Architecture of a tactile sensor suite for artificial hands based on FPGAs

Óscar Oballe-Peinado, José A. Hidalgo-López, José A. Sánchez-Durán, Julián Castellanos-Ramos and Fernando Vidal-Verdú

Dept. of Electronics
University of Málaga
Málaga, Spain
fvidal@uma.es

Abstract—Tactile sensing is a serious issue in assistive robotics or prosthetics. It requires the hands and body of the robot are equipped with many tactile arrays, and a large amount of data has to be acquired and transmitted to a central decision unit. Moreover, this has to be done in a very short time to improve dexterity and performance in service tasks. Therefore, the local electronics of a smart tactile patch must be powerful enough to acquire and pre-process data in real time to cope with the limited throughput of the communication buses. This paper presents an architecture based on FPGAs that implements a direct interface with the sensor without analog-to-digital converters and robust serial communications between the electronics in the finger tips, mid-digits and palm.

Keywords: Tactile sensor suite, direct connection sensor-FPGAs

I. INTRODUCTION

Tactile sensing in a humanoid robot involves a large area and many sensors are commonly distributed along its hands and body. Therefore, the communication buses must be able to convey the information provided by these and other sensors, as well as the commands for the actuators. Since a control sampled period of 2-100ms is required for a stable and smooth feedback control [1], the key issue behind this task is that a large amount of data must be processed and transmitted in a short time. FPGAs can carry out complex calculations in parallel, so they can accomplish high dynamic requirements as those demanded by this application.

Serial buses are chosen in humanoid robots to be housed in a reduced room and minimize mechanical and electrical interferences. For instance, an I2C serial bus is used in [2], although this bus does not guaranty a maximum latency, and a too long sampling time of the sensor system is mentioned as a problem for quick sensor feed backed motions. An common open strategy for humanoid robot control uses CAN bus based networks in the time-dependant sensory and device levels and LAN/WLAN Ethernet or USB based networks in the control and organization levels where strict real-time is not needed [3][4][5]. CAN bus can be shared by many devices and it guarantees a maximum latency. However, although appropriate for real time applications, this bus is not oriented to transfer large amounts of data. It takes 132us to transfer 8 bytes at 1Mbps under optimum conditions, so if the force against a tactel is encoded in 8 bits, it would take at least 135ms to read 1024 tactels. Moreover, the data cannot be transferred at the same time, so the data transfer is asynchronous [6].

Therefore, the essential information must be extracted and sent in a time as short as possible to fulfill real time requirements. A common pre-processing procedure consists in computing the ellipse that fits the tactile image the best [5]. These calculations are made locally and only the parameters related to this ellipse are transferred to the central computer. These algorithms can be executed at the same time that other tasks like data acquisition or calibration due the capability of FPGAs to work in parallel. The architecture we propose is composed of a set of FPGAs, one per tactile array, that communicate to each other through fast SPI buses. This tactile suite is able to scan all the arrays and compute the parameters of the ellipses that fit the tactile images registered by them at a rate of 200 frames per second. This rate can even allow the detection of slippage, which has the highest dynamic requirements in dexterous manipulation [7].

II. ARCHITECTURE OF THE TACTILE SENSOR SUITE

The authors have proposed two strategies for data acquisition based on direct connection tactile sensor—FPGA [8]. Both have been implemented for the work of this paper and are shown in Figure 1 and Figure 2. The proposed global architecture for the hand is shown in Figure 3. Please note that they are prototypes devoted to prove the concept and allow the development of software and evaluation of hardware. An advanced version much more compact will be implemented in the near future.

Figure 1 and Figure 2 differ basically in the way they implement data acquisition. A pin (and track) is dedicated per tactel in Figure 1 while Figure 2 addresses a whole row with the same pin. The reason behind this is crosstalk cancellation. If the number of tactels in the array is high, the alternative in Figure 2 can be better, although the external electronics is somewhat more complex, bulky, and costly. A double buffer is used in both cases to store the tactile image. The array is scanned and stored in the buffer by the acquisition module and pre-processing is carried out on these data by the pre-processing module as explained later. Then a second tactile image is acquired, pre-processed and stored in the other buffer. This way there is always a processed frame ready to be transmitted. This double buffer is implemented with a double port RAM memory, and there is a clock used by the communications and control module to read its content and another independent clock used by the sensing and pre-processing modules. Six boards of type ‘finger sensor’ in Figure 1 have been implemented for the mid-digits.
and fingertips and are able to scan 64 tactels each, while other board contains the palm sensor in Figure 2 able to scan 256 tactels plus a control module that performs the communications between the other boards and the host. Robust differential MLVDS signals [9] are used to implement the serial SPI bus. Seven tactile images corresponding to the seven tactile sensors are scanned and processed simultaneously.

III. PRE-PROCESSING ALGORITHM

Table I shows a set of parameters that are commonly calculated to provide a simplified representation of the tactile image [5][10]. The implementation of these calculations on the board of Figure 1 is illustrated in Figure 4.
Table 1. Parameters of the tactile image obtained by the preprocessing algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{00}$</td>
<td>$\sum_{x=1}^{N} \sum_{y=1}^{M} f(x, y)$</td>
</tr>
<tr>
<td>$M_{10}$</td>
<td>$\sum_{x=1}^{N} \sum_{y=1}^{M} y f(x, y)$</td>
</tr>
<tr>
<td>$M_{01}$</td>
<td>$\sum_{x=1}^{N} \sum_{y=1}^{M} x f(x, y)$</td>
</tr>
<tr>
<td>$M_{20}$</td>
<td>$\sum_{x=1}^{N} \sum_{y=1}^{M} x^2 f(x, y)$</td>
</tr>
<tr>
<td>$M_{11}$</td>
<td>$\sum_{x=1}^{N} \sum_{y=1}^{M} xy f(x, y)$</td>
</tr>
<tr>
<td>$M_{02}$</td>
<td>$\sum_{x=1}^{N} \sum_{y=1}^{M} y^2 f(x, y)$</td>
</tr>
</tbody>
</table>

- **Centroid X**
  \[
  \text{Centroid X} = \frac{M_{10}}{M_{00}}
  \]

- **Centroid Y**
  \[
  \text{Centroid Y} = \frac{M_{01}}{M_{00}}
  \]

- **CM_{20}**
  \[
  \text{CM}_{20} = \frac{M_{20} - \frac{M_{10}^2}{M_{00}}}{M_{00}}
  \]

- **CM_{11}**
  \[
  \text{CM}_{11} = \frac{M_{11} - \frac{M_{10}M_{01}}{M_{00}}}{M_{00}}
  \]

- **CM_{02}**
  \[
  \text{CM}_{02} = \frac{M_{02} - \frac{M_{01}^2}{M_{00}}}{M_{00}}
  \]

- **Square Root**
  \[
  \text{Square Root} = \sqrt{\left(\text{CM}_{20} - \text{CM}_{02}\right)^2 + 4\text{CM}_{11}^2}
  \]

- **Angle**
  \[
  \text{Angle} = \arctan\left(\frac{2\text{CM}_{11}}{\text{CM}_{20} - \text{CM}_{02}}\right)
  \]

Simple moments are calculated with the procedure in Figure 4(a). The values of $x^2$, $y^2$ and $xy$ are known beforehand because they do not depend on the tactile image. This is exploited to store them in the ROM memory of the FPGA together with $x$ and $y$. Then the simple moments are computed with a multiplier plus an adder and the results are stored in a RAM as indicated in Figure 4(a). Central moments and centroids are computed as depicted in Figure 4(b). Another multiplier, one adder plus a few multiplexers are used for this purpose. Finally, a CORDIC provides the two last parameters in Table 1 as Figure 4(c) shows. All building blocks in Figure 4 are implemented once and they are shared to obtain all the parameters in Table 1. The data acquisition (scanning of the array and analog to digital conversion), communication with the central processing unit (a PC in the tests that have been carried out) and the calculations in Table 1 are made in parallel. However, the latter are carried out sequentially because they take only 5.76us, i.e. only 0.11% of the time spent in scanning the array (5ms). Moreover, there are not enough resources in this specific FPGA to make them in parallel. It is a Xilinx Spartan-3C50AN-4TQG144 and the frequency has been set at 50MHz. An amount to around 60% of the flip-flops, 60% of the Look up Tables and up to 99% of the slices (including routed-through ones) is used. This means that other FPGA has to be chosen to implement more complex calculations. This is obviously possible, although the encapsulation would probably be BGA. Note also that these other calculations can be made in parallel with data acquisition as it has been done in the work of this paper.

Figure 4. Implementation of circuits on the FPGA to perform the calculations in the Table 1.
IV. RESULTS

Figure 5 shows the schedule that corresponds to the data acquisition plus the computation of the parameters in Table I. It is obtained from simulations since it shows internal signals. Figure 5(a) depicts a general view where the rows are scanned sequentially. Figure 5(b) shows a zoom of the last part of the schedule where most computations in Table I are performed (please note that the row labels are the same that those for the parameters in Table I). Note that partial results are obtained at the same time the rows are scanned, and final results are obtained once the last row is scanned. Then, all the parameters in Table I are computed in parallel with the scanning of the first row of the next frame (this takes 5.76 us). The parameters in Table I as well as the raw tactile data are provided at a rate of 200Hz to the central processing unit, i.e., they are ready to be transmitted at this rate to the central unit. It is done through USB to a Personal Computer in the version of this paper. The whole operation is synchronized by the control module at the Hand Controller so the result is deterministic. Table 2 shows a comparison of the proposed sensor suite with others that are reported or commercially available. Finally, Figure 6 shows the raw and preprocessed data (ellipses) obtained from these parameters for the whole sensor suite.

It is worth mentioning that if other algorithms must be implemented on this FPGA, an alternative consists in sending only the simple moments to the central processing unit. This fulfills the requirement of sending the essential information in a small package of a few bytes. This approach releases resources to implement other low level algorithms on chip, for instance some primitives to detect slippage or features of the tactile image [7][15] or algorithms for error correction [16].
Table 2. Comparison with other sensor suites

<table>
<thead>
<tr>
<th>Dextrous Hand</th>
<th>Number of tactels</th>
<th>Sample rate</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPS- Barret Hand</td>
<td>162 (24 Palm + 3 x 24 Mid-Digit + 3 x 22 Fingerprint)</td>
<td>30Hz</td>
<td>[11]</td>
</tr>
<tr>
<td>Shunk</td>
<td>504 (6 arrays of 14 x 6 tactels)</td>
<td>40Hz</td>
<td>[12]</td>
</tr>
<tr>
<td>Shadow (data for a fingertip)</td>
<td>34</td>
<td>20Hz</td>
<td>[13]</td>
</tr>
<tr>
<td>Gifu Hand III</td>
<td>859 (313 palm + 3 x 126 fingers + 105 thumb)</td>
<td>100Hz</td>
<td>[14]</td>
</tr>
<tr>
<td>Proposed sensor suite</td>
<td>640 (256 palm + 3 x 64 Mid-Digit + 3 x 64 Fingerprint)</td>
<td>200Hz</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6 Raw and pre-processed (ellipses) data obtained from the tactile sensor suite.

V. DISCUSSION

The results of this paper show the feasibility of an FPGA based architecture in scanning and pre-processing tactile data from a sensor suite mounted on an artificial hand at a high rate. The preprocessing is required if the communications are made through CAN bus because only a few bytes can be sent in a message. Other buses like EtherCAT begin to be applied to get high throughput at real time, so the whole raw data could be sent to the central unit. This way, this central unit has to cope with the processing of all tactile data besides of data from other sensors and commands to the actuators. The issue of how much processing has to be done locally and how much should be done by the central unit is not trivial. Nevertheless, smart sensors perform usually some processing, at least calibration and compensation. Since tactile sensors are very error prone, these correction procedures are mandatory and complex. This paper is a first step in this way and shows the realization of a common tactile preprocessing algorithm on FPGAs. There is obviously a compromise that involves the size of the sensor suite, the power consumption and the sample rate. The best solution is also closely related to the specific application. As stated in [17], a grid with 15 x 10 elements is ideally required for the fingertips, and it must respond as fast as 1ms. Generally speaking, the higher the number of tactels the longer it takes to scan and process the array. Both the scan and processing times can be reduced if we use a parallel approach, i.e. more pins of the FPGA are used for direct A/D conversion and more powerful devices are chosen to allow parallel pre-processing. As a result the size of the sensors increases, since larger devices and more complex wiring are required. This is achieved with BGA encapsulations and Printed Circuit Boards with more layers, although the cost increases. The implementation of an Application Specific Integration Circuit (ASIC) could reduce the price for high volume fabrication and also the power consumption, although its programmability is quite limited. This is the reason for us to work with FPGAs, because they achieve a performance that is closer to the ideal than other alternatives and are programmable devices.

ACKNOWLEDGMENT

This work has been partially funded by the Spanish government under contract TEC2009-14446-C02-01.

REFERENCES


