J Sign Process Syst (2013) 71:57–73
DOI 10.1007/s11265-012-0681-7

Hardware-Software Implementation of a Sensor Network for City Traffic Monitoring Using the FPGA- and ASIC-Based Sensor Nodes

Marek Wójcikowski · Robert Żaglewski · Bogdan Pankiewicz · Miron Kłosowski · Stanisław Szczepański

Received: 5 August 2011 / Revised: 19 May 2012 / Accepted: 22 May 2012 / Published online: 27 June 2012

Abstract This paper presents a prototype sensor network for monitoring urban traffic. The sensor network node, equipped with a low-resolution camera, observes the street and detects moving objects. Object detection is based on the custom video segmentation algorithm, using dual background subtraction, edge detection and shadow detection, running on dedicated multi-processor SoC hardware. The number and the speed of the detected objects are transmitted using a low-power license-free radio transceiver to another neighboring node. All the nodes create a self-organized network, data are aggregated at the nodes and passed further to the nodes closer to data sinks. Finally, information about the traffic flow is collected from the sinks and visualized on a PC. The prototype sensor network node has been realized in two versions: FPGA and ASIC. The ASIC version consumes approximately 500 mW and it can be powered from a photovoltaic solar panel combined with a single cell Li-Po battery. The comparison of power consumption of both versions has also been made. Apart from collecting traffic data, the proposed sensor network can gather environmental data, such as the temperature, the acoustic noise or the intensity of the sunlight. The set of 26 prototype sensors has been mounted on street lamp-poles on streets and tested in real conditions.

Keywords Wireless sensor networks · Sensor systems · Object detection

Index Terms—wireless sensor networks, sensor systems, object detection

M. Wójcikowski (✉) · B. Pankiewicz · M. Kłosowski · S. Szczepański
Gdańsk University of Technology,
Gdańsk, POLAND
e-mail: wujek@ue.eti.pg.gda.pl

R. Żaglewski
Intel Shannon Ltd.,
Shannon, Ireland

1 Introduction

Sensor networks are very useful in collecting data from large areas. Many potential examples of sensor networks can be found in the literature [1–4]. The sensor network can also be effectively used for urban traffic. Traffic monitoring systems are very useful in providing management of the traffic flow and they can help in increasing the throughput and safety of the transportation and provide valuable information for planning of future road developments. Typically, detection of moving vehicles is done by inductive loops, passive infrared sensors, magnetometers, microphones, radars or microwave sensors [5–8]. Also, high-resolution cameras connected to the monitoring center using high-bandwidth cables or fiber optic links are often used but collecting data from remote sensors is usually very expensive as far as installation and usage are concerned. In recent years, an automatic video detection has been becoming more popular in the literature. There are many algorithms and systems for this purpose. Unfortunately, most of them work with high-resolution images and require significant computing power, which is not suitable for low-power sensor networks. The paper [9] contains a survey of various image detection algorithms, from simple background subtraction to optical flow and stereo vision methods. In [10],

the authors present the algorithm dedicated for existing street cameras, which uses background subtraction with the Bayesian Network Classifier. The background subtraction is also used in the algorithm described in [11, 12], where pixels are modeled as the mixture of Gaussians and the pixel model is learned using the EM algorithm. The mixture of Gaussian for low-level tracking is also proposed in [13], while high-level position estimation is done using the Kalman filter. The PC-based system described in [14] utilizes the probability-based foreground extraction, with color image processing and shadow removal. Two separate algorithms for day and night operation are proposed in [15]: the spatio-temporal analysis is used during daytime, the morphological analysis of headlights is used at night, the collected information is processed with the forward chaining rule production system. Another approach is described in [16], where the outlines of the detected cars are represented as quadratic B-spline curves. The 3-D model consisting of the edge segments is matched with 2-D image segments in [17]. In [18] an adaptive background estimation on image divided into non-overlapped blocks is described, with PC-based implementation capable of 15 fps image processing. In [19, 20], FPGA-based implementations are proposed, where some regions of interest are defined and analyzed in the observed image. The self-contained computer system running Linux with the attached camera is described in [21], where the authors demonstrate the application of simple background detection algorithm. Traffic detection based on frame differencing, background subtraction and virtual induction coils are presented in [22]. [23] contains camera detection systems evaluation and optimal camera placement considerations. Other video detection systems use statistical approach using principal component analysis and independent component analysis [24], neural network approach with multilayer feed-forward neural network [25], support vector machine [26] or symmetry-based vehicle detection [27, 28].

The motivation of the presented work was to propose the low cost method (in terms of the hardware and the maintenance costs) of measuring the city traffic. The set of small, low-power, autonomous devices (i.e. sensor network nodes), taken out-of-the-box and attached to the street lamp-poles, is able to evaluate the traffic using their built-in cameras and to transmit traffic data to the central computer, using the self-organized radio network (even ad-hoc installation is possible in emergency situations). Data collected by the network can be used for many purposes, such as: informing citizens by local radio station or internet, detecting extraordinary traffic events, guiding the emergency vehicles to achieve their goal more quickly or interacting with the traffic lights to improve the traffic flow, resulting in a decrease in traffic jams and environmental pollution. Compared to the standard vehicle detection method with the inductive loops, the use of small sensor network nodes has many benefits: simple installation, lack of wire connections, independent radio communication, possibility to estimate the speed, direction and size of the vehicles, capture and transmission of single images to the monitoring center.

In this paper, a sensor network for monitoring the traffic of the vehicles on the street of a city is described. The design target is a low-power sensor network node capable of estimating the traffic flow, due to the carefully developed video detection algorithms and proper hardware/software co-design. The node of the sensor network is designed to be installed on street lamp-poles to observe the scene from the location high above the road. Analysis of the video stream locally by each node of the sensor network significantly reduces the amount of data which needs to be transmitted over the radio, further decreasing the energy usage. The nodes are capable of detecting car traffic; additionally, single images from the nodes' cameras can be sent, informing the operator, for example, about snow on the street.

The proposed sensor network node (Fig. 1) uses a typical low-cost camera, thus the installation near the street lamp is mandatory to enable observation in the night.

The sensor network can be quickly and simply installed without the need for expensive cabling or permissions, since the nodes have a built-in, low-power radio working on license-free frequency band (see Fig. 2) and they can be powered from the solar panel.

The layout of the paper is as follows: in section II, the authors present the algorithm and realization of the image processing for moving object detection. Section III describes the protocol and organization of the radio communication network. The details about the realization of the prototype nodes are given in section IV. The results of the sensor network operation and conclusions are presented in sections V and VI, respectively

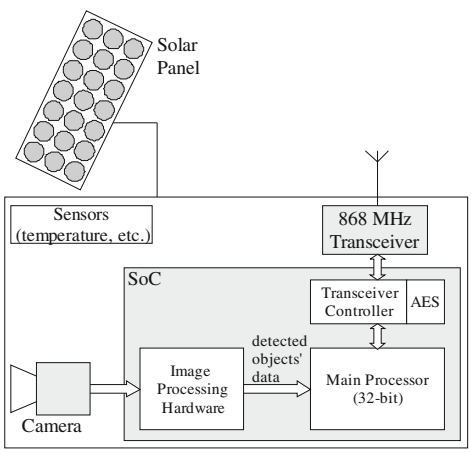


Figure 1 A conceptual diagram of the sensor network node.

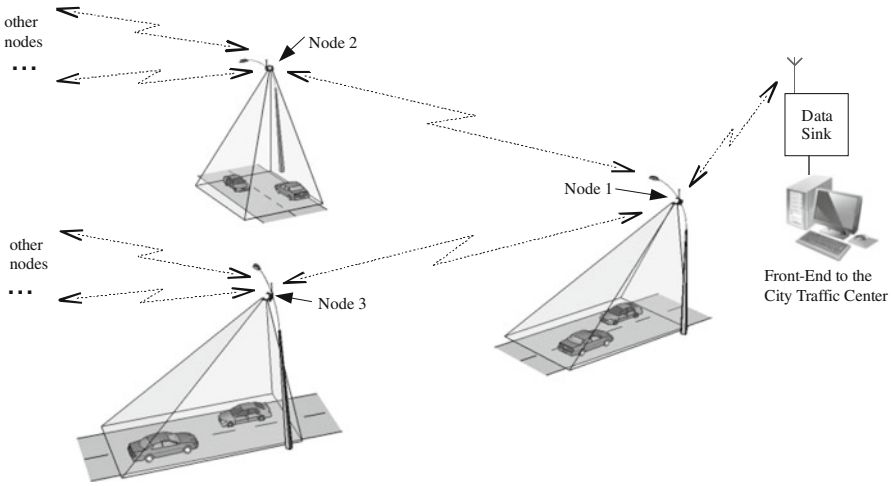


Figure 2 An overview of the sensor network application for collecting traffic data.

2 Moving Object Detection

Monitoring of the traffic with a camera is a very challenging task. It requires an image-processing algorithm capable of detecting cars under variable light conditions. Implementation of an image-processing algorithm in the sensor network node is further constrained by the limited resources of the node. The sensor network node is usually a low-power device using as simple hardware as possible to decrease the size and the power consumption. Due to this fact, a careful design of an image-processing algorithm which would fit into those limited hardware resources is very important. In the presented solution, a monochrome camera is used to reduce the complexity of the hardware by about 3 times at a cost of decreasing the segmentation sensitivity; additionally, the image resolution has been decreased to 128×128 pixels.

A. *Low–Level Image Processing*

In the presented implementation, the non-model-based approach for moving object detection with background subtraction is used, where each current frame from the camera is subtracted from the background image stored in the memory. Among the algorithms mentioned in section I, the background subtraction is the easiest to implement in

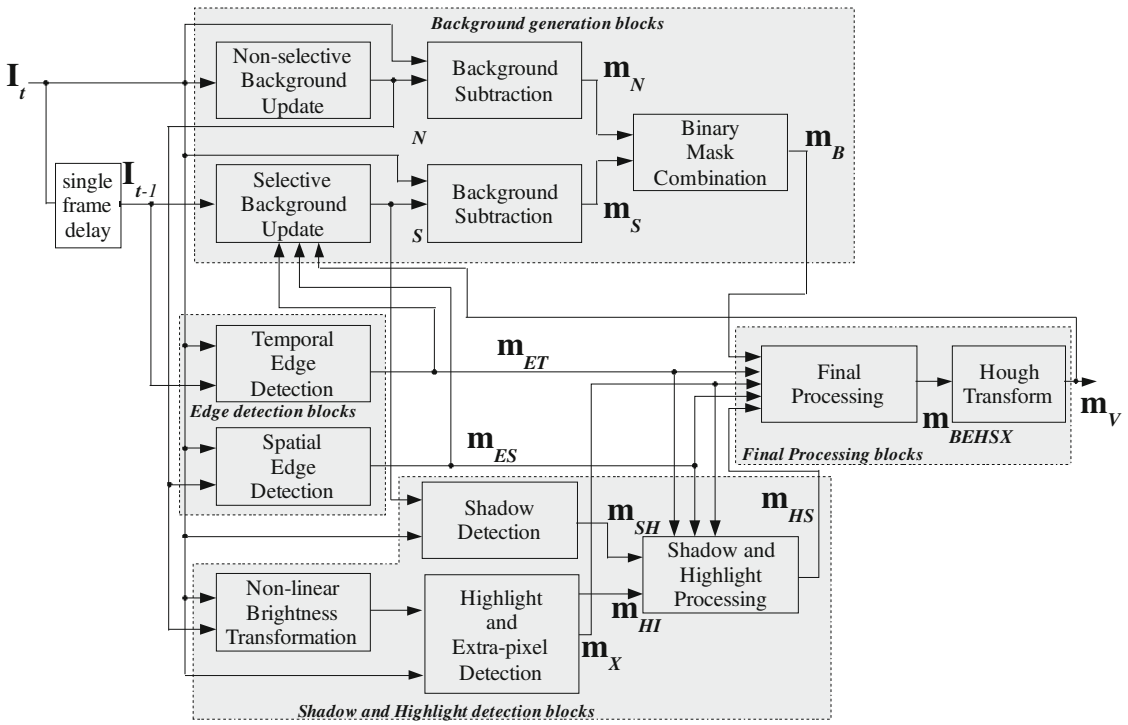


Figure 3 General diagram depicting the idea of the proposed image processing algorithm.

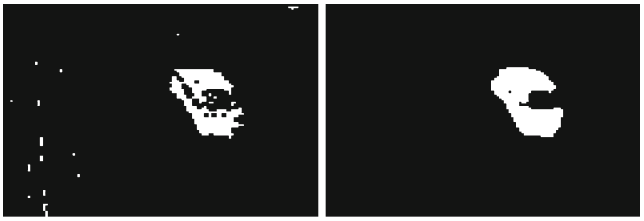


Figure 4 An example of the operation of the Hough transform block: **a**—image $\mathbf{m}_{BEHSX}$ before Hough block, **b**—image $\mathbf{m}_V$ after Hough block.

hardware, but it does not provide in its basic form the detection quality acceptable for the outdoor vehicle detection. The complete block diagram of the image-processing algorithm is shown in Fig. 3. The detailed description of the low-level image segmentation algorithm can be found in [29], where the earlier work of the authors is presented.

The authors decided to use two background models concurrently [30–33]: long-term with non-selective update and short-term with selective background update. In this way, the stopped objects are initially not included into the selective background, but after a while become a part of the non-selective background. After being part of the non-selective background, the stopped object is not detected any more, thus it can be quickly included into the selective background too.

Both background models assume single Gaussian distribution of pixel intensities for time t with the average brightness μ_t and the standard deviation σ_t updated using running mode, which can be very efficiently realized in hardware.

$$|I_t - \mu_t| > k \cdot \sigma_t \tag{1}$$

The detection results, using inequality (1), in the form of the masks: $\mathbf{m}_N$ and $\mathbf{m}_S$ from non-selective and selective background model, respectively, are combined into a single binary mask $\mathbf{m}_B$, using special combination of *and* and *or* operations, similarly as described in [32]. For the simplicity

Table 1 The assigned tasks for PPs.

Processor	Tasks
PP1	The non-selective background maintenance. Calculation of mask $\mathbf{m}_B$. Storing $\mathbf{I}_{t-1}$.
PP2	The selective background maintenance. Calculation of mask $\mathbf{m}_S$.
PP3	Calculation of masks $\mathbf{m}_{SH}$ and $\mathbf{m}_{ET}$.
PP4	Calculation of masks $\mathbf{m}_{ES}$, $\mathbf{m}_{HI}$ and $\mathbf{m}_X$.
PP5	Indexing of the detected objects (phase 1).
PP6	Indexing of the detected objects (phase 2). Generating image after indexation. Generation of the tables with objects' parameters.
PP7	Calculating the combination of the masks $\mathbf{m}_N$ and $\mathbf{m}_S$. Calculating the combination of the masks $\mathbf{m}_{ES}$ and $\mathbf{m}_{ET}$. Erosion of mask $\mathbf{m}_{SH}$.
PP8	Generating the final combination of masks. Erosion of the final mask $\mathbf{m}_{BEHSX}$.
PP9	Dilation of the final mask $\mathbf{m}_{BEHSX}$.

of the hardware, apart from the current pixel, instead of eight neighboring pixels, only the four previously analyzed pixels are used. To improve the segmentation quality, two edge detection blocks have been introduced: temporal edge detection and spatial edge detection, resulting in mask $\mathbf{m}_{ET}$ and $\mathbf{m}_{ES}$, respectively. To further increase the selectivity of the object detection, several additional blocks have been added, such as shadow detection block (mask $\mathbf{m}_{SH}$) or highlight detection block detecting highlights from the car lights in the night (mask $\mathbf{m}_{HI}$) and very dark pixels on a bright background (mask $\mathbf{m}_X$). The basic detection of shadows is done by comparing the decrease in brightness [34]. The Final Processing block from Fig. 3 is responsible for basic morphological operations (erosion followed by dilation, using 2×2 rectangular structuring element) to condition the final mask $\mathbf{m}_{BEHSX}$. The final detection result is

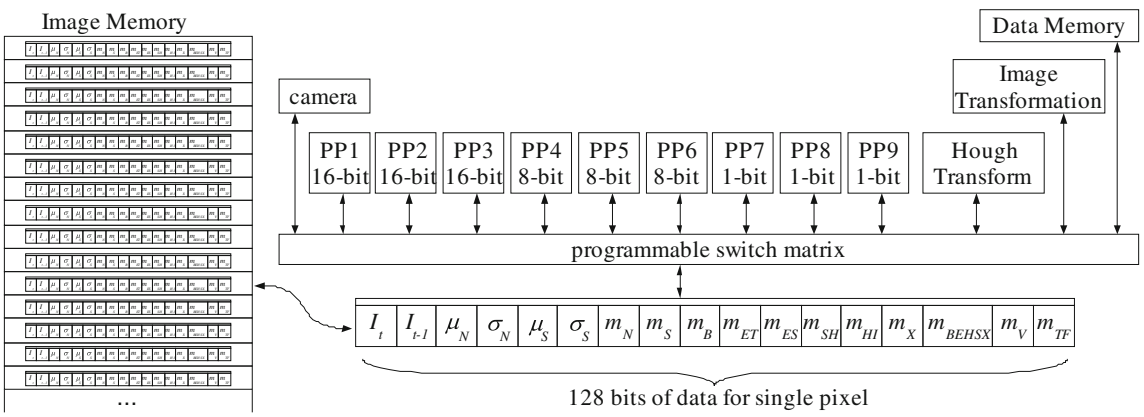


Figure 5 Block diagram of the image-processing hardware.

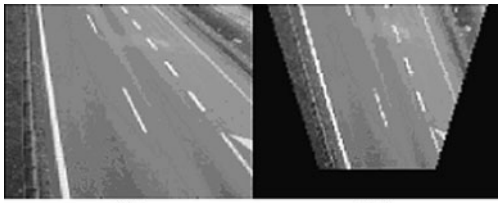


Figure 6 An example of the input image before (**a**) and after (**b**) the geometrical transformation canceling the perspective.

processed by the Hough transform based block, which helps to obtain convex, filled blobs representing the detected objects. The Hough transform with the rectangular structuring element improves the shape of the blobs (Fig. 4). The mask $\mathbf{m}_V$ is the final detected vehicle mask image of elements equal to 0 or 1, where 1 denotes the pixels belonging to the detected moving objects.

All the image-processing operations shown in Fig. 3 have been selected and adapted for efficient hardware implementation. The details of the pipelined implementation of the algorithm from Fig. 3, as well as the segmentation results, can be found in [29]. In this paper, instead of typical pipelined hardware, a set of concurrently working, dedicated Pixel Processors (PP) has been developed to enable the modifications of the algorithm in the ASIC implementation. The PPs have been designed as scalable modules in VHDL: 16-bit, 8-bit and 1-bit versions are provided. The system contains nine PPs, as shown in Fig. 5. The processors are connected to the common data bus via a programmable switch matrix, where the bus contains data of the currently processed pixel. Each processor can read any of 128 data bits, but it can write only to the exclusively assigned selection of bits, except from 1-bit processors which can write to each bit of the bus. The 1-bit processors have a built-in memory for caching the previously analyzed pixels - data from this memory are used to perform simplified morphological operations that require information about the neighboring pixels, such as erosion or dilation. The processing tasks have been divided among the processors, as listed in Table 1.

The camera can view the street from various angles. To estimate the speed and the size of the moving objects, the image must be geometrically transformed. An example of the transformation is shown in Fig. 6, presenting the input image before and after the transformation. In the real system, the mask $\mathbf{m}_V$ is transformed into the mask $\mathbf{m}_{TF}$. The image transformation is realized by the hardware block equipped with the memory block containing the image-mapping coordinates.

The detected blobs are labeled and their basic parameters are calculated, such as: object's boundaries, the center of the object and the area in pixels. The labeling and calculating of the parameters of the objects is done during pixel by pixel revision of the image by the selected PPs.

B. *High–Level Tracking*

The detection results from PPs in the form of the table containing the basic parameters of the blobs are passed to the main 32-bit processor of the system, where the objects are traced, classified and measured. The detected blobs are usually noisy and inaccurate, i.e. they can change shape, disappear, split or join with the other blobs. The main processor services the radio network and it also has to extract as much information as possible to infer the moving cars, but the limited

Table 2 The attributes assigned to each detected blob *B*.

Attribute	Description
t_{0B}	The time of creation of the blob *B*.
x_{0B}, y_{0B}	The coordinates of the center of the rectangle representing the blob *B* at the time t_{0B}.
x_B, y_B	The coordinates of the center of the rectangle representing the blob *B* at the current time.
X_B, Y_B	The current size of the rectangle representing the blob *B*.
$\overline{X}_B, \overline{Y}_B$	The average size of the rectangle representing the blob *B* since t_{0B}, updated at every frame.
$\overline{V}_B$	The average velocity of the blob *B* since t_{0B}, updated at every frame.
$\overline{\alpha}_B$	The average movement direction of the blob *B* since t_{0B}, updated at every frame.
P_B	The number of pixels of the real blob represented by the rectangle *B*.
q_B	The quality of the movement path of the blob *B*, calculated according to the equations: $q_{B,t_{0B}} = 0$ $q_{B,t} = q_{B,t-1} + 2Q - \left\|x_{B,t} - x_{B,t-1} - \overline{\Delta X}_{B,t-1}\right\| - \left\|y_{B,t} - y_{B,t-1} - \overline{\Delta Y}_{B,t-1}\right\|$ where: $\overline{\Delta X}_{B,t}$, $\overline{\Delta Y}_{B,t}$—the average differences between the traveled distance by the blob *B* at two consecutive time moments *t*-1 and *t* for the direction *x* and *y*, respectively; *Q*—acceptable change of $\overline{\Delta X}_{B,t}$, $\overline{\Delta Y}_{B,t}$ for consecutive frames, assumed value of Q=2. In further considerations, the blob's and model's time indexes will be omitted, unless necessary.

Table 3 The attributes assigned to each model *M*.

Attribute	Description
t_{0M}	The time of creation of the blob replaced by the model *M*.
x_{0M}, y_{0M}	The coordinates of the center of the rectangle *M* at the time t_{0M}.
x_M, y_M	The current coordinates of the center of the rectangle *M*, calculated as: $x_{M,t} = x_{M,t-1} + V_M\, cos(\alpha_M)$ $y_{M,t} = y_{M,t-1} + V_M\, sin(\alpha_M)$
X_M, Y_M	The size of the rectangle *M*.
V_M	The velocity of the model *M*.
α_M	The movement direction of the model *M*.
T_M	Time to live for the model *M*. At every time step, the value of T_M is decreased by 1. If T_M=0, the model is deleted (applicable only for the *short-term* model, as explained in the text).
S_M	The value informing about the size of the model, relative to the size of a typical car: 1=*small* when $w<0.5\cdot W_t$ 2=*normal* when $0.5\cdot W_t \le w \le 3\cdot W_t$ 3=*big* when $w>3\cdot W_t$ where: w is calculated as the width of car cast in the direction of its movement; W_t is the width of a typical car (2 m).

computing resources did not allow the use of the Kalman filter approach in real time, therefore a simplified method has been used. For that purpose, a set of *if-then* heuristic rules has been applied, working on simple rectangular models of the blobs. For every picture frame, each detected blob is always modeled as the rectangle *B* (denoted as *blob B* in the further text) with the assigned attributes shown in Table 2.

For simplicity, all the average values are calculated as running mean and the movement path is approximated with a straight line. The overlapping of the blobs in the current and the previous frame is detected by the PP and both such blobs are treated as representing the same moving object, providing the continuity of the existence of the blob's rectangle *B*. If the blob *B* disappears, the virtual model *M* is created, continuing the movement of the blob *B*, using the last known values for size, direction and speed of the disappeared blob *B*. The attributes of the model *M* are listed in Table 3.

A simple example of blob processing is shown in Fig. 7, where at the time *t*, a new blob has been detected and marked as B_t (the speed $\overline{V}B,t$ and the direction $\overline{\alpha}B,t$ are unknown at this time). At the next time step $t+1$, the blob changes its position and shape, but, due to the overlapping with the blob from time *t*, it is treated as the same moving blob, so the attributes of B_t are copied to B_{t+1}, also the first approximation of the speed of the blob $\overline{V}_{B,t+1}$ can be calculated, as well as $\overline{\alpha}_{B,t+1}$. At the time step $t+2$, the blob disappears, but its existence is represented by the newly created model M_{t+2} with the following attributes:

$$\begin{aligned} t_{0M} &= t_{0B} \\ (X_M, Y_M) &= \left(\overline{X}_{B,t+1}, \overline{Y}_{B,t+1}\right) \\ V_M &= \overline{V}_{B,t+1},\ \alpha_M = \overline{\alpha}_{B,t+1} \end{aligned} \tag{2}$$

When the size of the blob *B* changes (which could mean joining with another blob or splitting), new objects are created: a new model *M*, as a continuation of the *B*, and the new blob (or blobs in the case of splitting) representing the current blob. As an example, two moving blobs are

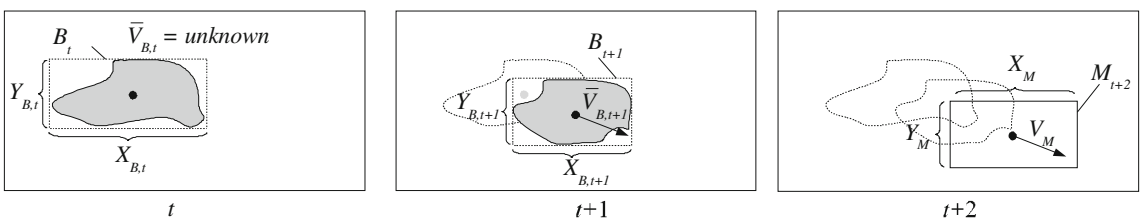


Figure 7 An example of transforming the blob *B* into the model *M*. At the time *t* a new blob has been detected and marked as B_t. Next, at $t+1$, the new blob overlaps with the blob from time *t*, so it is considered as the continuation of B_t. At the time $t+2$ the blob disappears, but the model M_{t+2} continues the movement of *B*.

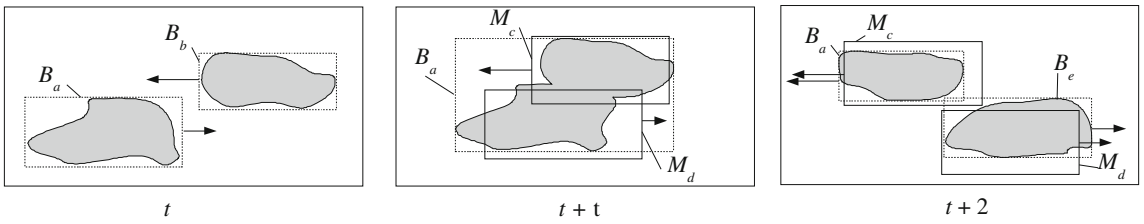


Figure 8 An example of blobs' tracking after blob joining.

shown in Fig. 8, represented by B_a and B_b at the time t. In the next frame at the time $t+1$, both blobs have joined together and from this moment B_a is representing the connected blobs. However, due to the size change of the blob B_a and disappearing of the blob B_b, the blobs B_a and B_b from time t have been converted into the models M_d and M_c, respectively. Later, at the time $t+2$, the blob B_a splits again into two blobs B_a and B_b; the overlapping models M_c and M_d are then absorbed by the blobs B_a and B_b, respectively, providing the continuity of their movement.

Having a set of blobs B and models M for each frame, the main processor executes the heuristic rules to extract data about moving vehicles. For example, if any blob and model are close together and have similar speed and direction of movement, depending on the situation, the model can be deleted or its position can be centered with the blob. Small models that are close to a bigger one and have similar speed and direction are deleted. The models that overlap with each other and move in the similar direction and with similar speed are consolidated. The most important rules concerning the blobs and models are summarized in Table 4 and below:

High level tracking rules:

rule #1 **If** *BlobSizeChange*($B_{a,t}$, $B_{a,t+1}$)≥50 % **then** $M_{a,t+1}$=*NewModelFromBlob*($B_{a,t}$)
$B_{b,t+1}$=*NewBlob*($B_{a,t+1}$)
DeleteBlob($B_{a,t+1}$)

rule #2 **If** *BlobDisappeared*(B_t) **then** M_{t+1}=*NewModelFromBlob*(B_t)

rule #3 **If** *Distance*(*B, M*)<D_{min} **and** $\left|V_M - \overline{V}_B\right| < V_{min}$ **and** *AngleDifference*($\overline{\alpha}_B, \alpha_M$)<$\alpha_{min}$ **and** $t_{0B} < t_{0M}$ **then** *BlobAbsorbsModel*(*B, M*).

rule #4 **If** *Overlap*(*B, M*)>50 % **and** $\overline{V}_B = 0$ **and** V_M=0 **then** *DeleteModel*(*M*)

rule #5 **If** *SpeedDifference*(V_{M_a},V_{M_b})<50 % **and** *AngleDifference*(α_{M_a}, α_{M_b})<α_{min} **and** *Distance*(M_a, M_b)<D_{min} **and** (S_{M_a} = small **or** S_{M_b} = small) **then** *ConsolidateModels* (M_a, M_b)

rule #6 **If** *SpeedDifference*(V_{M_a},V_{M_b})<50 % **and** *AngleDifference*(α_{M_a}, α_{M_b})<α_{min} **and** *Overlap*(M_a, M_b)<min (t − t_{0M_a}, 100) **then** *ConsolidateModels*(M_a, M_b)

rule #7 **If** *SpeedDifference*(V_{M_a},V_{M_b})<50 % **and** *AngleDifference*(α_{M_a}, α_{M_b})<α_{min} **and** (*Overlap*(M_a, M_b)<min (t − t_{0M_a}, 100) **or** *Overlap*(M_a, M_b)<min(*t* − t_{0M_b}, 100)) **then** *ConsolidateModels*(M_b, M_a)

Rules used for conversion of blob *B* into model *M*:

rule #8 **If** *NewModelFromBlob*(*B*) **and** $\overline{V}_B$ and $\overline{\alpha}_B$ are not known **then** create a new *short-term* model *M* using the parameters of *B* and $T_M = T_{MIN}$.

Short-term model is supposed to live shortly and it will disappear after T_{MIN} frames. It is used to track "blinking" moving blobs, for which it is difficult to estimate $\overline{V}_B$ and $\overline{\alpha}_B$ due to their short existence on the screen. T_{MIN}=6 has been set experimentally for frame rate 32 fps.

rule #9 **If** *NewModelFromBlob*(*B*) **and** $\overline{V}_B$ and $\overline{\alpha}_B$ are known **and** $\overline{V}_B < V_{\mathrm{MAX}}$ **and not** (*is_human(B)*) **and** $q_B > Q_{MIN}$ **then** create new model *M* using the parameters of *B*.

where:

V_{MAX}	the maximum allowed speed for the vehicle
D_{min}	the minimum allowed distance (assumed 4 pixels)
α_{min}	the minimum allowed angle, assumed α_{min}=45°

Table 4 Overview of situations when model is created and deleted. The details are presented in form of the rules in the text.

Action	Situation
Model is created	A blob disappears or changes its shape.
Model is deleted	A blob and a model are close together, move in similar direction with similar speed.
	A blob and a model overlap and their position is constant.
	Two models are close together, they move in similar direction with similar speed and one of them is small.
	Two models overlap and they move in similar direction with similar speed.

Table 5 The bit lengths of the transmission headers in different protocols.

Protocol/remarks	Description	Length [bits]
802.11	MAC header	240
802.15.4, (PHY+MAC header)	Beacon:	80–176
	Data:	96–192
	Ack	32
Protocol used in this paper. Header already contains its own 16-bit CRC. Each header (including Ack) contains additional data about the state of the transmitting node: battery state, external power supply, data buffer level.	Beacon	64
	Data	64
	Ack	64

The function *is_human()* is a simple and rough estimate, if the detected blob could be a person walking along the street (assuming that the camera is situated high above the road) and is calculated as:

$$\text{is\_human}(B) = \begin{cases} true & \text{when } \overline{V}_B < V_{MIN} \wedge 3 < \overline{Y}_B/\overline{X}_B < 6 \wedge P_{HMIN} < P_B < P_{HMAX} \\ false & \text{otherwise} \end{cases} \quad (3)$$

where:

V_{MIN} the maximum speed for walking human

P_{HMIN}, P_{HMAX} the constant defining the minimum and maximum number of pixels, respectively, for the blob recognized as a person. The values of P_{HMIN} and P_{HMAX} are calculated to represent the area of 0.75 m^2 and 3 m^2, respectively, at the image after geometrical transformation.

Q_{MIN} the constant defining the minimum allowed object's movement path quality q_B.

The quality q_B of the movement path of the blob B is calculated according to the equations:

$$\begin{aligned} q_{B,t_{0B}} &= 0 \\ q_{B,t} &= q_{B,t-1} + 2Q - \left|x_{B,t} - x_{B,t-1} - \overline{\Delta X}_{B,t-1}\right| - \left|y_{B,t} - y_{B,t-1} - \overline{\Delta Y}_{B,t-1}\right| \end{aligned} \quad (4)$$

where:

$\overline{\Delta X}_{B,t}$, $\overline{\Delta Y}_{B,T}$ the average differences between the traveled distance by the blob B at two consecutive time moments t-1 and t for the direction x and y, respectively.

Q acceptable change of $\overline{\Delta X}_{B,t}$, $\overline{\Delta Y}_{B,T}$ for consecutive frames, assumed value of Q=2.

$$BlobSizeChange(B_t, B_{t+1}) = max\left(\frac{|X_{B,t} - X_{B,t+1}|}{max(X_{B,t}, X_{B,t}+1)}, \frac{|Y_{B,t} - Y_{B,t+1}|}{max(Y_{B,t}, Y_{B,t}+1)}\right) \quad (5)$$

Distance (*B, M*) the shortest distance between the outlines of two rectangles: B and M. If the rectangles overlap, the distance is 0.

$$SpeedDifference(V_1, V_2) = \frac{|V_1 - V_2|}{max(V_1, V_2)} 100\% \quad (6)$$

AngleDifference (α_1, α_2) the difference between two angles α_1 and α_2.

Overlap (*B, M*) returns % of overlapping area between the rectangles B and M with respect to the area of B.

BlobAbsorbs Model(*B, M*) the attributes' values of the model M are copied to the blob B and the model M is deleted.

Consolidate Models (M_a, M_b) the models M_a and M_b are compared and the model which exists shorter is deleted, the remaining model is updated with the speed and moving direction as the average of the respective values from M_a and M_b.

Table 6 The comparison of the most important parameters of 802.15.4 and the protocol presented in this paper.

Description	802.15.4	The protocol used in this the presented sensor network node
Beacon interval	approx. 15 ms–4 min.	100 s
Access to the transmission medium	Contention based and contention free periods	Only contention free.
Number of channels	1 at 868 MHz 10 at 902–928 MHz 16 at 2400–2483.5 MHz	1 at 868.5 MHz
Data rate (at 868 MHz band)	20 kbps	38.4 kbps
Preamble length	32 bits	10 bits

Table 7 The general structure of the frame.

Field:	HEADER (64-bits)	PAYLOAD (0…1920 bits)	CRC (32 bits)
Data:	frame type, source and destination address, frame length, header's CRC	beacon data, 128-bit data packets (PKT)	CRC of the payload

The rules have been presented in a simplified form to aid the readability, in fact more conditions are checked, i.e. to protect from processing invalid data, division by 0, etc.

The moving models which leave the picture frame are filtered and only those considered as reliable are counted. The model is considered reliable if it comes from a blob that has been moving smoothly for some time (depending on the quality q_B of the movement path) or it has been often overlapping with blobs moving with similar size, speed and direction. The traffic flow data are constantly updated, the moving vehicles are classified according to their speed and direction—this information is periodically transmitted by the radio to a nearby node which is closer to the data sink.

3 Data Transfer in the Sensor Network

To avoid the overheads of the standard wireless communication protocols such as 802.11 or 802.15.4, the authors decided to develop and implement the proprietary protocol optimized for this application. The proposed protocol has shorter lengths of the headers (Table 5), moreover, the header already contains information about the state of the node (battery state, data buffer occupancy), which is important for neighboring nodes in routing the traffic. Also, data length is suited to carry only important information on car traffic. The simple comparison of the proposed protocol to the standard 802.15.4 is shown in Table 6. The sink nodes have a simple serial interface with text commands, so connecting the network to a PC is very easy.

The access to the transmission medium is based on decentralized Time Division Multiple Access (TDMA). Each node is equipped with a radio transceiver module using ISM 869.5 MHz band utilizing data rate of 38.4 kbps with Manchester encoding. The nodes' transmitters have the power of 500 mW (transmitter duty cycle <10 %), which helps in providing longer communication range in the harsh environment. The transmitter's power can be remotely changed by the software to the values 40 mW, 125 mW, 250 mW and 500 mW, if needed. The dedicated operating system of the node, running on the main 32-bit on-chip processor, controls all the operations of the node and collects data from the image-processing part and the PP processors.

A. *Transmission Time t_{tx}*

Each node has its own unique address and a table for storing data about its neighbors. All the nodes use the same period T_t of the transmission schedule, basing on their local clocks. Each node selects its own transmission start time t_{tx} in the TDMA transmission period T_t, providing:

$$\forall_i \left| t_{tx,i} - t_{tx} \right| > T_{MIN} \tag{7}$$

where $t_{tx,i}$ is the transmission start time of the other previously discovered node i (this can be also an indirect neighbor, i.e. the neighbor of the neighbor), T_{MIN} is a constant defined globally for the network, providing the required time distance between the transmissions. The times t_{tx} and $t_{tx,i}$ are relative to the node's local time source. The time distances of all the transmission start times $t_{tx,i}$ of the neighboring nodes are continuously monitored. When a node detects that among the neighboring nodes the condition (7) is not satisfied, it asks the problematic node to change its t_{tx}.

Table 8 The structure of the single 128-bit payload packet PKT.

Field:	PKT_ID (4-bits) (type of the packet)	PKT_ADDR (12 bits)	PKT_REALTIME (32 bits)	PKT_DATA (64-bits) (content depends on the PKT_ID)	PKT_CRC (16 bits)
Data:	PKTID_DATA1 (basic car traffic info)	Address of the source node	Time stamp	The number and the average speed of the detected cars in the last minute for each of four directions (N, E, W, S).	CRC of the payload
	PKTID_DATA2 (environmental data)			Ambient temperature, sun intensity, etc.	
	PKTID_DATA3 (node working parameters)			Node's battery state, number of detected neighbors, IDs of the first eight neighbors.	
	PKTID_DATA4 (camera image)			64 bits of camera image data.	

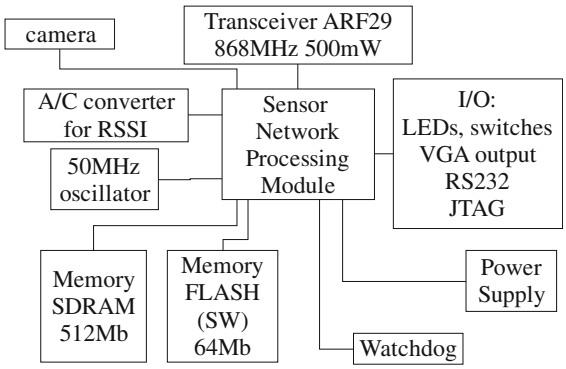


Figure 9 Block diagram of the prototype node.

B. *Beacon*

The beacon contains basic information about the node, such as the node's address, the distance of the node from the sink (measured in transmission hops), the node's battery condition, the node's data buffer occupancy and data about the node's neighbors. Beacon is transmitted at t_{tx} every N_B data transmission periods, typical value of N_B is from 3 to 10. Beacons are also used for establishing data links between the nodes. Each node infers from its neighbors' beacons the existence of the indirect neighboring nodes, which helps to prevent the hidden terminal problem [2].

C. *Startup and Discovery of the Neighbors*

At the startup, each node enters the discovery mode, when it continuously listens to the transmitted beacons, identifies neighbors and saves neighbors' transmission times $t_{tx,i}$. To find new nodes and accommodate the network topology changes, the discovery mode is periodically repeated. This is similar to the procedure used in S-MAC protocol [35].

There are four types of frames that can be transmitted through the network: beacon, data, acknowledge and configuration packets. All the frames have a 64-bit header of a similar structure. The beacon frame may contain information

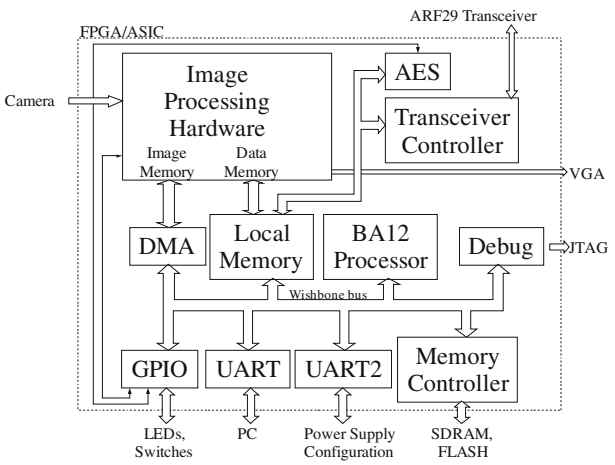


Figure 10 Block diagram of the Sensor Network Processing Module.

Table 9 The resources used in the FPGA implementation using Xilinx Virtex-4 XC4VLX60.

Resource	Total used
Flip Flops	10 325
LUTs (used as logic)	30 784
Total number of LUTs	31 676
Occupied Slices	19 220
18 Kb Block RAMs	157

for the neighboring nodes needed for organizing the network (Table 7). The data frame carries the encrypted 128-bit packets (PKT) with car traffic information (Table 8), which can be aggregated from several nodes and transmitted together in a single frame. The acknowledge frame, transmitted just after the successful reception of the data frame, is used to confirm the reception of data, so the data transmitting node can free its buffer.

Each node switches on its receiver at the time $t_{tx,i}$ to listen to the possible transmission of the i-th neighbor. If the receiving node is not an addressee of the data transmission, it switches off its receiver just after receiving the header, in order to avoid overhearing.

Car traffic data in the sensor network are regularly transmitted from the nodes to the data sinks using data frames. To provide the ability of sending information in the reverse direction, i.e. from the sink to distributed nodes, the configuration frame is used. The configuration frame is transmitted without acknowledgement. Each node, after receiving a new configuration packet, simply retransmits it several times. This simple mechanism causes a heavy load of radio links but it is used rarely and only for service purposes, such as configuring remote nodes' parameters, updating the firmware (FPGA nodes only) or the software.

Each data packet (PKT) contains the time stamp coming from the node's real clock. The time stamp is used for checking the validity of data at the host computer. The sensor network utilizes global real clock synchronization scheme. The sync packets are periodically generated by the main sink node and distributed through the network by each node using the beacons.

D. *Data Protection*

Each packet header is protected with 16-bit CRC, so the receiver can quickly verify the integrity of the header and the addressee of the packet without receiving the whole packet. Additionally, the payload has its own 32-bit CRC.

The header is transmitted using plain text, while the payload (PKT_DATA) is encrypted using AES with symmetric key. The hardware encryption, compared to the software encryption, has better energy efficiency measured in

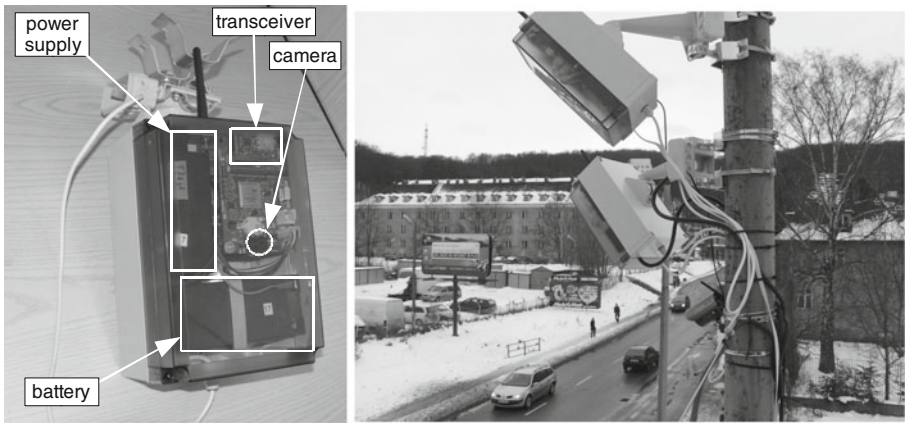


Figure 11 Prototype sensor network node with FPGA, **a**—picture of the node, **b**—two test nodes installed on a street lamp-pole.

energy/bit, thus each node has the hardware encryption block. Decryption is used occasionally by the nodes (except the sink nodes), so it has been implemented in the software.

The 3 prioritized AES keys have been used: KEY1 and KEY2 are the same for all the nodes, the KEY3 is different for each node. During regular transmission, the KEY1 is used. To change the KEY1, the KEY2 is required. At a regular time interval, the change of the KEY1 (protected by KEY2) is commanded by the sink node. The KEY3 enables changes to the KEY1 and KEY2 and it should be used in case of a node's takeover.

4 Hardware Realization of the Sensor Network

A. *Block Diagram of the Sensor Network Nodes*

The block diagram of the node's hardware, common for FPGA and ASIC prototype, is presented in Fig. 9. The main part of the hardware has been integrated into a Sensor Network Processing Module (SNPM), containing the custom microelectronic system with 32-bit processor BA12 from Beyond Semiconductor (the same class as Arm's ARM9™) and the peripherals connected to the Wishbone bus. The hardware moving object detection system, described in section II, has also been integrated, together with the additional hardware blocks providing quick AES encryption and control of the low level operations of the transceiver (Fig. 10).

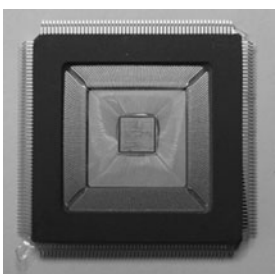


Figure 12 A photo of the manufactured ASIC realized in 130 nm UMC process packed in CQFP208 package.

B. *Hardware Implementation*

The first prototype sensor network node has been developed using Xilinx's XC4VLX60 FPGA. The main board contains all the elements from Fig. 10 with the camera MT9V111 from Micron and the transceiver ARF29 from Adeunis. The power supply has been implemented on additional boards using 12 V 6 Ah gel cell sealed lead acid battery. The resources used in FPGA implementation are listed in Table 9 and the pictures of the prototypes with the FPGA are shown in Fig. 11.

After the successful startup with the FPGA, the ASIC has been designed in 130 nm UMC CMOS process and manufactured through Europractice (Fig. 12), using the RTL code from the FPGA prototype. Both FPGA and ASIC provide almost the same functionality. The design of ASIC has been made using Cadence SoC software with Faraday L130FSG (LP and HS) library of digital gates. The code consisting of ~95 000 lines of VHDL and Verilog has been synthesized using Cadence RTL Compiler with clock gating optimization. DFT flip-flops and JTAG controller have been added to enable future tests. For the implementation, Cadence SOC Encounter GXL 6.2 has been used, with crosstalk and signal integrity analysis and power supply analysis (electromigration, IR drop). The parameters of the designed ASIC are listed in Table 10. Comparing Tables 9 and 10, a large difference in the number of Flip Flops can be observed. The reason is that small memories and shift registers in FPGA are implemented using LUTs, while ASIC

Table 10 The resources used in ASIC implementation using Faraday library and UMC 130 nm CMOS process.

Resource	Total used
Flip Flops	18 478
Primitives (gates, buffers, flip flops, etc.)	549 062
Chip area	25 mm^2
Memory blocks	75
Memory bits	1 661 952

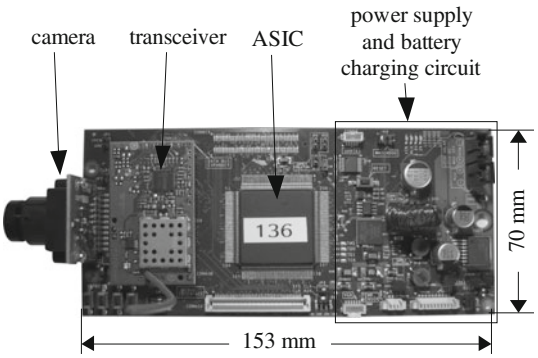


Figure 13 Photo of the ASIC version of the node.

implementation uses FFs for that purpose. Moreover, in the FPGA all the memory is composed of 18Kbit block memories, resulting in wasted bits. Each memory block used in ASIC is designed to exactly fit the required size.

The manufactured ASIC has been used to build the low-power version of the node shown in Fig. 13, which works with Li-Ion 3.7 V 3.5 Ah single cell battery and can be supplied by a solar panel of area of 0.5 m^2 (50 W peak power).

The set of the nodes has been installed on the street lamp-poles on several streets, as shown in Fig. 14. The detailed plan of the deployment is presented in the next section of the paper. The nodes have been using their batteries during the day; at night the batteries have been charging from the lamps' power supply. The ASIC node consumes less power than the FPGA counterpart and therefore it is capable of working with a solar panel, instead of using the lamp's power supply.

5 Simulation and Test Results

A. *Object Detection*

The vehicles detected by the system have been verified on-line by the human operator. The road selected for traffic detection tests had various traffic conditions: from traffic jams to speeding vehicles. The sensor network node installed above the road was detecting and counting passing vehicles. Concurrently, the human operator was manually counting the cars. The comparison of the detection for 100 cars for various conditions has been presented in Table 11. More results, including the object segmentation simulations and videos, are available on-line at http://www.ue.eti.pg.gda.pl/sn.

As can be seen from Table 11, the image detection system recognized 63–93 % of the moving vehicles registered by the human. There were typically 1–9 % additional false detections of non-existent vehicles. During the sunny day, the most important problem are the shadows, resulting in joining the blobs from different cars. The basic shadow detection used in this system is not able to detect all the shadows. On a cloudy day, the most errors come from the dark cars of gray level similar to the color of the road. In this situation, the edge detection blocks are very helpful in increasing the detected pixel rate. The detection quality is significantly lower in the night, where mostly the car lights are detected. The achieved accuracy is the result of the compromises made during the design, such as a processing of monochrome and low resolution images and 4-bit representation of the pixel values, to obtain a low-power operation and simpler hardware. The detection rates are satisfactory for the collecting of the statistical data on average traffic flow and getting the overall general picture of the traffic. However, the authors feel that improvements in accuracy should be the next step in the further development of the presented idea.

B. *Network Data Transmission*

The sensor network has been simulated to estimate the traffic parameters. The proposed sensor network is based on non-standard communication protocol and the use of popular simulation packages such as *ns2* would require the creation of the exact communication model of the node. Instead of this,

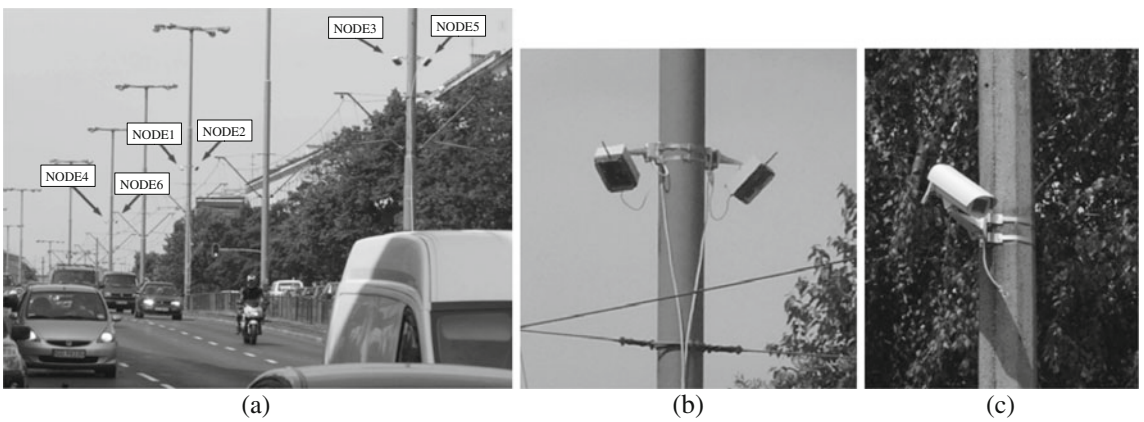
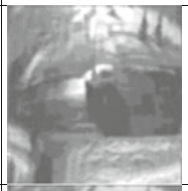
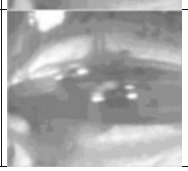


Figure 14 Photographs of the test nodes installed on the street lamp-poles, **a**—set of nodes along the street, **b**—two nodes with FPGA, **c**—the node with ASIC from Fig. 13 installed in a typical housing of an industrial security camera.

Table 11 Evaluation of vehicle detection rate for the sensor network node for various scenes. As the reference, 100 cars were counted by the human operator in each test.

Scene	Frame sample	% of vehicles detected correctly by the sensor network node	% of vehicles falsely overdetected by the sensor network node
Day, strong sun		93	8
Cloudy day		83	9
Night		63	1

the authors decided to embed the exact copy of the node's software into the custom simulator written in C++. In this way there is no need to create the separate simulation model of the node and the simulation results are very close to reality.

In the simulator, ideal radio links have been assumed, so all transmissions are successful. Assuming that each node regularly generates on average 9 bps of data stream containing information about the car traffic and basic node's information such as battery condition or node's neighbors, the maximal throughput for the data packets at the single radio link is 620 bps. Other parameters are listed in Table. 12.

Two basic network configurations of Fig. 15 have been simulated. For the network configuration shown in Fig. 15 (a), the delay and radio activity have been analyzed. Both source nodes were generating the test messages consisting of single 128-bit packet. The total average delay of packets was 2.7 s, when the source nodes generate packets every t_{gen}=0.5–2 s (Fig. 16). The activity of the radio receiver and transmitter is shown in Fig. 17(a) and (b), respectively. The

Table 12 Parameters of the proposed radio protocol.

Parameter	Value
Typical data stream generated by the node	9 bps
Maximal throughput—raw frames at single link (node to node)	651 bps
Maximal throughput—data packets at single link (node to node)	620 bps
Node's maximal number of neighbors	17

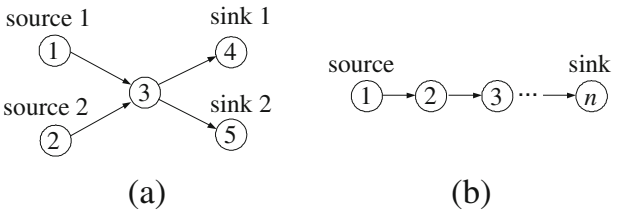


Figure 15 Test configurations of sensor network.

radio activity is highest for the middle node, as it has the largest number of neighbors. The average delay from the node 1 to the node n in the network configuration from Fig 15(b) is shown in Fig. 18. As can be seen, the average delay of packets per single hop in this configuration is almost constant and equal to approx. 1 s.

The effective radio range of the transceivers installed in real environment was about 900 m at the direct visibility of the antennas. The direct visibility can be easily achieved when the nodes are mounted on the street lamp-poles. After the installation of the nodes, the network organized itself and was correctly transmitting data to the sink, in the same way as during the simulation, except for sporadically occurring radio interference. The nodes have large data buffers for collecting unsent data (for at least 30 minutes); in the case of transmission problems, data is retransmitted at a later time. All the transmission times have been verified in practice and conform to the simulations. Apart from regular data about the car traffic, the operator can ask a node to send a single frame of the picture (Fig. 19), and uploading new firmware or software to the selected node, or all the nodes, is also possible.

C. *Power Consumption*

The comparison of the power consumption of FPGA and ASIC prototypes is shown in Fig. 20. As could be expected, the main power savings are at the core in ASIC version. The other blocks (SDRAM, camera, I/O) have also slightly lower power consumption in ASIC prototype, but it is

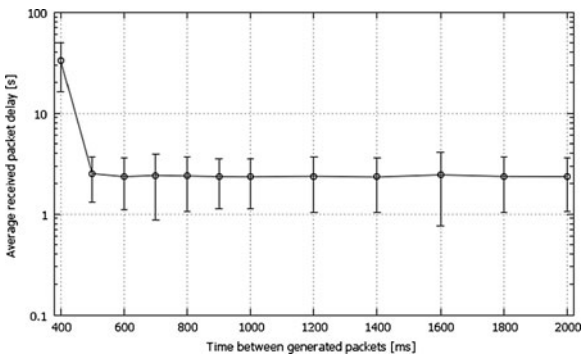


Figure 16 Simulation results of average delay of packets for the network configuration from Fig. 15(a).

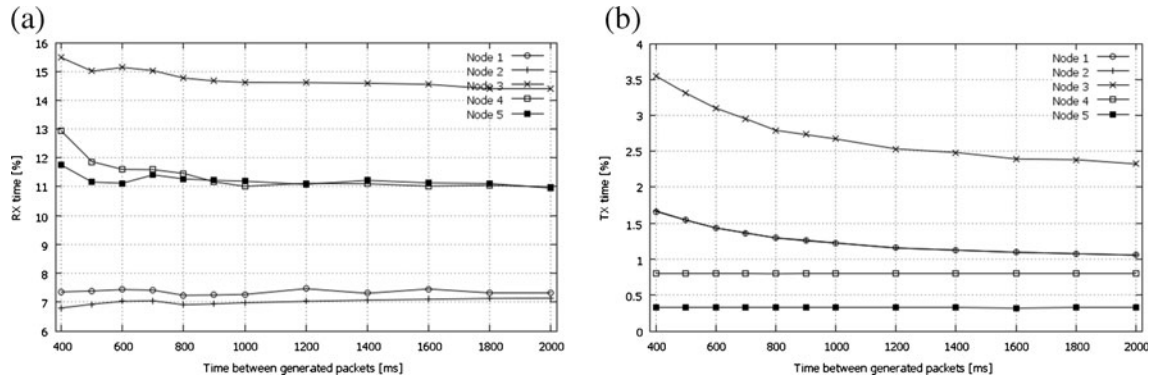


Figure 17 Simulation results of the activity of radio receiver **a** and radio transmitter **b** for the network configuration from Fig. 15(a).

only due to the fact that they work with lower supply voltage (3 V instead of 3.3 V used at FPGA prototype).

D. *The Sensor Network Installation*

The sensor network consisting of 26 nodes (22 FPGA-based and 4 ASIC-based) has been tested in real conditions. Data from the remote nodes are transferred to the sink node. The sink node is connected via RS-232 to the PC computer, which plays the role of operator's console, where the collected data are visualized. The map of the installation is shown in Fig. 21, the installed nodes are configured to measure only the traffic along the street, as indicated by the arrows on the map. For the deployment of the nodes, the following criteria have been chosen: the vicinity of the university, the possibility to test multi-hop radio transmission, one-way and two-way streets and the various congestion of the nodes' positions.

Each node accumulates information about the traffic and transmits it every 60 s. Finally, this information is presented at the console as simple histograms (Fig. 22), showing the number of vehicles detected for each direction. Each node maintains statistics about average tracks of the moving vehicles, so standing vehicles, for which the moving direction cannot be determined, shall be classified according to their location on the street. The project had a strictly scientific purpose, therefore the collected data were used only by the researchers, the controlling of the city lights was not introduced at this stage.

Apart from the current state of the traffic, the history of the traffic for each node and each direction can be displayed in the form of a time graph. The operator can also: configure a single parameter of each node, reset the node, update its software or download logs or single camera pictures, however these operations work slowly due to the low throughput of the radio network. Each node is also equipped with on-board temperature sensor and A/D converters for measuring the solar panel and battery voltages and currents. The proposed network protocol has dedicated data slots for transmitting 8-bit values of the measured temperature, noise, sun intensity and node's internal battery state.

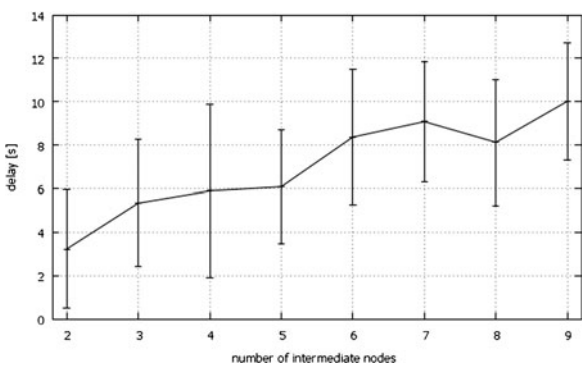


Figure 18 Simulation results of average delay of packets for the network configuration from Fig. 15(b) for $t_{gen}=1$ *s*.

6 Conclusions

This work represents an attempt to solve the problem of measuring street traffic using a sensor network. For the

Figure 19 The picture uploaded from the node.

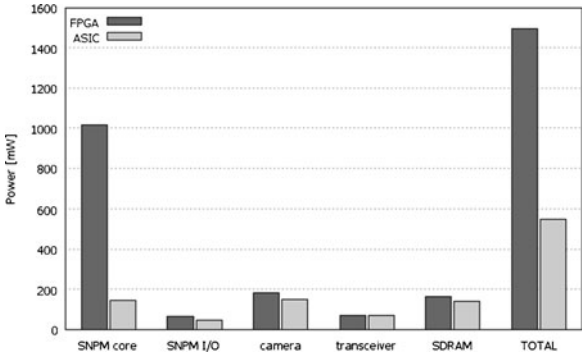


Figure 20 Power consumed by the main blocks of the sensor network node prototypes during typical work (detecting traffic and exchanging data with 5 neighbors). FPGA prototype works with the following power supply voltages: 1.2 V and 2.5 V for the core, 3.3 V for SNPM I/O, camera and SDRAM and 3 V for the transceiver. ASIC prototype uses 1.2 V for the core, 3 V for all other blocks. The power losses at the power supply and battery charging circuitry are not included.

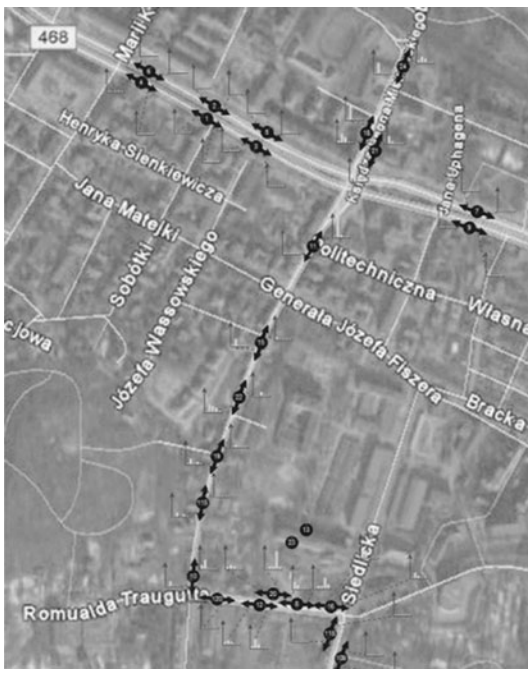


Figure 21 The location of the sensor network nodes (*black dots*) on the streets. The arrows indicate the directions of measured traffic.

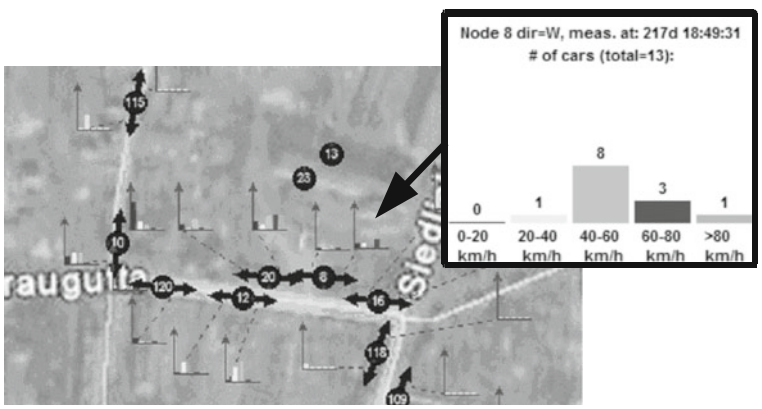
Figure 22 Histograms representing the traffic detected by a node.

purpose of video detection, dedicated multi-processor hardware and algorithms have been developed. The system has been realized as a relatively low-power device, with low hardware and software resource usage.

The video detection of the vehicles is realized using low-resolution images and simplified algorithms, thus its accuracy is not high, but it seems appropriate for a rough estimation of the traffic flow. The reduction of video data from 8 bits to 4 bits resulted in a decreased sensitivity for low-light scenes.

The sensor network nodes acquire data about car traffic and environmental parameters, such as temperature or the voltage from the solar panel, and they transmit them by the radio in the compressed form to the node closer to the sink. Other measured parameters, such as acoustic noise level, can be easily added, giving the possibility to monitor the noise in the area of the city. For that purpose, a self-organizing multi-hop radio network protocol has been designed, responsible for downloading data from the remote nodes to the sinks. Data are aggregated at the nodes to decrease the number of radio transmissions. The radio protocol also enables the uploading of data to configure the remote nodes. Despite using 500 mW transceivers, the transmission range was only 900 m, which requires dense location of the nodes. In some situations, it might be more convenient to install GSM transceivers for the remote nodes.

The sensor network node has been practically realized in two versions: FPGA and ASIC and compared with each other. The network consisting of 26 nodes has been tested in real conditions for more than 12 months. The achieved detection rate (as shown in Table 11) was enough to report the traffic flows in the monitored area. The radio transmission performance was very close to the simulation results. In the case of transmission problems, the large nodes' data buffers were able to buffer data locally for up to 30 min, waiting for the radio link improvement. The installation was successfully collecting data, which were recorded and visualized on a PC, proving the usefulness of this idea.



Acknowledgments This work was supported in part by the Polish Ministry of Science and Higher Education under R&D grant no. R02 014 01.

References

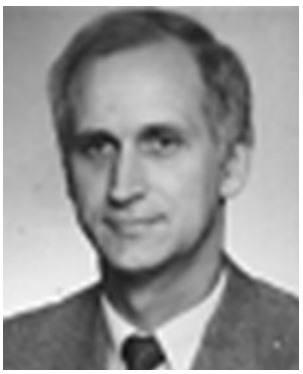
1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). "A survey on sensor networks". *Communications Magazine IEEE, 40*(8), 102–114.
2. Karl, H., Willig, A. (2007) Protocol and architecture for wireless sensor networks, John Wiley and Sons, Ltd.
3. Stojmenovic, I. (2005). Handbook of sensor networks, John Wiley and Sons, Inc.
4. Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks, 52*(12), 2292–2330.
5. Leduc, G. (2008). "Road traffic data: Collection methods and applications", JCR technical notes. European commission, joint research centre, institute for prospective technological studies
6. "Traffic Detector Handbook: 3rd edition", vol I, II, FHWA-HRT-06-139, FHWA-HRT-06-108, US Department of Transportation, Federal Highway Administration 2006.
7. Eng-Han Ng, Su-Lim Tan, Guzman, J.G. (2009). "Road traffic monitoring using a wireless vehicle sensor network", *Int. Symposium on Intelligent Signal Processing and Communications Systems ISPACS 2008*, pp. 1–4, 8–11
8. Yang, S.S., Kim, Y. G., Choi, H. (2007). "Vehicle identification using wireless sensor networks", *Proc. IEEE SoutheastCon, 2007*, pp. 41–46, 22–25.
9. Kastrinaki, V., Zervakis, M., & Kalaitzakis, K. (2003). A survey of video processing techniques for traffic applications. *Image And Vision Computing, 21*(4), 359–381.
10. Kumar, P., Ranganath, S., Weimin, H., & Sengupta, K. (2005). "Framework for real-time behavior interpretation from traffic video". *IEEE Transactions on Intelligent Transportation Systems, 6* (1), 43–53.
11. Friedman, N., Russell, S. (1997). "Image segmentation in video sequences: A probabilistic approach", *Proc. Thirteenth Conf. on Uncertainty in Artificial Intelligence* (UAI 97).
12. Stauffer, C., Grimson, W. (1999). "Adaptive background mixture models for real-time tracking", *Proc. IEEE Conf. Compu. Vis. Pattern Recognit*., pp. 246–252.
13. Veeraraghavan, H., Masoud, O., Papanikolopoulos, N. (2003). "Computer algorithms for intersection monitor", *IEEE Trans. Intell. Transp. Syst.*, no. 2, pp. 78–89.
14. Chiu, C., Ku, M., & Liang, L. (2010). A robust object segmentation system using a probability-based background extraction algorithm. *IEEE Trans. Circuits Syst. Video Technol., 20*(4), 518–528.
15. Cucchiara, R., Piccardi, M., & Mello, P. (2000). Image analysis and rule-based reasoning for a traffic monitoring system. *IEEE Transactions on Intelligent Transportation Systems, 1*(2), 119–130.
16. Ferrier, N.J., Rowe, S. M., Blake, A. (1994). "Real-time traffic monitoring", in Proc. 2nd *IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, pp. 81–88.
17. Koller, D., Daniilidis, K., & Nagel, H. H. (1993). Model-based object tracking in monocular image sequences of road traffic scenes. *Int. J. Comput. Vision, 10*(3), 257–281.
18. Zhou, J., Gao, D., & Zhang, D. (2007). Moving vehicle detection for automatic traffic monitoring. *IEEE Trans. Vehicular Technology, 56*(1), 51–59.
19. Gorgon, M., Pawlik, P., Jablonski, M., Przybylo, J. (2007). "FPGA-based road traffic videodetector", in *Proc. 12th Euromicro Conf. on Digital System Design, Architectures, Methods and Tools (DSD '09)*, Lubeck, Germany.
20. Pamula, W. (2009). "Vehicle detection algorithm for FPGA based implementation", *Computer Recognition Systems 3*, Springer Berlin/Heidelberg, 57: 585–592.
21. Chen, P., Ahammad, P., Boyer, C., Shih-I Huang, Lin, L., Lobaton, E., Meingast, M., Oh, S., Wang, S., Yan, P., Yang, A. Y., Yeo, C., Chang, L-C., Tygar, J. D., Sastry, S. S. (2008). "CITRIC: A low-bandwidth wireless camera network platform", *Second ACM/IEEE International Conference on Distributed Smart Cameras ICDSC 2008*, pp.1–10.
22. Xu, L., Bu, W. (2011) "Traffic flow detection method based on fusion of frames differencing and background differencing", Second Int. Conf. Mechanic Automation and Control Engineering (MACE), pp.1847–1850.
23. Kwon, C.H., Park, H.J. (2004). "A Study on Camera-Detector System for Traffic Control in Korea", Lecture Notes in Computer Science, Conceptual Modeling for Advanced Application Domains 3289: 566–576, Springer.
24. Wang, C., & Lien, J.-J. J. (2008). Automatic vehicle detection using local features—A statistical approach. *IEEE Transactions on Intelligent Transportation Systems, 9*(1), 83–96.
25. Junior, O. L., & Nunes, U. (2008). *"Improving the generalization properties of neural networks: An application to vehicle detection"* (pp. 310–315). Oct: Proc IEEE Conf. Intell. Transp. Syst.
26. Sun, Z., Bebis, G., & Miller, R. (2006). Monocular precrash vehicle detection: Features and classifiers. *IEEE Transactions on Image Processing, 15*(7), 2019–2034.
27. Arrospide, J., Salgado, L., Nieto, M. and Jaureguizar, F. (2008). "On-board robust vehicle detection and tracking using adaptive quality evaluation," in Proc.IEEE Int. Conf. Image Process. pp. 2008–2011.
28. Chan, Y., Huang, S., Fu, L., & Hsiao, P. (2007). *"Vehicle detection under various lighting conditions by incorporating particle filter"* (pp. 534–539). Oct: Proc IEEE Conf. Intell. Transp. Syst.
29. Wójcikowski, M., Żaglewski, R., & Pankiewicz, B. (2012). FPGA-based real-time implementation of detection algorithm for automatic traffic surveillance sensor network. *Journal of Signal Processing Systems, 68*(1), 1–8.
30. Cucchiara, R., Grana, C., Piccardi, M., Prati, A. (2000). "Statistic and knowledge-based moving object detection in traffic scenes", in *IEEE Proc. Intell. Transp. Syst.*, Dearborn, MI, pp. 27–32.
31. Elgammal, A., Harwood, D., Davis, L.S. (2000). "Non-parametric model for background subtraction" in *European Conf. Computer Vision*, Dublin, Ireland, vol. II, pp. 751–767.
32. Duque, D., Santos, H., Cortez, P. (2005). "Moving Object Detection Unaffected by Cast Shadows, Highlights and Ghosts", in *Proc. IEEE Int. Conf. Image Processing*, pp. 413–416.
33. Porikli, F., Ivanov, Y., Haga, T. (2008). "Robust abandoned object detection using dual foregrounds", *EURASIP J. Adv. Signal Process*.
34. Cucchiara, R., Granna, C., Piccardi, M., Prati, A., Sirotti, S. (2001). "Improving Shadow Suppression in Moving Object Detection with HSV Color Information", in Proc. IEEE Intell. Transp. Syst. Conf., Oakland, CA, pp. 334–339.
35. Ye, W., Heidemann, J., & Estrin, D. (2002). *"An energy-efficient mac protocol for wireless sensor networks"*. New York: Proc INFOCOM 2002 IEEE Press.

Marek Wójcikowski received his M.Sc. degree in Electrical Engineering in 1993 from the Technical University of Gdańsk, Poland. In 1994, he was on leave at the University of Karlsruhe, Germany. In 200, he received his Ph.D. degree in Electrical Engineering from the Technical University of Gdańsk, Poland. Since 1994, he has been with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. His research interests lie in the design of microelectronic systems and sensor networks.

Robert Żaglewski received his M.Sc. degree in Electrical Engineering in 2007 from the Technical University of Gdańsk, Poland. In the years 2007–2008, he was with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. He is now with Intel Shannon Ltd, Shannon, Ireland. His research interests lie in FPGA and ASIC design.

Bogdan Pankiewicz received his M.Sc. degree in Electrical Engineering in 1993 from the Technical University of Gdańsk, Poland. In 2002, he received his Ph.D. degree in Electrical Engineering from the Technical University of Gdańsk, Poland. Since 1994, he has been with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. His research interests lie in the design of analog and digital integrated circuits.

Miron Kłosowski received his M.Sc. degree in Electrical Engineering in 1994 from the Technical University of Gdańsk, Poland. In 2001, he received his Ph.D. degree in Electrical Engineering from the Technical University of Gdańsk, Poland. Since 2001, he has been with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. His research interests lie in the applications of FPGAs in microelectronic systems.



Stanisław Szczepański received the M.Sc. and Ph.D. (with honors) degrees in electronic engineering from the Gdańsk University of Technology, Gdańsk, Poland, in 1975 and 1986 respectively. In 1986, he was a Visiting Research Associate with the Institute National Polytechnique de Toulouse (INPT), Toulouse, France. From 1990 to 1991, he was with the Department of Electrical Engineering, Portland State University, Portland, OR, on a Kosciuszko Foundation Fellowship. From August to September 1998, he was a Visiting Professor with the Faculty of Engineering and Information Sciences at the University of Hertfordshire, Hatfield, U.K. He is currently an Associate Professor and Head of the Department of Microelectronic Systems, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology. He has published more than 140 papers and holds two patents. His teaching and research interests are in circuit theory, fully integrated analog filters, high-frequency transconductance amplifiers, analog integrated circuit design in bipolar and CMOS technology, and current-mode analog signal processing.