Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

Development Framework For Hardware Implementation Of Digital Neural Network Used In Smart Sensors

Attila Buchman[1], Ştefan Oniga[1]

Abstract: This paper presents a development platform for the implementation of digital neural network used in Sensor System for an Artificial Hand. The use of neural networks to add learning and adaptive behavior to smart sensors is essential and the FPGA implementation is an easy an attractive way for hardware implementation. This platform was developed in order to provide a fast prototyping environment using reconfigurable devices (FPGA) and a microcontroller. The microcontroller is used to implement the Data Acquisition System and to adapt signal sensors to neural network input requirements. The reconfigurable device (XC2S50 Xilinx) is used to implement the neural networks and other logic blocks of the same application. The System Generator tool for Simulink allow the easy generation of hardware Description Language (HDL) code that can be synthesized for implementation in the Xilinx family of FPGA devices. The developed framework allows device communication with a PC in order, to perform the offline learning task or, to transfer data for analysis. Software is designed to manage the communication protocol with Matlab via parallel port.

Key words: Data Acquisition System, Field Programmable Gate Arrays; neural network; smart; adaptive; learning.

I. INTRODUCTION

The goal of our work was to provide a hardware support for the FPGA implementation of neural networks. A reconfigurable FPGA
device (XC2S50) was used to implement the neural networks while the System Generator Tool for Simulink was used for generation of Hardware Description Language code from a system representation in Simulink. In our specific application we used two data sources:

1. A data glove (DTG) providing information about fingers and hand position
2. Force sensing resistors (FSR) to detect contact with an object.

These data sources are of different nature: while the DTG is a digital data source, the FSRs are analog data sources.

According to these considerations we realized a data acquisition board (DAQB) capable of handling these two data types and of ensuring a proper interface both with a personal computer (PC) and the XC2S50. The framework has he block diagram shown in fig.1.

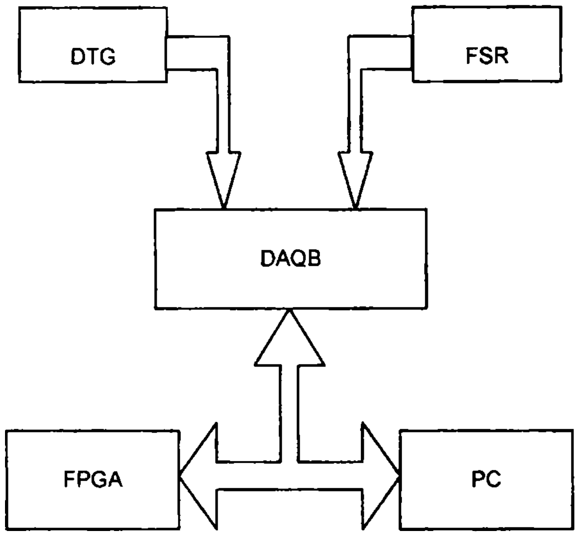


Fig. 1. Framework Block Diagram

II. THE DATA ACQUISITION BOARD

We decided to make our own DAQB in order to perfectly fit the application requirements. The main component of our board is the AduC 812 microconverter (uC). This chip is in fact an embedded system built around an 8051-compatible microcontroler core supported by 8Kb Flash/EE program memory, 640 bytes Flash/EE data memory and 256 bytes data SRAM on-chip. The digital data can be acquired via the uC standard UART Serial Port. The analog signal sources can be connected to the on-chip 8-channel 12-bit successive approximation analog to digital converter (ADC). Fig.2 shows a block diagram of the DAQB.

[1] Universitatea de Nord Baia Mare, str. Victor Babeş nr. 62A
e-mail abuchman@ubm..ro

BUPT

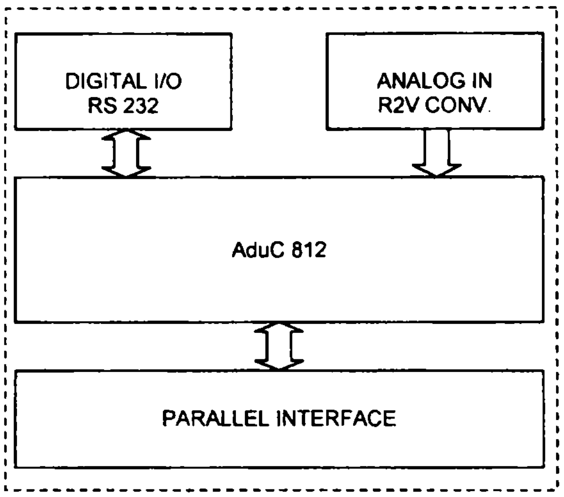


Fig. 2. DAQB Block Diagram

1. The RS 232 compatible digital I/O uses the on-chip UART of the microconver. A MAX 202 transceiver was added on board in order to convert the 0 +5V input to the ±12V needed for RS-232 output levels. This interface is ment for communication with the DTG. In accordance with the 5DT data glove specification the UART was programmed to work at at 19200 bps (8 data bits, 1 stop bit, no parity). This is a bi-directional interface, allowing DAQB to send reset or request for data commands to the glove and receive data packets from the glove.

2. The analog input consists of six identical signal-conditioning stages in order to interface the uC internal ADC to six FSR attached to the five fingers and the palm of the hand. This is a unidirectional interface used only for reading the value of the FSRs.
3. The AduC 812 is the main component of the DAQB. The code running in the internal program memory was written in C and compiled with Keil uVision2 compiler. Thus the uC is able to initiate communication with the DTG and temporarily stores the received data. Also converts the analog inputs and store the result of conversion. When the PC or the FPGA asks for data the uC provide this data on a parallel data interface.
4. The parallel data interface uses 11 lines in order to ensure appropriate communication between the DAQB and the PC (via LPT1 printer port) or the FPGA. In table 1 we show how these ten lines connects to a standard LPT1 printer port. Data7-Data0 forms the actual data to be transmitted, while the last to signals are used for implementing a handshaking protocol.

Table 1.

ADuC pin	LPT1 pin	Description
P2.0	2	Data0
P2.1	3	Data1
P2.2	4	Data2
P2.3	5	Data3
P2.4	6	Data4
P2.5	7	Data5
P2.6	8	Data6
P2.7	9	Data7
P3.5/T1	13	Rdy
P3.6/WR	1	Ack

III. DATA STRUCTURE AND HANDLING

The glove resets to command mode if receives the byte 0x41H and confirm resetting sending 0x55H to the data acquisition board. After that the data acquisition board sends 0x43H to the glove and in response will receive a 9-byte data packet with the fallowing structure:

$\boxed{\text{Header}}$ $\boxed{\text{f1}}$ $\boxed{\text{f2}}$ $\boxed{\text{f3}}$ $\boxed{\text{f4}}$ $\boxed{\text{f5}}$ $\boxed{\text{pitch}}$ $\boxed{\text{roll}}$ $\boxed{\text{checksum}}$

$\boxed{\text{Header}}$ = 0x80H for any data packet.

$\boxed{\text{f1-f5}}$ are flexure values assigned to the thumb, index, middle, ring and little fingers. Each flexure value has a decimal range of 0 to 255, with a low value indicating an unflexed finger, and a high value indicating a flexed finger.

$\boxed{\text{Pitch}}$ and $\boxed{\text{roll}}$ are byte values ranging from 0 to 255. A value of 128 indicates that the hand is untilted in that axis.

After the data packet has been sent, the glove stops sending, until a new request to send (0x43H) is received. Since the DTG has no handshaking protocol during the transmission of a data packet the uC must store the whole 9-byte packet into the internal data memory. However, when it comes to send the data to the PC (or FPGA) only the seven bytes (f1, f2, f3, f4, f5, pitch, roll) of relevant information will be sent.

The internal ADC first converts the analog inputs. After completing the conversion the 12-bit result is stored in a register pair as shown in table 2.:

Table 2

ADCDATAH		ADCDATL	
MSB	LSB	MSB	LSB
Channel nr.	Bit 11 to 8	Bit 7 to 4	Bit 3 to 0

For our purposes it is useless to store these to bytes as they are. We can throw away the channel number information by shifting ADCDATAH four bits to the left. We do the same thing with de LSB of the result by shifting ADCDATAL four bits to the right. Concatenating these two registers we obtain the conversion result with 8-bit resolution. The SNR is thus improved and the result can be stored into a single memory location.
Now we have 13 bytes of information (seven from the DTG and six from the FSR) in the uC memory, ready

BUPT

to be sent to the PC or FPGA. This data packet has the following structure:

$\boxed{\text{f1}}\ \boxed{\text{f2}}\ \boxed{\text{f3}}\ \boxed{\text{f4}}\ \boxed{\text{f5}}\ \boxed{\text{pitch}}\ \boxed{\text{roll}}\ \boxed{\text{fsr(f1)}}\ \boxed{\text{fsr(f2)}}\ \boxed{\text{fsr(f3)}}$

$\boxed{\text{fsr(f4)}}\ \boxed{\text{fsr(f5)}}\ \boxed{\text{fsr(f6)}}$

where fsr(f1-f5) are data acquired from the FSRs attached to the fingertips and fsr(f6) is the data corresponding to the FSR attached to the palm of the hand.
In order to the data transfer accomplish properly the following communication protocol must be respected (see fig. 3) for each and every byte:

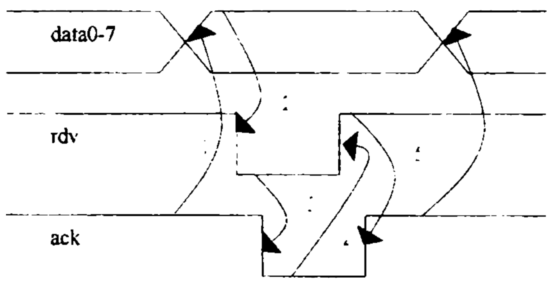


Fig. 3. Parallel Communication Protocol

1. The uC check if ack line is HIGH (if not wait until is so) and put a valid data on the parallel port.
2. rdy line is reset to a LOW level in order to inform the receiver that a new byte is available.
3. The uC check if ack line is LOW (if not wait until is so) thus confirming that the data was received.
4. rdy goes HIGH and the uC expects that in response...
5. ...ack line goes high and a new data can be sent.

This 5-step communication protocol ensures that some hazardous transition on ack or rdy lines will produce no effect.

IV. MATLAB INTERFACE EXAMPLE

When communicating with a PC it is possible to transfer data from the DAQB directly into mat lab's workspace. This is very useful in the implementation phase of a neural network when using Neural Network Toolbox. The input data set for training the network is stored in a matrix in workspace. The DAQB offers the data to be written in that matrix.
For example let's see how can we read the 13 data bytes. 50 consecutive times and store the readings in a 50x13 matrix in mat lab's workspace. One line of this matrix is the 13-byte data packet delivered by the DAQB. One column of this matrix represents 50 time samples of one particular sensor. The following matlab script running on a PC will do the job:

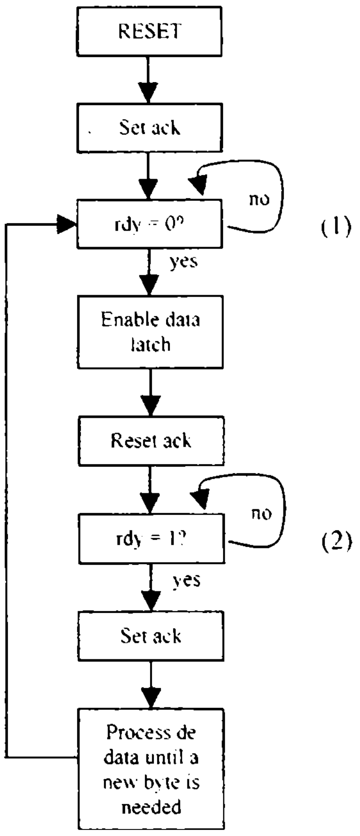
```
matport('Outport', 890, 252);
matport('Outport', 888, 255);
dio = digitalio('parallel','LPT1');
addline(dio,0:7,'in');
addline(dio,9,'in');
addline(dio,13,'out');
a=zeros(50,13);
ack = 0;
putvalue(dio.line(11),ack);
for i=1:50,
    for j=1:13,
        while getvalue(dio.line(9))==1;
        end
        b=getvalue(dio.line(1:8));
    a(i,j)=binvec2dec(b);
        ack = 1;
        putvalue(dio.line(11),ack);
        while getvalue(dio.line(9))==0;
        end
        ack = 0;
        putvalue(dio.line(11),ack);
    end
end
```

In order to gain direct access to PC I/O ports we used NTPort Library 2.5 Evaluation Version. Thus we added matport.dll to Matlab's Work folder. In fact the matlab script starts with two calls of this function:
The first call sets the LPT1 data buffer in input mode. The second call writes 255 to LPT1 output buffer because a LOW level on these lines would force to 0 the incoming data. From now on we can use the matlab digitalio function in order to define a digital input/output object on LPT1.
The next statements assign 3 ports to dio digital input/output object. The first port has 8 lines and is the input data port (LPT1 pin 2-9) the second input port has only one line, rdy (LPT1 pin 13) and the last port is a one line output port for sending ack to the data acquisition board (LPT1 pin 1).
The input data string is just a binary vector for matlab. In order to store-it in workspace as a decimal number we must specifically instruct matlab to do so (see example script).

V. INTERFACING WITH XC2S50 FPGA

The code running on the uC makes no difference between a PC and any other hardware connected to his parallel interface. As long as the communication protocol is respected the DAQB will provide data to no matter who ask for it.
For this reason we need to implement not only the neural network but also a parallel port and communication protocol circuitry on the same FPGA. This extra circuit consumes very few resources so it is not a bargain at all. The communication logic must work according to the following algorithm:

BUPT



The above diagram executes with the speed given by the XC2S50 clock except for stages 1 and 2 where a response from the DAQB is expected. The code running on the uC is so written that rdy automatically responds in the next instruction cycle to any changes in ack line. So the maximum delay introduced by steps 1 or 2 is about 1μs in length. In these conditions the transfer time of a single byte can be as short as 2.5 – 3us. But after 13 bytes were transferred from the uC internal data memory it takes about 6.5ms for the DAQB to acquire another set of 13 data. This time can be used by the XC2S50 to process the previously acquired data.
In the actual stage of implementation the neural network implemented in FPGA is used for gesture recognition. Each 13-byte data packet is associated to a gesture. In response to the input data the network displays the gesture number

VI. CONCLUSIONS

1. This development framework is very flexible due to the fact that both the DAQB and the FPGA are in-system reprogramable. The same hardware can fulfill various tasks.
2. The serial I/O port may be used for communication not only with the actual sensor system but also with any other hardware that has a RS 232 serial interface.
3. The use of other sensor than FSR is possible but, depending on the particular sensor type, some additional signal conditioning may be required.

REFERENCES

[1] ***, *ADuC812 MicroConverter*, Analog Devices Inc., 2003
[2] R. Holcer, L. Michaeli, J. Šaliga, *The Test Of The A/D Converters Embedded On Two Microcontrollers*, Measurement Science Review, Volume 1, Number 1, 2001
[3] ***, *5DT Data Glove 5 User's Manual*, Fifth Dimension Technologies, February, 2000

Foto 1. A picture of the entire system

BUPT