Microprocessors and Microsystems
journal homepage: http://www.elsevier.com/locate/micpro

Electronic Nose System on The Zynq SoC Platform

Amine Ait Si Ali[a,b], Hamza Djelouat[a], Abbes Amira[a], Faycal Bensaali[a], Mohieddine Benammar[a], Amine Bermak[c]

[a] *College of Engineering, Qatar University, Doha 2713, Qatar*
[b] *Department of Computer and Information Sciences, University of Northumbria, Newcastle, NE2 1XE, UK*
[c] *College of Science and Engineering, Hamad bin Khalifa University, Doha, P.O. Box: 5825, Qatar*

ABSTRACT

Electronic nose or machine olfaction are systems used for detection and identification of odorous compounds and gas mixtures. An electronic nose system is mainly made of two parts, the sensing part which takes the form of a single or a set of sensors and the processing part which takes the form of some pattern recognition algorithms. As an alternative solution to pure software or hardware implementation of the processing part of a gas identification system, this paper proposes a hardware/software co-design approach using the Zynq platform for the implementation of an electronic nose system based on principal component analysis as a dimensionality reduction technique and decision tree as a classification algorithm using two different sensors array, a 4×4 in-house fabricated sensor and a commercial one based on 7 Figaro sensors, for comparison purpose. The system was successfully trained and simulated in MATLAB environment prior to the implementation on the Zynq platform. Various scenarios were explored and discussed including the investigation of different combination of principal components as well as the utilization of drift compensation technique to improve the identification accuracy. High level synthesis was carried out on the proposed designs using different optimization directives including loop unrolling, array partitioning and pipelining. Hardware implementation results on the Zynq system on chip show that real-time performances can be achieved for proposed EN systems using hardware/software co-design approach with a single ARM processor running at 667 MHz and the programmable logic running at 142 MHz. In addition, using the designed IP cores and for the best scenarios, a gas can be identified in 3.46 μs using the 4×4 sensor and 0.55 μs using the Figaro sensors. Furthermore, it has been noticed that the choice of the sensor array has an important impact on performances in terms of accuracy and processing time. Finally, it has been demonstrated that the programmable logic of the Zynq platform consumes much less power than the processing system.

1. Introduction

Gas monitoring is one of the critical challenges in the gas industry. A single leakage of explosive gas is enough to destroy the whole gas plant. Moreover due to the recent environmental degradation exposure of hazardous gases is a challenging task in different parts of the world (Yamazoe and Miura (1994)). Therefore, the concept of Electronic Nose (EN) is introduced based on humans olfactory (Pearce et al. (2006)), it becomes widely adopted not only in gas industry but in different fields such as in food industry (Schaller et al. (1998)), storage of gases (Kharaka et al. (2009)) and the computation of the air pollutant (Zampolli et al. (2004)). An EN system can be described by two main blocks for data collection and data processing. The first block is called the sensing block and it takes the form of a single or a set of sensors. The second block is called the processing block and it takes the form of some pattern recognition algorithms. Most of the existing implementation of EN systems are software based. Taking in consideration the fact that the information provided by the EN is critical in many applications and must be in real-time, many EN have been implemented on multiple hardware platforms for acceleration. Each

**Corresponding author:
e-mail: amine.aitsiali@qu.edu.qa
amine.ali@northumbria.ac.uk (Amine Ait Si Ali)

of the pure software pure hardware solution have advantages and disadvantages. An implementation of an EN system that takes advantages of both software and hardware is highly required. In this paper, a hybrid implementation of an EN based on PCA for dimensionality reduction and DT for classification using two different sensor arrays is presented. One of them is an in-house 4×4 fabricated sensor while the other one is based on 7 Figaro sensors. The main aim is to present an innovative Hardware/Software (HW/SW) co-design approach for an EN implementation on the Zynq platform by implementing computationally intensive algorithms on the FPGA part and the remaining on the processor using HLS over hardware description language to reduce the prototyping and development time. The second aim is the comparative study for the two types of sensors (in-house one and commercial one) and their impact on the EN System. The main objective of this paper is to present an innovative way to implement pattern recognition algorithms on hardware to accelerate the execution time while minimising power consumption as well as design and development time, this is very important for critical applications. A hardware/software approach is used for the implementation of various pattern recognition algorithms on the Zynq SoC using high level synthesis. A case study of EN is taken into consideration as a critical application to evaluate the performances of the proposed innovative method while PCA and DT are taken into consideration as the pattern recognition algorithms. It is worth mentioning that this paper is the continuity of previous works (Akbar et al. (2015)) and (Ait Si Ali et al. (2015)). The work presented in (Akbar et al. (2015)) is concerned with RFID transmission, it deals with how the data is collected and transmitted to the processing unit, the EN which is implemented on the processing unit is briefly mentioned without much details. Only one type of sensor is used to collect different datasets at different temperatures. The work presented in (Ait Si Ali et al. (2015)) uses also one type of sensor. This paper has a considerable new contribution since two type of sensors are used as well as some different gases. It shows how the performances of an EN can be affected by the choice of sensor. Furthermore, the hardware implementations of pattern recognition algorithms are evaluated and described in much more details. The remainder of this paper is organized as follows. Section 2 describes the exiting work along with shortcomings. In Section 3, an overview of the EN system is presented including experimental setup, data collection, feature extraction, dimensionality reduction using PCA and classification using DT. Section 4 is concerned with the implementation of the EN on the Zynq platform. A brief description of the Zynq platform is given as well as details about the implementation and design flow using HLS and Vivado suite. In section 5, implementation results are presented and discussed. Section 6 concludes the paper.

2. Related Work

A portable EN with sensor array made of thick-film of oxide based semiconductor sensing materials is introduced for gas applications in (Hong et al. (2000)). Further, a gas identification system using eight tin-oxide gas sensors implemented on Field Programmable Gate Arrays (FPGA) is presented in (Benrekia et al. (2013)). Euclidean Normalization is used for the preprocessing of the data prior to the classification which is performed using Multilayer Perceptron (MLP) algorithm. Whereas in (Far et al. (2009)) a temperature modulation scheme is used for gas identification. Self-Organized Map (SOM) technique is used as a preprocessing technique to transform the data into 2D image then the image moments and Linear Discriminant Analysis (LDA) are used to extract feature from the images before the classification. In the solution presented Far et al. (2009) a 16-array sensor is used. Sensitivity of sensors is one of the major concerns in EN systems. Therefore, Arshak et al. in (Arshak et al. (2004)), analyses different sensors used for EN systems. The main problem with the sensors used in current EN system is their lack of selectivity. Therefore, in order to overcome the problem of selectivity the number of sensors in the EN is increased to get more signatures for the same gas such as in (Guo et al. (2007)) where a 4×4 array of tin-oxide based gas sensor is developed. However increasing the number of sensors will also increase the data size which improves classification but also increases the computation complexity. Therefore, a feature reduction algorithm is required to extract the most relevant information from the data. Different research approaches have already been presented for feature reduction like Independent Component Analysis (ICA) (Li et al. (2005)) and multidimensional scaling (Chandrasiri et al. (1999)). However, Principal Component Analysis (PCA) is one of the most common techniques used in machine learning for dimensionality reduction (Honeine (2012)). PCA was used in (Bravo et al. (2010)) for the detection of moving object where the training and testing parts are both implemented on a Xilinx Virtex-II based FPGA platform. Cascaded adders and dividers have been used to compute the mean of the training data while a hardware architecture is presented for the computation of Eigenvectors needed in Jacobi method. Furthermore, in (Perera and Li (2011)) a hardware implementation of the PCA based learning algorithm is presented. The implementation was carried out on a Xilinx Vertex-6 based FPGA platform in which partial dynamic reconfiguration is used to implement the mean and covariance steps of PCA algorithm. Along with the feature reduction in an EN system, an efficient classifier is required to discriminate the type of gas. Therefore, in (Shi et al. (2008)) a Committee Machine (CM) is proposed to identify the gases. The CM is based on five commonly used classifiers including K Nearest Neighbors (KNNs), MLP, Radial Basis Function (RBF), Gaussian Mixture Model (GMM) and Probabilistic Principal Component Analysis (PPCA). Whereas, in (Brahim Belhouari et al. (2004)) a gas recognition system is designed and executed on a processor using the comparative analysis between the classification obtained from the density model and the discriminant function. The density estimator used KNN, GMM and Generative Topographic Mapping (GTM) based classifier, while RBF, MLP and Generalized Linear Discriminant Analysis (GLDA) based classifier is used for determining the discriminant function. Another software implementation of an EN is presented in (Kim et al. (2012)), eight commercial sensors are combined to create a sensor array, Smoothed Moving Average (SMMA) is used

to preprocess the data along with normalization to make the data in a range between 0 and 1. In this solution, Genetic Algorithm (GA) and Artificial Neural Network (ANN) are combined to develop a Neural Genetic Classification Algorithm (NGCA). A hardware implementation on both FPGA and Application-Specific Integrated Circuit (ASIC) of an EN system that uses a 16-array sensor and based on PCA and Binary Decision Tree (DT) is presented in (Li and Bermak (2011)). The conducted literature review has shown that the EN systems have been implemented either using a uniprocessor-based software approach or a hardware-based implementation approach to accelerate the slow software-based approach to meet the real-time requirement. Hardware platforms such as FPGAs have been used for this purpose. With the emergence of novel platforms such as the Xilinx Zynq which holds in the same chip a processor and an equivalent of FPGA allows an efficient and quick hybrid based implementation approach especially when associated with High Level Synthesis (HLS) Tool. In a hybrid implementation, computationally intensive blocks of the EN system can be executed on hardware while the remaining non-complex tasks can be executed on a processor in a software manner. This approach will not only reduce the power consumption of the hardware but also will free more space in the FPGA part for other hardware acceleration related tasks.

3. System Design

3.1. System Overview

The proposed EN system consists of three main parts as shown in figure 1. The first part is for sensing where two types of SnO_2 based sensor array are evaluated. The first sensor array is based on seven commercial Figaro sensors (Sensors (2015)) while the second is a 4×4 in-house fabricated sensor array (Guo et al. (2007)). The second part of the EN system is for dimensionality reduction where PCA is used to reduce the size of the input vector without affecting the overall performance of the system. The last part is the classification where a decision is made to identify a given gas. For the proposed EN system, a DT classifier is used in this part. It is worth mentioning that the system has to be trained offline and tested online. The training is performed in MATLAB while the testing is done in both MATLAB and on the Zynq platform where the EN is to be implemented on.

3.2. Data Collection

Data collection is an important step which consists of the gathering of the data required for training, testing and validation. The experimental setup used in the laboratory to collect the data is shown in figure 2. It consists of a gas chamber where the sensor array is placed. The gas chamber has two orifices, one to serve as an input for the in-flow of gases and the other one as an exhaust to evacuate the gases. Multiple gases are stored in various cylinders and connected to the gas chamber individually through several Mass Flow Controllers (MFCs). A control unit is connected to the MFCs to control the in-flow of gases and to the sensor array via a Data Acquisition (DAQ) system to collect and sample the response of the sensor array. Two

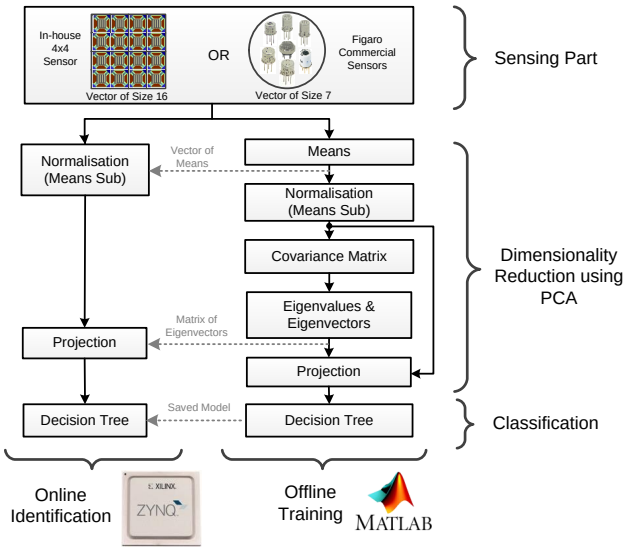

Fig. 1. Building blocks of the proposed EN system

Table 1. Sensors specifications

Sensors	Target Gases	Concentration (ppm)	Power (mW)
TGS826	NH_3	30-300	833
TGS2442	CO	30-1000	14
TGS2600	H_2	1-30	210
TGS2602	C_2H_6O	1-30	280
TGS2610	C_4H_{10} & C3H8	500-10000	280
TGS2611	CH4	500-10000	280
TGS2620	Alcohol & Solvent vapors	50-5000	210
In-house 4x4	CO H_2 C_2H_6O CH_4	25-260 25-800 50-200 500-5000	352

data sets are collected, one using the 4×4 in-house sensor array and the other one using the seven Figaro sensors. Specifications of the used sensors in terms of target gases, typical concentration detection range and power consumption are listed in table 1. It is worth mentioning that the target gases in the case of the Figaro sensors illustrate the main gas that the sensor can detect. However, the same sensor can detect multiple other gases. As an example, carbon dioxide can be detected by many of the mentioned Figaro sensors.

It is worth mentioning that the sensors starting by TGS are the commercial Figaro sensors used to form a sensor array. The layout of the 4×4 sensor is made of 16 sections organized in four rows and four columns where each section represents one sensor. To modify the response of the SnO_2 sensing film for each part, a post treatment is performed to improve the selectivity by depositing various noble metal additives on the sensing film. The post treatment is realized by metal doping and ion implantation. Three noble metals (Platinum (Pt), Palladium

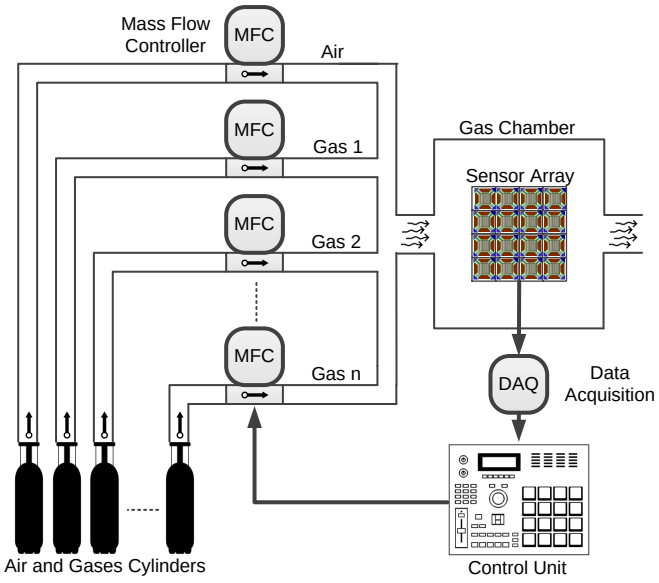

Fig. 2. Experimental setup

Table 2. 4×4 **sensor array post treatment scheme**

Row	Column 1	Column 2	Column 3	Column 4
1	N/A	Pt	Pd	Au
2	B	Pt/B	Pd/B	Au/B
3	P	Pt/P	Pd/P	Au/P
4	H	Pt/H	Pd/H	Au/H

(Pd) and Gold (Au)) are combined with three ions (Boron (B), Phosphorus (P) and Hydrogen (H)). A summary of the post treatment is shown in table 2.

Six different gases are targeted in total, two using the 4×4 sensor array (Carbon monoxide (CO) and Ethanol (C_2H_6O)), three using the Figaro sensors (Carbon dioxide (CO_2), Propane (C_3H_8) and Ammonia (NH_3)) and one common to both (Hydrogen (H_2)). The choice of gases is for evaluation purpose, specific gases can be targeted for specific applications where the EN system can be easily adapted to handle new types of gases. For the first data set, 60 patterns are collected using the 4×4 sensor array representing two cycles of 10 concentrations (20, 40, 60, 80, 100, 120, 140, 180 and 200 parts per million (ppm)) for the three gases (CO, C_2H_6O and H_2). Before each gas concentration injected for 250 seconds, air is injected for 750 seconds. For the second data set, 200 patterns are collected using the seven Figaro sensors. 30, 40, 50 and 80 patterns are collected for C_3H_8, CO_2, NH_3 and H_2 respectively. Those patterns are representing five cycles of eight concentrations (25, 50, 75, 100, 125, 150, 175 and 200 ppm) for CO_2, while it is representing three cycles, five cycles and eight cycles of 10 concentrations (25, 50, 75, 100, 125, 150, 175, 200, 225 and 250 ppm) for C_3H_8, NH_3 and H_2 respectively. Each injection of C_3H_8 which lasts for 250 seconds was preceded by air injection for 500 seconds while each injection of CO_2, NH_3 and H_2 which lasts for 500 seconds was preceded by an air injection for 750 seconds. The injection time is chosen to give enough time to sensors to reach the baseline in the case of air and the Steady State (SS) in the case of gas mixture. When sensors are exposed to a given gas, the voltage measurement will settle at a specific value, this value is called the SS. It is worth mentioning that the sampling time for the 4x4 sensor array is one second while it is 10 seconds for the Figaro based sensor. Part of the raw responses for one cycle of various concentration of H_2 for both sensors are shown in figure 3. H_2 is chosen for comparison purpose since it is the only common gas between the two sensors. The voltage measurement of the 4×4 sensor array decreases when exposed to the target gas while it increases for the Figaro sensors.

3.3. Feature Extraction

There are various techniques to generate descriptive parameters from the raw responses of the sensors (Bermak et al. (2006)). The most common used feature is the SS. SSs corresponding to all gases and concentrations are extracted manually form the data set by taking the values corresponding to the end of each gas injection period. Figure 4 represents the SSs generated in one cycle for both 4×4 and Figaro based sensors. The SSs are organized in vectors of size 16 and 7 for the 4×4 sensor array and the 7 Figaro sensors respectively. The SSs are then given to the system as an input. The other important values extracted from the raw signals are the baselines which are used for drift compensation. When the gas chamber is flushed with air, the voltage measurement of the sensor will settle at a specific value, this value is called the baseline. Baselines are also extracted manually form the data set by taking the values corresponding to the end of each air injection period. It is worth mentioning that 50% of the collected data is used for training and the remaining 50% is used for testing.

3.4. Dimensionality Reduction Using PCA

The first step in the EN is the dimensionality reduction. The aim is to reduce the number of features taken in consideration which is the SSs in this case. The objectives are to reduce the size of the input vector from 16 in the case of the 4×4 sensor array and from 7 in the case of the Figaro sensors to two or three without affecting the performance of the EN system. The classification accuracy when dimensionality reduction is applied should be higher or at least equal to the initial accuracy when dimensionality reduction is not applied. This preprocessing technique is very important since it reduces the computation complexity considerably in most cases and it is also considered as a feature selection technique which helps to increase the classification accuracy. PCA is one the most used techniques in EN systems (Marco and Gutiérrez-Gálvez (2012)). PCA reduces the dimensionality of the data by projecting it into a new space where the axes are classified in a decreasing order of importance. The first principal components PCA1 which are corresponding to the first axe are the ones that keep the most dissimilarity between the samples of the data then followed by PCA2, PCA3 ... etc. As it can be seen in figure 1, there are two parts when PCA is used as a dimensionality reduction technique. The training part is performed offline while the testing one is executed in both offline and online for verification purpose. The

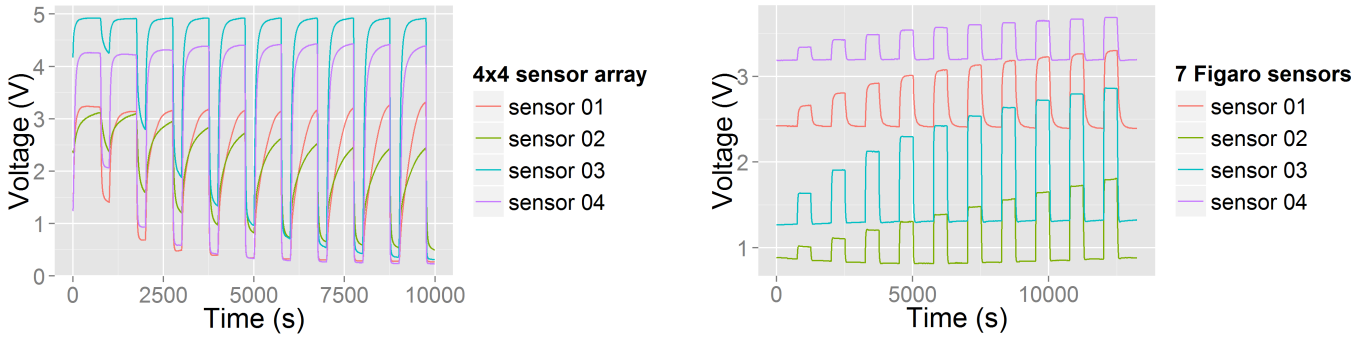

Fig. 3. Sensors responses for one cycle of H_2

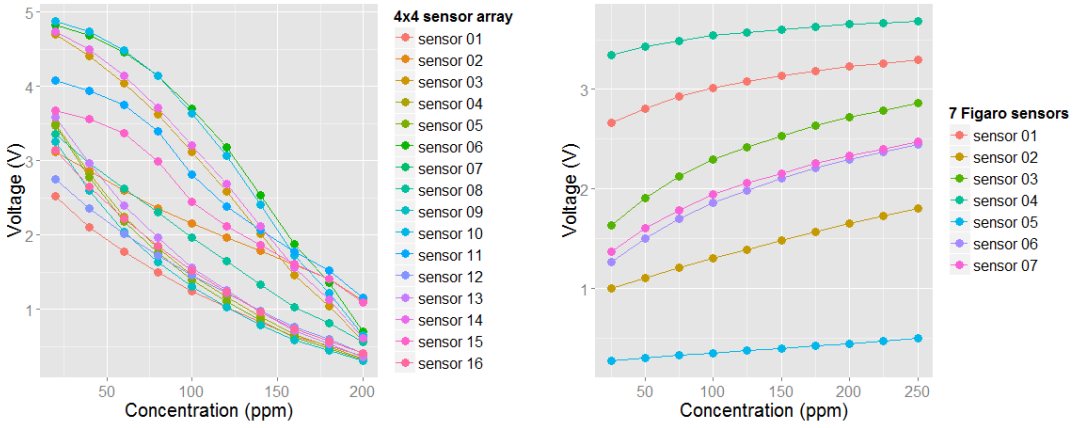

Fig. 4. Steady states for one cycle of H2

first step for the training part is to organize the collected training data set in an $M \times N$ matrix A as shown in equation (1).

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \tag{1}$$

Where ($a_{mn} \in R$) such as the number of rows m represents the dimension which is the number of sensors (16 for the 4×4 sensor array and 7 for the Figaro based sensor) and the number of columns n is the number of training samples (30 for the 4×4 sensor array and 100 for the Figaro based sensor). The output of the sensor i at a given concentration for a given gas j is represented by a_{ij}. The mean $\bar{\mu}_i$ for each dimension is then computed as shown in equation (2).

$$\bar{\mu}_i = \frac{\sum_{j=1}^{n} a_{ij}}{n} \tag{2}$$

The result is saved in a vector U of size m as shown in equation (3).

$$U = \begin{bmatrix} \bar{\mu}_1 \\ \vdots \\ \bar{\mu}_m \end{bmatrix} \tag{3}$$

The following step is the normalization which consists of the subtraction of the mean from each element and saving the result in an $M \times N$ matrix B as shown in equation (4).

$$B = \begin{bmatrix} a_{11} - \bar{\mu}_1 & \cdots & a_{1n} - \bar{\mu}_1 \\ \vdots & \ddots & \vdots \\ a_{m1} - \bar{\mu}_m & \cdots & a_{mn} - \bar{\mu}_m \end{bmatrix} \tag{4}$$

The third step in the PCA training is the computation of the covariance $M \times N$ matrix COV according to equation (5).

$$COV = \frac{B \times B^T}{n - 1} \tag{5}$$

Where B^T is the transpose of B. The next and most computationally intensive part of the PCA training algorithm is the computation of eigenvalues and eigenvectors. It has been well described in (Ait Si Ali et al. (2013)), the resulting $M \times M$ matrix V (6) corresponds to the m eigenvectors of size m organized in columns and in decreasing order of importance (v_{i1} are values of the most important eigenvector while v_{im} are the values of the last and least important eigenvector).

$$V = \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{mm} \end{bmatrix} \tag{6}$$

Finally, the m principal components are computed in the projection step using equation (7) and saved in $M \times N$ where V^T is the transpose of V.

$$PCA = V^T \times B \tag{7}$$

Figure 5 shows the visualization of the training data set when it is reduced from 16 dimensions in the case of 4×4 sensor array and from 7 dimensions in the case of the Figaro based sensors to two dimensions (PCA1 and PCA2) in both cases. The testing part of PCA requires only two steps which are the normalization and the projection, the computations are based on parameters calculated and saved in the training phase, these parameters are the vector of means and matrix of eigenvectors. Figure 6 shows the visualization of the testing data set when it is reduced to two dimensions.

3.5. Classification Using DT

The second block in the EN system is the classification using DT. The binary DT classifier is chosen for its simplicity, yet with good performance. The input of the learning algorithm is a set of labeled data and the output is a binary tree, it is a supervised learning technique. The generated tree is used for the classification of a new input. The classification process starts at the root decision node of the tree and ends at one of the leaf nodes which represents a specific class passing by intermediate decision nodes. The DT training is performed in MATLAB using various scenarios. In other words, the training is achieved and a tree is generated when the input of the classifier is the SSs without preprocessing and dimensionality reduction and when dimensionality reduction is applied using various combinations of PCAs. It is worth mentioning that the Delta which is the difference between the SS and the baseline is also considered as a feature for identification to overcome the sensor drift problem which is considered as a drift compensation technique (Gutierrez-Osuna (2002)). Sensor drift is one of the major concerns in current gas sensors. Sensor drift is defined as slow temporal variations of the sensor response when exposed to the same mixture. The DT training algorithm requires two inputs, one is the matrix of predictor values containing the training dataset in terms of SSs, deltas or PCAs where rows represent observations and columns represent features. The second input is a vector with the same number of rows as the matrix of predictor. Each row of the vector represents the class of the corresponding row of the matrix of predictor. It is worth mentioning that the classification problem deals with three classes corresponding to three gases when the 4×4 sensor is used while in the case of 7 Figaro sensors, four gases are used. A summary of all scenarios and class labels are presented in table 3. Results of the training as well as details about the different DTs are shown in tables 4, 5, 6 and 7. Those tables have been obtained after performing the training and testing in MATLAB, the tables show the classification accuracy for each tree which is obtained using formula (8), the number of selected predictors which correspond to the dimension that the generated tree is using after training is performed and tree characteristics in terms of nodes, leaves and depth. As an example for dimensions, in table 4 it can be seen that the initial dimension for SS is 16 while for two PCAs it is two. It is worth mentioning that various combinations of PCAs are used to evaluate the performance. However, it can be seen from tables 4, 5, 6 and 7 that the training is performed for up to 5 PCAs for the 4×4 sensor array and up to 4 PCAs for the Figaro sensors when the starting point is the SS while the training is performed for up to 5 PCAs for the 4×4 sensor array and up to 3 PCAs for the Figaro sensors when the starting point is the Delta between the SS and the baseline because it starts generating the exact same tree. The accuracy is computed according to formula (8). The DT learning algorithm does not take in consideration all predictors for the generated tree model in all cases. The algorithm will select the best predictors to successfully classify all observations as it can be seen in the tables. Furthermore, it can be noted that the 100% accuracy has been reached many times in the case of 7 Figaro sensors as mentioned in tables 6 and 7. When the SS are used, the 100% accuracy is reached only when two PCAs are being used as predictors. However, when the deltas are being used, the 100% accuracy is reached in all cases with and without PCA which results in the fact that in this particular scenario PCA is not required as it does not improve the accuracy.

Table 3. Classification dimensions and class labels

Type of sensor	Dimension of inputs	Class ID	classification labels
4 × 4	16	1	Hydrogen (H_2)
		2	Carbon monoxide (CO)
		3	Ethanol (C_2H_6O)
7 Figaro	7	1	Hydrogen (H_2)
		2	Carbon dioxide (CO_2)
		3	Propane (C_3H_8)
		4	Ammonia (NH_3)

Table 4. DTs for 4×4 sensor array using steady states with and without PCA

DT characteristics	Classification features				
	SS	2 PCAs	3 PCAs	4 PCAs	5 PCAs
Classification accuracy	73.33%	80%	90%	80%	80%
No. of predictors	16	2	3	4	5
No. of selected predictors	4	2	3	3	3
No. of trees nodes	9	11	9	7	7
No. of trees leaves	5	6	5	4	4
Tree depth	3	4	3	2	2

Table 5. DTs for 4×4 sensor array using delta between baselines and steady states with and without PCA

DT characteristics	Classification features				
	Delta	2 PCAs	3 PCAs	4 PCAs	5 PCAs
Classification accuracy	63.30%	83%	86%	96.66%	96.66%
No. of predictors	16	2	3	4	5
No. of selected predictors	4	2	3	3	3
No. of trees nodes	9	11	11	7	7
No. of trees leaves	5	6	6	4	4
Tree depth	3	4	4	2	2

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of testing observations}} \times 100 \tag{8}$$

4. System Implementation

The Zynq System-on-Chip (SoC) platform is chosen for the hardware acceleration of the gas identification system due

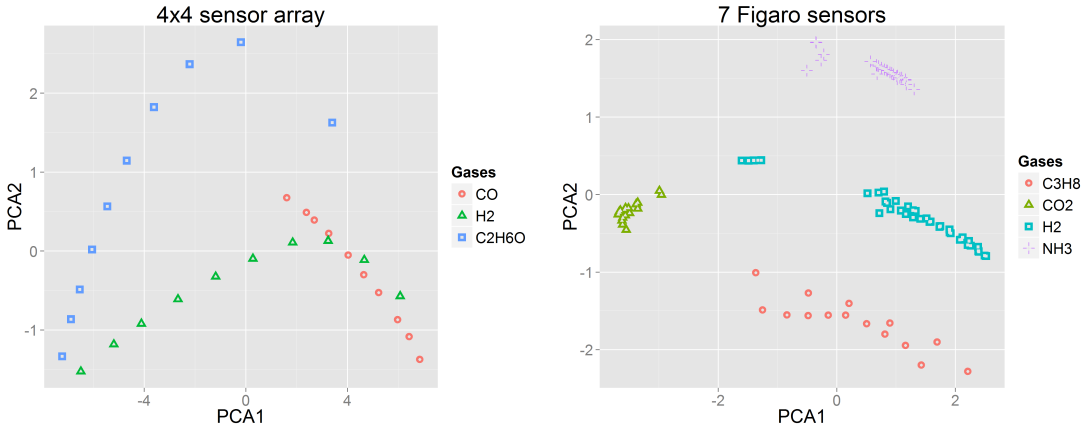

Fig. 5. Dimensionality reduction using PCA1 and PCA2 for the training data

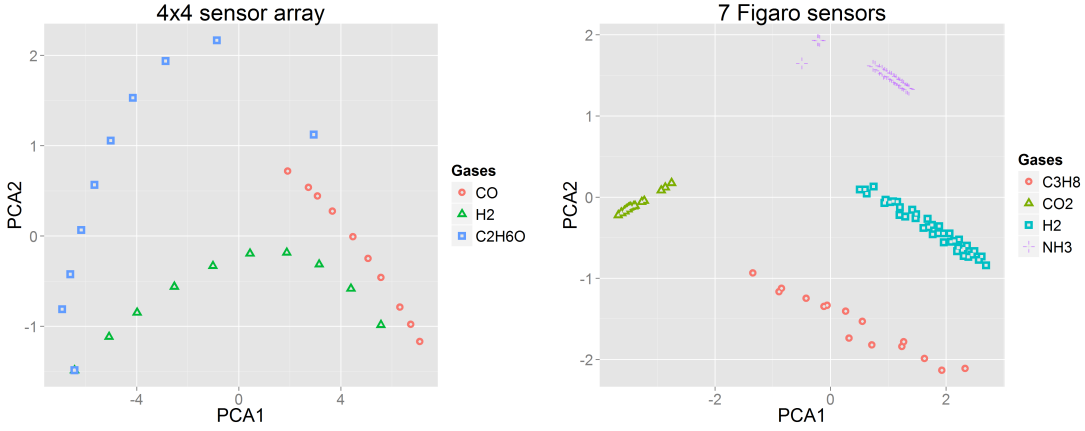

Fig. 6. Dimensionality reduction using PCA1 and PCA2 for the testing data

Table 6. DTs for 7 Figaro sensors using the steady states with and without PCA

DT characteristics	Classification features			
	SS	2 PCAs	3 PCAs	4 PCAs
Classification accuracy	99%	100%	99%	99%
No. of predictors	7	2	3	4
No. of selected predictors	3	2	3	3
No. of trees nodes	9	7	9	9
No. of trees leaves	5	4	5	5
Tree depth	4	3	3	3

Table 7. DTs for 7 Figaro sensors using Delta between steady states and baselines with and without PCA

DT characteristics	Classification features		
	Delta	2 PCAs	3 PCAs
Classification accuracy	100%	100%	100%
No. of predictors	7	2	3
No. of selected predictors	1	2	2
No. of trees nodes	7	7	7
No. of trees leaves	4	4	4
Tree depth	3	3	3

to its flexibility and suitability for a HW/SW co-design approach. The Xilinx Zynq-7000 all programmable SoC combines a traditional FPGA based on Xilinx 7-series forming the Programmable Logic (PL) with a dual core ARM Cortex-A9 processor forming the Processing System (PS). The PL is based on Artix-7 or Kintex-7 with different variants. Details about the Zynq-7000 all programmable SoC can be found in (Xilix (2014d)). The combination of the PS and the PL inside the same chip makes platforms based on the Zynq SoC suitable for HW/SW co-design approach. Especially when associated with Xilinx Vivado HLS tool, Vivado IP Integrator and Software Development Kit (SDK) which will allow a high level of abstraction with benefits in terms of performance, cost and power compared to a conventional FPGA or processor implementations. The starting point for a HW/SW co-design on the Zynq SoC is the system design where all the desired requirements and specifications of the top-level system and various subsystems are set. The following stage is the software profiling where a software code is executed in a processor and with the help of a profiling tool such as GProf, computationally intensive parts of the program can be identified. The designer can then decide which parts are to be executed on hardware (i.e. PL) and which ones are to be executed on the ARM processor (i.e. PS). The

third step consists of the design of the IP Cores for hardware acceleration of the computationally intensive parts. IPs can be designed using various approaches. One approach is the use of Vivado HLS where the IPs are created and simulated using C, C++ or SystemC and then exported. Another approach is the use of Xilinx System Generator. The IPs can also be designed using a hardware description language (HDL) such as VHDL or Verilog directly. In all cases the IPs are exported and added to Vivado IP Catalog. A hardware system is then created in Vivado using IP Integrator where all blocks are interconnected including Zynq SoC and the previously created IPs. The following stage is to export the hardware design to SDK, here a software code is to be written and executed on PS. The software corresponds to the parts of the design not implemented on PL and also to manage the IPs implemented on the PL. The PL is programed from SDK.

4.1. System Architecture

A simplified architecture of the EN on the Zynq SoC for a HW/SW co-design implementation when used in the context of a multisensing platform for gas application is shown in figure 7. This platform consists of a Radio Frequency Identification (RFID) tag that contains a gas sensor and temperature sensor the second part is the processing unit which is implemented on the Zynq SOC. The third component is the RFID reader which is connected to the processing unit to read and transmit data from the RFID tag to the processing unit. PCA and DT are executed on the PL while data acquisition and results visualization blocks are executed in the PS. At this stage, while the RFID tag is still under fabrication, data acquisition block is not implemented and instead, data is stored in the processing system and directly read from memory to be able to evaluate the system within the Zynq board. The choice of RFID is due to the fact that this paper is part of a larger ongoing research project which aims at the development of a low power reconfigurable self–calibrated multi-sensing platform for gas application. Part of this project is the integration of a temperature and gas sensor into an RFID tag. Sensors that can be integrated with RFID to monitor physical and chemical parameters such as temperature, pressure and gas are extremely critical for a very wide range of applications in environmental monitoring, security, military and more importantly in the gas industry. The integration of sensors with RFID will allow the deployment of a new generation of smart devices leading to a massive wave of new emerging applications. Recently, a significantly increasing interest in integrating sensors with RFID is being witnessed, leading to the concept of sensor tag. A sensor tag is an RFID tag, which contains a sensor to monitor physical parameters while supporting the same identification function of a normal RFID tag.

4.2. System Implementation on the Zynq SoC using Vivado

The implementation of the EN based on PCA as a dimensionality reduction technique, and DT as a classifier on the Zynq SoC using HLS requires many steps. First the system is designed, simulated, evaluated and validated in MATLAB. Then Vivado HLS is used to create the corresponding Register Transfer Level (RTL) design for the system's algorithm described in

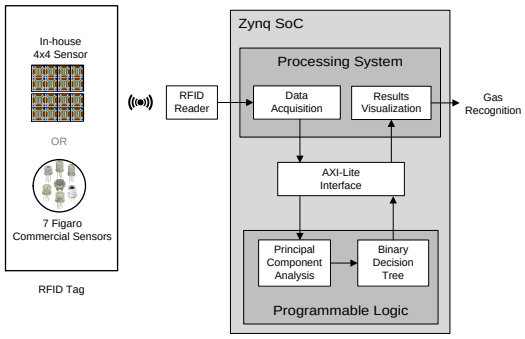

Fig. 7. EN system architecture

C. The next step is the hardware development performed in Vivado using IP Integrator. The final stage is the software development and FPGA programing realized in SDK. All versions of HLS, Vivado and SDK used for the design and implementation are the latest at the time of writing which is the 2015.1 one. The prototyping board being used is the Zynq ZC702.

IP Design Using Vivado HLS. First of all, the C source code corresponding to the gas identification system is written. The code consists of a function called "Predict". The input of the function is a vector of 16 floating-point elements in the case of the 4×4 sensor array and 7 floating-point elements in the case of the Figaro sensors. The output of the function is an integer: "1" for CO, "2" for C_2H_6O and "3" for H_2 in the case of 4×4 sensor array. On the other hand, it is "1" for C_3H_8, "2" for CO_2, "3" for H_2 and "4" for NH_3 in the case of Figaro sensors. Within the function the vector of means and required eigenvectors are declared and initialized, only needed eigenvectors are stored. The vector of means is used for normalization. The normalization consists in the subtraction of each mean from the corresponding value of the input vector. The projection consists of the multiplication of the normalized vectors by the eigenvectors of same size, the resulting floating-point values are used by the DT to classify the input. The DT takes the form of a succession of If and Else statements resulting in the output being "1", "2" or "3" for the 4×4 sensor array and "1", "2", "3" or "4" for Figaro sensors. Algorithm 1 shows the pseudo code written for one of the four solutions implemented in HLS.

A second C file is needed for testing since in Vivado HLS the testbench is also written in C. The C testbench takes the form of the main C function that will execute the "Predict" function and self-check the results. It is worth mentioning that Vivado HLS allows the user to export the IP to IP Catalog (For Vivado), Pcore (For Embedded Development Kit (EDK)) or system generator. Drivers related to the designed piece of hardware are included in the IP package. They will be used by the software managing the core from the processor. Different optimization directives are applied including loop unrolling, array partitioning and pipelining. The first "Unroll" directive applied to the loop where the mean and projection are computed is very powerful. It allows loops to be executed in parallel having dedicated hardware resources for each loop. Each array in the func-

Algorithm 1 E-Nose (PCA+DT) Prediction Algorithm

Input: Sensor Array responses $X[16]$
Output: Gas_ID: "1" for CO, "2" for C_2H_6O and "3" for H_2

function PREDICT(X)
  $Mean[16] \leftarrow$ The means of each dimension computed in the training phase of PCA
  $Eigen1[16] \leftarrow$ The Eigenvector corresponding to the 1^{st} dimension of PCA
  $Eigen2[16] \leftarrow$ The Eigenvector corresponding to the 2^{nd} dimension of PCA
  $Eigen3[16] \leftarrow$ The Eigenvector corresponding to the 3^{rd} dimension of PCA
  $PCA1 \leftarrow 0$
  $PCA2 \leftarrow 0$
  $PCA3 \leftarrow 0$
  for $i = 0$ to 16 **do** ▷ Next is computation of PCA1, PCA2 and PCA3
    $X[i] \leftarrow X[i] - Mean[i]$ ▷ PCA normalization
    $PCA1 \leftarrow PCA1 \times Eigen1[i]$ ▷ PCA projection
    $PCA2 \leftarrow PCA2 \times Eigen2[i]$
    $PCA3 \leftarrow PCA3 \times Eigen3[i]$
  end for
  if $PCA3 < 0.134946$ **then** ▷ next is Decision Tree Model
    if $PCA2 < 0.177746$ **then**
      if $PCA1 < 3.63144$ **then**
        $Gas\_ID \leftarrow 2$
      else
        $Gas\_ID \leftarrow 1$
      end if
    else
      $Gas\_ID \leftarrow 1$
    end if
  else
    if $PCA1 < 4.31419$ **then**
      $Gas\_ID \leftarrow 3$
    else
      $Gas\_ID \leftarrow 1$
    end if
  end if
end function

tion can be considered as one entity having limited data ports for data transfer or multiple entities using the "Array Partition" where each entity is having its own data ports. "Array Partition" is applied to the input array which results in the breakdown of the array into various sub arrays. The "Pipeline" directive is applied to the top level function "Predict" to allow pipelining of all instructions and sub function existing inside. The last directive which is "AXI Lite" is very important since it will help interconnecting the IP core designed in Vivado HLS for an implementation in the PL with the Zynq PS. The "AXI Lite" directive is applied to the top function "predict" under "Resource" and to the input array along with the output variable under (Interface). Details about Vivado HLS can be found in (Xilix (2014c)).

Hardware Design Using IP Integrator in Vivado. The hardware block design when using Vivado IP Integrator is shown in figure 8. It is worth mentioning that when creating the block design in IP Integrator, the Zynq PS IP and the HLS IP are added manually while two extra IPs are automatically added to the design (Processor System Reset and AXI Interconnect) when running the block and connection automation. All interconnection between different blocks are also made. In total, four IPs are used. The first one is the "ZYNQ7 Processing System", it is for configuration purpose only and will not be implemented on the PL, it is actually corresponding to the fixed PS on the Zynq chip, anything related to the PS is to be specified in this IP. The second IP called "Predict" is the one designed, simulated and exported in HLS, it is corresponding to main blocks of the EN which are PCA and DT. The third IP named "AXI Interconnect" is used to interconnect an AXI memory mapped master device which the PS in our case with a memory mapped slave device which is the AXI-Lite compatible HLS core called "Predict" in this solution. The last IP which is the "Processor System reset" provides a customized resets for the entire system including the PS, the AXI interconnect core and the Predict core from HLS. The block design can be seen in figure 8. Details about Vivado IP Integrator can be found in (Xilix (2014a)).

Software Design Using SDK. SDK is used for software development, an application to get the basic setting and initialization of the platform including the Universal Asynchronous Receiver/Transmitter (UART) to print results in the terminal is created. The C source code is then modified to read/write data from/to the HLS core implemented on the PL. The communication with the hardware present in the PL is performed by calling some read and write data functions that exist in driver files which were automatically created and exported for various Operating Systems (OS) including Linux and the lightweight Standalone OS. Details about SDK can be found in (Xilix (2014b)).

5. Implementation Results and Analysis

5.1. HLS Results and Analysis

The implementation of the EN on the Zynq SoC is performed and evaluated for the best four scenarios corresponding to the simulation done in MATLAB. The first one is done when the EN is based on the 4x4 sensor array and the input is the SSs, in this case the best solution with the highest accuracy of 90% is when the dimensionality reduction is applied and PCA 1, 2 and 3 are used as predictors. Results of the implementation in terms of resources used, clock cycle and latency are shown in table 8. The second scenario is when the EN is also based on the same 4×4 sensor array. However, the input of the system is the Delta between the SSs and the baseline. In this case the best solution with an accuracy of 96.6% is when the dimensionality reduction is applied and PCA 1, 2 and 4 are used as predictors. The results of this implementation are shown in table 9. The third scenario is when the EN is based on the 7 Figaro sensors and the input is the SSs. The best 100% accuracy is obtained when dimensionality reduction is applied and PCA 1 and 2 are used as predictors. The results of this implementation are shown in

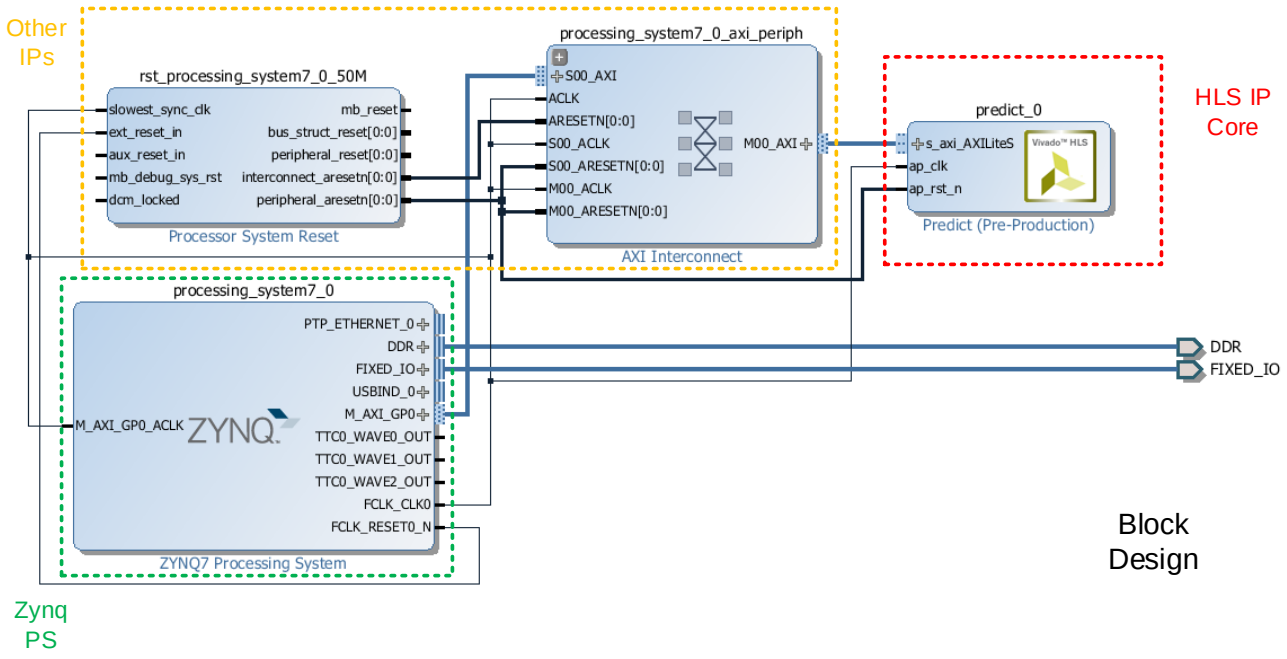

Fig. 8. Block design implementation on the chip

table 10. The fourth and last scenario is when the EN is based on the 7 Figaro sensors and the input is the Delta between the SSs and the baseline. The best solution with an accuracy of 100% is obtained in all cases, with and without dimensionality reduction. The solution without dimensionality reduction is chosen because the generated decision tree uses the output of one sensor only as a predictor and ignores the others which reduce the computation. The results of this implementation are shown in table 11.

Table 8. Hardware usage and performance results for EN based on 4×4 sensor array using steady states (without drift compensation) and PCA 1, 2 and 3

Hardware resources and performance	Optimization directives			
	Without Directives	Unroll Loops	Array Partitioning and Pipelining	AXI Lite Interface
BRAM 18K	0	0	0	8
DSP48E	15	15	136	68
FF	2089	1989	14730	9803
LUT	3925	4830	27208	14676
Max frequency (MHz)	142	142	142	142
Latency (clock cycles)	261	95	94	96

As it can bee seen from tables 8, 9, 10 and 11 a step-by-step optimization strategy has been used. Each table contains four columns where the first one is to show the implementation results when the defaults HLS settings are used and none of the optimization directives is applied. The second one shows the obtained results when the loops are unrolled for a better latency. It is worth mentioning that in the scenario presented in table 11 the corresponding C code does not have any loop, this is why the unroll directive is not applicable. The third column shows the obtained results when the input array corresponding

Table 9. Hardware usage and performance results for EN based on 4×4 sensor array using the Delta for drift compensation and PCA 1, 2 and 4

Hardware resources and performance	Optimization directives			
	Without Directives	Unroll Loops	Array Partitioning and Pipelining	AXI Lite Interface
BRAM 18K	0	0	0	8
DSP48E	15	5	136	68
FF	1793	1433	14729	9804
LUT	3314	2980	27075	14670
Max frequency (MHz)	142	142	142	142
Latency (clock cycles)	261	179	94	95

to the sensors data is partitioned and the entire code is pipelined when possible. The last column is to show the final results when the AXI lite encapsulation is used which will make the interconnection of the HLS designed IP with the PS possible via AXI-Interconnect. The gradual improvement in terms of latency and interval can be easily seen. It is worth mentioning as well that most of LUTs, flip-flops and DSP48E are used to create the instances of the proposed architecture corresponding to the C code including adders, multipliers and AXI interfaces. The other few LUTs and flip-flops are used for multiplexers, expressions or registers. Most of the resources are used for the implementation of PCA and the remaining few for DT. HLS converts the C Code to a successions of control and operation steps where each of them take a given number of clock cycles.

5.2. Hardware Implementation Analysis

Figure 9 shows the chip layouts of the implemented designs on the Zynq SoC for the four best EN solutions using HW/SW co-design approach with a single ARM processor running at 667 MHz and the programmable logic running at 142 MHz.

Table 10. Hardware usage and performance results for EN based on Figaro sensors using steady states (without drift compensation) and PCA 1 and 2

Hardware resources and performance	Optimization directives			
	Without Directives	Unroll Loops	Array Partitioning and Pipelining	AXI Lite Interface
BRAM 18K	0	0	0	0
DSP48E	10	10	84	84
FF	1640	1533	8387	8721
LUT	3042	3327	14764	15316
Max frequency (MHz)	142	142	142	142
Latency (clock cycles)	117	50	46	46

Table 11. Hardware usage and performance results for EN based on Figaro sensors using the Delta for drift compensation without PCA

Hardware resources and performance	Optimization directives			
	Without Directives	Unroll Loops	Array Partitioning and Pipelining	AXI Lite Interface
BRAM 18K	0	N/A	0	0
DSP48E	0	N/A	0	0
FF	303	N/A	560	666
LUT	764	N/A	1637	1805
Max frequency (MHz)	142	N/A	142	142
Latency (clock cycles)	4	N/A	2	2

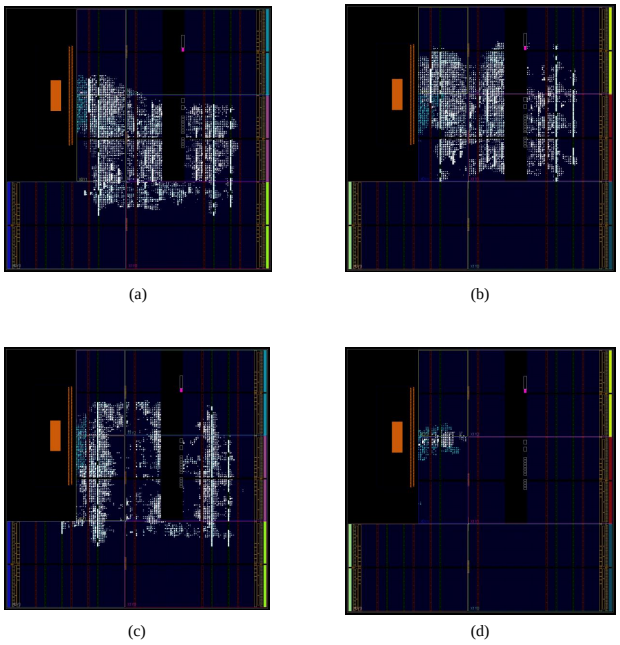

Fig. 9. Chip layouts. (a) 4×4 **In-house sensor (Steady states). (b)** 4×4 **In-house Sensor (Delta). (c) 7 Figaro sensor (Steady states). (d) 7 Figaro sensor (Delta)**

Table 12. Summary of best EN solutions implemented on the Zynq SoC using a single ARM processor running at 667 MHz and the PL running at 142MhZ

EN Characteristics	Type of sensor array			
	In-house 4x4		7 Figaro	
System input	Steady states	Delta	Steady States	Delta
PCA	Yes	Yes	Yes	No
Feature for-	PCA 1,-	PCA 1,-	PCA 1-	7 steady-
-classification	-2 and 3	-2 and 4	-and 2	-states
Accuracy	90%	96.66%	100%	100%
Latency (clock cycles)	96	95	46	2
Execution time (μs)	3.46	3.85	1.81	0.55

The presented solutions are summarized in table 12 where the mentioned execution time exclude the time spent by the IP core to access the memory to read and write data. It is worth mentioning that figure 9 illustrates the full implemented designs that include the IP core designed in HLS which takes the biggest part of the resources used as shown in white as well as the other two small IPs (Processor system reset and AXI interconnect) shown in blue. It can be seen that in the case of the 4×4 sensor similar amount of resources are used in both solutions (with and without drift compensation), using drift compensation improved the identification accuracy from 90% to 96.66%. However, this will involve periodic calibration to be able to update the baselines for each sensor used for the computation of the deltas which might be required in critical applications where accuracy specification is higher. In the case of the Figaro sensors the big difference in the amount of resources used between both solutions (with and without drift compensation) is explained by the fact that PCA is not applied when drift compensation technique is used. This is due to the DT alone being able to provide the similar 100% accuracy. However, this is at the cost of periodic calibration as well. It can also be seen in table 12 that the set of Figaro sensors outperform the in-house ones in terms of accuracy and execution time. Better accuracy is explained by the good response to target gases for Figaro keeping in mind that the in-house one consumes less power, is smaller in size and is made to be integrated into an RFID tag. Better execution time is explained with the fact that in the case of Figaro only 7 sensors are used while it is 16 in the case of the in-house one. In addition, in the last scenario PCA is not used at all. Therefore, the projection is not performed which saves time.

5.3. Power Consumption and Analysis

There are two types of on-chip power consumption which are the device static and the dynamic. The device static power consumption is related to transistor leakage dissipated on-chip from sources of voltage when the device is switched off with a power supply connected to it or when the device is switched on and not being used, it is device dependent and not related to the implemented design, it varies with process, voltage and junction temperature. The dynamic on-chip power consumption is related to the implemented design and it is consumed by fluctuating power as the implemented design runs, which includes PS power, clocks power, signals power, logic power, BRAM power and DSP power. Details about dynamic on-chip power consumption for the implemented EN systems are shown in figure 10. It can be seen in figure 10 that the PS consumes much more power than the PL, this is due to the fact that the ARM dual core Cortex-A9 based PS has much higher running frequency than the PL and it runs drivers and control programs. It can also be seen in figure 10 that within the PL, the HLS design is the portion that consumes most of the power in all scenarios except in the case of Figaro sensors where PCA is not used and where the design corresponds only to the decision tree model.

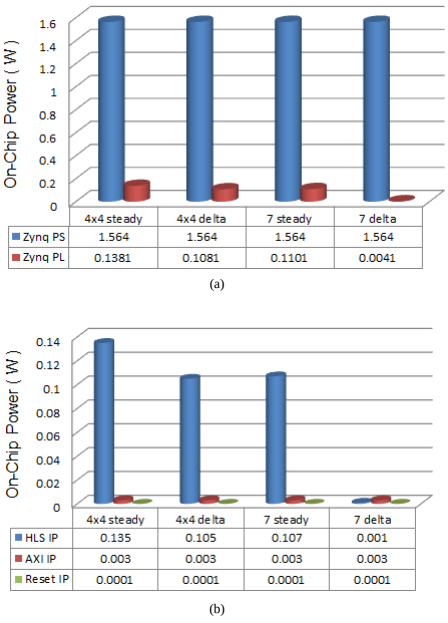

Fig. 10. Dynamic power consumption for the EN based on both 4×4 sensor array and 7 Figaro sensors. (a) Zynq PS/PL power consumption. (b) Power consumption within the Zynq PL

6. Conclusion

In this paper, an EN system based on PCA as a dimensionality reduction and DT as a classifier is presented. A comparative study is performed for the deployment of an In-house fabricated 4×4 sensor array over 7 commercial Figaro sensors. Results from the comparison have shown how modifying the type of sensor can have a positive impact on the performances of the EN in terms of accuracy and execution time. Simulations and training are performed in MATLAB environment while the hardware implementation is realized on the heterogeneous Zynq SoC platform. A HW/SW co-design approach using high level synthesis is chosen over a conventional implementation on FPGA using HDL or on a processor. Different scenarios are evaluated in MATLAB. However, only the best solutions that provide the highest accuracy in term of classification are implemented on the Zynq SoC. The interconnection between the two parts of the Zynq platform are realized via an AXI4-Lite interface. Results from the hardware implementation on the Zynq SoC show that real-time performances can be achieved for proposed EN systems using hardware/software co-design approach with a single ARM processor running at 667 MHz and the programmable logic running at 142 MHz. In Addition, the PL is consuming much less power compared to the PS. Ongoing research is focusing on the design and implementation of new classifier using different PS-PL interfaces.

Acknowledgment

This paper was made possible by National Priorities Research Program (NPRP) grant No. 5 - 080 - 2 - 028 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Ait Si Ali, A., Amira, A., Bensaali, F., Benammar, M., 2013. Hardware pca for gas identification systems using high level synthesis on the zynq soc, in: Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on, IEEE. pp. 707–710.

Ait Si Ali, A., Amira, A., Bensaali, F., Benammar, M., Akbar, M.A., Hassan, M., Bermak, A., 2015. Design and implementation of a gas identification system on zynq soc platform. Sensors & Transducers 10, 9758–9764.

Akbar, M.A., Zgaren, M., Ait Si Ali, A., Amira, A., Benammar, M., Bensaali, F., Sawan, M., Bermak, A., 2015. Gas identification using passive uhf rfid sensor platform. Sensors & Transducers 194, 42–53.

Arshak, K., Moore, E., Lyons, G., Harris, J., Clifford, S., 2004. A review of gas sensors employed in electronic nose applications. Sensor Review 24, 181–198.

Benrekia, F., Attari, M., Bouhedda, M., 2013. Gas sensors characterization and multilayer perceptron (mlp) hardware implementation for gas identification using a field programmable gate array (fpga). Sensors 13, 2967–2985.

Bermak, A., Belhouari, S.B., Shi, M., Martinez, D., et al., 2006. Pattern recognition techniques for odor discrimination in gas sensor array. Encyclopedia of Sensors 10, 1–17.

Brahim Belhouari, S., Bermak, A., Wei, G., Chan, P., 2004. Gas identification algorithms for microelectronic gas sensor, in: Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE, IEEE. pp. 584–587.

Bravo, I., Mazo, M., Lázaro, J.L., Gardel, A., Jiménez, P., Pizarro, D., 2010. An intelligent architecture based on field programmable gate arrays designed to detect moving objects by using principal component analysis. Sensors 10, 9232–9251.

Chandrasiri, N.P., Park, M.C., Naemura, T., Harashima, H., 1999. Personal facial expression space based on multidimensional scaling for the recognition improvement, in: Signal Processing and Its Applications, 1999. ISSPA'99. Proceedings of the Fifth International Symposium on, IEEE. pp. 943–946.

Far, A.B., Flitti, F., Guo, B., Bermak, A., 2009. A bio-inspired pattern recognition system for tin-oxide gas sensor applications. Sensors Journal, IEEE 9, 713–722.

Guo, B., Bermak, A., Chan, P.C., Yan, G.Z., 2007. An integrated surface micromachined convex microhotplate structure for tin oxide gas sensor array. Sensors Journal, IEEE 7, 1720–1726.

Gutierrez-Osuna, R., 2002. Pattern analysis for machine olfaction: a review. Sensors Journal, IEEE 2, 189–202.

Honeine, P., 2012. Online kernel principal component analysis: A reduced-order model. Pattern Analysis and Machine Intelligence, IEEE Transactions on 34, 1814–1826.

Hong, H.K., Kwon, C.H., Kim, S.R., Yun, D.H., Lee, K., Sung, Y.K., 2000. Portable electronic nose system with gas sensor array and artificial neural network. Sensors and Actuators B: Chemical 66, 49–52.

Kharaka, Y.K., Thordsen, J.J., Hovorka, S.D., Nance, H.S., Cole, D.R., Phelps, T.J., Knauss, K.G., 2009. Potential environmental issues of co 2 storage in deep saline aquifers: geochemical results from the frio-i brine pilot test, texas, usa. Applied Geochemistry 24, 1106–1112.

Kim, E., Lee, S., Kim, J.H., Kim, C., Byun, Y.T., Kim, H.S., Lee, T., 2012. Pattern recognition for selective odor detection with gas sensor arrays. Sensors 12, 16262–16273.

Li, Q., Bermak, A., 2011. A low-power hardware-friendly binary decision tree classifier for gas identification. Journal of Low Power Electronics and Applications 1, 45–58.

Li, S.Z., Lu, X., Hou, X.W., Peng, X., Cheng, Q., 2005. Learning multiview face subspaces and facial pose estimation using independent component analysis. Image Processing, IEEE Transactions on 14, 705–712.

Marco, S., Gutiérrez-Gálvez, A., 2012. Signal and data processing for machine olfaction and chemical sensing: a review. Sensors Journal, IEEE 12, 3189–3214.

Pearce, T.C., Schiffman, S.S., Nagle, H.T., Gardner, J.W., 2006. Handbook of machine olfaction: electronic nose technology. John Wiley & Sons.

Perera, D.G., Li, K.F., 2011. Fpga-based reconfigurable hardware for compute intensive data mining applications, in: P2P, Parallel, Grid, Cloud and In-

ternet Computing (3PGCIC), 2011 International Conference on, IEEE. pp. 100–108.

Schaller, E., Bosset, J.O., Escher, F., 1998. electronic noses and their application to food. LWT-Food Science and Technology 31, 305–316.

Sensors, F., 2015. Figaro Sensors. http://www.figarosensor.com/. [Online; accessed 30-September-2016].

Shi, M., Bermak, A., Chandrasekaran, S., Amira, A., Brahim-Belhouari, S., 2008. A committee machine gas identification system based on dynamically reconfigurable fpga. Sensors Journal, IEEE 8, 403–414.

Xilix, I., 2014a. Vivado design suite user guide: Designing ip subsystems using ip integrator. ug994, v2014.4. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_3/ug994-vivado-ip-subsystems.pdf. [Online; accessed 30-September-2016].

Xilix, I., 2014b. Vivado Design Suite User Guide: Embedded Processor Hardware Design. UG898, v2014.4. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_3/ug898-vivado-embedded-design.pdf. [Online; accessed 30-September-2016].

Xilix, I., 2014c. Vivado design suite user guide: High-level synthesis. ug902, v2014.4. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_3/ug902-vivado-high-level-synthesis.pdf. [Online; accessed 30-September-2016].

Xilix, I., 2014d. Zynq-7000 all programmable soc: Technical reference manual. ug585, v1.8.1. http://www.xilinx.com/support/documentation/userguides/ug585-Zynq-7000-TRM.pdf. [Online; accessed 30-September-2016].

Yamazoe, N., Miura, N., 1994. Environmental gas sensing. Sensors and Actuators B: Chemical 20, 95–102.

Zampolli, S., Elmi, I., Ahmed, F., Passini, M., Cardinali, G., Nicoletti, S., Dori, L., 2004. An electronic nose based on solid state sensor arrays for low-cost indoor air quality monitoring applications. Sensors and Actuators B: Chemical 101, 39–46.