Mid Sweden University

This is an accepted version of a paper published in *IEEE transactions on circuits and systems for video technology (Print)*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the published paper:

Imran, M., Khursheed, K., Lawal, N., O'Nils, M., Ahmad, N. (2012)
"Implementation of wireless Vision Sensor Node for Characterization of Particles in Fluids"
IEEE transactions on circuits and systems for video technology (Print), 22(11):
1634-1643

Access to the published version may require subscription.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-14389>

<http://miun.diva-portal.org>

Implementation of Wireless Vision Sensor Node for Characterization of Particles in Fluids

Muhammad Imran, Khursheed Khursheed, Najeem Lawal, Mattias O’Nils and Naeem Ahmad

Abstract—Wireless Vision Sensor Networks (WVSNs) have a number of wireless Vision Sensor Nodes (VSNs), often spread over a large geographical area. Each node has an image capturing unit, a battery or alternative energy source, a memory unit, a light source, a wireless link and a processing unit. The challenges associated with WVSNs include low energy consumption, low bandwidth, limited memory and processing capabilities. In order to meet these challenges, our research is focused on the exploration of energy efficient reconfigurable architectures for VSN. In this work, the design/research challenges associated with the implementation of VSN on different computational platforms such as micro-controller, FPGA and server, are explored. In relation to this, the effect on the energy consumption and the design complexity at the node, when the functionality is moved from one platform to another are analyzed. Based on the implementation of the VSN on embedded platforms, the lifetime of the VSN is predicted using the measured energy values of the platforms for different implementation strategies. The implementation results show that an architecture, where the compressed images after pixel based operation are transmitted, realize a WVSN system with low energy consumption. Moreover, the complex post processing tasks are moved to a server, with reduced constraints.

Index Terms—Reconfigurable architecture, Image processing, Wireless vision sensor networks, Wireless vision sensor node.

I. INTRODUCTION

Wireless vision sensor networks have proved to be useful in many applications, including industrial control and monitoring [1], surveillance [2][3], environmental monitoring [4] and personal care [5]. The application area, which is the focus of this work, is in the monitoring of industrial machines. Traditional imaging systems used a wired link, centralized network, high processing capabilities, unlimited storage and a wall power supply. In many applications, the wired solution results in high installation and maintenance costs. However, a wireless solution is the preferred choice as it offers less

maintenance, infrastructure costs and greater scalability. Due to technological developments, smart camera networks, usually referred to as WVSNs, can perform complex tasks using limited resources such as batteries or alternative energy sources, embedded platform, a wireless link and a limited storage facility. WVSNs produce large amounts of data in two dimensional forms and this places higher requirements on the power, bandwidth, memory and on the processing capabilities. The limited bandwidth of the wireless link makes it difficult to transmit all the visual data. There is a need for local processing on the node so that the amount of data required to be transmitted is reduced and only relevant information is sent over the wireless link. Energy consumption is a critical factor in a WVSN as the sensing nodes are powered by batteries or through alternative energy sources. Therefore the focus of researchers is on the reduction of energy consumption so as to maximize the lifetime of the VSN [6].

In this work, the energy consumption of different VSN implementation strategies are analyzed for software and hardware platforms in order to prolong the lifetime of the node. It must be noted that by software platform, we mean a micro-controller and by hardware platform, we mean FPGAs (Field Programmable Logic Arrays). After analyzing the strategies on these computational platforms, the results are verified by implementing the different processing strategies of the VSN on a hardware platform, which has small sleep power consumption together with a software platform and the server. The proposed VSN resulted in a greater lifetime under a set of resource constraints such as the limitations with respect to energy, local processing capabilities and bandwidth for wireless transmission.

Following the introduction, Section II provides a recap of related work whereas Section III considers the experimental system. Section IV covers the analysis of the VSN on the software and the hardware-software platform, Section V presents the target architecture, Section VI discusses the final implementation of the VSN and Section VII summarizes the conclusions drawn from this work.

II. RELATED WORK

WVSNs have been designed and implemented on both a micro-controller/micro-processor and on a FPGA. The first discussion will be with regards to three WVSN systems that have been developed on a micro-controller/micro-processor

Manuscript received March 07, 2011. This work was supported by STC@miun, Sweden and Higher Education Commission (HEC), Pakistan.

Muhammad Imran, Khursheed Khursheed, Najeem Lawal, Mattias O’Nils and Naeem Ahmed are with Department of Information Technology and Media, Mid Sweden University, Sweden. (phone: 060-148416; fax: 060-148456; e-mail: muhammad.imran, khursheed.khursheed, najeem.lawal, mattias.onils, naeem.ahmed@miun.se).

Copyright (c) 2012 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubpermissions@ieee.org.

namely FireFlye Mosaic [5], CMUcam3 [7] and SensEye [3]. FireFly Mosaic [5] is a wireless camera which consists of a wireless sensor platform. It uses a real-time distributed image processing infrastructure with a collision free TDMA based communication protocol. FireFly is a low-cost, low power sensor platform that uses a real time operating system and an expansion board. However, a CMUcam3 [7] was used with FireFly, which possesses both limited RAM and a reduced computational speed, which is essential for more complex image processing systems. SensEye [3] is a multi-tier of heterogeneous wireless nodes and cameras which aims at low power, low latency detection and low latency wakeup. In this case, low power elements were used in order to wake-up high power elements. It should be noted that these software solutions can have a high power consumption and can exhaust the energy consumption quickly if operated on batteries.

Application Specific Integrated Circuits (ASICs) often provide a better performance and low power consumption for dedicated applications but, the design may be expensive due to Non-Recurring Engineering (NRE) costs and that there is no flexibility in relation to accommodating changes and upgrades in the future [8]. Reconfigurable design technologies are required to provide a competitive level of performance while still having the flexibility to incorporate design changes.

Due to the rapid development in semiconductor technology, the single chip capacity of an FPGA increases greatly at the same time as its power consumption is significantly reducing [9]. The advantages and disadvantages of FPGA technology and its suitability for computer vision tasks were discussed in detail in [10] and its optimization in [11]. A design methodology for mapping the computer vision algorithm onto an FPGA through the use of a coarse grain reconfigurable data flow graph was discussed in detail in [12].

WVSN platforms that have been developed using FPGAs together with micro-controllers will now be discussed. In [13] the authors have designed a novel VSN, based on a low cost, low power FPGA plus a micro-controller System On Programmable Chip (SOPC). The authors in [14] proposed an approach involving integrating the hardware content analysis engine into smart camera systems. The content analysis engine consists of dedicated accelerators and a programmable morphology coprocessor to provide functionality relating to object segmentation, object description and tracking.

In [15], the design principles for the video node are presented in the context of a long-lifetime. The authors have developed a prototype to validate the approach. In the system, a low power flash FPGA and a CMOS camera are used. The imager used is a 128×64-pixel binary contrast based sensor. However, for the majority of applications, a binary image sensor is used and this small resolution would not be feasible. In [16] an image sensor node is presented which includes an embedded processor, a memory, a CMOS image sensor, image processing on an FPGA, an RF module and a power unit, but, no experimental verification was provided in relation to how the power consumption is reduced in the presence of an

embedded processor.

Compared to the work of other researchers, our research is focused on the exploration of energy efficient and reconfigurable architectures for a VSN by partitioning the vision tasks among three computational platforms namely software, hardware platforms and the server. The challenges associated with the implementation of VSN on both hardware and software platforms are shown in this work. The effect on the energy consumption and design complexity at the node, when the functionality is moved from one platform to another, is investigated. This study can be used to determine, which general programmable and low energy architecture is suitable for VSN implementation and what are the relevant design/research challenges.

III. EXPERIMENTAL SYSTEM

The main task for this system is to develop image processing and analysis methods that are able to discover particles in the fluid, extract the features of these particles and analyze them.

The following are the main components and vision tasks involved in this work.

A. System description

In industry, a network of cameras is used in order to obtain real-time data for monitoring and control. These networks are wire based for both data and control transmission and for power supply purposes. The wired solution in an industrial application usually results in high installation and maintenance cost. In this situation, wireless solutions are attractive as they offer lower system and infrastructure costs, improve the product quality and provide simple upgradability and physical mobility [1].

In this work, we are interested in monitoring the health status of hydraulic machines used in industry, as these become worn out after a certain amount of time. The failure of a hydraulic machine can be predicted by means of the detection of magnetic particles in the fluid, flowing in the machine. At the present time, this is monitored manually, which means that the machines must be stopped by the engineers in order to check the fluid quality and to determine whether any of the magnetic particles have detached from the engine. A machine stoppage reduces the production rate in addition to the labor and time costs involved. If the smart cameras detect particles in the fluid while the machine is running and transmit data to the user over the wireless link, then this would be an entirely new scenario for the manufacturers of hydraulic machines. This will reduce the maintenance costs and will increase the productivity of the machine.

B. Generalization of vision systems

At this point, some common characteristics of wireless vision systems will be presented. Although it is difficult to generalize the image processing flow, some common vision tasks employed in different vision systems implemented on smart cameras will be provided. These vision tasks include

frame subtraction[7], segmentation [2],[6],[7],[15], filtering[7],[15],[16], labeling [15], classification [14][15] and post processing such as compression[5][7]. These types of systems [2],[6],[15],[7],[16] have similar operations and information reduction characteristics. The approach in this work is useful for all such systems.

C. Processing platforms

All vision tasks for our application were implemented on both the software and hardware platforms and the server. The software platform, in this case, is the SENTIO32 platform which is basically a development of the original SENTIO platform [17] with a high performance processing capability. SENTIO32 has a 32 bit AVR micro-controller, AVR32, AT32UC3B0256 [18]. Moreover, the SENIO32 has a CC2520 RF transceiver [19] with 2.4 GHz IEEE 802.15.4.

For the analysis of VSN on hardware platforms, two FPGAs, namely, ACTEL AGL600V5 FPGA [20] and Xilinx Spartan6 XC6SLX9L FPGA [21] were used. In order to test all cases, the Xilinx FPGA was selected as it has sufficient logic cells to accommodate each of the processing strategies for the power calculation. The dynamic power of Xilinx was used and the sleep power of the ACTEL FPGA as this has the smallest sleep power consumption currently available in the market. A serial flash memory of 64 Mbits [22] is used for the background storage. All possible strategies are analyzed in Table 3 and the results are verified by implementing the processing strategies, which fit on the ACTEL FPGA. The energy values are then measured for the implemented strategies and are shown in Table 4.

Following this discussion, now a description of the algorithm flow is given. We have developed two algorithms for the magnetic particles analysis, in which one approach is object based while the other is pixel based. Both algorithms have the same flow, the exception being the pre-processing and the bubble remover step. In the pixel based method, the background is captured at the initial stage and is stored in the memory, while, in the object based method, the background is generated in real-time using a spatial filter. In the pixel based method, bubble removal is performed after morphology, while, in the object based method, bubble removal is performed after classification. Fig. 1 shows the algorithm flow for the different processing strategies in which each vision task is represented by a symbol. The dashed lines represent a possible combination of processing strategies for the final implementation of the VSN and this is shown in Table 4.

D. Image capturing

The image is captured using a Micron Imaging MT9V032, which is a 1/3-inch wide-VGA format CMOS active-pixel digital image sensor. The camera is programmed through an I2C interface for a resolution of 640×400 in real time. The camera clock is running at a frequency of 13.5 MHz, with 30 frames per second. The captured image using the CMOS sensor is represented by caption I in Fig. 1.

E. Pre-Processing

In the object based method, the background is generated using an averaging filter, while, in the pixel based method, the

Fig. 1. Algorithm flow of different possible processing strategies

background is captured and stored in a flash memory at the initial stage. An averaging filter with a mask of 21x21, used for the background generation took 5 minutes on the software platform, which is not realistic for the implementation of VSN. Instead of this approach, the preference was to store the background image in the flash memory. The background is stored in the flash memory via a Serial Peripheral Interface (SPI) at an initial stage. The background image is accessed via the SPI and then subtracted from the current frame in order to detect objects, which could either be magnetic particles or air bubbles.

F. Segmentation

The process of spatial partitioning of an image into mutually exclusive connected image regions is known as image segmentation [23] and the typical segmentation algorithms include thresholding and clustering, boundary detection and region growing. In this application, thresholding has been used as the objects that must be detected are white and the background is relatively black. All pixels having a gray scale value less than a predefined threshold are assigned a zero value and all other pixels in the image are assigned the value one. The resulting image after segmentation is a binary image. The image after segmentation is shown by caption II in Fig. 1 in which noise is visible.

G. Morphology

A morphological operation is then performed on the segmented image in order to remove one or two pixel false objects. The morphology operation used in our algorithm is erosion followed by dilation using a mask of 3×3. During the erosion and dilation, two complete rows must be stored in the line buffers so that we have the necessary neighborhood information for the operation. The morphology is a spatial domain image processing operation in which the neighborhood processing of the values of a group of pixels in an input image is required in order to compute only one pixel in the output image. This type of processing requires a processing mask or kernel, which defines the operation. In this process, the output image size is the same as that of the input. At the image boundary, the central pixel is copied into the invalid boundary pixel. Fig. 2 (B) shows the memory architecture that provides the necessary storage of the previous pixel values in order to

Fig. 2. Part a) shows 3x3 neighborhood processing, Part b) shows its implementation using line buffers

facilitate the sliding window shown in Fig. 2 (A). The line buffers must hold the pixels in a row. In the figure, P_{ij} represents the pixel data at the i -th row and j -th column in the neighborhood and R represents the register. The size of each element in this buffer depends on the dynamic range of the pixel. The line buffers are implemented in the FPGA using block RAMs. The other groups of elements, with down arrows, are the registers where the morphological operation is performed. The resultant image after morphology is shown by caption III in Fig. 1 in which noise is removed.

H. Bubble Remover

The flowing fluid in the system might contain air bubbles, which must be identified and removed as these are unwanted objects and the only interest is in relation to the magnetic particles. Bubbles are removed either by the pixel based method or by an object based method. In the pixel based method, the bubbles are removed after the morphology. Bubbles can be identified as moving objects, so if an object changes its location in two consecutive frames, this confirms that it is a bubble. The corresponding pixels in two consecutive frames are compared and if their binary values are different, which confirms that it is a bubble, then a zero is placed at that pixel location. In this manner, the bubbles are identified and removed. For bubble removing, in the pixel based method, the entire frame must be stored so that the previous frame can be compared with the current frame.

In the object based method, the bubbles, as well as the particles, are treated as objects at the initial stage and the bubbles are removed after the classification stage. Bubbles are removed by comparing the area and location of the objects in the current frame with the objects of the previous frame. Objects whose area and location are not same will be treated as bubbles and hence removed. The challenge associated with this method is that, sometimes, due to changes in the illumination, the area of the object could either decrease or increase in consecutive frames and thus the possibility exists that the magnetic particles might be treated as bubbles. We have dealt with this challenge by introducing a flexibility of one to three pixel variations in the area and location of the objects in consecutive frames. The image after the bubble removal step is depicted by caption IV in Fig. 1. In images I, II and III, the bubbles are visible but these are removed in image IV by applying the bubble removing step.

I. Labeling and Classification

During labeling, the pixels belonging to the same image component are assigned a unique label. After labeling, an image component is described in terms of a region's features such as its area, mean gray value or position [24]. This feature information can then be used for the classification of the image components.

J. Image Compression

Since the data that is available after segmentation is in a binary format, a bi-level compression scheme was used. There are many compression schemes for bi-level images such as CCITT group3 1D and 2D [25], CCITT group4 [25] and JBIG2 [26]. The CCITT group4 has been used in this case as it provides a compression ratio that is sufficient to show the effect on energy consumption, when compressed images are transmitted and when uncompressed images are transmitted. However, the comparisons of bi-level compression schemes do not fall within the scope of this paper.

The CCITT group4 is a lossless compression algorithm which encapsulates data in a TIFF file format. The coding scheme uses a two-dimensional line-by-line coding method in which the position of each changing picture element, rather than alternating white and black runs in each scan line, is considered. In this manner, the vertical features in the source image can be used to achieve better compression ratios. The position of each changing pixel element on the current coding line is coded with respect to the position of a corresponding reference element situated on either the coding line or on the reference line, which is immediately above the coding line. After the coding line has been coded, it becomes the reference line for the next coding line [25].

Three line buffers are used in order to implement the CCITT group4 compression. Two of these are used to act as the stored reference line and the coding line and the third line buffer is used for saving the current row data. When the data is being saved, it then becomes the coding line and the previous coding line becomes the reference line, while the previous reference line becomes the saving line for the next row and is overwritten by the new pixel data. In our case, the CCITT group4 could be performed after the segmentation, morphology or bubble removal as shown in Fig. 1.

The CCITT group4 compression produces varying compressed data based on the input image. Thus, the amount of data after segmentation, morphology and bubble removal is different. The CCITT group4 encoded data may be different on the software and hardware platforms, because in the software platform, the available C library was used, which uses all the header information of the TIFF file format. For the hardware implementation, a group4 compression scheme was developed in which the header information of only the CCITT group4 is used. Using the CCITT group4 compression, the data being transmitted is reduced to few hundred bytes. This, in turn, will lead to a low energy consumption for the overall system as the communication process is the main factor involved in the energy consumption [1].

K. Communication

The final results are transmitted to the server using an IEEE 802.15.4 compliant transceiver [19], embedded in the SENTIO32 platform.

IV. ANALYSIS OF VSN ON SOFTWARE AND HARDWARE-SOFTWARE PLATFORM

In our previous work [27] it was shown that, if the compressed binary image after segmentation is transmitted from the VSN to the server, it will have a low energy consumption. The reason given was that, at this stage the energy consumption, due to the processing and the communication, falls in such proportions that it results in a lower energy consumption. Performing the same vision tasks on the hardware reconfigurable platform, with small sleep power consumption, further improves the results [28]. We have extended the previous work [29] and analyzed different design/research challenges when the VSN is implemented on both software and hardware platforms and these have been verified for different strategies.

The vision tasks for the different processing strategies are shown in Fig. 1, where each task is represented by a symbol such as image capturing by A, pre-processing by B and so on. These symbolized letters are used in Table 1, Table 2, Table 3 and Table 4 in order to visualize all the possible partitioning strategies on the software, hardware and the server. For example, in Table 1, ABC represents the energy consumption of the vision tasks, image capture, preprocessing and segmentation on the software solution and ABC in Table 2 shows the energy consumption for the same vision tasks when implemented on the hardware platform.

A. Energy calculations of VSN modules

In order to calculate the energy consumption of the vision tasks, it is necessary to calculate the execution time of the vision tasks on the hardware and software platforms. The execution time of all the vision processing strategies involved in this application, such as ABC or ABCDEFG are calculated on both the software and hardware-software platforms as shown in Table 1 and Table 2.

In order to calculate the execution time on the software platform, a high signal was sent to one of the output pins of the SENTIO32 at the initiation of the vision task and this was then returned to low at the conclusion of the task. During this operation a time stamp was recorded using the logic analyzer. The execution time for the operation performed on the hardware platform is determined by the camera clock speed. For this work, the CMOS camera is programmed for a resolution of 640×400 and the operating clock frequency is 13.5 MHz. The horizontal low line sync is required to be considered while calculating the execution time on the hardware platform. The execution time for each strategy on the hardware platform is calculated by using formula (1) where Lt is the latency of each tasks, f is frequency, R represent rows, C represents columns and Ls represents low line sync. In this case, R is 400, C is 640, Ls is 32 and f is 13.5 MHz.

TABLE 1. ENERGY CONSUMPTION OF INDIVIDUAL STRATEGY ON SOFTWARE

Individual Modules	Time (ms)	Energy (mJ)
ABC	584.60	45.34
ABCH	929.68	72.10
ABCD	2911.69	225.80
ABCDH	3256.77	252.56
ABCDE	3114.19	241.51
ABCDEH	3459.27	268.27
ABCDEFG	5724.90	443.97

TABLE 2. ENERGY AND AREA OF EACH STRATEGY ON HARDWARE

Individual strategies	Time (ms)	Energy (mJ)	Logic cells
ABC	0.01991	0.038	705
ABCH	0.01996	0.066	1190
ABCD	0.01996	0.061	1388
ABCDH	0.01996	0.089	1873
ABCDE	0.01996	0.064	1406
ABCDEH	0.02001	0.093	1891
ABCDEFG	0.01996	0.118	2279

$$T = (R \times (C + Ls) + Lt) / f \quad (1)$$

The time spent on transmitting the results to the server is calculated as

$$T_{\text{IEEE}} = (X + 19) \times 32 \times 10^{-6} + 192 \times 10^{-6} \text{ (sec)} \quad (2)$$

where X is the number of bytes transmitted, 19 bytes are the overhead involved due to the header information, 32 μsec is the processing time of one byte while 192 μsec is the settling time of the radio transceiver.

The energy consumption of the external flash light, used to achieve a sufficiently high signal to noise ratio, is 0.085mJ, which is included in the energy calculation of the embedded platform for each strategy. The power consumption of the IEEE 802.15.4 transceiver is 132mW while that of the AVR32 is 77.55mW, when operating. The total energy spent on sending data over the wireless link is the combination of the individual energy consumption of the IEEE 802.15.4 transceiver and the software platform because both are running when the data is communicated to the server. The power consumption of the CMOS camera is 160 mW and its processing time is 33.3 ms, so its energy consumption for processing one image is 5.3 mJ.

The time spent and the energy consumed by each strategy running on the software platform is given in Table 1. The time spent for performing vision tasks on the hardware platform is calculated using formula (1), and this is used to determine the energy consumption of the individual modules in Table 2, using the Xilinx Spartan6 FPGA. The Xilinx power measuring tool, called the XPower Analyzer [30] was used for the power analysis of post-implement placing and routed designs. It supports entries from a user constraint file (PCF) so as to provide an accurate definition of the operating clock frequencies as well as the stimulus data (VCD) generated from the HDL simulators to determine the device power and the individual net power utilization. Input stimuli of dimensions 640×400 have been provided. In relation to the analysis purposes, the sleep power consumption of the ACTEL FPGA was used as it has a sleep power consumption of 5 μW . The cost of each individual strategy in terms of logic cells is also shown in Table 2.

A. Partitioning tasks between hardware, software and Locality

In this section we will discuss different challenges when VSN tasks are partitioned among the software platform, hardware platform and the server. In relation to this, the energy consumption and design complexity of the VSN, when functionality is moved from one platform to another, is investigated. All possible scenarios involved in the software, hardware and the server are analyzed and displayed in Table 3, where some are discussed in detail. The maximum achievable frequency (fps), energy consumption, cost in terms of FPGA logic cells, block RAMs and data transmitted from VSN for each strategy is given in Table 3. It must be noted that when processing is performed on the hardware or software platform, the communication, is performed on the software platform as an IEEE 802.15.4 transceiver is embedded with the AVR32 on SENTIO32 platform.

The different VSN implementation strategies are firstly analyzed on the software platform. Strategy28 to Strategy34 in Table 3 are software implemented strategies. The energy consumption for each strategy is shown in Fig. 3 and in Fig. 5, the processing energy is represented by E_SOFTWARE_PROC. The communication energy is a combination of the energy consumption of the software platform, when it is running for communication, represented by E_SOFTWARE_COM and the energy consumption of the transceiver, represented by E_IEEE 802.15.4. The absolute energy consumption in Fig. 3 shows the trend of different VSN implementation strategies. Fig. 5 shows the relative energy consumption of each strategy, in which the energy contribution of the VSN modules such as processing (E_SOFTWARE_PROC) or communication (E_SOFTWARE_COM+ E_IEEE 802.15.4) can be observed.

By looking at all the strategies implemented on the software platform, shown in Fig. 3 and Fig. 5, it must be noted that when the binary data (32000 bytes) is transmitted from the VSN to the server, the energy consumption is higher due to the higher communication energy. This is depicted by Strategy28, Strategy30 and Strategy32. The communication energy consumption on the software platform can be reduced by transmitting compressed binary data or by sending only features. Strategy29, Strategy31 and Strategy33 show that when compressed data is transmitted from the VSN, the communication energy is lower. The processing energy would increase to some extent, as compression is a complex vision task. It must be noted that more vision processing on the software implemented VSN will increase the energy consumption as shown by Strategy34. In this strategy all tasks are performed on the VSN and only the features are transmitted, but, the overall energy is still high due to the high processing energy even if the communication energy is small.

It is evident in Fig. 3 that Strategy29 has a low energy consumption. The reason for this is that the proportion of energy consumption due to the processing and communication is such that this results in the low energy consumption shown in Fig. 5. This strategy offers a general and low energy architecture on a software platform as complex vision tasks

(DEFG) are moved to the server. The design complexity for this strategy is moved from the embedded software platform to the server with reduced constraints.

After the analysis of the software implemented VSN, the strategies in which VSN is implemented on the hardware/software platforms will be included for discussion. The introduction of the hardware platform for vision tasks would improve the results. In order to provide a comparison with respect to the software implemented strategies shown in Fig. 3 and Fig. 5, the strategies with similar vision tasks on the hardware platform are shown in Fig. 4 and Fig. 6. Fig. 4 and Fig. 6 show the energy consumption for different processing strategies on the hardware platform in absolute and relative graphs respectively. It can be observed that the energy consumption due to processing on the hardware platform (E_HARDWARE_PROC) is small in all cases as compared to the software implemented vision processing denoted by E_SOFTWARE_PROC. The reason is that when implementing vision tasks on a parallel computing platform such as an FPGA, the execution time of the tasks is small, which results in a low energy consumption.

TABLE 3. ESTIMATED ENERGY VALUES AND OTHER MEASURES OF EACH VSN STRATEGY ON HARDWARE/SOFTWARE PLATFORM

Strategy	VSN Tasks		Server Tasks	Max Freqency (fps)	E(mJ) Node	logic cells	FPGA brams	data sent
	Hardware	Software						
1	ABC	HP	DEFG	2.89	40.55	705	N.A.	1218
2	ABC	DHP	EFG	0.37	221.45	705	N.A.	1282
3	ABC	DEHP	FG	0.34	231.62	705	N.A.	458
4	ABC	P	DEFG	0.97	220.20	705	N.A.	32000
5	ABC	DP	EFG	0.43	400.67	705	N.A.	32000
6	ABC	DEP	FG	0.39	416.37	705	N.A.	32000
7	ABC	DEFGP	N.A.	0.19	405.02	705	N.A.	114
8	ABCH	P	DEFG	44.3	10.21	1190	3	680
9	ABCD	EHP	FG	1.82	51.18	1388	4	458
10	ABCD	P	EFG	0.97	220.22	1388	4	32000
11	ABCD	EP	FG	0.97	235.93	1388	4	32000
12	ABCD	EFGP	N.A.	0.35	224.57	1388	4	114
13	ABCDH	P	EFG	59.5	9.03	1873	7	500
14	ABCDE	P	FG	0.97	220.23	1406	5	32000
15	ABCDEH	P	FG	82.0	8.07	1891	8	356
16	ABCDEFG	P	N.A.	224	6.47	2279	12	114
17	AB	CHP	DEFG	1.75	54.39	702	N.A.	1218
18	AB	CDHP	EFG	0.34	238.89	702	N.A.	1282
19	ABCD	HP	EFG	2.89	41.00	1388	4	1282
20	AB	CDEHP	FG	0.32	249.07	702	N.A.	458
21	ABCDE	HP	FG	2.89	35.48	1406	5	458
22	AB	CP	DEFG	0.97	237.65	702	N.A.	32000
23	AB	CDP	EFG	0.39	418.11	702	N.A.	32000
24	AB	CDEP	FG	0.36	433.82	702	N.A.	32000
25	AB	CDEFGP	N.A.	0.18	422.46	702	N.A.	114
26	ABCDE	FGP	N.A.	0.38	208.87	1406	5	114
27*	A	BP	CDEFG	0.12	1748.0	329	N.A.	256000
28	A	BCP	DEFG	0.97	263.41	329	N.A.	32000
29	A	BCHP	DEFG	1.10	83.76	329	N.A.	1218
30	A	BCDP	EFG	0.34	443.88	329	N.A.	32000
31	A	BCDHP	EFG	0.31	264.66	329	N.A.	1282
32	A	BCDEP	FG	0.32	459.58	329	N.A.	32000
33	A	BCDEHP	FG	0.29	274.84	329	N.A.	458
34	A	BCDEFG	N.A.	0.17	448.23	329	N.A.	114
35*	A	P	BCDEFG	0.12	1722.2	329	N.A.	256000
36*	AB	P	CDEFG	0.12	1722.2	702	N.A.	256000

Fig. 3. Absolute energy consumption of each strategy on software platform

Fig. 4. Absolute energy consumption of each strategy on hardware/software platform

Fig. 5. Relative energy consumption of each strategy on software platform

Fig. 6. Relative energy consumption of each strategy on hardware/software platform

For the hardware implementation, it is the communication energy consumption depicted in Fig. 6 by $E_{\text{SOFTWARE_COM}}$ and $E_{\text{IEEE 802.15.4}}$ which is required to be addressed. Depending on the requirement and budget involved in an application, it is possible to implement any of the 36 strategies mentioned in Table 3, except those strategies denoted by the symbol * as they generate large amounts of data (256000 bytes). All strategies have their own pros and cons. By looking at all the strategies, it can be observed that the energy consumption is small for strategies 8, 13, 15 and 16.

In Strategy16, all the vision tasks for this application are processed on the VSN using the hardware platform and only features are transmitted (P, Radio) using the software platform. The cost of this strategy is high compared to other strategies as 2279 logic cells and 12 block RAMs are required. Similarly, there is also a high design time for this strategy as it involves complex post processing tasks on the hardware platform. In strategies 15 and 16, the previous frame must be stored in order to remove the bubbles in the current frame. Storing a binary frame (640×400) requires 256k bits of memory which may prove to be large for some specific development boards. Following the analysis, an architecture used for the final implementation is now presented.

V. TARGET ARCHITECTURE

The target architecture for our work is presented in Fig. 7 which includes a CMOS camera, a light source, FPGA, micro-controller, flash memory, radio transceiver and the server. The vision tasks can be performed on the ACTEL FPGA, micro-controller or at the server side. Communication is handled on the SENTIO32 platform, which has AVR32 micro-controller and IEEE 802.15.4 compliant transceiver. The board to board communication between ACTEL and SENTIO32 is handled using an RS232 with a baud rate of 115200. Buffers are required when data is communicated between different boards due to the different processing speeds. The sleep power consumption affects the results as the VSN is switched to sleep power mode when it is not transmitting data. Therefore the selection of a suitable processing platform is necessary.

Fig. 7. Target architecture of VSN

ACTEL FPGA has the lowest sleep power consumption ($5\mu\text{W}$) currently available in the market. In addition, the ACTEL flash FPGAs allow the user to quickly enter and exit from the Flash Freeze mode. This occurs almost instantly (within $1\mu\text{s}$) and the device retains the configuration and data in registers and RAM [20].

VI. IMPLEMENTATION OF VSN

In order to validate the simulation results of Table 3, the strategies from 1 to 13 were implemented on the ACTEL FPGA, as these strategies fit on the target platform. The measured energy values are shown in Table 4. The energy consumption of each VSN strategy is measured when it is running on the FPGA together with the AVR32 micro-controller. The Strategy16, which has a low energy consumption, is extrapolated by the difference factor of the simulated and measured power values of the FPGA. The strategies in Table 3, in which the hardware platform sends gray scale data to the software platform or to the server, are not feasible for implementation as the amount of data is quite large. The architecture for our work is shown in Fig. 7. The demonstration setup is shown in Fig. 8 where the results after segmentation are shown on the display unit. In the figure, the setup is represented by capital letters including A1 for the light, B1 for the CMOS camera, C1 for the machine setup, D1 for the LCD display unit and E1 for the ACTEL FPGA.

The power consumption of the ACTEL FPGA is measured, when it is running a particular strategy, using the two output pins of the ACTEL platform. The current is measured by using a shunt resistor with a value 0.33Ω between the two pins. The FPGA utilization for each strategy on the ACTEL FPGA is shown in Table 5 where the core is the logic used, IO is the input and output pins, global is the clock network resources and RAM/FIFO are the memories used.

The result of each strategy is verified by reconstructing and comparing the images on the server using MATLAB. The maximum achievable frequency of each implementation strategy is shown in Table 4. It must be noted that in strategies 4, 10, and 11, the transceiver speed is limiting the frame rate. In these strategies, the time spent on task execution is small as the vision tasks are implemented on the hardware platform but, the time spent on transmission is large due to the large amount of data transmission (32000 bytes). In strategies 7 and 12, the transmission time is small but, the execution time is high, as complex vision tasks are implemented on the software platform. The execution time on the software platform is limiting the frame rate. The strategies 8, 13 and 16 have higher frame rates as complex vision tasks are implemented on the hardware platform and the amount of data being transmitted is small. The amount of data required to be communicated for each strategy is shown in Table 4 and the differences for each strategy are based on the CCITT group4 compression, which produces varying compressed data based on the incoming data from each of the vision tasks.

The lifetime for the different VSN strategies is predicted using 4 AA batteries and is shown in Fig. 9. It is assumed that the batteries would provide a constant performance and zero leakage current. When VSN transmits data after 5

TABLE 4. MEASURED ENERGY VALUES OF EACH VSN STRATEGY ON HARDWARE/SOFTWARE PLATFORMS

Strategies	VSN Tasks		Server Tasks	Max frequency (fps)	Energy (mJ) Node	Bytes sent
	Hardware	Software				
1	ABC	HP	DEFG	2.89	40.6	1218
2	ABC	DHP	EFG	0.37	221	1282
3	ABC	DEHP	FG	0.34	231	1014
4	ABC	P	DEFG	0.97	220	32000
5	ABC	DP	EFG	0.43	400	32000
6	ABC	DEP	FG	0.39	416	32000
7	ABC	DEFGP	N.A.	0.19	405	114
8	ABCH	P	DEFG	44.3	10.5	680
9	ABCD	EHP	FG	1.82	51.3	1014
10	ABCD	P	EFG	0.97	220	32000
11	ABCD	EP	FG	0.97	236	32000
12	ABCD	EFGP	N.A.	0.35	224	114
13	ABCDH	P	EFG	59.5	9.57	500
16	ABCDEFG	P	N.A.	224	7.19	114

TABLE 5. ACTEL FPGA RESOURCE UTILIZATION AND ENERGY CONSUMPTION

	ABC	ABCH	ABCD	ABCDH	Available
CORE	556	2357	1602	2778	13824
IO (W/ clocks)	20	20	20	20	177
GLOBAL	6	6	6	6	18
RAM/FIFO	8	17	16	21	24
Energy(mJ)	0.152	0.758	0.455	1.061	

minutes, the lifetimes of VSNs are 5 years for Strategy8, 5.1 years for Strategy13 and 5.2 years for Strategy16. It must be noted that the differences in lifetimes and the energy consumption are small in these strategies, whereas, the design complexity and costs are large. In Strategy16, the vision tasks (ABCDEFG) are implemented while in Strategy8, the vision tasks ABCH are implemented on the VSN. Strategy8 has a small design complexity and cost compared to Strategy16 and provides a more general solution because the complex post processing vision tasks are moved to the server. Implementing the vision tasks on general programmable platforms, such as a server, has small design complexity compared to FPGA implementation. This approach can be used for other smart camera systems as many vision tasks are common in different VSN applications as discussed in section III (B). The energy contribution of each sensor module for Strategy8 is shown in Fig. 10 for different sample periods.

Fig. 8. Industrial setup with ACTEL platform

It must be noted, when the VSN transmits data after a short duration such as after every 2 sec (0.03 minutes), then at this sample rate, the communication energy is high as shown by (E_IEEE 802.14.5, E_SOFTWARE_COM), which must be addressed. The communication energy can be reduced by sending compressed data, by sending data when new changes are detected or by sending data of regions of interest. For applications, where the node transmits data after a long period such as after 5 minutes or 15 minutes, the sleep energy depicted by (E_SLEEP_PLATFORM) is the dominant factor. It can be seen in Fig. 10 that after a sample period of 5 minutes the sleep energy starts to dominates. The sleep energy can be reduced by selecting platforms with a small sleep power.

Fig. 9. Life time of VSN for different strategies implemented on hardware/software platform.

Fig. 10. Energy contribution of different modules of VSN for Strategy8 over different sample periods.

VII. CONCLUSION

Different VSN implementation strategies were analyzed on different computational platforms such as a micro-controller and an FPGA together with a server. The results are verified by implementing VSNs for different strategies and measuring the energy values. The VSN architecture included an FPGA, a micro-controller, an IEEE 802.15.4 compliant transceiver, a flash memory, a light source and a CMOS camera. The vision tasks are partitioned in the three target platforms including the

FPGA, micro-controller and server, in order to determine its effect on the energy consumption and the design complexity. This work will assist the designers and researchers in relation to knowledge concerning the challenges associated with the implementation of VSN. Moreover, the study can be used to determine which general programmable and low energy architecture is suitable for a VSN when functionality is moved from one platform to another.

For a software implemented VSN, the processing energy is high and it is better to perform the initial vision tasks on the VSN and then transmit the compressed binary data to the server for processing. In this way, the energy consumption is reduced as well as the design complexity being moved from the embedded platform to the server with reduced constraints. The challenge in relation to a software implemented VSN is to reduce the processing energy while still having a small design complexity on the node.

In relation to a hardware implemented VSN, the processing energy is reduced and the results are improved compared to those for a software implemented node. However, there is large space for further improvements if the communication is handled efficiently. It has been shown in this work that there can be two approaches for VSN implementation on the hardware platform. In one approach, all vision tasks for a specific application are processed and the final features can be transmitted to the user. In the second approach, some initial vision tasks are processed on the VSN and then compressed binary data is transmitted to the server for processing. In this manner, complex post processing tasks are moved to the server. It is concluded that the difference, between the two approaches, in terms of energy consumption and lifetime, is small but the difference in terms of design complexity is high. The difference in design complexity is high because implementing vision tasks on general programmable platforms such as a server has a small design complexity as compared to an FPGA implementation.

The challenge in relation to the hardware platform is in the reduction of communication energy while still having a small design complexity at the node. The communication energy can be reduced by reducing the amount of data being transmitted. The amount of data can be reduced by transmitting compressed data, by sending regions of interest or by transmitting data when new changes are detected. By addressing all these issues, the lifetime of the VSN can be prolonged while still having a low design complexity at the node.

ACKNOWLEDGMENT

M. Imran thanks Kotte Hari Babu and Ambatipudi Radhika for discussion relating to the measurement setup.

REFERENCES

- [1] K K. S. Low, W. N. Win, M. J. Er, "Wireless Sensor Networks for Industrial Environments", *Intl Conf. on Intelligent Agents, Web Technologies and Internet Commerce*, Austria, Nov, 2006.
- [2] L. Ferrigno, S. Marano, V. Paciello, A. Pietrosanto, "Balancing computational and transmission power consumption in wireless image

- sensor networks". *IEEE Intl Conf. on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, Italy, July, 2005.
- [3] P. Kulkarni, D. Ganesan, P. Shenoy, Q. Lu. "SensEye, A multi-tier camera sensor network". *13th annual ACM Intl Conf. on Multimedia*, Singapore, pp. 229 – 238, 2005.
 - [4] H. Sato, K. Kawabata, T. Suzuki, H. Kaetsu, Y. Hada, Y. Tobe, "Information Gathering by wireless camera node with Passive Pendulum Mechanism", *Intl Conf. on Control, Automation and Systems*, ICCAS 2008.
 - [5] A. Rowe, D. Goel, R. Rajkumar. "FireFly Mosaic, A vision-enabled wireless sensor networking system". *28th IEEE Intl. Real-Time Systems Symposium. Pages*, pp. 459-468, 2007.
 - [6] I.F. Akyildiz, T. Melodia, K. Chowdhur, "A survey on wireless multimedia sensor networks", *Elsevier-Computer Networks*, Mar. 2007.
 - [7] A. Rowe, A. Goode, D. Goel, I. Nourbakhsh, "CMUcam3: An Open Programmable Embedded Vision Sensor", <http://cmucam.org/>, 2007.
 - [8] N. K. Ratha, A. K. Jain, "Computer Vision Algorithms on Reconfigurable Logic Arrays", *IEEE Transactions on Parallel and Distributed Systems*, vol 10, pp. 29-43, Jan. 1999.
 - [9] L. Shang, A. S. Kaviani, K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family", *ACM/SIGDA 10th Intl. Symposium on Field-Programmable Gate Arrays*, pp. 157– 164. 2002.
 - [10] W. J. MacLean. "An Evaluation of the Suitability of FPGAs for Embedded Vision Systems". *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp.25-25 Jun. 2005.
 - [11] N. Lawal, B. Thörnberg, M. O' Nils, "Power-aware automatic constraint generation for FPGA based real-time video processing systems", *Proc. IEEE Norchip Conf.*, 2007, pp. 124-128, 2007.
 - [12] M. Sen, I. Corretjer, T. Lv, S.S.Bhattacharyya, F. Haim, W. Wolf, S. Saha, S. Jason. "Dataflow-Based Mapping of Computer Vision Algorithms onto FPGAs". *Hindawi Publishing Corporation EURASIP Journal on Embedded systems*, vol. 2007, Jan 2007.
 - [13] C.H. Zhiyong, L. Y.Pan, Z. Zeng, M.Q.-H Meng. "A Novel FPGA-Based Wireless Vision Sensor Node". *Proc. the IEEE Intl Conf. on Automation and Logistics Shenyang, China*. 2009.
 - [14] W. Chan, J. Chang, T. Chen, Y. Tseng, S. Chien, "Efficient Content Analysis Engine for Visual Surveillance Network", *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 693-703, vol.19, 2009.
 - [15] L. Gasparini, R. Manduchi, M. Gottardi, D. Petri, "An Ultralow-Power Wireless Camera Node: Development and Performance Analysis", *IEEE Transactions on Instrumentation and Measurement*, vol PP, pp1-9, 2011.
 - [16] Z. Y. Cao, Z. Z. Ji, M. Z.Hu, "An image sensor node for wireless sensor networks", *Intl Conf. on Information Technology: Coding and Computing*, vol. 2, pp. 740-745, 2005.
 - [17] F. Linnarsson, C. Peng, B. Oelmann. "SENTIO: A Hardware Platform for Rapid Prototyping of Wireless Sensor Networks". *IECON 2006-32nd Annual Conf. of IEEE Industrial Electronics*, 2007.
 - [18] AT32UC3B0256, AVR32, Available: <http://www.atmel.com/>
 - [19] CC2520 transceiver, Available: <http://www.ti.com/>
 - [20] IGLOO video kit (2009), Available: <http://www.actel.com/>
 - [21] Spartan-6 family overview (2010), Available: <http://www.xilinx.com/>
 - [22] Numonyx Serial Flash Memory, Available: <http://www.micron.com/>
 - [23] N. R. Pal and S. K. Pal, "A Review on Image Segmentation Techniques," *Pattern Recognition*, vol. 26, pp. 1,277–1,294, Sept. 1993.
 - [24] A. W. Malik, B. Thörnberg, X. Cheng and N. Lawal, "Real-time Component Labelling with Centre of Gravity Calculation on FPGA", *The Sixth Intl Conf. on Systems, St. Maarten*, 23-28 Jan, 2011.
 - [25] TIFF, Revision 6.0, 1992, Available: <http://www.itu.int/itudoc/itu-t/com16/tiff-fx/docs/tiff6.pdf>
 - [26] JBIG2, Available: http://jbig2.com/jb2com_toc.html
 - [27] K. Khursheed, M. Imran, M. O' Nills, N. Lawal. "Exploration of Local and Central Processing for a Wireless Camera Based Sensor Node". *Proc. IEEE Intl Conf. on Signal & Electronic System*, Gliwice, Poland, Sept. 2010.
 - [28] M. Imran., K. Khursheed, M. O' Nills, N. Lawal. "Exploration of Target Architecture for a Wireless Camera Based Sensor Node". *28th Norchip Conference*, Finland, Nov. 2010.
 - [29] K. Khursheed, M. Imran, A.W. Malik, M. O'Nils, N. Lawal, B. Thörnberg, "Exploration of Tasks Partitioning between Hardware Software and Locality for a Wireless Camera Based Vision Sensor Node", *6th Intl. Symposium on Parallel Computing in Electrical Engineering*, UK, pp. 127-132, 2011.
 - [30] Xilinx power tools tutorial (2010), Available: <http://www.xilinx.com/>