J Real-Time Image Proc (2016) 11:155–166
DOI 10.1007/s11554-013-0331-8

ORIGINAL RESEARCH PAPER

Fast image recognition based on *n*-tuple neural networks implemented in an FPGA

Petr Burian · Radek Holota

Received: 13 July 2012 / Accepted: 9 February 2013 / Published online: 26 February 2013

Abstract This paper deals with the design and the implementation of an image recognition system based on FPGA devices. It explores an *n*-tuple methodology using node 'grouping' and the possible advantages offered by this little-known technique. The paper is based on the implementation of this concept by an FPGA device. A novel approach to the organization of the neural networks data in the *n*-tuple memory is introduced. The system was tested on a real-world recognition task—the recognition of road signs. The test results are presented and discussed. It is concluded that the designed system may be a powerful part of more complex equipment for the solution of many recognition issues.

Keywords Image recognition · FPGA · Neural network · *N*-tuple · Qsys interconnect

1 Introduction

At present, the requirements for electronic systems to interact with humans are becoming more and more complex. A growing demand for partial or fully autonomous systems suggests that this is an area in which neural networks are applicable. Often, one of the main tasks of neural networks is to recognize the object within the neighbourhood and to opt for the best action based upon image understanding. This objective is usually very difficult and time consuming. For this reason, specialised HW systems are designed. Hence, the main topic of this presented work is the implementation of *n*-tuple neural networks on an FPGA. The *n*-tuple methodology was chosen because, in hardware, it is very fast and is comparable with other conventional methods in performance. The detailed comparison in different applications is published by Morciniec and Rohwer [1].

The original *n*-tuple methodology of Bledsoe and Browning [2] in 1959 was not realised until later that it could be implemented by means of 'deterministic' logic nodes. Hence this may be considered to be one of the oldest pattern recognition techniques based on logic node neural networks. This method was later popularised by Aleksander [3, 4] and realised in HW [5–7]. In the early 1990s, several SW simulations were created, which led to the design of an improved SW system [8]. It included an image pre-processing stage [9] and image recognition stage [10]. The system supported the recognition of binary, greyscale, and colour images. The novel derivative for this colour image recognition was published in [11, 12].

The *n*-tuple methodology was originated by Bledsoe and Browning in 1959 for the purpose of recognising printed characters and subsequently hand-written characters. Nevertheless, the use of the *n*-tuple method is not limited only to this purpose. In the past, systems utilising *n*-tuple nodes were used in many different applications, e.g. face recognition [13], texture recognition [14], control and automation [15, 16], or in medicine [17].

P. Burian
Regional Innovation Centre for Electrical Engineering, University of West Bohemia, Pilsen, Czech Republic
e-mail: burianp@rice.zcu.cz

R. Holota (✉)
Department of Applied Electronics and Telecommunications, University of West Bohemia, Pilsen, Czech Republic
e-mail: holota5@kae.zcu.cz

2 Background

2.1 *N*-tuple methodology

In principle, the *n*-tuple technique is equivalent to that of using Single Layer Networks (SLNs) consisting of 'deterministic' logic nodes. The term 'deterministic logic nodes' was originally used by Aleksander and Stonham over 30 years ago. It simply means that an *n*-tuple node can perform all 2 to the power of '*n*' logic functions and that, after training, it can only respond to those training patterns and, therefore, it is 'deterministic'. The logic nodes of these SLNs are realised by Random Access Memory (RAM). Sometimes it is described as RAM-based networks [18] in the literature. Each logic node consists of a RAM with an *n*-bit address space (see Fig. 1). The following paragraphs describe the *n*-tuple methodology in a simple way. An exact mathematical description of the *n*-tuple recognition method can be found in many articles, e.g. [13, 19].

The pattern which is applied to the address inputs consists of '*n*' sample points. These points are taken pseudo-randomly from the input data. The pattern applied to the input is termed an "*n*-tuple". All the logic nodes have to be initialised (to zero state) before the training phase. During training, each RAM stores a logic '1' to the position given by relevant sampled *n*-tuple data words. The recognition phase is based on reading the stored data from the address given by *n*-tuple data words which are sampled from the input image by the same pseudo-random sequence. During recognition, the RAM operates as a look-up table without requiring any arithmetic or logical operations. Each SLN (or "discriminator") is composed of *n*-tuple nodes (the number of nodes is defined by '*k*') and the response is given by the summation of all logic nodes in the layer. The bit width of response depends on the number of nodes (see Fig. 1).

In principle, "*n*-tuple" classifier systems operate in a multi-discriminator configuration (see Fig. 2). In the training mode, each discriminator is trained on each class of patterns. The class is dedicated to one object and is formed by the set of similar images of this object (e.g. with different rotation, scale, noise). The number of discriminators is equal to the number of classes which are defined for future classification. The classification mode can be started when all discriminators have been trained. Each discriminator gives a response to an unknown input image. The input image is assigned to the class which corresponds to the discriminator with the highest response.

The discriminators' responses are usually represented either numerically (by the number of pass logic nodes or the percentage value) or as a bar graph display. In general, each bar in the bar graph display represents the degree of compliance within a given class. Usually, the responses are also displayed in the training mode. The monitoring of these responses gives an indication of the amount of training required and assists in determining possible over-training or under-training of the neural networks.

2.2 Network with grouped *n*-tuple nodes

It is possible to utilise grouping methods in these types of SLNs as in [8, 10]. The effect of grouping is to increase the differences among responses to the discriminator (for the given class) and other discriminators. The principle of this method is based on creating groups of *n*-tuple nodes in each SLN. Each group consists of the given number of nodes, a summation unit, and a threshold unit (see Fig. 3). The final response of each discriminator then constitutes the sum of the groups' responses.

2.3 Utilisation and memory requirements

Generally, an SLN composed of *n*-tuple nodes or grouped *n*-tuple nodes can be used for the recognition of binary images. In the case of greyscale images, it is necessary to use a suitable method for converting them into binary

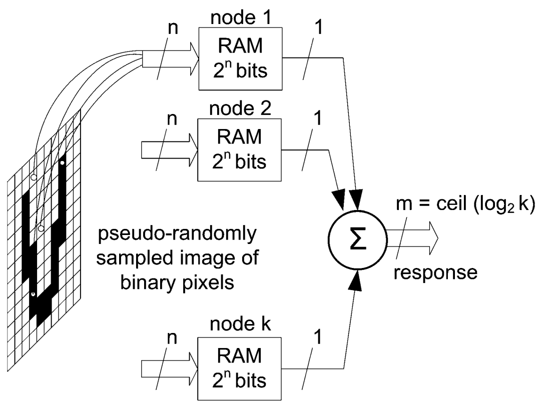

Fig. 1 Single layer network architecture

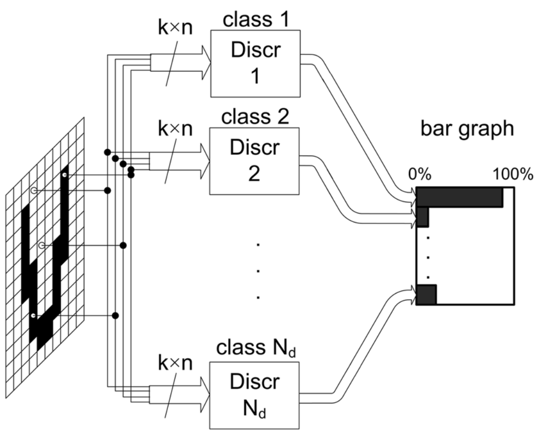

Fig. 2 Multi-discriminator configuration

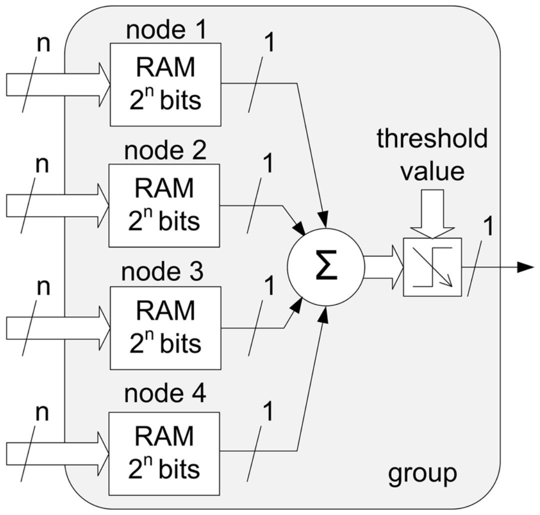

Fig. 3 Principle of grouping

images. In order to perform colour recognition, the networks are composed of "trixel" *n*-tuple nodes (TNT nodes) as shown in [8, 10].

The memory requirements are given by the following formula:

$$\mathrm{Mr} = \frac{N_{\mathrm{d}} \cdot 2^{n} \cdot N_{\mathrm{p}}}{n} \,(\text{bits}); \qquad (1)$$

where N_{d} is the number of discriminators, N_{p} the number of selected pixels, n the type of tuple ($n = 8 \Rightarrow$ 8-tuple), and Mr the memory requirements in bits

The variable N_{p} has to be chosen in accordance with the size of the tuple node and the group size.

3 HW implementation

3.1 General description

This recognition system is designed as a System on the Programmable Chip (SoPC) architecture based on an FPGA device. Several Qsys [20] components form the system (Fig. 4). The Qsys interconnect ensures connectivity and mutual communication. The Camera Unit together with the Frame Buffer obtains image data from the camera module TRDB-D5M (5 Mega Pixel Digital Camera Package from Terasic Corp.). It also provides the conversion of the raw image data to RGB (red, green, blue) colour model and BW (black/white) format. The Frame Buffer selects and stores data needed for image recognition. In addition, the component may send full image data to other components in the system. The Training Unit and Recognition Unit are the next main (in the line with the Camera Unit) components of the system; they are discussed later. Further, the system contains two memory controllers—the SRAM and SDRAM controllers. The external SRAM memory (2 MB–1 M × 16 bit) is connected to the SRAM Controller and is used for storage of the neural networks data. The latter controller—the SDRAM Controller—manages data transfers to the 64-MB (32 × 16 bit) SDRAM. This memory is used for video storage in this project. Data from this memory can be displayed on the PC monitor by means of the VGA Controller. The VGA Controller works with a resolution of 800 × 600 at 75 frames per second (fps). The remaining three components control the training and recognition process and communicate with the supervisory system. The core of the supervisory software is implemented in the softcore CPU NIOSII; this 32-bit processor controls the operations of all the other components. This processor also communicates with the high-level system (PC) via a UART (implemented by the UART Controller). The on-chip memory serves as the program and operational memory for the NIOSII. The special user application on the PC controls the whole image recognition process via the UART and NIOSII processor.

3.2 Camera unit with frame buffer

This is the key component of the system. It obtains data from the external camera module and provides two outputs which are the data stream for the video memory and image data for *n*-tuple processing in the appropriate format. The block diagram of this component is shown in the Fig. 5. This component is connected to the camera module TRDB-D5M via the Avalon Conduit Interface. It captures image data in RAW format (resolution 800 × 600) and can convert this data to the following formats: RGB 16 bit, Greyscale and BW. The selected format depends on the setting of the component by means of the Avalon Slave Port.

The conversion to a BW image uses a fixed threshold that is set via the Avalon Slave Port or an automatic threshold detection. The final RGB vectors enter the FIFO memory and can be sent to the video memory by means of the Avalon Master Port. This function makes it possible to observe the captured image.

The remaining parts of the component create an appropriate data format for subsequent processing by the Training and Recognition Units. The authors of this project designed and implemented the new approach of image data buffering. Compared with the conventional architecture [21], this approach does not need a buffer for full image frame. They usually store a full frame to memory and then pixels for processing are selected; afterwards, these pixels are randomly combined to form the *n*-tuples. The main disadvantage of these techniques is higher memory requirements.

The Pixel Selector selects pixels for the next processing stage. The full image contains 480,000 (800 × 600) pixels of which 5 % are selected (i.e. 24,000 pixels). The pixel

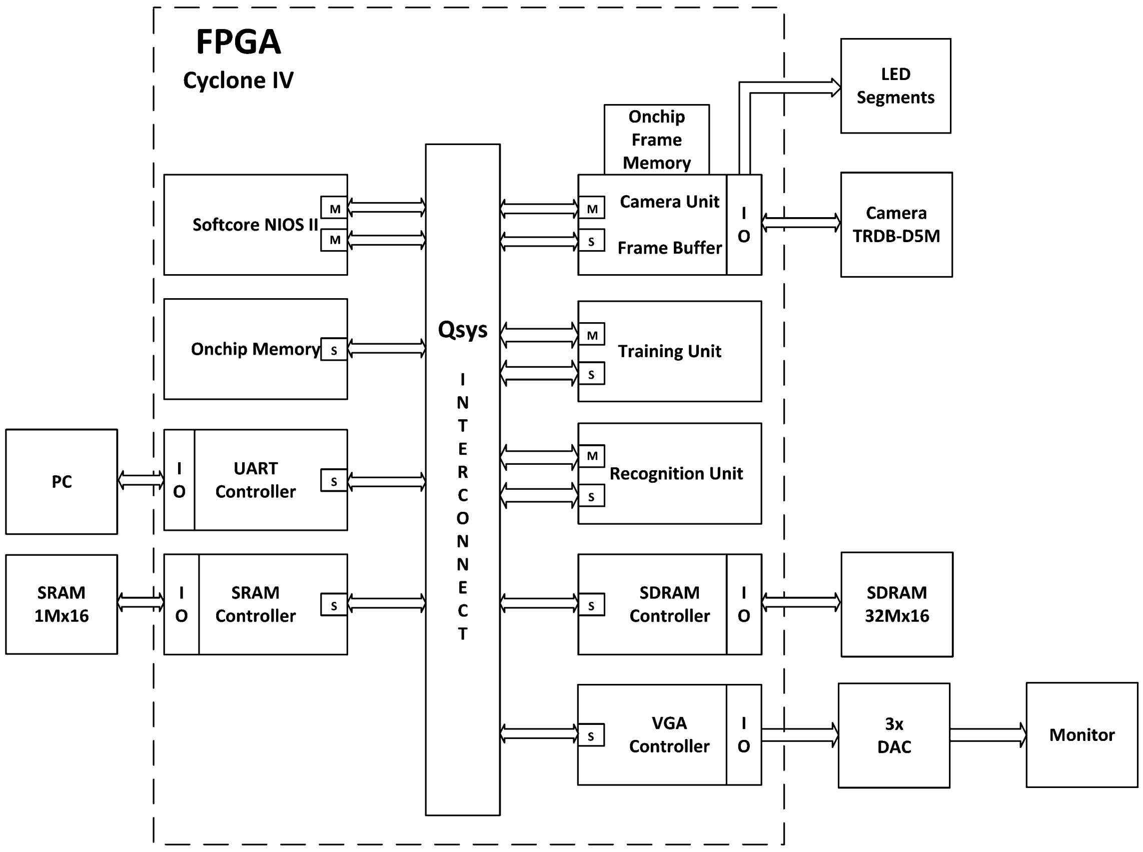

Fig. 4 Block diagram of the system

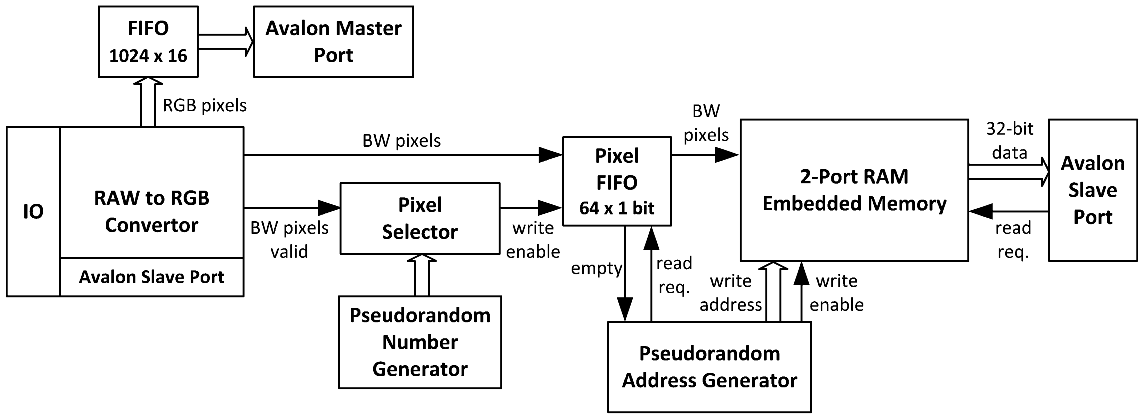

Fig. 5 Camera unit with frame buffer

data are fed into the Pixel FIFO, and the Pixel Selector generates a "write enable" signal which allows writing of pixels into the FIFO memory. The image frame is divided into 15 segments in which pixels are selected with variable intensity. The pixels more closely placed to the centre of the frame have preference over those near the borders. The Selector detects the start of the frame and then selects 24,000 pixels from each frame. The optional module for performing image data processing can be inserted between the RAW to RGB Convertor and Pixel FIFO.

After selection of the pixels, it is necessary to assemble them in a pseudo-random manner before applying them to the n-tuple memory nodes. In this case, an 8-tuple is used. The random composition is made by means of an embedded memory which is configured as a two-port RAM memory with a capacity of 24,000 bits. The write port consists of a 1-bit data input and a 15-bit address. The read port consists of a 32-bit data output with a 10-bit address input. The component generates a random write address; it means that pixels are stored in random locations within the memory. The correct write address and control signals are created by the Pseudo-random address generator (Fig. 6). A simple 15-bit Linear Feedback Shift Register (LFSR) with a minimal polynomial of $x^{15} + x^{14} + 1$ is used. This pseudo-random generator generates random numbers in the range of 0–32,766; however, correct write addresses are in the range of 0–23,999. For this reason, it is necessary to check the value of the address vector. The LFSR also implements a digital comparator which flags (by the signal address valid) whether the generated address vector is valid. Further, the generator contains a “gen. enable” signal which enables/disables its function and thereby issues the next random number (address). The writing into memory is allowed if several conditions are satisfied. If the Pixel Fifo is not empty, the Address generator is activated and the generated address vector is correct, the memory write enable signal will be set to high in the next clock cycle (via the D flip-flop). Also, the generated address vector is registered by means of a 15-bit register. If the memory write enable signal is high, the pixel of the image will be stored at a random position in the memory. If the “gen. enable” signal is high then the next random address will be released in the next clock cycle and, if the generated address vector is not correct or if the pixel is not stored in memory. In other words, this signal is high if the next address vector is needed; either the current value of the vector is not accepted or the new address vector for another memory write is demanded. This approach makes it possible to use a simple pseudo-random generator implemented as a LFSR. If the generator gives incorrect values, the pixel data are held in the Pixel Fifo memory. It is necessary to note that the implemented logic works with a 110-MHz clock in this project. It is a higher frequency than 50 MHz which is the maximum pixel rate. It must also be taken into account that only 5 % of pixels are processed this way; thus the delays caused by waiting for the correct address vector are irrelevant. The pixels are stored in memory and form 3,000 8-tuple nodes. As mentioned earlier, the read data memory port is 32-bit wide which means 750 double-words of data. This configuration of memory was chosen for more effective data transfers inside the system; four 8-tuples can be read within one read transfer operation. The read port of the memory is connected to the Qsys Interconnect via the Avalon Slave Port. It implements an auto-increment read pointer in which the whole memory is mapped as one 32-bit word in the memory space and a new read address is released automatically. This function simplifies the address calculation to other components in the system; the main benefit of this component is that a buffer for a full image is not needed. It represents a significant reduction of memory requirements. The amount of reduction is dependent on the coverage of an input image. In our case, the reduction of memory requirements is approximately 95 % within a 5 % coverage. Another advantage is lower time consumption because the random selection and forming of image pixels are performed already during data storing.

3.3 Training unit

The purpose of this component is to train neural networks consisting of 8-tuple nodes (each node composed of 8 image pixels). The component reads 8-tuples from the embedded memory in the Camera Unit that define the memory position (for neural network data) where logic high will be stored. Other memory positions remain unchanged. Before training the first image frame, it is necessary to clear the neural networks memory to ensure logic low value on all positions. In this project, neural network data are stored in an external SRAM memory with a 16-bit memory data bus. The memory requirements for one discriminator are 768,000 (256 × 3,000) bits, where 3,000 is the number of 8-tuple nodes each of which requires 256 bits. On account of a more effective access to the memory, the following structure of the neural network data in the memory was designed—see Table 1: The data for a particular discriminator (class) are not stored in the continuous memory area but the “column structure”. Data of particular discriminators are organized as columns within the memory area. For example: data for the discriminator (class) 1 is stored in the LSBs of the memory words, class 2 in bits with index 1, and so on. In Table 1, it is shown that 512 (0x200) bytes are needed for keeping one n-tuple of 16 discriminators. For that reason, the memory offset for the next n-tuple is always 0x200. Because of this designed memory structure, the formula for the memory requirements (1) has to be modified in the following way:

$$M_r = \frac{f_{mo} \cdot W_b \cdot 2^n \cdot N_p}{n} \text{(bits)}; \qquad (2)$$

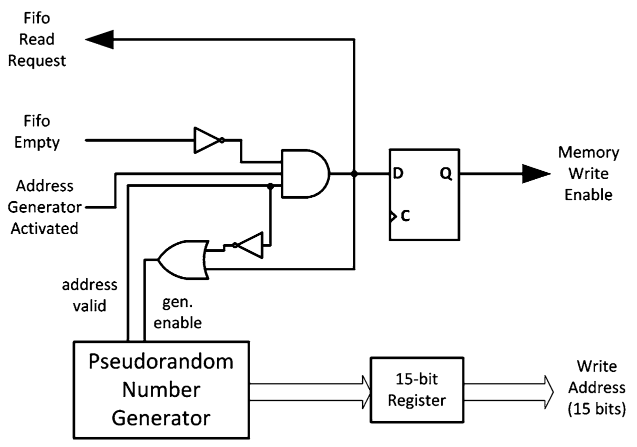

Fig. 6 Pseudorandom address generator

where W_b width of memory data bus, N_p the number of selected pixels, n size of tuple ($n = 8 \Rightarrow$ 8-tuple), M_r memory requirements in bits and f_{mo} memory organization factor

Memory organization factor is defined as

$$\min\left\{f_{mo} \in \mathbb{N} | f_{mo} \geq \frac{N_d}{W_b}\right\}; \quad (3)$$

where W_b is width of memory data bus and N_d the number of discriminators.

From this formula, it is obvious that the memory is fully and effectively utilised if the number of discriminators is an integral multiple of the memory data bus width.

The authors of this project assumed the use of 16 classes; this means that 12.288 Mbit of memory are needed. The process of training is as follows: at first, 8-tuples are read (4 × 8-tuples in one 32-bit vector) from the embedded memory. The word on the position defined by the value of the 8-tuple and its order is read from the SRAM memory. In this word, the appropriate bit (determined by the chosen class) is set to high. Afterwards, the modified word is written back on its position. Training is complete once all 8-tuples (in this case 3,000 ⇒ 750 vectors) are read. In addition, the component can determine the networks' responses to the training data. This response may be useful for training process control.

The component is controlled by means of control registers via the Avalon MM interface. By these registers, the user can start training and setup values (grouping and group threshold) for response calculations.

3.4 Recognition unit

This component determines the responses of the neural networks based on the currently unknown images. As well as the Training Unit, it reads n-tuple data from the embedded memory. The data are loaded into a small FIFO memory; this means that the delay caused by their reading occurs only at the beginning of the recognition process. The rest of the process is determined only with the latency of the SRAM (memory for neural networks data). The component's Avalon ports are able to work with any type of memory (an appropriate controller is needed). Responses of (up to) 16 networks may be calculated in parallel because of the designed data structure of the memory (introduced in previous chapter). The word from the SRAM memory represents neural network data for 16 discriminators (depends on the width of the memory bus). Hence, only one read operation is required for each n-tuple. From this, the time needed for obtaining the responses of W_b (width of memory data bus in bits) classes can be derived. The requirements T_r recalculated to this number of discriminators is given by the following formula:

$$T_r = \frac{(M_l \cdot N_n) + 8}{f_s} (\mu s); \quad (4)$$

where M_l is latency of reading from memory (in clock cycles), N_n the number of n-tuples and f_s the frequency of system clock (in MHz). Note to formula: constant 8 represents latency of component's pipeline.

It follows that the time needed for evaluation of one discriminator is the same as the time for the evaluation of W_b discriminators. For that reason, the real-time consumption related to one discriminator depends also on a relationship between the number of discriminators and the width of the memory data bus.

The final computed responses are obtainable via the component's response registers which are mapped by means of the Avalon Memory Mapped Interface. The grouping is supported by the component as well. The component contains a few control registers for the component's setting. The desired group size and group threshold can be set; the permitted range is from 1 (without grouping) up to 15. However, it is necessary to take into account that the group size must be a divisor of the total number of used n-tuples (in given class). In this project, 3,000 8-tuples are used and this means that grouping factors of 7, 9, 11, 13 and 14 are not utilisable. The value of the grouping factor has no impact on the recognition speed.

3.5 System control

As mentioned in section HW implementation—General description, the whole system is controlled by the softcore CPU NIOSII. The firmware only controls the operation of the components and ensures communication with the supervisory system (the PC in this case) via the Avalon MM Interface. However, all sophisticated operations related to image processing and n-tuple operations are performed by

Table 1 Memory structure

Word index	Memory address					Class
	0x000000	0x000200	0x000400	…	0x176E00	
0 (LSB)	1st tuple	2nd tuple	3rd tuple	…	3,000th tuple	1
1	1st tuple	2nd tuple	3rd tuple	…	3,000th tuple	2
2	1st tuple	2nd tuple	3rd tuple	…	3,000th tuple	3
3	1st tuple	2nd tuple	3rd tuple	…	3,000th tuple	4
…	1st tuple	2nd tuple	3rd tuple	…	3,000th tuple	…
15 (MSB)	1st tuple	2nd tuple	3rd tuple	…	3,000th tuple	16

the above-introduced components. This means that no CPU's computing time is needed for these operations. The utilisation of a softcore processor is very profitable and a modern way to control the components in the FPGA devices. It enables simpler debugging and testing of the system. The processor also can perform more complex control algorithms which can be difficult for a user. This was exploited to control the training process when training on real images from the camera (not during the test images training). An algorithm was tested so that the processor could terminate the training process if defined conditions were satisfied. The use of two termination conditions was tested. The first condition limited the maximum number of frames. Second, the training can be terminated if a defined number of successive frames yield responses which are equal or higher to the defined response value. The parameters of this algorithm may be set by the supervisory system. Of course, this algorithm might be implemented by the supervisory system as well. The recognition process is very simple and can be described in a few steps:

1. The supervisory system sends command requiring recognition. It also transfers grouping and group threshold parameters.
2. The CPU sends a command to the Camera Unit to capture a frame from the camera and process it to a suitable form for the recognition process.
3. The CPU sets the grouping parameters and starts the recognition process by setting the start bit in the control register of the Recognition Unit.
4. The CPU detects activity of the Recognition Unit by means of its status register. If the recognition process is finished, the CPU reads the responses and sends them to the superior system. Afterwards, the process can continue by step 1 or be finished.

The process of training is very similar; instead of the Recognition Unit, the Training Unit is used.

4 Experimental results

In this section, a real-world recognition task was used to test the designed HW system. The performed tests had two main goals. The first objective was to verify the HW system. The second objective was to present the possibilities of system utilisation in a real recognition task and to show the influence of different system settings.

4.1 Description of recognition task

As mentioned above, a real-world recognition task was chosen. This task was road signs recognition which is a really difficult problem. The presented system should be a powerful part of a more complex system for the solution of this problem. The main task of the recognition system was to classify the images of signs with a slightly different position, rotation and size. So the supervisory system should include processing for the sign detection and processing for the normalization of position, rotation and size.

4.2 Database of input images

For the presented tests, artificially generated images were used for testing due to the fact that one of the tasks was to verify the implementation of neural networks in HW. For that reason, a program named SourceImageGen for generating the datasets of images with different positions and rotation was created. The test database included 11 classes—road signs (see Fig. 7). The resolution of source images was 800 $\times$ 600 pixels.

There were six datasets for each class which differed in the position of the sign and its angle of rotation. These datasets were generated by SourceImageGen and the parameters for each dataset are shown in Table 2. The program output are the binary files with pseudo-randomly mapped and thresholded pixels, which are used for recognition from the image of a road sign. This feature significantly reduces the amount of data which have to be transferred to the HW system. The program performs the same pseudo-random selection and forms the data in the same way as in HW. That ensures the possibility to upload binary data directly to the HW system memory (2-port RAM Embedded Memory in Camera Unit—see Fig. 5) through the JTAG interface.

4.3 Tests and results

The tests can be divided into three groups. The first group includes the datasets 1 and 2 where the angle of rotation

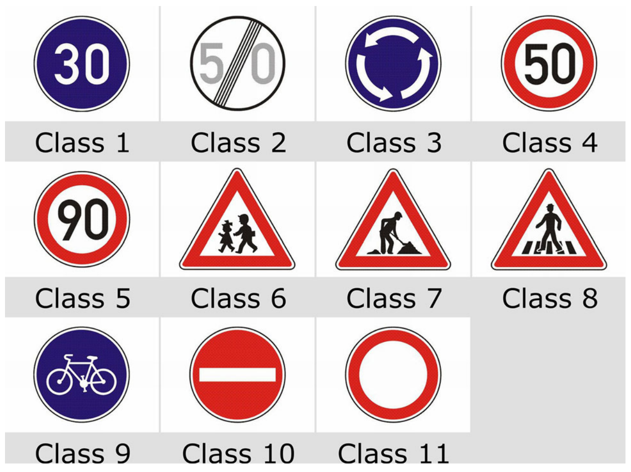

Fig. 7 Road sign classes

was varied. The second group works with datasets 3 and 4 where the position was varied. The last group used the datasets 5 and 6 where both variations were recombined. Table 2 shows that the differences between the two datasets in the same group were small. The reason for this was the possibility to use different datasets in the training and classification modes. A detailed overview of the tests is shown in Table 3. Each test included several measurements with different neural network settings—G1T1 (Group size 1, group threshold 1), G4T1, G4T2, G4T3, G4T4, G8T8, G15T13.

Tests 1, 3 and 5 were performed to verify the neural network implementation by the FPGA device. According to the theoretical assumptions, all discriminators must have a 100 % response for given classes in the cases where the same datasets are utilised in both modes. This verification was successful for all classes and tests.

Test 6 was chosen for the illustration of results which are reached by this recognition system. For the illustration of the obtained results, a bar graph is presented in Fig. 8. It shows the maximum and minimum levels of responses in the case that input images belong to the class 4 (speed limit—50) and G8T8 neural network settings. The figure indicates the similarity between Class 4 and Class 5 (speed limit—90) but the difference between the minimum response of the Class 4 discriminator and the maximum response of the Class 5 discriminator is still sufficient for reliable recognition.

Table 2 Datasets parameters

Dataset	Parameters		
	Position	Rotation	Number of variation
1	Constant	±5°, step 1°	11
2	Constant	±4.5°, step 1°	10
3	±2 px, step 1 px	Constant	25
4	±3 px, step 2 px	Constant	16
5	±1 px, step 1 px	±5°, step 1°	99
6	± 1px, step 1 px	±4.5°, step 1°	90

Table 3 Summary of tests

Test	Used datasets	
	Training mode	Classification mode
1	Dataset 1	Dataset 1
2	Dataset 1	Dataset 2
3	Dataset 3	Dataset 3
4	Dataset 3	Dataset 4
5	Dataset 5	Dataset 5
6	Dataset 5	Dataset 6

The bar graph in Fig. 9 shows the levels of responses for different neural network settings. Each column displays the range of discriminator's responses to correct class (full fill), to other classes (hatched fill) and the differences (dotted fill). By comparing of each column, it is possible to observe a benefit of grouping method with a higher group threshold against grouping with a too low group threshold or without grouping (G1T1). On the other hand, too high group threshold in combination with a higher group size can cause significant reduction of responses to correct class. It means that the differences between correct and other classes can be decreased.

The second bar graph (Fig. 10) shows the differences between responses to correct and other classes for different neural network settings (see legend of graph) and for all tests. The bar graph clearly shows that the differences are really high for appropriate neural network settings and, therefore, the designed system could be successfully used for road sign recognition system in the future.

It is impossible to say generally which network setting is the best or acceptable. The suitable network parameters are dependent on many factors like target application, quality of image acquisition (noise, clutter, etc.), similarity of classes and others.

4.4 System performance

This part specifies the time consumption and the memory requirements of the recognition process. Initially, the parameters of HW system are summarised. Eleven test classes were used. Each image consisted of 480,000 pixels, matching the resolution of 800×600 pixels. For the recognition process, 5 % of pixels were selected; this represents 24,000 pixels. If the formula (1) is applied to these values, the memory requirement is 8.448 Mb. This value is the minimum needed for representation of the discriminators. However, because of the exploited memory organization introduced in the chapter HW Implementation, the formula (2) must be used. By this formula, 12.288 Mb of memory is required. This value is equal to that using 16 discriminators. The difference between these two values represents an overhead of the memory organization. It can seem to be markedly disadvantageous. However, it is only a trade-off between the memory requirements and the time required to obtain the responses. If a large number of discriminators are assumed, then this overhead is marginal.

The SRAM memory with a 16-bit data bus and a reading latency of three matched clock cycles was used. By using formula (4), the response times for the discriminators can be calculated; this is 81.89 μs from 1 up to 16 discriminators. If 11 discriminators are used, then the time requirement for one discriminator is longer and this means that a time of 81.89 μs is required to obtain all 11

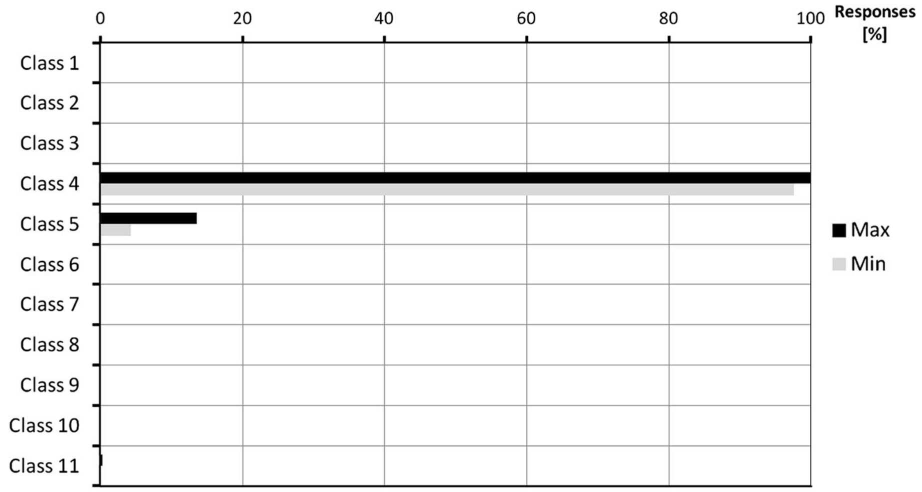

Fig. 8 Responses for 'Class 4' images and G8T8 configuration

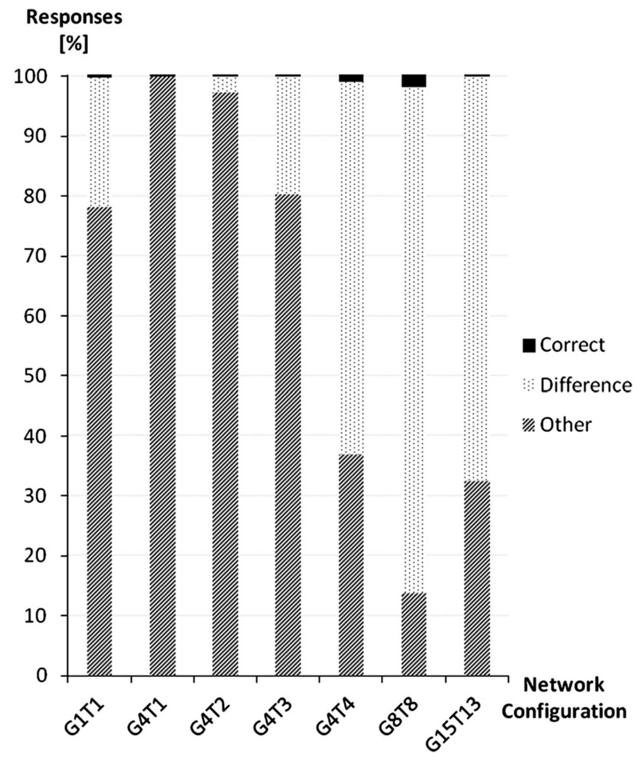

Fig. 9 Levels of responses for different neural network settings

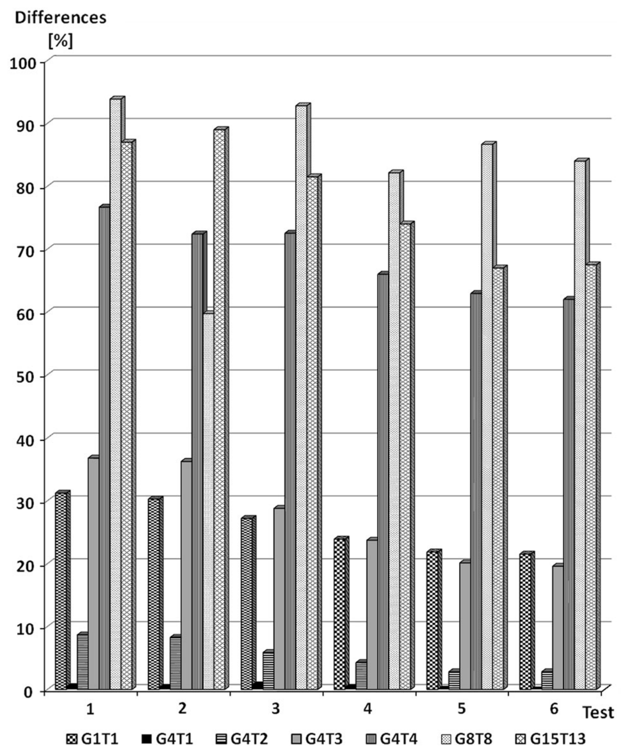

Fig. 10 Comparison of minimum differences

responses. This time does not include the time needed to control the components. In theory (image processing is not taken into account), the designed system may evaluate over 12,000 unknown images per second. The same number of images can be evaluated even if 16 discriminators are used. From another point of view, the system with a frame rate of 1 fps may obtain responses of up to roughly 195,000 discriminators per second. Indeed, enough memory space is assumed.

The following chart generally summarizes the previous two paragraphs. It shows the trend of two coefficients (defined by the authors) depending on the number of classes. The first of them is the Coefficient of Acceleration (CA); it means the speed benefit of the designed memory organization. In other words, it expresses the ratio between the number of memory read operations needed for recognition not using (sequential organization is assumed) and using our designed memory organization. The latter coefficient is the Coefficient of Memory Requirements (CMR). It expresses the ratio between memory requirement needed for our and for classical sequential memory organization. It is clear from the chart that our designed memory

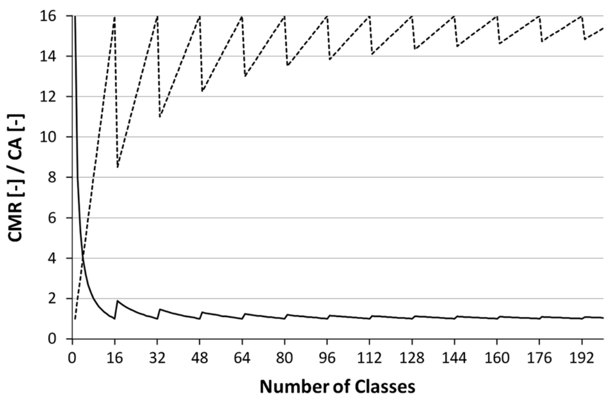

Fig. 11 Coefficients of memory requirements (CMR—*solid*) and acceleration (CA–*dashed*)

organization is beneficial to the cases where the number of classes is in multiples of bus width or for larger numbers of classes. If we consider a large number of classes, we obtain the coefficient of acceleration approaching 16, and the CMR approximately 1. We may say that for an infinite number of classes, the speed benefit of our memory organization is equal to the width (16 in our case) of used memory; memory requirements remain unchanged (identical with sequential organization). However, this is only theoretical consideration.

The training process takes a significantly longer time. The Training Unit has to perform 3,000 read–modify–write cycles. In the experiments, the used SRAM memory has a writing latency of four clock cycles. In total, 21,007 clock cycles are needed. This matches approximately to 190 µs (Fig. 11).

4.5 FPGA resources

The FPGA resources are acceptable. The Recognition Unit needs 905 logic elements (LEs) and the Training Unit only 241 LEs; each unit consumes 128 bits of embedded memory. In comparison with these, the Camera Unit is the most demanding; it requires 1,784 LEs and 60,302 bits of memory. However, these values include a component part for streaming the full image to the SDRAM memory to display it. As long as this functionality is not needed, some resources could be saved. The whole system (shown in Fig. 4) consumes 7,125 LEs. The maximal frequency of this design is approximately 130 MHz. All tests and measurements were performed on a Cyclone IV device from Altera Corp. The development kit DE2-115 from Terasic Corp. was exploited.

5 Comparison with other methods and implementations

This section presents and compares the obtained results in two domains: in recognition performance domain and from the point of view of the processing speed. For comparison, the following conventional classification algorithms were chosen:

- Nearest neighbour algorithm
- Minimum mean distance algorithm
- *K*-nearest neighbour algorithm

These methods were implemented in software by the National Instruments Vision Builder for Automated Inspections (NI VBAI). All mentioned methods used the sum distance metrics (city block metrics). In the case of the *k*-nearest neighbour algorithm, the tests with three different *k*-parameters were performed.

Furthermore the special software application in MATLAB was programmed. It implements the image recognition based on the *n*-tuple method by the same way as done by the hardware described above. The results obtained by this application can be used to verify the presented hardware implementation and to compare software and hardware solutions.

The hardware system introduced in [16], which is based on the same *n*-tuple method, was chosen to compare system the performance of the two different architectures.

The benchmarks use the Test 2 which is described in Table 3. In this kind of test +different datasets are exploited for the training phase (Dataset 1) and the classification phase (Dataset 2). Each run of the test was repeated 1,000 times and the results were averaged to obtain relevant data.

Table 4 summarizes the results of the benchmarks. It is evident that the time requirements of the classification algorithms realised by NI VBAI are almost identical. The achieved speeds are around 170 fps. In terms of recognition performance, the minimum mean distance algorithm reached the smallest false rate (FR) from the tested conventional methods, but still did not reach 0 % like the *n*-tuple method. The misclassification was observed mainly in the cases of these road signs: speed limit 50/speed limit 90 or children/pedestrian crossing.

The software implementation of the *n*-tuple method in MATLAB produced the same responses like the HW realisation but the time requirements were significantly higher. All software tests were performed on a PC Intel (R) Core2 Quad CPU Q9550 @ 2.83 GHz, 3 GB RAM.

Table IV also presents the results of two hardware implementations. However, it is necessary to note that the

Table 4 Comparison of speeds and false rates

Tools	Classification algorithm	Parameters	Speed (fps)	FR (%)
NI VBAI	Nearest neighbour	Sum distance metrics	169	7.3
	Minimum mean distance	Sum distance metrics	169.5	1.8
	k-nearest neighbour	Sum distance metrics, $k = 3$	173.3	6.3
		Sum distance metrics, $k = 6$	170.5	9.1
		Sum distance metrics, $k = 10$	168.6	8.2
Matlab	*N*-tuple method	$n = 8$, coverage 5 %	37	0
HW—Burian	*N*-tuple method	$n = 8$, coverage 5 %, memory read latency 3	12,211	0
HW—Bonato [16]	*N*-tuple method	$n = 8$, coverage 5 %	12,247	0

speed of the HW realisation by Bonato [16] is estimated for the same configuration (number of *n*-tuple nodes). This HW architecture is based on the use of the embedded memory inside the FPGA; it makes it possible to implement memory configuration optimized for given *n*-tuple network.

From the values in Table 4 it is clear that the time requirements of Bonato's architecture are slightly higher. However, the amount of embedded memory is significantly limited and it does not allow the implementation of larger number of classes and higher input image resolution. The novel memory organization presented in this article solves this disadvantage and makes it possible to use a conventional external memory with similar time requirements.

6 Conclusion and future work

This paper has introduced a new hardware implementation of *n*-tuple neural networks based on an FPGA device. The presented novel structure for memory organization offers effective access to the data memory for the processing of more classes at one time. It enables very fast image recognition in a relatively simple HW architecture. This article also provides a brief overview of the *n*-tuple theory with grouping methods and presents the possibility of its utilisation for the road sign recognition. The system performance and the required FPGA resources are included for the estimation of possible system performance.

The comparison of the presented hardware solution with other methods and architectures is summarized in Table 4. The values in this table show that our approach is comparable to the solution published in [16] and it is not limited by the size of the embedded memory. This advantage makes it possible to recognize the images in high resolution and/or higher number of classes.

Unfortunately, the comparison of the 'learning' capability of *n*-tuple networks with conventional feature extraction systems is not particularly fair to both techniques. The classical *n*-tuple classifiers have the generalisation properties (i.e. their probabilistic nearest matches). They can tolerate variations in the input image. It is also of a great importance that the parameters of the object in the image do not have to be analysed. On the other hand, the conventional methods—namely Feature Extraction—usually provide concise and reasonably accurate measurements of an object within an image. However, it is difficult to determine how many and which features should be extracted after the initial edge detection (i.e. perimeter, area, shape factor, min/max enclosing rectangles, centre of area, min/max radius, etc). From the theoretical point of view, it is evident that the conventional methods mentioned above require a lot of different operations which are needed for feature extraction. Contrarily, the *n*-tuple classifiers only need reading from memory and simple 'add' instructions in the recognition phase. For these reasons, the time requirements should be lower.

This research should lead to a complex recognition system placed inside a car which would facilitate the tracking and recognition of road signs. Nevertheless, the developed HW implementation of an *n*-tuple neural network is not limited only to this purpose and could be used for miscellaneous tasks of image recognition.

Acknowledgments This research has been supported by the European Regional Development Fund and the Ministry of Education, Youth and Sports of the Czech Republic under the Regional Innovation Centre for Electrical Engineering (RICE), Project No. CZ.1.05/2.1.00/03.0094. This research has been also supported by the Department of Applied Electronics and Telecommunications at the University of West Bohemia.

References

1. Rohwer, R., Morciniec, M.: A theoretical and experimental account of *n*-tuple classifier performance. Neural Comput **8**(3), 629–642 (1996). doi:10.1162/neco.1996.8.3.629
2. Bledsoe, W.W., Browning, I.: Pattern recognition and reading by machine. In: Papers presented at the 1–3 December 1959, eastern joint IRE-AIEE-ACM computer conference [IRE-AIEE-ACM '59 (Eastern)], pp. 225–232. ACM, New York. doi:10.1145/1460299.1460326 (1959)

3. Aleksander, I., Albrow R.C.: Pattern recognition with adaptive logic elements. In: IEE conference on pattern recognition, pp. 68–74 (1968)
4. Aleksander, I., Stonham, T.J.: Guide to pattern recognition using random-access memories. IEEE J Comput Digital Tech 2(1), 29–40 (1979). doi:10.1049/ij-cdt:19790009 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4808602&isnumber=4808593)
5. Wilkie, B.A.: A stand-alone, high resolution adaptive pattern recognition system. Ph.D. Thesis, Department of EEE of Brunel University, UK (1983)
6. Aleksander, I., Stonham, T.J., Wilkie, B.A.: Computer vision systems for industry: wisard and the like. Digit Syst Ind Autom **4**, 305–320 (1982)
7. Aleksander, I., Thomas, W.V., Bowden, P.A.: WISARD. A radical step forward in pattern recognition. Sens Rev (1984)
8. Holota, R., Trejbal, J., Wilkie, B.A.: Software realisation of a colour image recognition system with an image normalisation stage. In: Proceedings of the University of West Bohemia, sect. 1, vol. 4/2000, pp. 75–86. (ISBN 80-7082-718-1, ISSN 1211-9652)
9. Holota, R.: Position, size and rotation normalisation, diploma project report. Dept. of Electrical Engineering, University of West Bohemia, UK (2000)
10. Trejbal, J.: Software realisation of a colour image recognition system using *n* tuple and MIN/MAX node neural network, diploma project report. Dept. of Applied Sciences, University of West Bohemia, UK (2000)
11. Wang, Y.S., Griffiths, B.J., Wilkie, B.A., Silverwood, P.A., Norgate, P.: Complex and coloured object inspection. Comput Ind **25**(2), 125–130 (1994). doi:10.1016/0166-3615(94)90043-4
12. Wang, Y.S., Griffiths, B.J., Wilkie, B.A.: A novel system for coloured object recognition. Comput Ind **32**(1), 69–77 (1996). doi:10.1016/S0166-3615(96)00065-6
13. Lucas, S.M.: Real-time face recognition with the continuous *n*-tuple classifier. High performance architectures for real-time image processing (ref. no. 1998/197). IEE colloquium on 12 Feb 1998, pp. 11/1–11/7 doi:10.1049/ic:19980051 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=667494&isnumber=14675)
14. Hepplewhite, L., Stonhamm, T.J.: Texture classification using *n*-tuple pattern recognition. Pattern recognition. In: Proceedings of the 13th international conference on 25–29 Aug 1996, vol. 4, pp. 159–163. doi:10.1109/ICPR.1996.547253 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=547253&isnumber=11505) (1996)
15. França, H.L., da Silva, J.C.P., De Gregorio, M., Lengerke, O., Dutra, M.S., França, F.M.G.: Movement persuit control of an offshore automated platform via a RAM-based neural network. Control automation robotics and vision (ICARCV). In: 11th international conference on 7–10 Dec 2010, pp. 2437–2441. doi:10.1109/ICARCV.2010.5707913 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5707913&isnumber=5707203) (2010)
16. Bonato, V., Sanches, A., Fernandes, M., Cardoso, J., Simoes, E., Marques, E.: A real time gesture recognition system for mobile robots. In: international conference on informatics in control, automation and robotics, 25–28 August 2004, pp. 207–214. Setúbal, Portugal
17. Pattichis, C.S., Schizas, C.N., Sergiou, A., Schnorrenberg, F.: A hybrid neural network electromyographic system: incorporating the WISARD net. Neural networks. In: IEEE world congress on computational intelligence 1994. In: IEEE international conference on 27 Jun–2 Jul 1994, vol. 6, pp. 3478–3483. doi:10.1109/ICNN.1994.374894 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=374894&isnumber=8561) (1994)
18. Austin, J.: RAM-based neural networks. Prog Neural Process **9**, 252 (1998). (ISBN: 978-981-02-3253-5, ISSN: 2010-2895)
19. Rohwer, R.J.: Two bayesian treatments of the *n*-tuple recognition method. In: Fourth international conference on artificial neural networks, 26–28 Jun 1995, pp. 171–176. doi:10.1049/cp:19950549 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=497811&isnumber=10614) (1995)
20. Altera Corp.: Qsys interconnect (chapter of quartus II handbook 11.1). http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf (2011)
21. Mitchell, R.J., Bishop, J.M., Minchinton, P.R.: Optimising memory usage in *n*-tuple neural networks. Math Comput Simulation **40**(5–6), 549–563 (1996). doi:10.1016/0378-4754(95)00006-2

Author Biographies

Petr Burian is currently a Ph.D candidate at the University of West Bohemia in Pilsen, the Czech Republic. He received his M.Sc. degree in Electronics from the same university. He works as a seminar lecturer at the Department of Applied Electronics and Telecommunications and as a FPGA design engineer at the Regional Innovation Centre for Electrical Engineering. His main fields of activity are evolutionary design, evolvable hardware, image recognition and FPGA design.

Radek Holota is a lecturer at the University of West Bohemia in Pilsen, the Czech Republic. He received his M.Sc. and Ph.D. degrees in Electronics from the same university. The final project of his M.Sc. degree was defended at Brunel University in London under the supervision of Assoc. Prof. Bruce A. Wilkie in 2000. His current research interests include computer vision, pattern recognition, image processing and artificial intelligence.