory is parameterized by time
- If $p_{ref}$ is a geometric reference path (no temporal parameterization) then we get the “path tracking problem” {e.g. $p_{ref}=(x,y)$ where $(x,y)$ are the Cartesian coordinates of the robot}
- For the Dubin’s Car model:

$$
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix}
= \begin{bmatrix}
\cos \theta \\
\sin \theta \\
\kappa
\end{bmatrix} v
$$

$p_{ref}=(x_{ref}, y_{ref})$, where $(x,y)$ is the middle point of the robot axis.

- Speed is constant
- Control input is the curvature $\kappa$
Constraints

- Non-holonomic: *rolling without slipping*
- Rigidity: *robot dimensions remain fixed*
- Input bounds: *robot cannot turn while stopped (bounded $\kappa$)*
- Quantization: *robot quantizes states and inputs*
  - No explicit command for curvature. Curvature command is issued through the control of the linear and angular velocities
  - Angular velocity quantized to 1 deg/sec/bit resolution
  - Linear Velocity quantized to 1 mm/sec/bit resolution
  - **RESULT**  → Turning radius: $\kappa^{-1} = R = \frac{c v}{\omega}$ is quantized
Quantization presents a problem for the control input. If the curvature is bounded to $|\kappa| \leq 1 \text{ m}^{-1}$, a velocity of $v=100 \text{ mm/sec}$ results to 6 available input commands. Bounding constraint reduces further the available levels. Performance severely degraded. Results to oscillations. Putting $R=1$ and solving w.r.t $\omega$ we get the available quantization levels:

$$L_{num} = \left\lfloor v \cdot \frac{180}{1000 \cdot \pi} \right\rfloor$$

Linear dependence on $v$. Obvious solution is to increase velocity but if $v$ is large, dynamics might come into play.
Setting the Speed

To estimate an acceptable error level between the curvature computed by the DFLP and the actual curvature the robot follows, we draw the maximum relative error versus all available speeds over all available inputs i.e.,

\[ 100 \cdot \max(\kappa_{FLC} - \kappa_{ACTUAL}) / \kappa_{ACTUAL}, \forall \phi_1, \phi_2 \]

In this way one can see the maximum possible relative error for each speed between the actual and the desired curvature.

An acceptable trade-off speed seems to be 1000 mm/sec (1 m/sec) where the error drops below 1.745%.

For this speed, the available quantization levels are $Lnum=57$. As a result, the robot’s speed was set to 1000 mm/sec in all field experiments.
Control Strategy

- Path sampled under fixed sampling spacing $\Delta s$
- Inputs: Angle error $\phi_1$
  - Heading error $\phi_2$
- Output: Curvature $\kappa$

- Spatial Window
  - Offset (Start at "offset" points from closest)
  - Order (iterate over "order" path points)
  - Step (skip "step" points at each iteration)
- Output: Mean of curvatures

$$\kappa = \frac{\kappa_1 + \kappa_2 + \ldots + \kappa_n}{n}$$
FPGA Development Platform Specifications

- The SoC Application was implemented on a Spartan-3 Development Platform

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

Kyriakos M. Deliparaschos, George P. Moustris, Spyros G. Tzafestas
Fuzzy Path Tracker (DFLP IP Core)

The DFLP IP Core is fully parameterized. The selected 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.

Characteristics

- **TS Zero-Order Fuzzy Logic Controller**
- 2 Inputs, 1 Output
- 9 Triangular MFs per Input
- 5 Singleton Output MFs
- 81 Fuzzy Rules
- Implication method: Product
- Defuzzification method: Weighted average

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy Inference System (FIS) type</td>
<td>Takagi-Sugeno zero-order type</td>
</tr>
<tr>
<td>Inputs</td>
<td>2</td>
</tr>
<tr>
<td>Input resolution</td>
<td>12 bit</td>
</tr>
<tr>
<td>Outputs</td>
<td>1</td>
</tr>
<tr>
<td>Output resolution</td>
<td>12 bit</td>
</tr>
<tr>
<td>Antecedent Membership Functions (MF’s)</td>
<td>9 Triangular or Trapezoidal shaped per fuzzy set</td>
</tr>
<tr>
<td>Antecedent MF Degree of Truth (α value) resolution width</td>
<td>8 bit</td>
</tr>
<tr>
<td>Consequent MF’s</td>
<td>81 Singleton type</td>
</tr>
<tr>
<td>Consequent MF resolution</td>
<td>8 bit</td>
</tr>
<tr>
<td>Max. no. of fuzzy inference rules</td>
<td>81 (no. of fuzzy sets no. of inputs)</td>
</tr>
<tr>
<td>AND method</td>
<td>MIN (T-norm operator implemented by minimum)</td>
</tr>
<tr>
<td>Implication method</td>
<td>PROD (product operator)</td>
</tr>
<tr>
<td>MF overlapping degree</td>
<td>2</td>
</tr>
<tr>
<td>Defuzzification method</td>
<td>Weighted average</td>
</tr>
</tbody>
</table>
SoC Architecture - High Level Diagram

- Microblaze™ is licensed as part of the Xilinx Embedded Development Kit (EDK) and is based on RISC architecture.
- It is a soft core, meaning that it is implemented using general logic primitives rather than a hard dedicated block on the FPGA.
- The DFLP IP core is connected to the Microblaze™ via the FSL bus.
- The processor connects to the OPB bus for access to a wide range of different modules, and communicates via the LMB bus for a fast access to BRAM inside the FPGA.
SoC Architecture – Detailed Diagram

- Component pipeline registers (CPR)
- Path synchronization registers (PSR)
- ‘U’ blocks represent the different components of the DFLP IP core
- $U_{\text{fpga\_fc}}$ component is embedded in flc_ip top structural entity wrapper
- Two Digital Clock Manager (DCM) Modules (DCM_0 and DCM_1) used for the different clocks production
- SoC achieves a system clock (DCM_0) operating frequency of 14.140ns or ~71 MHz, while DCM_1 is mainly used for clocking the external DDR RAM

Kyriakos M. Deliparaschos, George P. Moustris, Spyros G. Tzafestas
U_fpga_fc alone was synthesized using Synplify Pro synthesizer tool. The rest of the design components were synthesized by Xilinx Synthesis Tool (XST) through the EDK Platform Studio. The DFLP IP core alone for the selected parameters on this application occupies 1600 (6%) LUTs, 4 Block Multipliers (MULT18X18s), 12 64x1 ROMs (ROM64X1), and 56 256x1 ROMs (ROM256X1).

<table>
<thead>
<tr>
<th>Resource</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSCANs</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>BUFGMUXs</td>
<td>6</td>
<td>8</td>
<td>75%</td>
</tr>
<tr>
<td>DCMs</td>
<td>2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>External IOBs</td>
<td>121</td>
<td>487</td>
<td>24%</td>
</tr>
<tr>
<td>LOCed IOBs</td>
<td>120</td>
<td>121</td>
<td>99%</td>
</tr>
<tr>
<td>MULT18X18s</td>
<td>11</td>
<td>32</td>
<td>34%</td>
</tr>
<tr>
<td>RAMB16s</td>
<td>16</td>
<td>32</td>
<td>50%</td>
</tr>
<tr>
<td>Slices</td>
<td>4021</td>
<td>13312</td>
<td>30%</td>
</tr>
<tr>
<td>SLICEMs</td>
<td>668</td>
<td>6656</td>
<td>10%</td>
</tr>
<tr>
<td>Total LUTs:</td>
<td>5,956</td>
<td>26,624</td>
<td>22%</td>
</tr>
</tbody>
</table>
Top-Level Control Program

Written in C and executed in the Microblaze™ soft processor core

- Implements the autonomous control logic of the P3 robot
- Receives odometry information from the robot and issues steering commands outputted by the FL tracker
- Treats synchronization and timing requirements
- For the communication with the outside world (Robot, Laptop) uses two I/O channels, one serial and one Serial2USB bridge, both having 16-byte input and output buffers

(a) Path tracking control algorithm
(b) “fix” algorithm to avoid 16-bit integer coordinates overflow when their range is exceeded
Hardware/Software Co-design Flow

Software Development Flow

- Software Project Build
- SW Configuration
- Automatic SW Library Generation
- Software Compilation
- Executable
- Debug

Hardware Development Flow

- Specify Processor, Bus, and Peripherals
- HW Configuration
- Automatic HW Platform Generation
- Xilinx FPGA Implementation Flow
- Bitstream
- Download to FPGA

Virtual System Model

Specify Processor, Bus, and Peripherals

Automatic SW Library Generation

Automatic HW Platform Generation

Virtual System Model

Download to FPGA

Hardware/Software Co-design Flow

Kyriakos M. Deliparaschos, George P. Moustris, Spyros G. Tzafestas
DFLP IP Core design Flow

Mathworks - Matlab

Autonomous SoC for Fuzzy Robot Path Tracking

Kyriakos M. Deliparaschos, George P. Moustris, Spyros G. Tzafestas
Matlab GUI Application

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.

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.
Experiments

Experiments took place inside the NTUA campus. The goal was to assess the overall efficiency of the system and particularly the fuzzy tracker. In order to log the actual position of the robot during the runs, a DGPS antenna and receiver were 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.

Localization was done through odometry, not DGPS. The DGPS was used only to get the actual position. 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.
Showcase
Conclusions

► A novel SoC for the path following task of autonomous non-holonomic mobile robots was presented.

► 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, e.g. the quantization of available steering commands. 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.