J. Low Power Electron. Appl. **2011**, *1*, 45-58; doi:10.3390/jlpea1010045

OPEN ACCESS

Journal of Low Power Electronics and Applications
ISSN 2079-9268
www.mdpi.com/journal/jlpea

Article

A Low-Power Hardware-Friendly Binary Decision Tree Classifier for Gas Identification

Qingzheng Li and Amine Bermak *

Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST), Clear Water Bay, Kowloon, Hong Kong, China; E-Mail: liqingzh@ust.hk

* Author to whom correspondence should be addressed; E-Mail: eebermak@ust.hk; Tel.: +852-2358-8992.

Received: 13 December 2010; in revised form: 1 March 2011 / Accepted: 2 March 2011 / Published: 9 March 2011

Abstract: In this paper, we present a hardware friendly binary decision tree (DT) classifier for gas identification. The DT classifier is based on an axis-parallel decision tree implemented as threshold networks—one layer of threshold logic units (TLUs) followed by a programmable binary tree implemented using combinational logic circuits. The proposed DT classifier circuit removes the need for multiplication operation enabling up to 80% savings in terms of silicon area and power compared to oblique based-DT while achieving 91.36% classification accuracy without throughput degradation. The circuit was designed in 0.18 μm Charter CMOS process and tested using a data set acquired with in-house fabricated tin-oxide gas sensors.

Keywords: binary decision tree; classifier; gas identification; hardware implementation

1. Introduction

The last decade has witnessed an increasing interest in gas identification based on gas sensor arrays and pattern analysis for machine olfaction applications [1]. The multivariate response of an integrated gas sensor array can be used as a signature to detect and identify a wide range of odors. Conventional gas identification processing would typically involve signal pre-processing, feature extraction (dimensionality reduction) and classification [1]. Such processing is not well geared towards low cost compact hardware implementation because of the relative complexity involved in feature extraction methods such as principal component analysis (PCA) and linear discriminate analysis (LDA) and pattern recognition algorithms such as artificial neural networks (ANN), K nearest neighbor (KNN) and support vector machines (SVM) [2]. Furthermore, various classification algorithms such as ANN and SVM require large amounts of memory to store various coefficients and complex non-linear computations. Efficient hardware realization of gas identification algorithms would enable the integration of processing circuitry together with the sensor array. The fully integrated electronic-nose-on-chip has the potential to offer significant advantages in terms of reduced power consumption, lower manufacturing cost and increased portability for hand-held applications. A decision tree (DT) [3] based classifier offers a number of advantages such as smaller number of coefficients for storage and linear computation, making it a viable candidate for on-chip integration . Previously reported hardware implementations of DTs include: not only FPGA implementations [4] but also silicon integrations [5,6]. Only the two latter works target gas identification. Both implemented binary oblique DTs using a combination of custom VLSI chip and complex programmable logic device (CPLD) chip. This paper presents a compact single-chip DT classifier based on an axis-parallel DT architecture, which completely removes the need for multiplication operations. To achieve high throughput, the DT classifier is implemented as a layer of threshold logic units followed by a layer of combinational logic units [6]. Compared with prior works including non-DT hardware implementations [7,8], this approach provides a good trade-off between implementation complexity and classification accuracy.

The paper is organized as follows. Section 2 introduces binary decision tree classifiers and examines prior hardware implementations. Section 3 describes the proposed axis-parallel DT architecture and silicon implementation. Experimental results and performance evaluation are discussed in Section 4. Finally, a conclusion is given in Section 5.

2. Binary Decision Tree Classifiers

2.1. Overview

Decision tree is a classification model where the target function is represented by a sequence of queries based on the attributes of input test patterns. The sequence of queries is reflected in a tree structure with finite depth. The classification of a particular pattern begins at the root decision node and terminates in the leaf node where a single class label is specified as the classification result. Each decision node contains different tests involving one or more attributes of the test patterns. Depending on the output of the decision node, only one branch of the decision node is selected and the child (descendent) node is visited. This process is repeated until a leaf is reached.

Figure 1(a) depicts a binary decision tree diagram with 4 decision nodes A, B, C, D and three classes $F1$, $F2$, $F3$. In a given node, a function can be represented in the following general format:

$$f(A_1, A_2, A_3, ..., A_n) = 0 \tag{1}$$

where f is a function of n attributes (A_1, A_2, A_3,...,A_n). Depending on the value of function f, either the left or right branch of the node is activated. If only one attribute A_i is considered in one node, we can simplify the function as

$$f = A_i + w_i \tag{2}$$

where w_i is the threshold value of the attribute A_i. This function corresponds to the case of axis-parallel DTs (see e.g., ID3 [9] and C4.5 [10] or J48 in Weka [11]). On the other hand, oblique DTs (OC1 [12]) are represented by the following function

$$f = \sum_{i=1}^{n} w_i \times A_i + w_{n+1} \tag{3}$$

where w_i represents the weight value of the attribute of A_i and w_{n+1} is the threshold value of the function f.

While quadratic polynomial nonlinear DTs correspond to the function

$$\begin{aligned} f_{2nd} &= (\sum_{i=1}^{n} w_i \times A_i + w_{n+1}) \times (\sum_{j=1}^{n} w_j^{'} \times A_j + w_{n+1}^{'}) \\ &= \sum_{i=1}^{n}\sum_{j=i}^{n} W_{i,j} \times A_i \times A_j + \sum_{i=1}^{n} W_i \times A_i + W \end{aligned} \tag{4}$$

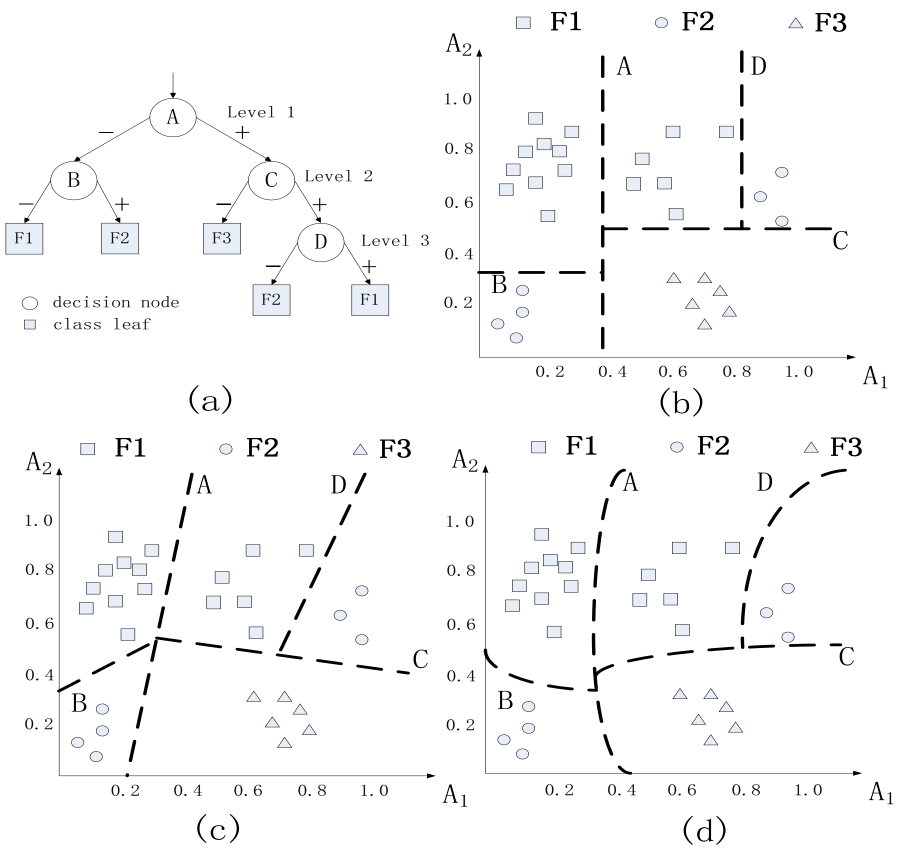
where either A_i or A_j refers to a single attribute and the new weight $W_{i,j} = w_i \times w_j^{'} + w_i^{'} \times w_j$; $W_i = w_{n+1} \times w_i + w_{n+1}^{'} \times w_{n+1}$ and the new threshold $W = w_{n+1} \times w_{n+1}^{'}$.

Figure 1 depicts the three commonly used decision tree classification methods for the three-category problem. The two coordinate axes represent the two attributes A_1 and A_2 of test patterns. The decision boundaries of different decision tree methods have their own properties. For axis-parallel DTs, partition lines are parallel with the attribute axes. For oblique DTs, partition lines are oblique straight lines and the decision boundaries are curves for nonlinear DTs.

The three aforementioned DT classifiers are actually intimately related. For example, Equation (4) can be embedded into Equation (3) by introducing a new set of attributes corresponding to the products $A_i^{m_i} \times A_j^{m_j}$, with $m_i < 2$ and $m_j < 2$. Nonlinear DTs with polynomial hypersurfaces can be regarded as oblique DTs in the higher-dimension attribute space. In addition, axis-parallel DT can be treated as a special oblique tree where the weights for all attributes are zeros except for a single attribute whose weight is 1.

Nonlinear DTs are more complex than the other two types of DTs. Oblique DTs are normally much smaller than axis-parallel DTs due to their flexibility at each decision node. However, finding the best oblique DT is an NP-complete problem [12].

Figure 1. (**a**) DT diagram of depth 3; (**b**) Axis-parallel DT classification method; (**c**) Oblique DT classification method; (**d**) Nonlinear DT classification method.



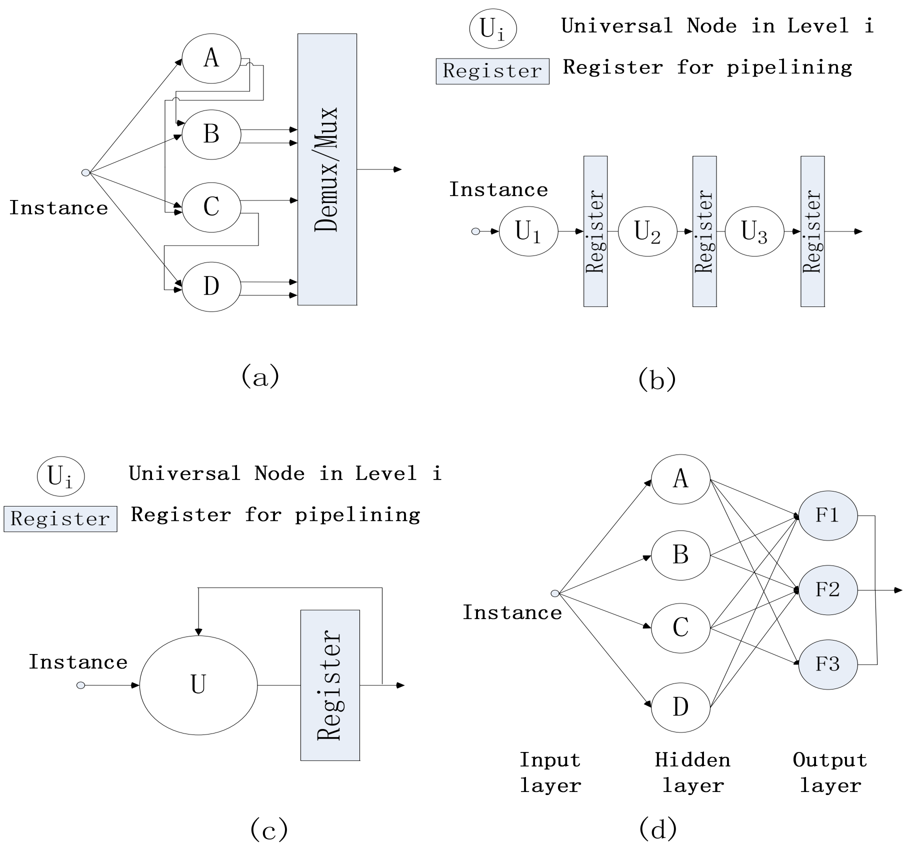
2.2. *Hardware Considerations*

A straightforward approach to the hardware implementation of DTs would be to implement every DT node as a separate module as proposed in [13]. The idea is illustrated [Figure 2(a)] for the case of the DT diagram of Figure 1(a). The disadvantage of this approach is large latency, due to the fact that a new test pattern cannot be fed into the module before the classification output has propagated. To overcome the large latency problem, a universal node DT architecture [Figure 2(b)], using registers to pipeline each stage, was proposed in [14]. Although it reduces hardware cost by folding each level, it exhibits a relatively low throughput as a result of serial processing [13,14].

A second approach [6] to the hardware implementation of DTs is based on the equivalence between DTs and threshold networks [15]. It exhibits a relatively high throughput since the signals need only to propagate through the threshold unit and the combinational logic layers. In the threshold network architecture [Figure 2(d)], each decision node is implemented as a processing element in the threshold unit layer. Each element of the output layer is adopted to obtain the class-belonging according to the Boolean function. Obviously, this architecture requires more hardware resources to process the decision node, in parallel, and achieves higher classification speed.

In this paper, we propose to explore axis-parallel DT based classifiers, to remove the need for multiplication operations and enable a low cost compact implementation with a classification accuracy comparable to prior works [6]. OC1 tool is used to find the possibly optimal weight values by using a randomized algorithm with simulated annealing or heuristics methods [12].

Figure 2. Reported DT hardware implementations with: (**a**) representing a decision tree implemented on FPGA; (**b**) and (**c**) representing a decision tree implemented on an IP core; (**d**) representing a network ensemble of decision trees implemented using a 3D chip.

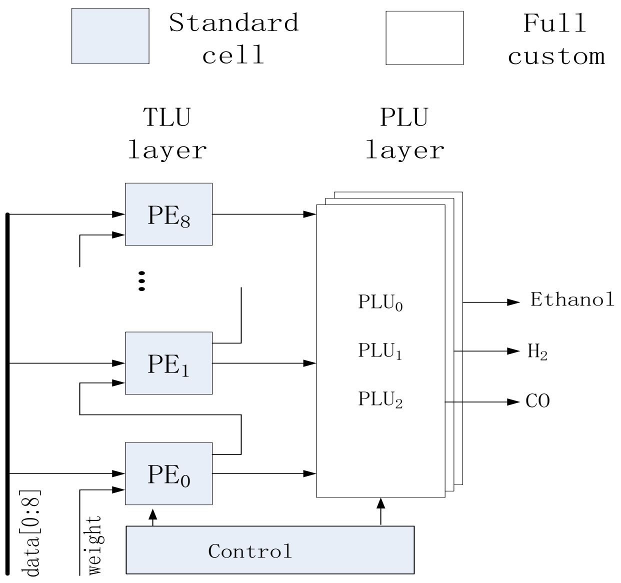


3. Proposed DT Classifier Implementation

The proposed DT classifier illustrated in Figure 3 comprises two types of building blocks—threshold logic units and programmable logic units. In Figure 3, each of the processing elements (PE) represents a threshold logic unit (TLU). The programmable logic unit (PLU) is implemented using a compact tree-based logic circuit. The control unit generates the synchronous control signals for both processing elements and programmable logic units.

There are 9 processing elements in the TLU layer and 3 programmable logic blocks using the tree structure. Each processing element is here a pipelined digital comparator (subtracter). Each programmable logic is configured for a given gas identification task. To minimize the area cost and avoid input and output port limitation, we chose a bit-serial architecture to process elements. The digital weight precision is important for classification performance. Research works in [6] suggests that 10 bit precision achieves the required accuracy for data quantization. In our design, the control block generates the data control signals and synchronizes the two layer building blocks. For one operation, 10-bit data are serially fed into the first layer and the valid signals are transported to the following layer once every 11 clock cycles.

Figure 3. Two layer neural network architecture to implement the axis-parallel DT classifier. The first layer is the TLU layer to implement the decision node function. The second layer is the PLU layer to implement the logic function for each class.



3.1. *Threshold Logic Unit*

For an oblique DT architecture, each node requires to realize the function given by Equation (3). The sum of the weighted feature is compared with a stored threshold value w_{n+1} to make the decision on the node. With the increase of the number of attributes n, hardware resources increase linearly. By using pipelining and folding techniques, hardware resources can be minimized but throughput decreases linearly.

In the proposed axis-parallel DT architecture (Figure 4), each node generates the function given by Equation (2). The latter only requires a single digital comparator in place of the multiplication and accumulation (MAC) circuits. The processing element associated to each DT node is thus simplified significantly.

However due to the same classification problem, the built tree's node number and tree's depth are typically much larger than that of the oblique DT structure.

3.2. *Programmable Logic Unit*

In prior works [5], a full custom chip was designed to implement the TLU layer for the oblique decision tree algorithm while AND-OR array-based combinational logic in CPLD was used to implement the output layer of the classifier. In this paper, we propose to design a tree-structure combinational logic circuit to replace the AND-OR array-based combinational logic circuit.

As shown in Figure 5, our proposed circuit for tree-based programmable logic circuit includes four parts: (i) a dynamic current mode tree-based decoder, (ii) two lines of transistor switches to enable programmability, (iii) a differential dynamic comparator to amplify the low-swing signal and (iv) latches in the input and output stages to synchronize the data with the TLU layer.

Figure 4. The processing element of the axis-parallel decision tree. The processing element includes 10-bit register to store the coefficients, 3 1-bit registers for pipelining, one 1-bit adder with an invertor for the substraction operation and one tri-state buffer for controlling the output data. S_{in} and S_{out} represent the input and output ports of the serial 10-bit shift register.

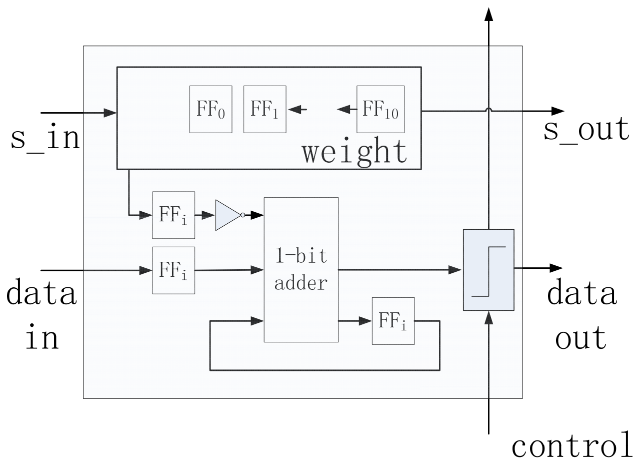


Figure 5. Overview of the proposed tree-based programmable logic circuit.

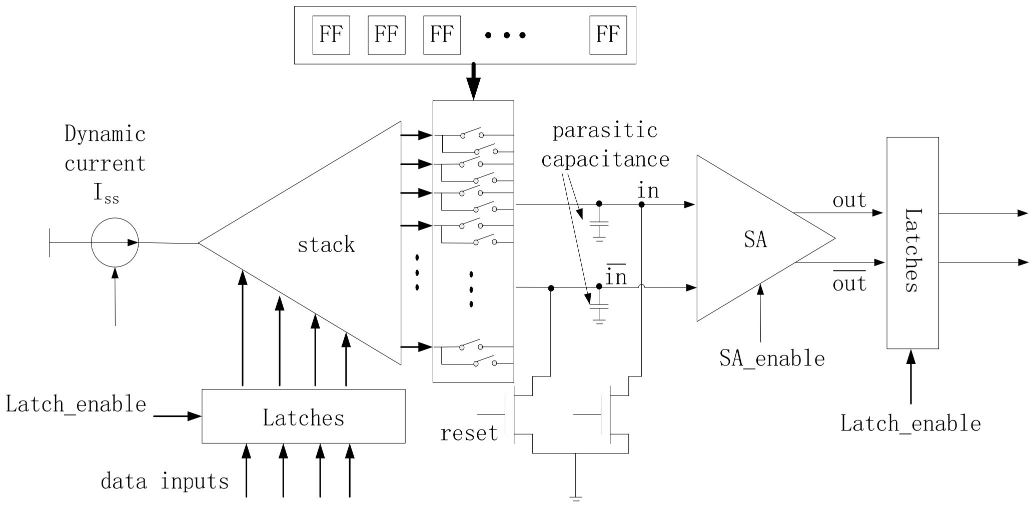
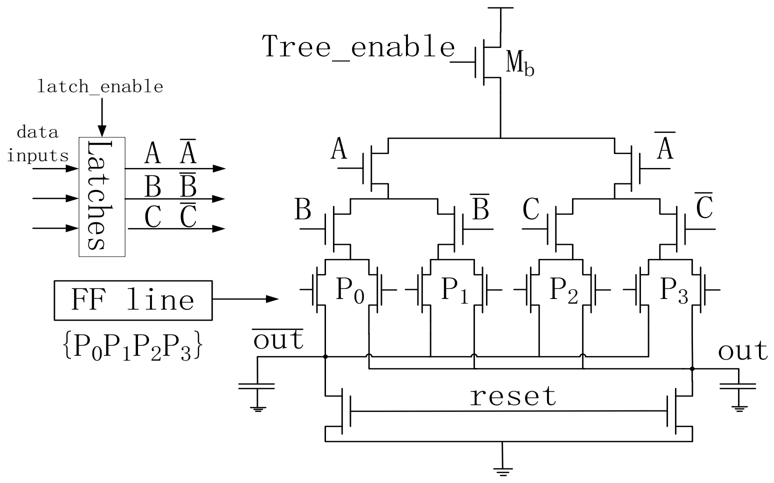


Figure 6 shows the stacked transistor tree circuit. This tree-based structure is the same as the DT diagram whereby only one branch will be turned on from the root to the leaf. There are two lines of switches that group all the stacked paths to differential outputs similarly to current based logic circuits [16]. To avoid static current consumption, the current source is controlled and the resistive load [16] is replaced by the parasitic capacitance. The operational principle is as follows. Firstly, the two lines of switches are configured according to the values stored in the registers. The stable input data come from the input latches. When *reset* is high, the differential outputs of the stacked transistor are pulled down to 0 (low). When *reset* is low and *tree_enable* is high, one of the differential outputs is conditionally charged.

Figure 6. The stacked transistor tree circuit. The FF line represents a line of registers to control the P_0, P_1, P_2, P_3 values and determine the differential outputs (*out* and $\bar{out}$) values.



The comparator used in our proposed circuit is illustrated in Figure 7. Because the accuracy of the comparator plays a critical role in the proposed circuit, transistors M_{s1} and M_{s2} are included to avoid hysteresis or delayed response in the resetting phase. The operational principle is as follows. When the clock is high, the comparator is operated in the resetting mode and both outputs (*out* and $\bar{out}$) are pulled to 0 (low) and the bias transistor M_b is turned off to avoid the static current. On the other hand, when the clock is low, the circuit will execute the comparison of differential inputs. The outputs of the comparator will be buffered and latched to the next stage.

Figure 7. The dynamic comparator circuit with no static current consumption. Two operation phases: (i) the reset phase; (ii) the comparison phase.

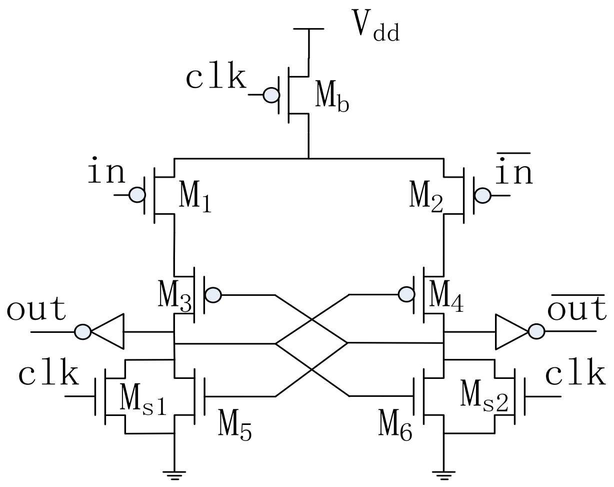
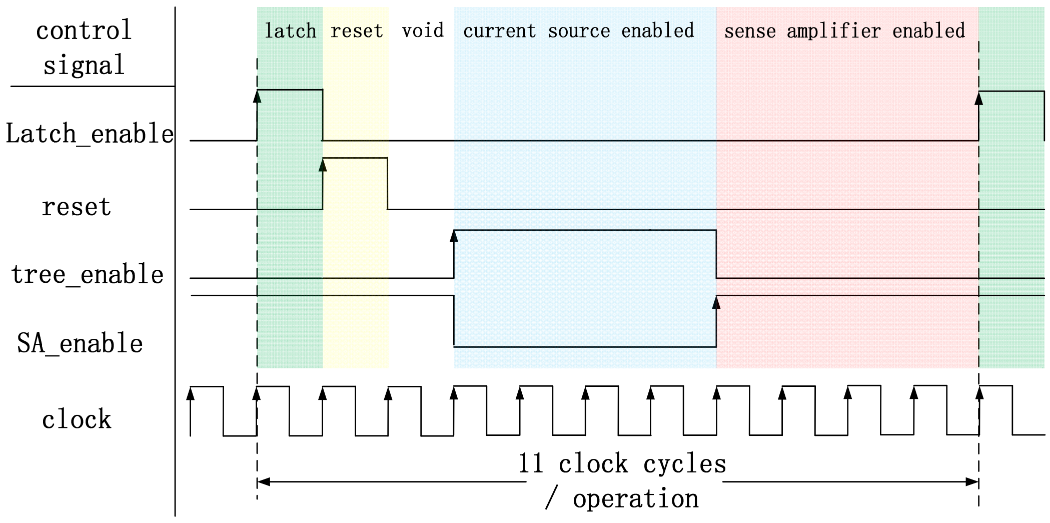


Figure 8 shows the timing waveform of our proposed circuit. The circuit is synchronous with the front-end TLU, whose data is serially processed and the valid signals appearing once every 11 clock cycles. The latches are used between the TLU and the programmable circuit to maintain the signals. There are four control signals in our circuit, which are generated by a synchronous control block.

$Latch\_en$ signal is used to activate the latches when the output signals from TLU are valid. Next, the $reset$ signal turns on the two reset transistors to discharge both differential buses of stacked tree and make them equal. After that, the root transistor turns on when $Tree\_enable = 1$, the current flows from one specified path and charges one bus. Next, SA_enable activates the sense amplifier to amplify the low-swing signal. The latches in the output stage store the correct data for the next stage.

Figure 8. Timing diagram of the proposed programmable circuit with the four control signals $Latch\_enable$, $reset$, $tree\_enable$ and SA_enable.



4. Results

4.1. Experiment Setup

Figure 9 shows the experimental equipment used to acquire the data from our in-house fabricated tin-oxide gas sensor array. The gas delivery system includes four mass flow controllers (MFC), three of which are for the target gases and one for dry air. The gas sensor array was placed into a gas chamber, in which the gas concentration is adjusted by selecting the correct flow rate. Voltages across the sensors are measured and stored in the data acquisition board. This data was used to evaluate the performance of the proposed VLSI friendly DT classifier.

Figure 10 shows an example of measured voltage response across a gas sensor. A test gas with a fixed concentration is injected periodically and the injection phase is followed by a cleaning phase by injecting fresh dry air. After the cleaning phase, the signal will settle down at the baseline value which is the reference value before gas injection. When the test gas is injected into the chamber, the voltage across the sensor changes as a result of variations in its conductivity. After several minutes of gas injection, the sensor output is sampled for concentrations ranging from 20 ppm to 200 ppm. CO, H_2 and $Ethanol$ are chosen as the test gases. Figure 10 shows the response of a gas sensor array comprising 12 individual tin oxide gas sensors fabricated in our design house. The overlapping curves of different gas sensors illustrate the low selectivity and non-linearity of the gas sensor array response, which adds to the complexity of the gas classification problem.

Figure 9. Experiment setup for gas identification.

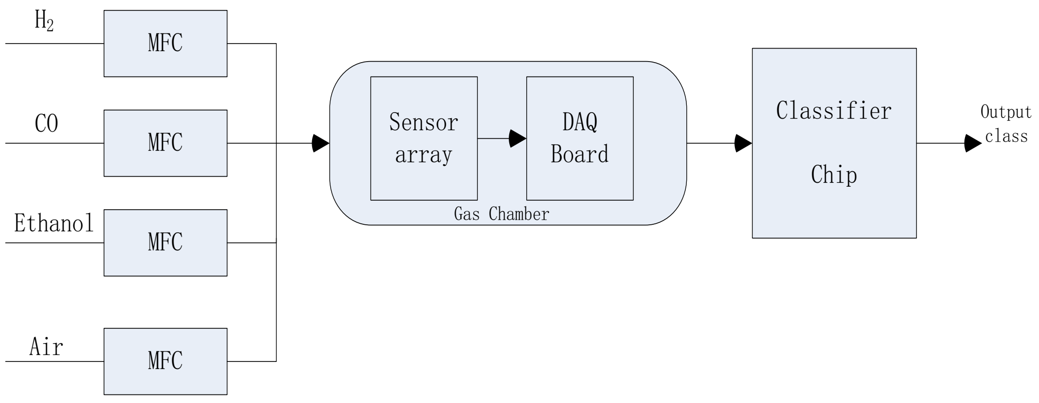
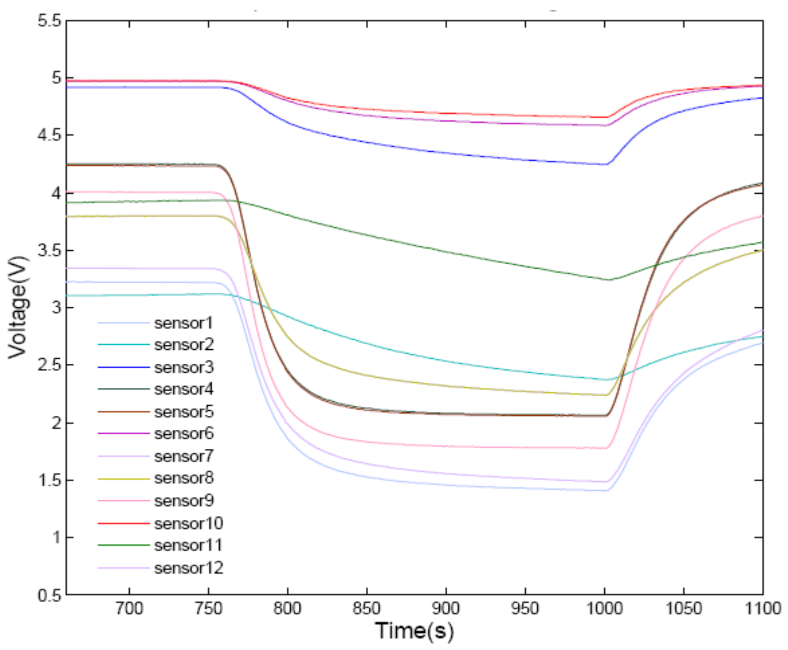


Figure 10. Example of response of the gas sensor array.



4.2. Classification Performance

Principal component analysis (PCA) is often used to pre-process the initial data and decrease the dimensions of the feature vector [8]. Figure 11 shows the PCA pre-processing of the gas sensor array multivariate response for the case with two principal components—PC_1 and PC_2. In this section, we compare the performance of the proposed binary decision tree classifier with and without PCA pre-processing. 12 gas sensors are selected to build the sensor test vector. The dimensions of test vectors are reduced to 2, 3 and 4 by using PCA program implemented in MATLAB. Both original and PCA pre-processed data were fed into the OC1 software tool to build the axis-parallel decision tree and the oblique decision tree for gas classification. Table 1 shows classification performance and hardware complexity using the axis-parallel decision tree and the oblique decision tree classifiers with different number of principal components (PC) and without PCA, respectively.

Figure 11. PCA pre-processing of the gas sensor array multivariate response.

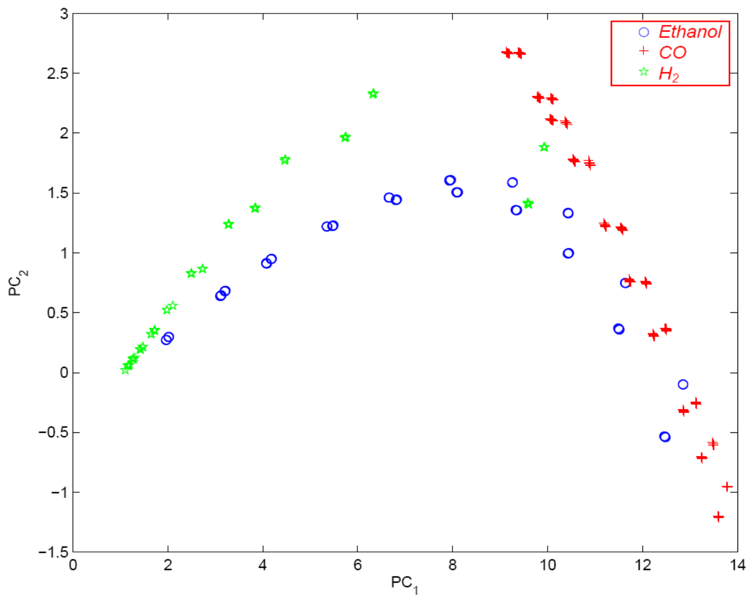


Table 1. Hardware cost and classification accuracy comparison for 3 gases (CO, H_2 and $Ethanol$) with a total of 600 data-sets for 10 different concentrations ranging from 20 ppm to 200 ppm.

PC No.	**DT Type**	**Accuracy**	**Leaves**	**Node**	**Mul**	**Add/Sub**	**Memory (No. of Coefficients)**
2	axis-parallel	88.33%	13	12	0	12	12
2	oblique	89.24%	9	8	16	16	24
3	axis-parallel	92.73%	10	9	0	9	9
3	oblique	90.30%	5	4	12	12	16
4	axis-parallel	99.55%	7	6	0	6	6
4	oblique	94.55%	3	2	8	8	10
no PCA	axis-parallel	91.36%	10	9	**0**	**9**	**9**
no PCA	oblique	92.58%	6	5	60	60	65

According to our simulation results, both decision tree based classifiers benefit from the PCA algorithm. In particular, oblique decision tree hardware complexity can be significantly decreased by reducing the number of attributes in the test vector by using the PCA. However, PCA requires more hardware resources such as matrix multiplications. For example, if we want to select m principal components from n attributes, the PCA computation includes $m \times n$ multiplications and $m \times (n-1)$ additions. From Table 1, one can note that with the increase of the number of principal components, the accuracy of the classifier increases. However, the requirements for hardware resources also increase. The axis-parallel decision tree classifier requires less computational units and memory bits compared to the oblique decision tree classifier. The trade-off between computational units and memory bits suggests that axis-parallel decision tree is a better choice for gas identification applications.

4.3. VLSI Implementation

The proposed classifier was designed using Charter 0.18 μm CMOS process. The TLUs and the control unit were modeled using verilog HDL and synthesized using Design Compiler. The generated netlist was fed into SOC Encounter and the layout was generated through auto placement and routing using the standard digital cell library. The tree-structure programmable unit was designed in full-custom using Cadence tools. Figure 12 shows the layout of the complete classifier.

Figure 12. Layout of the classifier and interface circuits. Note that the PLUs are full-custom designs whereas the TLUs are designed using standard cells.

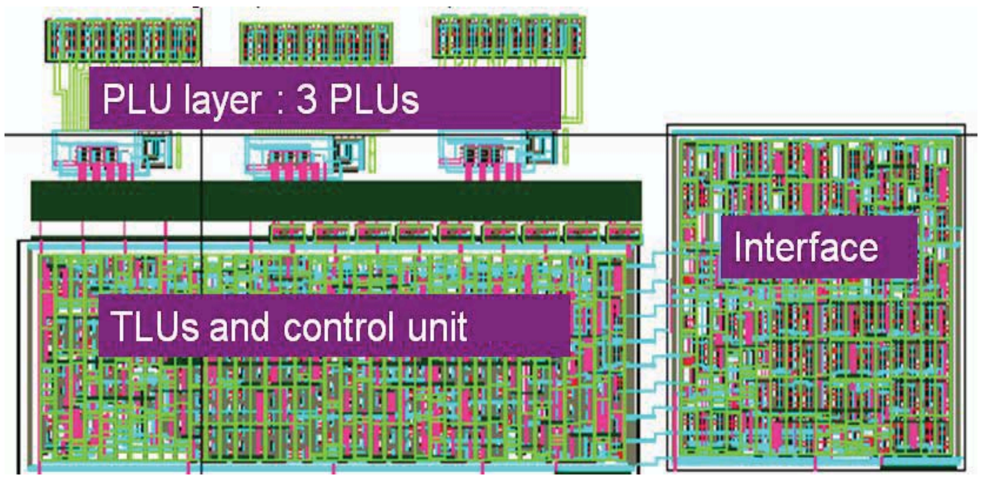


Table 2 compares the synthesized performance of oblique DT's Front-end TLU blocks together with our designed TLU blocks. Compared with prior art, this work significantly improves the hardware implementation cost in terms of power consumption, delay and silicon area as illustrated in Table 2.

Table 2. Performance of Front-end TLU blocks using Design Compiler(DC).

	DT Type	**Process**	**clk (ns)**	**Delay (ns)**	**Power (mW)**	**Area (10^3 μm^2)**
Reference [6]	oblique	0.18 μm	5	1.61	25	100
This work	axis-parallel	0.18 μm	5	**0.9**	**3.1**	**8.9**

Table 3 summarizes the overall performance in terms of hardware implementation cost and detection rate. To compare with previously reported FPGA implementation of gas classifiers [8], we also synthesized our classifier model using the same Xilinx Virtex II FPGA chip. According to our simulation results, our proposed axis-parallel DT classifier saves more than 90% FPGA resources compared with the committee machine method reported in [8]. It also saves up to almost 80% resources compared with the oblique DT classifier. Our proposed classifier only consumes 3.1 mW at a clock frequency of 100 MHz.

Table 3. Gas classification performance comparison.

Classification Method	**GMM** [7]	**Committee Machine** [8]	**Oblique DT** [6]	**This Work (Axis-Parallel DT)**
Sensor type	2×2 SnO_2 array	2×2 SnO_2 array	4×4 SnO_2 array	4×4 SnO_2 array
Target gas species	CH_4,CO,H_2, CO–CH_4, CO–H_2	CH_4,CO,H_2, CO–CH_4, CO–H_2	$Ethanol$, CO, H_2	$Ethanol$, CO, H_2
Detection rate	92%	94%	92.58%	91.36%
FPGA Implementation				
No. of Slice FF	N/A	12146	1176	**233**
No. of 4-LUT	N/A	20115	1269	**160**
ASIC Implementation				
process	0.25 μm	N/A	0.18 μm	0.18 μm
Area	1.69 mm^2	N/A	0.1 mm^2	**0.028** mm^2
Power	N/A	500 mW	25 mW	**3.1** mW

5. Conclusions

In this paper, a low-power single-chip binary decision tree based classifier is proposed for gas identification applications. The classifier circuit uses an axis-parallel architecture to remove the need for multiplication operations and enables a low cost compact implementation with a classification accuracy comparable to prior art. Without PCA pre-processing, the proposed classifier can still achieve 91.36% classification accuracy without throughput degradation while saving up to 80% silicon area and power consumption. The performance of the circuit can easily be enhanced by using Boosting or Bagging techniques [6], but this would result in larger silicon area since a larger number of processing nodes will be required. The classifier circuit was successfully validated using 0.18 μm Charter CMOS process and in-house fabricated tin-oxide gas sensor array.

Acknowledgment

This work was supported by a Research Grant from Hong Kong University of Science and Technology (Research Project Competition grant reference: RPC 10EG20).

References

1. Gutierrez-Osuna, R. Pattern analysis for machine olfaction: A review. *IEEE Sens. J.* **2002**, *2*, 189–202.
2. Belhouari, S.B.; Bermak, A. Gas Identification using density models. *Pattern Recognit. Lett.* **2005**, *26*, 699–706.

3. Beriman, L.; Freidman, J.H.; Olshen, R.A. *Classification and Regression Trees.* Wadsworth International Group: Belmont, CA, USA, 1984; pp. 203–215.
4. Struharik, R.J.R.; Novak, L.A. Evolving decision trees in hardware. *J. Circuits Syst. Comput.* **2009**, *18*, 1033–1060.
5. Bermak, A.; Martinez, D. A reconfigurable hardware implementation of tree classifier based on a custom chip and CPLD for gas sensors applications. In *Proceedings of the IEEE TENCON*, Chiang Mai, Thailand, October 2004; Volume 1, pp. 32–35.
6. Bermak, A.; Martinez, D. A Compact 3D VLSI Classifier using Threshold Network Ensembles. *IEEE Trans. Neural Networks* **2003**, *14*, 1097–1109.
7. Shi, M.; Bermak, A. An Efficient Digital VLSI Implementation of Gaussian Mixture Models-based classifier. *IEEE Trans. Very Large Scale Integr. Syst.* **2006**, *14*, 962–974.
8. Shi, M.; Bermak, A.; Belhouari, S.B.; Chan, P.C.H. Gas Identification based on a committee machine for Microelectronics gas sensors. *IEEE Trans. Instrum. Meas.* **2006**, *55*, 1786–1793.
9. Quinlan, J.R. Induction of Decision Trees. *Mach. Learn.* **1986**, *1*, 81–106.
10. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann Publishers: San Francisco, CA, USA, 1993.
11. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA Data Mining Software: An Update. *SIGKDD Explor.* **2009**, *11*, 10–18.
12. Murthy, S.K.; Kasif, S.; Azlzberg, S. Asystem for induction of oblique decision trees. *J. Artif. Intell. Res.* **1994**, *2*, 1–33.
13. Lopez-estrada, S.; Cumplido, R. Decision tree based FPGA architecture for texture sea state classification. In *Proceedings of IEEE Conference on Reconfigurable Computing and FPGAs*, San Luis Potosi, Mexico, September 2006; pp. 1–7.
14. Struharik, R.J.R.; Novak, L.A. Intellectual property core implementation of decision trees. *IET Comput. Digit. Tech.* **2009**, *3*, 259–269.
15. Sethi, I.K. Entropy net: from decision trees to neural nets. *Proc. IEEE* **1990**, *78*, 1605–1613.
16. Alioto M.; Palumbo, G. *Model and Design of Bipolar and MOS Current-Mode Logic: CML, ECL and SCL Digital Circuits*; Kluwer Academic Publishers: Norwell, MA, USA, 2005.