Reconfigurable FPGA architecture for computer vision applications in Smart Camera Networks

Luca Maggiani, Claudio Salvadori, Matteo Petracca, Paolo Pagano, Roberto Saletti

Reconfigurable FPGA Architecture for Computer Vision Applications in Smart Camera Networks

Luca Maggiani *†, Claudio Salvadori*, Matteo Petracca†, Paolo Pagano†, Roberto Saletti‡,
* TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, Italy

† National Laboratory of Photonic Networks, CNIT, Pisa, Italy

‡ Department of Information Engineering, University of Pisa, Pisa, Italy

***Abstract*—Smart Camera Networks (SCNs) is nowadays an emerging research field which represents the natural evolution of centralized computer vision applications towards full distributed and pervasive systems. In such a scenario, one of the biggest effort is in the definition of a flexible and reconfigurable SCN node architecture able to remotely support the possibility of updating the application parameters and changing the running computer vision applications at run-time. In this respect, this paper presents a novel SCN node architecture based on a device in which a microcontroller manages all the network functionality as well as the remote configuration, while an FPGA implements all the necessary module of a full computer vision pipeline. In the paper the envisioned architecture is first detailed in general terms, then a real implementation is presented to show the feasibility and the benefits of the proposed solution. Finally, performance evaluation results prove the potential of hardware software codesign in reaching flexibility and reduced latency time.**

I. Introduction

Smart Camera Networks (SCNs) is an emerging research field which represents the natural evolution of centralized computer vision applications towards full distributed systems. Indeed, in SCNs the application logic is not centralized, but spread among network nodes: every SCN node has the capability to (i) pre-process images to extract significant features, and (ii) aggregate data to understand the surrounding environment. In such a scenario a strong cooperation among nodes is necessary, as well as the possibility of having pervasive and redundant SCN devices [1]. These main requirements are nowadays addressed by the research community by proposing SCN nodes based on low-complexity, low-power and low-cost devices able to exchange information through wireless communications.

In designing pervasive SCNs based on low-end devices, one of the biggest efforts is the porting (or the redefinition) of complex PC-based computer vision algorithms to embedded devices, as described in [2]. In the above mentioned paper a GMM-based background subtraction algorithm is implemented over a low-complexity, low-memory and low-power microcontroller while analyzing both the performance gap with respect to a state-of-the-art implementation (based on a floating point arithmetic), and its capability in performing real-time image processing (namely the capability to process images at around 25 fps) in isolation. Another important issue in the SCNs scenario is the definition and realisation of a flexible and reconfigurable node architecture able to update the application parameter and/or change the target application at run-time. In respect of such a problem, this paper proposes a camera node architecture based on a microcontroller, able to handle both the high level operations and the network communications, and an FPGA, devoted at processing complex computer vision algorithms (i.e., pixel-wise operations and/or machine learning blocks). The proposed architecture can have benefit from the high degree of parallelism that an FPGA-based solution makes available, thus permitting to optimise the algorithms at very low level (i.e., hardware level). In this sense it is possible to use the programmable logic to perform elaboration directly on the data stream, as a pixel appears at input port, and without buffering frames. This behaviour might be defined as a *streaming* method because the data is treated as a continuous flow of information.

Hardware programming in SCNs represents a radical different view with respect to pure software based solutions. However, nowadays no standard node architectures are available, although non-reconfigurable custom hardware based solutions targeted to specific applications have been proposed. The works of [3], [4], [5] and [6] follow this concept: the hardware is programmed from scratch in order to find an optimised solution for a specific problem. More in detail, [3], [4] and [5] present an implementation of the Histogram of Oriented Gradients (HOG) on streaming video flow, while [6] extracts aggregated features into a covariance matrix. This work aims at overcoming the state-of-the-art limits in FPGA-based SCN nodes by proposing, implementing and evaluating the performance of an innovative and reconfigurable architecture for designing computer vision pipeline inside nodes embedding FPGAs. Thus, by choosing a fixed set of hardware blocks (namely a set of computer vision algorithms able to define a certain group of consistent pipelines), our architecture permits at run-time to (i) change one block in the pipeline, (ii) instantiate a new application in parallel with another one already running, (iii) remove unused pipelines, and finally (iv) configure certain blocks with new parameters. All the above features give the possibility of creating a flexible and configurable solution without changing the whole FPGA bitstream. The paradigm that permits to implement all these feature is called hardware software codesign [7], and represents the state-of-the-art of

the FPGA programming because merges the flexibility of software programming with the parallel computation allowed by hardware modules.

The proposed approach well fits with the Internet of Things (IoT) vision, in which each node of a global network shares a set of resources among nodes. In this direction, a given hardware block written in a generic Hardware Description Language (HDL), and instantiated inside the proposed SCN node architecture, can be abstracted as network resource and made available to other network nodes. This approach permits to see each hardware block (e.g., computer vision algorithm) as an abstract functional module, following the model-based design reasoning. Any computer vision pipeline can be easily created by connecting these blocks, thus reaching a higher level of abstraction. Although such abstract vision is similar to a model-based design approach, the proposed architecture is not affected by additional computation overheads, because each block is generated independently and the processing time strictly depends on its own optimisation level.

The rest of the paper is organized as follows. First, a detailed description of the architecture is given in Section II. In Section III an implementation performed by using the Altera Design Tools is detailed, in Section IV performance results are shown, finally Section V concludes the paper.

II. Architecture Description

Every complex computer vision application can be seen as a pipeline of functional blocks. Following this view, and by using a set of hardware modules (i.e., hardware computer vision library) which implement some elaboration functions, this paper provides a reconfigurable solution able to combine pipeline items to implement different computer vision applications (e.g., tracking, recognition tasks, etc.).

The above introduced dynamic configuration is addressed in state-of-the-art approaches by using software oriented solutions. In such a vision it is straightforward to modify both configuration parameters and applications at run-time, at the cost of avoiding possible low-level optimizations. Instead, the use of a pure hardware based approach results in the realization of static and monolithic hardware pipelines optimized only for a single application. To overcome the above depicted limitations, while keeping the possibility of dynamic configuration, in this work we present a mixed solution, which takes advantages from hardware optimisation and still considers a software-based configuration. Along this line, we propose a microcontroller and a re-configurable FPGA architecture controlled by a SoftCore. This platform is designed to become the basement for a broad range of computer vision applications by allowing the configuration of computer vision pipelines at runtime.

A. Architecture of a SCN node

The proposed SCN node is mainly based on a microcontroller and an FPGA. The former is in charge of all network communications (i.e., data transmission and feature sharing among nodes), moreover it plays the role of high level controller able to manage remote configuration requests. The latter implements the reconfigurable hardware pipeline. This hybrid structure allows to speed-up the elaboration performance with respect to pure software oriented solutions, while keeping the possibility of a remote configuration.

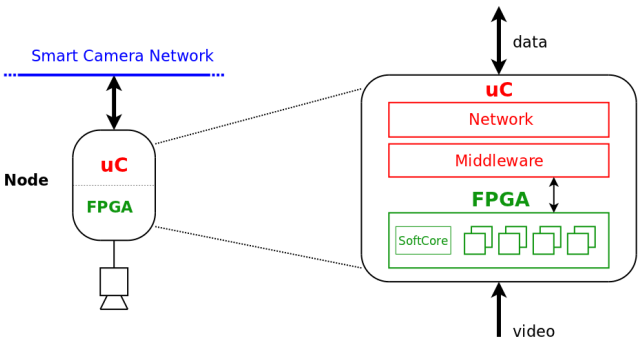


Figure 1: Proposed smart camera network node.

The proposed SCN node architecture is depicted in Figure 1, where it has been reported the high level architectural view of the node, as well as a more detailed representation in which the FPGA communicates with software components run by the microcontroller. The dual layer structure realises a separation between the computer vision heavy processing operations, performed by the FPGA, and the remote node interface, represented by the software abstraction into the microcontroller. In the FPGA, reconfigurable hardware modules that perform optimised pixel elaborations are implemented, as well as a SoftCore in charge of controlling modules parameters and the whole computer vision pipeline. The microcontroller, instead, gives a high level abstraction of the whole system by providing network based interfaces to control configuration parameters of elementary blocks, pipeline composition, and data transmission. Where IoT compliant SCN nodes are necessary, the microcontroller provides an additional abstraction of the system through a middleware solution, thus permitting to show all the internal FPGA modules as resources for other network devices.

B. Internal FPGA architecture

The proposed FPGA architecture called *Camera_OneFrame*,is detailed in Figure 2. It permits to create a full reconfigurable pipeline in a SCN node. The whole architecture is designed focusing on interconnection modules, called *RouteMatrix*, and functional blocks, called *Elab*. The former realises the connections between the functional blocks, while the latter implement basic computer vision algorithms that can be part of a specific elaboration pipeline tunable at run-time. In the left side of the figure four video input ports, labeled as *VideoStream* are shown. This architecture allows to have a multi-camera video inputs that can be parallel processed as a continuous pixel flow. The data flow, composed by one or more video streams, is captured and then processed by successive steps, represented by Elab blocks.

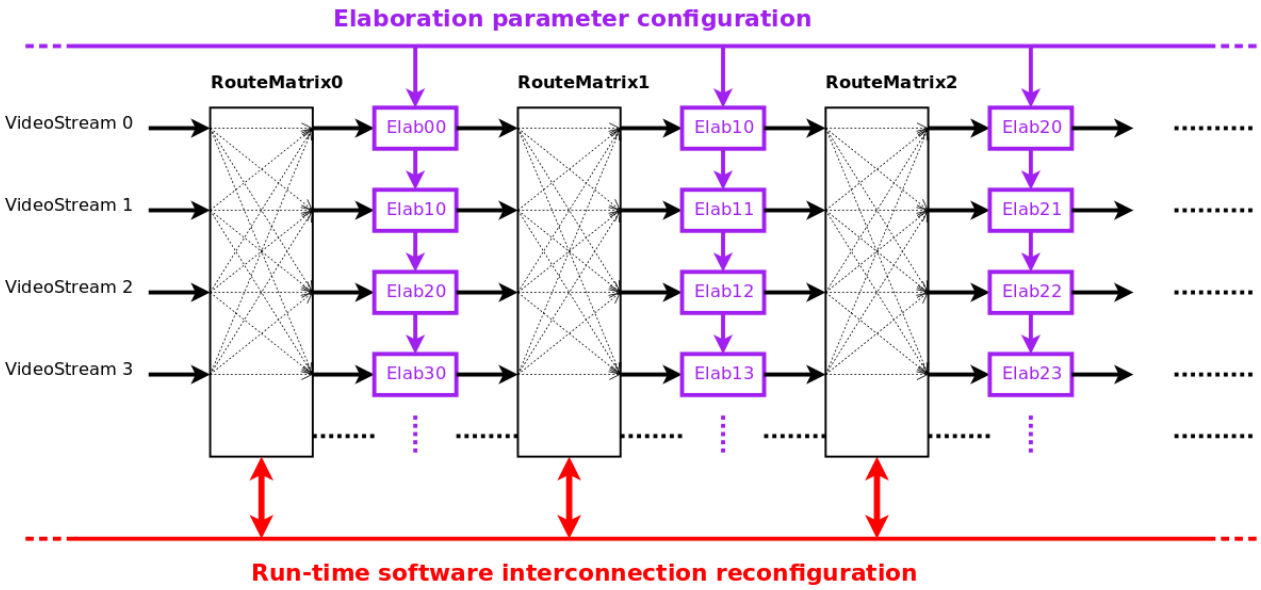


Figure 2: Camera_OneFrame architecture.

In this respect, the captured flow has not an associated semantic, so that an user interacting with a configuration manager can select and compose the appropriate modules suited to the desired application. For instance, through the IoT middleware the desired pipeline can be remotely selected. Every parameter of each Elab block in the architecture as well as the connection among the blocks are managed by the SoftCore created into the FPGA.

The described internal FPGA structure can be expanded as a function of a set of possible applications. Indeed, the number of input and output ports, even the number of elaboration steps can be abstracted as parameters and then configured during the hardware compilation. This allows to modify the number of the parallel data flow, the amount of pipeline steps, or the elaboration blocks, without modifying the HDL instances, but easily inserting them as a new Elab instance using for example a graphic interface tool.

An image processing application has to be divided into functional blocks, algorithm steps, to unveil the internal data-path requested. Each functional block performs an associated algorithm, implemented by a hardware module available into a Hardware Library tool. This library contains a certain amount of functional blocks, defined using HDL, and then collected into a package made available as high level resource. After the instantiation phase, the system is ready to be compiled and programmed in the FPGA as a bitstream. Afterward, though the FPGA bitstream is statically programmed, the system still keeps the flexibility through the software configuration of blocks and connections. This leads to a dynamic and adaptive system, but optimised at the same time, because of the software re-configuration.

The proposed internal FPGA architecture introduces two degrees of freedom: (i) it is possible to grab the requested functional operation from a library, without any knowledge of HDL languages as it was in a model-based unit, and (ii) it is possible to configure the system at run-time to perform new pipelines with the already installed hardware modules configured and connected through the SoftCore.

C. *RouteMatrix module*

RouteMatrix is the core of the internal FPGA architecture: it is the connection point between hardware configuration and software programming. The logic behind our approach can be seen as a 3D-multiplexer, having a $N \in \mathbb{N}$ inputs and $M \in \mathbb{N}$ outputs (Figure 3), and configured by a $N \times M$ matrix through a dedicated bus (the red lines in Figure 3). In this way, the RouteMatrix internal logic guarantees that each output is connected to a selected $m-th$ input vector (with $m \in [1, M]$), to avoid data collision. On the other side every input vector can be connected to several outputs, in order to generate two or more twin sub-pipelines from the same source.
More in details, the RouteMatrix module permits to:

- define the routing path of a data stream (Figure 4a);

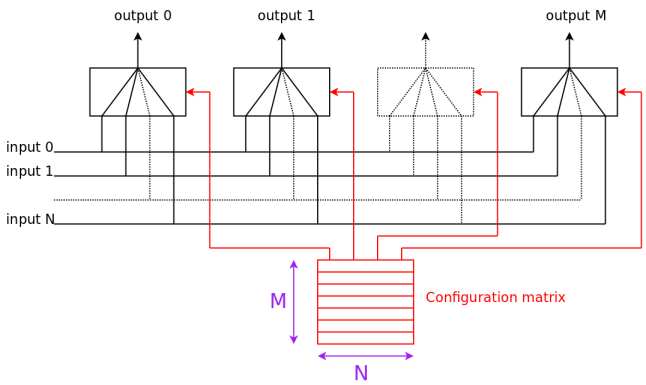


Figure 3: RouteMatrix internal architecture.

- handle the execution of parallel pipelines with different data sources (Figure 4b);
- split a data stream in two or more pipelines (Figure 4c).

D. Elab module

In this paper we define a hardware block as the implementation of a certain computer vision algorithm in any HDL language (e.g., Verilog, VHDL, CAPH [8]). Every block requires at least one input and one output with the related data-valid signals, that notify the validity of the data to the following blocks. The data-valid signals are necessary for enabling the streaming paradigm: in this direction the proposed architecture does not require to specify any data latency and processing time. Finally, the proposed architecture provides a dedicated bus for managing the FPGA region mapped as internal memory of the system. Such a memory can be directly addressed by the SoftCore for elaboration blocks configuration purposes.

III. Implementation

As described in the previous sections, the SCN architecture is based on a microcontroller and a FPGA: the former is in charge of managing the network communication and acts as high level resource controller, while the latter performs heavy processing operations by providing a reconfigurable internal architecture. To really evaluate the benefits of such architecture a real implementation has been performed by using two commercial off the shelf boards: the SeedEye [9] and the Terasic DE0-nano [10].

The SeedEye board is an advanced device, specially targeted to wireless sensor networks, based on PIC32MX795F512L @80MHz provided by Microchip [11] and embeds: a wireless transceiver compliant with the IEEE802.15.4 standard, an IEEE802.3 interface and an USB interface. As discussed in [1], this board is a state-of-the-art solution for IoT networks, providing enough memory and computational resources to execute both IoT protocols and complex middleware solutions performing distributed applications.

The Terasic DE0-nano board embeds an Altera Cyclone IV FPGA, with 22000 Logic Elements (LE), 600 Kbits of on-board memory and 144 9x9 bits DSP modules. In such FPGA we included an Altera NIOSII SoftCore as data flow controller embedded into the elaboration logic and connected to the Elab and RouteMatrix blocks by using an internal bus, called Avalon Memory Mapped (or Avalon-MM). As described in Section I, this approach permits to have a greater flexibility with respect to a pure hardware based solution. In fact, every elaboration block can be mapped on the Avalon-MM bus to be addressed from the NIOSII as a standard memory location. This main feature allow us to: (i) set up the block interconnection at run-time, using some dedicated registers in the RouteMatrix instances, and (ii) control the elaboration parameters for every block inserted into the elaboration pipeline.

A. Hardware library

The *Camera_OneFrame* architecture allows a user to instantiate hardware elaboration blocks without modifying their internal HDL behaviour. This result is realized according to the model-based design view as a consequence of the hardware abstraction. To the end of validating this claim a minimal HDL library has been developed. The library implements the following functions: (i) the *VideoSampler*, which realises the CMOS camera interface; (ii) *RemoteImg*, which performs a serial acquisition of an image from an UART link; (iii) *GradientHW*, that implements a spatial gradient extraction, and (iv) *HistogramHW*, which extracts the histogram of an image divided in cells of a configurable size (as used in the HOG algorithm [12]). Each block optimises a specific function by exploiting the advantages given by the hardware parallelism and the internal DSP modules embedded into the FPGA. Moreover, a complete run-time reconfiguration of all block parameters is allowed through a connection with the Avalon-MM bus. In Table I the FPGA occupancy of every block is described in terms of LE, on-chip RAM, and DSP elements by considering the EP4CE22 FPGA embedded on the Terasic DE0-nano board.

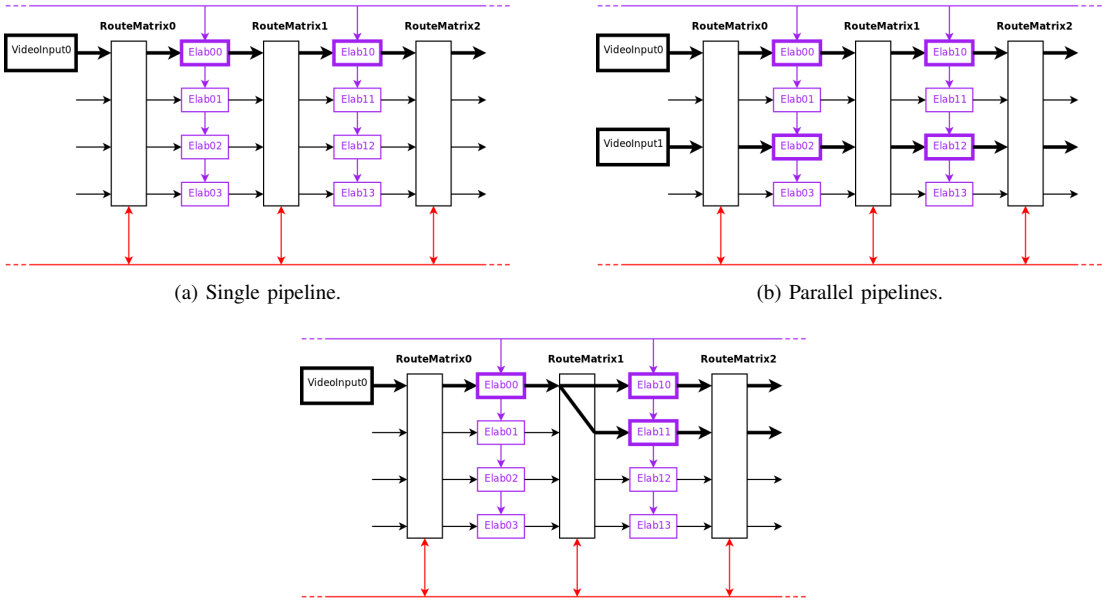
All the developed hardware library functions are compliant with the SoPC Builder [13] tool provided by Altera as a part of the Quartus II software edition [14]. SoPC Builder abstracts every HDL module as a functional entity into a graphical interface, thus helping its instantiation in the proposed architecture. In Figure 5 the design flow for a generic computer vision application is shown using the SoPC Builder tool: (i) the application is chosen (ii) and divided into functional elements; then (iii) the specific HDL blocks are instantiated using the above mentioned tool, and finally (iv) the code is compiled into a bitstream.

IV. Evaluation and Results

In this section the performance of the proposed architecture is evaluated in terms of data output latency. Particularly, we propose a suite of test cases aimed at validating all the claims described in the previous sections by using the modules presented in Section III. More in detail, in the test cases (shown in Figure 6) the following modules have been instantiated: two VideoSampler, three GradientHW, a HistogramHW, three RouteMatrix instances and finally a NIOSII SoftCore. In this scenario the proposed architecture permits to generate several combinations of the considered blocks, thus performing a set of applications fitting inside a rather small FPGA size. In fact,

Table I: Hardware library occupancy.

	Logic elements	RAM footprint (Bytes)	DSP (9x9 bit)
VideoSampler	200 (0,9%)	512	0
RemoteImage	200 (0,9%)	1	0
GradientHW	1200 (5,4%)	640	32
HistogramHW	850 (4,0%)	16384	0
RouteMatrix (x1)	400 (1,8%)	40	0



(a) Single pipeline.

(b) Parallel pipelines.

(c) Pipeline splitting.

Figure 4: RouteMatrix functionality.

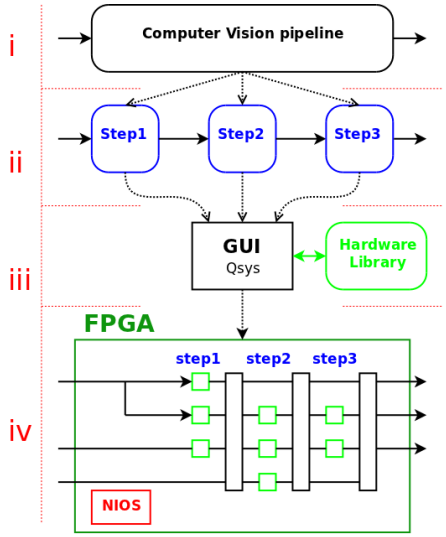


Figure 5: Example of the architecture design flow.

considering the occupancy data reported in Table I, and considering the additional overhead due to the NIOSII SoftCore and the Avalon-MM bus, the whole bitstream occupies only 31% of the LE, the 38% of on-chip memory and the 51% of the 9x9 DSP modules.

The all possible cases of the test suite are reported in Figure 6, where it is shown that the proposed architecture can: (i) handle a single pipeline (Figure 6a), (ii) handle two parallel pipelines using two cameras (Figure 6b) and (iii) split a data stream to follow two pipelines (Fig. 6c).

In order to evaluate the output latency of each pipeline, the time latency of each architecture block is shown in Table II in terms of FPGA clock cycles. These values measure the amount of clock cycles needed by data to flow inside each block, considering that they are implemented according to the streaming paradigm, without buffering a whole image.

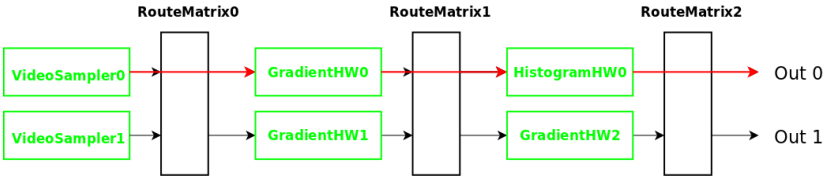
Table II: Hardware module processing latency.

	Latency (clock cycles)
VideoSampler	0
RemoteImage	0
GradientHW	2
HistogramHW	2560
RouteMatrix	1

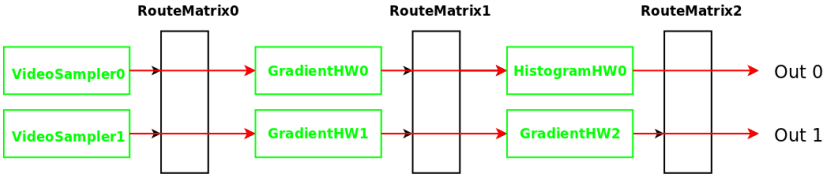
As reported in the table above, every module has a constant data delay, as a consequence of the hardware realisation based on the HDL description. More in detail, the VideoSampler and RemoteImage modules do not introduce any latency time, because they do not perform any elaboration, but only manage clock speed conversions. The GradientHW, instead, introduces a latency time 2 clock cycles. The largest latency values are related to the HistogramHW module, that introduces a latency dependent on image size and cell size: in the contingent case, using 8x8 pixels cells and Q-VGA images, it is of 2560 clock cycles. The last implemented block, RouteMatrix, only introduces a latency of 1 clock cycle.

The overall system performance in time delay can be evaluated as sum of the latencies of the modules inserted into a specific pipeline plus the one introduced by the architecture structure elements (namely the RouteMatrix), as shown in the following equation:

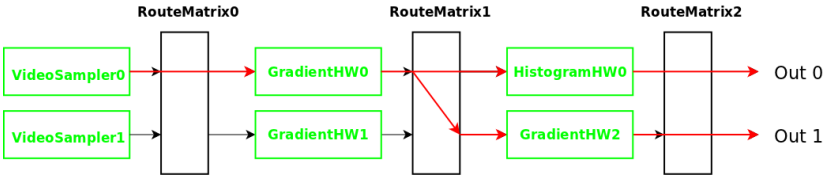
$$L_{total} = \sum_{i=0}^{N-1} L_i^B + M \cdot L^{RM} \quad (1)$$



(a) Test 1: handling a single pipeline.



(b) Test 2: handling two pipelines.



(c) Test 3: split a data stream to follow two pipelines.

Figure 6: Test cases.

Table III: Tests case results.

	Latency (clock cycles)		Time (μs)	
	Out_0	Out_1	Out_0	Out_1
Test 1	2565	-	102,600	-
Test 2	2565	9	102,600	0,360
Test 3	2565	9	102,600	0,360

where L_i^B is the latency of the i-th elaboration block, L^{RM} the latency of RouteMatrix, M the number of RouteMatrix iterations, and N the depth of the considered pipeline.

The latency time value in number of clock cycles can be easily converted in time delay by defining the frequency of the FPGA master clock. In our design, all the system runs at a frequency of 50MHz, thus the values of Table II can be expressed as delay time by multiplying them for the clock period (in this case $40ns$). Table III shows the delays for all the test cases depicted in Figure 6. For all the performed experiments the latency in terms of clock cycles and delay time is evaluated following the rule described in Equation 1.

V. Conclusions

In this paper, we present an innovative architecture concept for a smart camera network node. By leveraging the hardware software codesign concept we propose a hybrid SCN node composed by a microcontroller and a reconfigurable FPGA architecture controlled by an internal SoftCore. Concerning the FPGA architecture, it represents a flexible and reconfigurable solution into a static FPGA bitstream. The flexibility and the reconfigurability is enabled by providing the possibility of composing computer vision pipelines as well as configuring the elaboration block parameters. Moreover, in the proposed design the computer vision algorithms are realized with modules of an HDL library, thus helping programmers in the development of IoT based applications leveraging computer vision capabilities.

In the paper the proposed architecture is first presented by detailing the internal reconfigurable FPGA architecture. Then, a real implementation on commercial off the shelf devices is discussed in respect to three possible applications based on a three stage pipeline. Finally, performance evaluation results are presented in terms of latency time and FPGA occupancy. Results show that the high level abstraction realised by the architecture does not decrease the elaboration performance, indeed shows a constant output latency and an optimized solution, compared to a pure-software application.

References

[1] P. Pagano, C. Salvadori, S. Madeo, M. Petracca, S. Bocchino, D. Alessandrelli, A. Azzar, M. Ghibaudi, G. Pellerano, and R. Pelliccia, "A middleware of things for supporting distributed vision applications," in *Proceedings of Workshop on Smart Cameras for Robotic Applications*, June 2012.

[2] C. Salvadori, D. Makris, M. Petracca, J. M. del Rincón, and S. A. Velastin, "Gaussian mixture background modelling optimisation for micro-controllers," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7431, pp. 241–251.

[3] K. Negi, K. Dohi, Y. Shibata, and K. Oguri, "Deep pipelined one-chip fpga implementation of a real-time image-based human detection algorithm," in *Proceedings of International Conference on Field-Programmable Technology*, December 2011, pp. 1–8.

[4] T. Cao, D.Elton, and G.Deng, "Fast buffering for fpga implementation of vision-based object recognition systems," *Journal of Real-Time Image Processing*, vol. 7, no. 3, pp. 173–183, 2012.

[5] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architectural study of hog feature extraction processor for real-time object detection," in *Proceedings of IEEE Workshop on Signal Processing Systems*, October 2012, pp. 197–202.

[6] S. Martelli, D. Tosato, M. Cristani, and V. Murino, "Fpga-based pedestrian detection using array of covariance features," in *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, August 2011, pp. 1–6.

[7] M. Chiodo, P. Giusto, A. Jurecska, H. Hsieh, A. Sangiovanni-Vincentelli, and L. Lavagno, "Hardware-software codesign of embedded systems," *IEEE Micro*, vol. 14, no. 4, pp. 26–36, 1994.

[8] J. Serot, F. Berry, and S. Ahmed, "Implementing stream-processing applications on fpgas: A dsl-based approach," in *Proceedings of the 2011 21st International Conference on Field Programmable Logic and Applications*, ser. FPL '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 130–137. [Online]. Available: http://dx.doi.org/10.1109/FPL.2011.32

[9] Scuola Superiore Sant'Anna and Evidence s.r.l., "SeedEye board," http://www.evidence.eu.com/products/seed-eye.html, 2011.

[10] Terasic Technologies Inc., "DE0-nano," www.terasic.com, 2012.

[11] Microchip, "PIC32MX3XX/4XX Family Data Sheet," ww1.microchip.com/downloads/en/DeviceDoc/61143E.pdf, 2008.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2005, pp. 886–893.

[13] Altera, "SoPC Builder User Guide," http://www.altera.com/literature/ug/ug_sopc_builder.pdf, 2010.

[14] Altera, "Quartus II Software," http://www.altera.com/products/software/quartus-ii/about/qts-performance-productivity.html, 2012.