Highly Scalable Monitoring System on Chip for Multi-Stream Auto-Adaptable Vision System

Ali Isavudeen, Nicolas Ngan, Eva Dokladalova, Mohamed Akil

On-chip Monitoring for Self-Aware Multi-Stream Vision System

Ali Isavudeen[1,2], Nicolas Ngan
[1]Safran Electronics and Defense
Groupe Safran, Eragny, France
Email: {ali.isavudeen, nicolas.ngan}@safrangroup.com

Eva Dokladalova, Mohamed Akil
[2]Laboratoire Informatique Gaspard Monge, Equipe A3SI
CNRS-UMLV-ESIEE (UMR 8049), Noisy-le-Grand, France
Email: {eva.dokladalova, mohamed.akil}@esiee.fr

***Abstract*—The integration of multiple and technologically heterogeneous sensors (infrared, color, etc) in vision systems tends to democratize. Thus, the advanced driver assistance, 3-D vision, inspection systems or military equipment benefit from this multi-modal perception allowing to improve the resulting quality and robustness; or simply enabling the new applications. According to the applicative context, the parameters of each sensor can dynamically vary as well as the number of 'active sensors' used at the moment. This makes the design of computing resources very arduous task in the context of latency critical application. The proposed solution is based on the self-awareness of such vision system. We propose an original on-chip monitor, completed by an observation and command network-on-chip allowing the system resources supervision and their on-the-fly adaptation. We present the evaluation of the proposed monitoring solution through FPGA implementation. We estimate the cost of the proposed solution in the terms of surface occupation and latency. We show that the proposed solution guarantees a processing of 1080p resolution frames at more than 60 fps.**

***Keywords*—on-chip monitoring, self-aware, auto-adaptive architecture, router, vision, multi-stream, FPGA.**

I. Introduction

More and more embedded vision systems involve multiple, and often heterogeneous, image sensors such as color, infrared or low-light sensor. This trend is motivated by the need to improve the robustness of the applications or by the new industrial usage. To illustrate, we can cite the frequent case of color and infrared image fusion from day and night vision cameras, frequently used in surveillance and security context [1]–[3]. Another example is the fusion of low-light and infrared images enabling color night vision system [4]. Also, the ADAS[1] and UAV[2] systems benefit from such multi-modal approaches increasing the capabilities of such systems [5], [6].

Also, the modern multi-sensor vision systems (Fig. 1) have to provide numerous functionalities as photo capture, face detection, image fusion, depth estimation [3] or moving object tracking [7]. These applications impose the different performance requirements in terms of frame rate, frame resolution or processing latency. It means that according to the applicative context, the parameters of each sensor can dynamically vary. In addition, the number of 'active sensors' used at the moment can dynamically change with respect to the luminosity conditions or the applicative requirements. This makes the design of computing resources very arduous task, especially in the context of latency critical application.

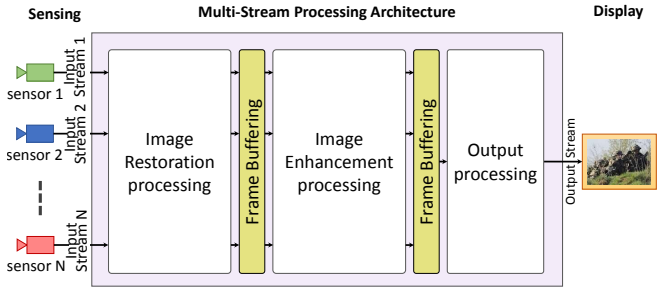

Fig. 1: Multi-sensor embedded vision system.

The complexity of the computationally efficient hardware design for multi-sensor systems is illustrated by the numerous publications dealing with some individual key problems of this challenge. Thus in [8] the authors develop and implement an FPGA-based scalable and resource-efficient multi-camera IP core for image reconstruction, however, the performance decreases with the number of sensors. In [9] the authors focus on the minimization the cost of multi-modal sensing systems, they make the information delivered by sensors available for use by different applications. A runtime reconfigurable SoC is proposed in [10] but it remains limited only to two sensors at the time. In [11] we introduce an auto-adaptable architecture for multi-sensor vision system, but the control and command part is not developed.

Generally speaking, the existing multi-sensor hardware propositions suppose to know the working parameters in advance and the computing system is designed for some given trade-off or even for the worst-case configuration. It results into the multiplication of processing chains specific to each sensor (Fig. 2). If we take into account that not all the sensors are used at the same time, such solutions become very costly and inefficient. For these reasons, we propose to consider dynamically adaptive architecture, based on self-awareness principle [12] and allowing on-the-fly reorganization of the computing capabilities of the architecture.

However, the dynamic reorganization of a heterogeneous multi-stream architecture could raise the latency and the data management issues. To reduce the impact on the processing latency and to add the support of multi-stream data management, we introduce an original on-chip system monitor. Its role is to observe the system and to decide in the real-time when to perform the required runtime adaptations of ressources.

In the past, some interesting monitoring approaches have

[1]Advanced driver assistance systems
[2]Unmanned Aerial Vehicles

been proposed with the similar objectives. For instance, in [13], authors present a Multiprocessor System-on-Chip monitoring solution for frequency scaling. Another monitoring method for Partial and Dynamic Reconfiguration application is presented in [14]. In [15], authors propose a network on chip monitoring based on programmable probes. However, these solutions are not directly applicable to the heterogeneous multi-stream architectures. We demonstrate it in [16], where we present a first dedicated monitoring for on-the-fly pixel frequency fine-tuning for multi-sensor systems. Nevertheless, the scalability of this previous solution was limited.

In this paper, we propose a highly scalable on-chip monitoring system for runtime adaptation of heterogeneous multi-stream architecture. This solution is based on a network on chip, dedicated to collect system observation and to route adaptation command.

The paper is organized as follows. Section II presents the specific challenge of heterogeneous multi-stream vision system design. Then, the proposed Monitoring solution is presented in Section III. Performance evaluation on hardware implementation is given in Section IV while Section V draws the conclusion of the paper.

II. Multi-Stream Vision System Design Challenge

We consider a vision system with multiple and heterogeneous image sensors. These sensors differ from their frame rate, frame size (resolution) or type (color, infrared, low-light). In standard approach each data stream has a restoration, enhancement and output processing stage before getting displayed (Fig. 2). The restoration stage is sensor specific, i.e. a color image stream needs white balance processing while an infrared stream needs heavier contrast enhancement.

In general, linear pipeline implementation of Processing Element (PE) is adopted for lowest latency processing performance. Notice that every latency critical application are inevitably integrated as an optimized pipeline, with several programable features. To illustrate, we can cite corner detecton [17] or mathematical morphology co-processor published in [18]. Such single processing pipeline works at a different pixel frequency and has its own frame rate and frame size. These parameters are tailored according to the characteristics of the sensor (Fig. 2).

Such static and fully pipelined architecture does not allow any runtime modification of these parameters. Nevertheless, we have to consider a multi-context application and the dynamic

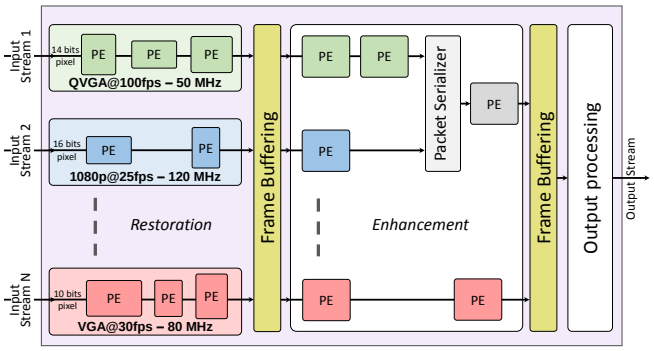

Fig. 2: Pipelined static multi-stream architecture

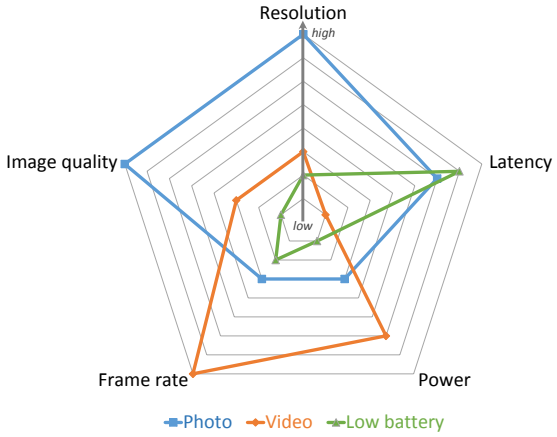

Fig. 3: Different use-cases and required performances

variation of the sensor parameters. Figure 3 allows to compare the different set of parameters and the associated performance requirements we need to support.

In *Video* mode, the context requires the highest frame rate and the lowest latency that the architecture can provide. Whereas, in *Photo* mode, the resolution is the crucial parameter. This mode expects the highest resolution and image quality at expense of a low frame rate. Finally, a third use-case with a *low battery* context is illustrated. In this case, the vision system can provide a quite good frame rate and resolution performance but with the lowest power consumption. This use-case occurs when the end-user is in the end of his operation with the lowest level of the battery.

To resume, we wish to optimize resource utilization while enabling runtime context variation. Hence, we need a dynamically adaptive architecture with the capability to reorganize its structure/data stream management according to the use-case requirements. The proposed monitoring system is designed to perform on-the-fly adaptation of such heterogeneous multi-stream systems. The attention is paid to the streams management and synchronisation during the adaptations. It also guarantees the data coherency. Notice that the proposed on-chip monitoring solution withstands multi-pipeline architecture with multiple clocking domains.

III. On-Chip Monitoring System

In our proposition, the Monitor is used to collect runtime status of the architecture (processing resources and hardware controllers). The runtime status is called *Observations* (Fig. 4).

When a dynamic context switching operation is requested, the Monitor compares the observed system status with the required performances and it adapts the concerning part of the architecture through adaptation *Commands*. According to the considered adaptation, the Monitor may have to load configuration data from *Configuration memory*. Adaptation commands may target processing pipelines or hardware controllers of the architecture. Also, the Monitor supports the partial dynamic reconfiguration, the Monitor only needs to check bitstream memory address in the *Configuration memory*.

The Monitor communicates with processing pipelines thanks to dedicated network on chip. It allows to collect the observation data (OBS) and to send the adaptation commands (CMD). Each processing element (PE) of is bound to a router.

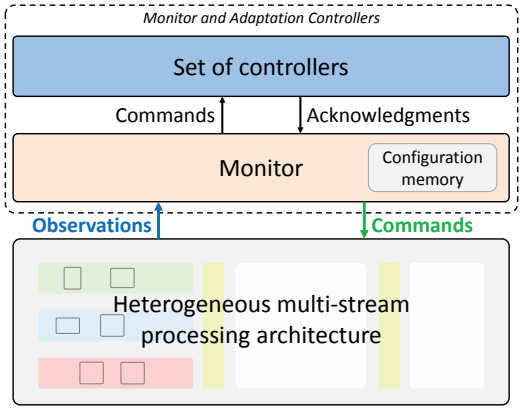

Fig. 4: On-Chip monitoring principle

The PEs on the extremities of a pipeline (the first and last) are bound to a specific R_M router (Monitoring router) while internal PEs are bound to a R_S router (Simple router).

R_M is the interface router between processing pipelines and the Monitor. Each R_M router is connected to the Monitor through a CMD channel and an OBS channel (Fig. 5). Observation data reach the Monitor through OBS channel while the Monitor sends commands to the pipelines through CMD channel. OBS and CMD channels of R_M routers are enough to reach all the PEs of a pipeline. A R_S router of a PE conveys its observation data to its right side neighbour router until reaching the ending R_M router. This later conveys the observation data to the Monitor. In the same way, an adaptation command toward an internal PE is sent to the beginning R_M router of the concerning pipeline. This R_M router forwards the command to its right side neighbour router until reaching the target PE.

Number of PEs and pipelines in figure 5 are given only as an example to put ideas down. For reasons of clarity, only Restoration and Enhancement processing stages are presented in this figure. But, the concept remains valid for Output processing stage too. In figure 5, we can see that the ending R_M routers have not their CMD channel. Actually, as mentioned before, the beginning R_M router is enough to convey CMD data to all the PEs of the pipeline. However, the CMD channel

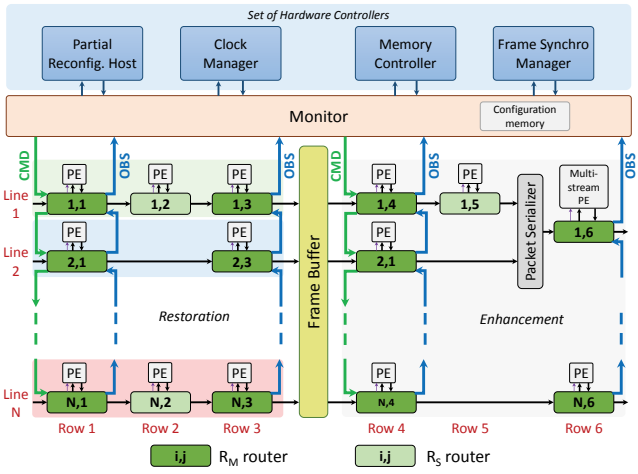

Fig. 5: Network on chip for monitoring of multi-sensor vision system

of the ending R_M routers can be activated in case of a high latency-critical application. Some pipelines may have less PEs than others (ie : pipeline in line number 2). In this case, they will have less R_S routers but still two boundary R_M routers.

To reduce the implementation cost, we adopt the principle proposed in [11] where the adaptation commands were encapsulated into the data stream header. We complete it by adding also the Observations into the header packets (Fig. 6). We propose to use a common communication interface and protocol between PEs, routers and the Monitor. This interface is quite similar to ALTERA Avalon Streaming Interface or XILINX AXI4-Stream interface. This communication protocol is depicted in figure 6.

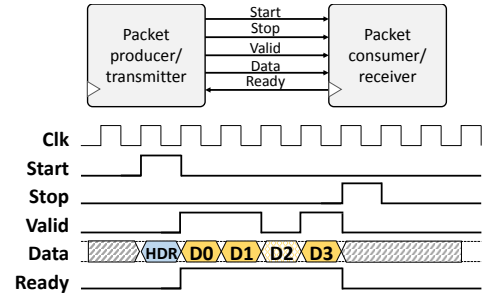

Fig. 6: Communication interface and protocol

A Start and Stop signals indicate respectively the beginning and the ending of a data packet. Between Start and Stop signals, there are a given number of data phits (payload). A Valid signal indicates the validity of the value presented in Data. The value of Data while Start is high represents the packet header. The header has a size of one phit. Ready signal is used as back pressure signal to prevent data loss. By the way, using a back pressure signal reduces buffer memory footprint.

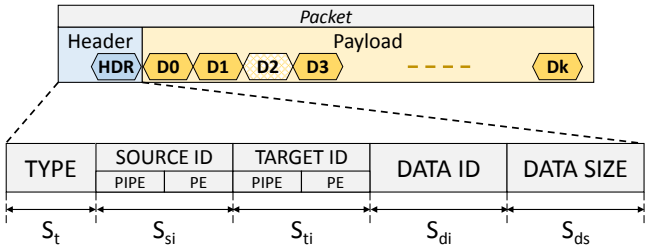

Fig. 7: Packet header description

Packet header details are given in figure 7. The packet header has five fields : *Type*, *Source ID*, *Target ID*, *Data ID* and *Data size*. *Type* indicates whether the packet is a pixel (PIX), an observation (OBS) or command (CMD) packet. *Source* and *Target IDs* give information respectively about the producing and the targeting component of the packet. An OBS packet has necessarily the Monitor's ID as Target ID. Meanwhile, as a CMD packet comes necessarily from the Monitor, its Source ID is the Monitor's one. *Data ID* is used to distinguish several OBS or CMD data respectively from or toward a same PE. Finally, *Data size* gives the number of data phits.

We added a fourth type of packet : frame synchronization packet (SYN). In traditional architecture, frame synchronization signal are distributed by a single wire. Here, we rather use a SYN packet from the Frame Synchronization Controller to synchronize PEs of a pipeline.

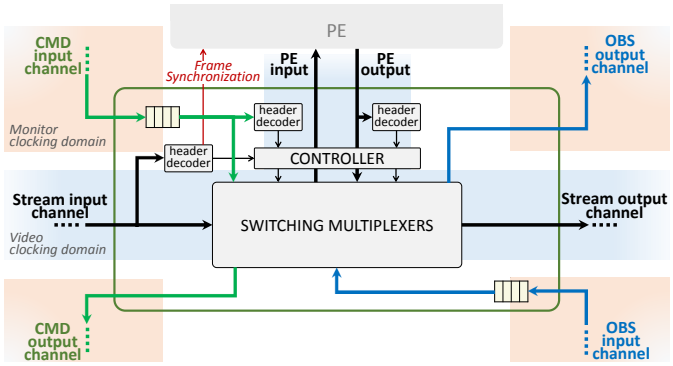

Fig. 8: R_M router internal structure

Figure 8 depicts the internal structure of R_M router. R_S router has a similar structure without CMD and OBS channels. Three dedicated header decoders are used to decode the header of packets entering from Stream input channel, CMD input channel and PE output interface. Header information, Start and Stop signals are used to synchronize and control the set of multiplexers of a router. CMD and OBS channels work in Monitor clock domain whereas Stream channels work in video clock domain.

R_M or R_S router has exclusively one of the following set of configurations $S_{CFG} = \{CFG_1, CFG_2, CFG_3, CFG_4, CFG_5, CFG_6\}$ (Fig. 9). Two more monitoring purpose routing are possible in parallel with one of the previous configurations : FWD_{CMD} and FWD_{OBS}. There are used to forward CMD or OBS packet toward upper or lower pipeline without altering the processing of the current pipeline. Configurations CFG_5, CFG_6, FWD_{CMD} and FWD_{OBS} are specific to R_M router.

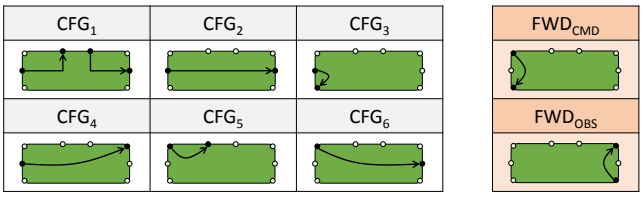

Fig. 9: Set of routers configurations

Packet routing mechanism of R_M is described in figure 10. At the initialization of the system, a R_M router is in default CFG_1 configuration. When a new packet reaches the R_M router (Start signal rising), the packet header is decoded by the R_M router. Then, the router's configuration will depend on the *Type* of the packet. If the type is PIX, the router is configured as CFG_1. In case of SYN packet, the router takes CFG_1 configuration and launches *Frame Synchronization* signal.

If it is an OBS packet, the router checks whether the OBS Output channel is already busy. As long as the OBS Output channel is busy, the Ready signal is set to low to keep the OBS packet. Once the OBS Output channel is free again, the router takes FWD_{OBS} configuration. In case of CMD packet, the router checks whether the CMD Output channel is already busy. If the CMD Output channel is free, the configuration will depend on *Target ID*. According to the Target ID, the router will be configured in CFG_1, CFG_2 or CFG_3. Whatever is the Type, a packet routing process ends when Stop signal rises.

For multi-stream Processing Element, such as color-infrared streams fusion, a Packet Serializer is used to buffer and interlace both streams. One whole frame line of the first pipeline is forwarded to the multi-stream PE before forwarding the next line of the second pipeline. As the Packet Serializer has to deal with twice the bandwidth of a single router, its frequency is at least twice the frequency of a router.

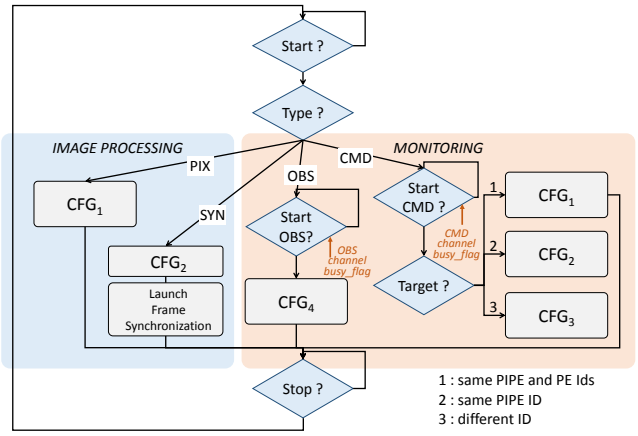

Fig. 10: R_M routing mechanism

IV. Hardware Prototyping and Evaluation

The presented monitoring solution has been implemented in an ALTERA Cyclone V FPGA (5CGXFC7D6F). Performance of this solution has been evaluated through two major scenarios presented in the following paragraphs.

A. Use-case 1: Runtime frame characteristics modification

Context : The application requires the sensor frame rate or resolution modification.

Observation : The monitor verifies the present sensor characteristics.

Adaptation : If necessary, the monitor takes the decision and intiates an on-the-fly pixel clock frequency adaptation.

Controller : Frame synchronisation manager for PLL reconfiguration in FPGA (ALTERA).

When the outdoor context changes (environment type or luminosity condition), the Monitor should choose the appropriate sensor among the available sensors of the vision systems. According to the operational context, it could even be a couple of sensors (ie : color-infrared image fusion, multi-focal image fusion). Consequently, characteristics of the input stream, especially the frame rate and the resolution, would change on-the-fly. In the same way, when the Region-of-Interest (ROI) is rescaled, the resolution of the processed stream could change.

Instead of scaling the architecture's Processing Element with the highest worst-case frequency, we propose to dynamically rescale the pixel clock frequency according to the runtime context requirements. According to the frame rate and resolution of the stream (observation data from the sensor), the Monitor computes the minimal required pixel clock frequency of a given pipeline of the architecture. Then, the Monitor fine-tunes the current clock frequency if its value does not fit with the required one. Some early results have been presented in a previous work [16].

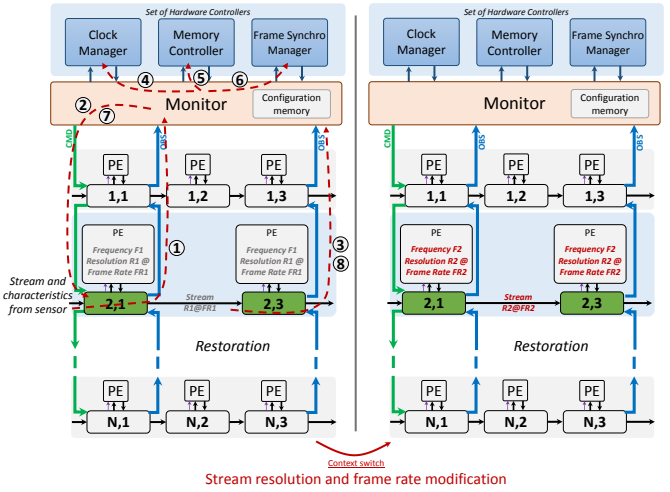

Fig. 11: Use-case 1 : stream frame rate and resolution adaptation

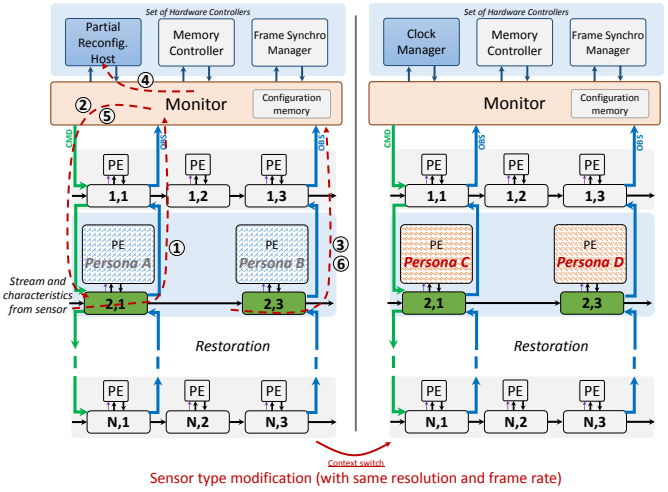

Fig. 12: Use-case 2 : stream type modification

The adaptation of the clock frequency consists in reconfiguring the PLL corresponding to the concerned clock (ALTERA reconfigurable PLL). The Monitor also adapts the Frame Synchronization signal's period. Before any frequency value adaptation, the Monitor sends command to the concerned pipeline to freeze the interface of the PEs.

① Observation of the characteristics of the sensor from the sensor board.
② Freezing command toward PEs of the concerned pipeline in case of any characteristic modification.
③ Freezing operation success information from PEs.
④ New required frequency computation and frequency adaptation command toward the Clock Manager (then PLL Reconfiguration).
⑤ Frame resolution modification command toward the Memory Controller.
⑥ Frame period time modification command toward the Frame Synchronization Manager.
⑦ End of freezing command toward PEs of the concerned pipeline (once all adaptation are completed).
⑧ End of freezing operation success information from PEs.

Notice that this concept is suitable for recent adaptive frame rate and resolution sensor technology.

B. *Use-case 2: Runtime sensor type switching*

Context : New sensor connected to the system, switching between sensor types used in the application.

Observation : Processing pipeline characteristics, sensor specific informations.

Adaptation : The monitor initiates coarse-grain dynamic re-allocation of computation resources.

Controller : Partial and Dynamic Reconfiguration host of FPGA (ALTERA).

This use-case illustrates the context of sensor type switching while the frame rate and resolution values remain unchanged. When the outdoor luminosity condition changes, the type of the sensor ought to be adapted. For instance, the vision system shifts to the infrared sensor for night vision when it is getting night. Sensor-specific pre-processing depends on the type of the sensor. In a static architecture, when one of the vision system's sensor is not used, its pre-processing resources are not re-usable for another active sensor. In our dynamically adaptive architecture, we propose to deploy sensor-specific pre-processing in reconfigurable resources. In case of sensor type switching, these resources would be re-allocated for the new active sensor.

① Observation of the characteristics of the sensor from the sensor board.
② Freezing command toward PEs of the concerned pipeline in case of any characteristic modification.
③ Freezing operation success information from PEs.
④ Decision of the new required image pre-processing and PE adaptation request to the Partial Reconfiguration Host.
⑤ End of freezing command toward PEs of the concerned pipeline (once reconfiguration is completed).
⑥ End of freezing operation success information from PEs.

For evaluation purpose, we simulated luminosity condition switching scenarios (day, evening, night). When the luminosity condition changes, the Monitor check the current active sensors. If the required sensor is not active, it shifts the sensor and adapts the sensor-specific image pre-processing pipeline. The pipeline adaptation consists in Partial and Dynamic Reconfiguration of FPGA.

As the prototype is implemented in Altera Cyclone V FPGA, the Monitor sends reconfiguration request to the Altera PR Core (cyclonev_prblock) through a PR Host. The PR Core returns back a PR success or failure feedback to the Monitor. Once again, before any partial reconfiguration, the Monitor sends command to the concerned pipeline to freeze the interface of the PEs.

V. Latency Cost Evaluation

The proposed solution have been described in HDL and evaluated with a HDL simulator (ModelSim). Sensor pixel streams have been simulated thanks to image vector input files. Several values of frequency, frame rate and resolution have been tested.

Any packet crosses a R_M or R_S router with a minimal

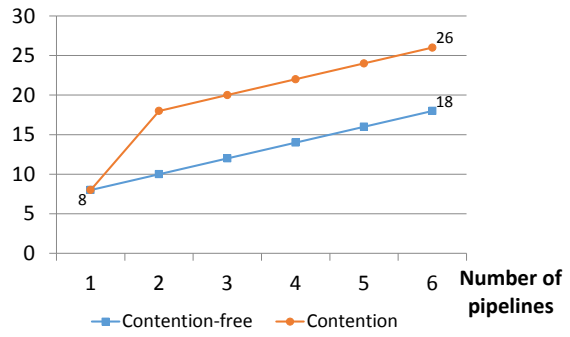

Fig. 13: Monitoring packet routing latency

latency of 2 cycles. These minimal 2 cycles can increase up to 8 cycles in case of contention in the router. In figure 13, latency performance of typical pipelines are presented. We evaluated the latencies in case of 1 to 6 pipelines. For each case, the worst-case routing path latency has been reported. The blue curve presents contention-free scenario whereas the orange one presents the highest contention scenario. In case of a single pipeline, there is no contention. For 3 pipelines-based multi-stream architecture, we have a worst-case latency of 20 cycles. That is to say, a CMD packet from the Monitor takes at most 20 cycles to reach the farthest PE. In the same way, an OBS packet from the farthest PE takes at most 20 cycles to reach the Monitor. Besides, in addition to the interlacing operation latency, the Packet Serializer adds an extra two cycles latency.

A. *Synthesis results*

Our synthesis results are based on Altera Cyclone V FPGA (5CGXFC7D6F) implementation with a 32 bits data size. The header fields sizes of this implementation are given in table I. Area overhead of the presented monitoring solution is given in table II.

Field	S_t	S_{si}	S_{ti}	S_{di}	S_{ds}
Size (bits)	2	8	8	10	4

TABLE I: Header implementation in 32 bits data

The area utilization of the monitoring solution has been compared to a typical multi-stream reference design. This reference design needs 13 R_M, 4 R_S and 1 Packet Serializer. Area overhead comparison is given between brackets. In this reference design, the proposed monitoring solution has less than 7% of overall area overhead.

Component	ALUT	Register	Memory (bit)
R_S	6	144	0
R_M	248	408	512
Packet Serializer	39	42	40 960
Monitor	151	164	0
In reference design (%)	3 438 (6.7%)	6 086 (2.9%)	41 584 (0.9%)

TABLE II: Monitoring solution area overhead

The memory footprint of the Packet Serializer can be improved by reducing the interlacing granularity. Otherwise, as R_M and R_S routers have a relative low area overhead, the solution is easily scalable for architectures with more than 4 pipelines. In case of 64 bits data, we got the following synthesis results. R_M (ALUT:289, Regs:620, Mem:1024) and R_S (ALUT:10, Regs:280, Mem:0).

Nb. of pipeline	1	2	**3**	**4**	6
Clk_Monitor	218	196	**177**	**157**	123
Clk_Video	237	223	**210**	**198**	193

TABLE III: Frequency performance (MHz)

Frequency performance of the proposed solution is presented in table III. Typical multi-stream architectures have 3 or 4 pipelines. Results in table III show a maximum affordable frequency of 157 MHz for monitoring clock (Clk_Monitor) and **198 MHz** for pipeline clock (Clk_Video) in case of 4 pipelines. Within this performance, we can deal with 1080p resolution up to 60 frame per second.

VI. Conclusion

In this paper, we introduced an original on-chip monitoring solution for dynamically adaptive multi-stream vision architecture. This solution is based on a dedicated network on chip for monitoring observation and adaptation. It supports architecture with numerous heterogeneous pixel streams and multiple clocking domains. Evaluations on FPGA implementation show fair latency performance with a relatively low area overhead. Future works will focus on the extension of the proposed network on chip for pixel stream datapath flexibility.

References

[1] Y. Yuan, H. Xu, Z. Miao, F. Liu, J. Zhang, and B. Chang, "Real-time infrared and visible image fusion system and fusion image evaluation," in *Photonics and Optoelectronics (SOPO), Symposium on*, 2012.

[2] S. Yang, W. Liu, C. Deng, and X. Zhang, "Color fusion method for low-light-level and infrared images in night vision," in *Image and Signal Processing (CISP), International Congress on*, 2012.

[3] K. Hisatomi, M. Kano, K. Ikeya, M. Katayama, T. Mishina, Y. Iwadate, and K. Aizawa, "Depth estimation using an infrared dot projector and an infrared color stereo camera," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[4] S. Yang, W. Liu, C. Deng, and X. Zhang, "Color fusion method for low-light-level and infrared images in night vision," in *2012 5th International Congress on Image and Signal Processing*, Oct 2012, pp. 534–537.

[5] C. Bahlmann, M. Pellkofer, J. Giebel, and G. Baratoff, "Multi-modal speed limit assistants: Combining camera and gps maps," in *2008 IEEE Intelligent Vehicles Symposium*, June 2008, pp. 132–137.

[6] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli, "Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 1556–1561.

[7] J. Serrano-Cuerda, M. T. Lopez, and A. Fernandez-Caballero, "Robust human detection and tracking in intelligent environments by information fusion of color and infrared video," in *Intelligent Environments (IE), 2011 7th International Conference on*, 2011, pp. 354–357.

[8] O. W. Ibraheem, A. Irwansyah, J. Hagemeyer, M. Porrmann, and U. Rueckert, "A resource-efficient multi-camera gige vision ip core for embedded vision processing platforms," in *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec 2015, pp. 1–6.

[9] M. Darms and H. Winner, "A modular system architecture for sensor data processing of adas applications," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, June 2005, pp. 729–734.

[10] E. Dokladalova, R. Schmit, S. Pajaniradja, and S. Amadori, "Carvision: SOC architecture for dynamic vision systems from image capture to high level image processing," in *MEDEA DAC*, no. 1, France, 2006, p. 10pp., electronic version (4 pp.). [Online]. Available: https://hal-upec-upem.archives-ouvertes.fr/hal-00622297

[11] N. Ngan, E. Dokladalova, and M. Akil, "Dynamically adaptable noc router architecture for multiple pixel streams applications," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 1006–1009.

[12] H. Giese, T. Vogel, A. Diaconescu, S. Gtz, N. Bencomo, K. Geihs, S. Kounev, and K. Bellman, *State of the Art in Architectures for Self-Aware Computing Systems*. Springer, 2017, ch. 8, pp. 237–275.

[13] M. H. Diana Goehringer, Mounir Chemaou, "Invited paper: On-chip monitoring for adaptive heterogeneous multicore systems," *Reconfigurable and Communication-centric Systems-on-Chip (ReCoSoC)*, 2012.

[14] X.-W. Wang, W.-N. Chen, C.-L. Peng, and H.-J. You, "Hardware-software monitoring techniques for dynamic partial reconfigurable embedded systems," *ICESS, International Conference on Embedded Software and Systems Symposia*, 2008.

[15] L. Fiorin, G. Palermo, and C. Silvano, "A monitoring system for nocs," in *Proceedings of the Third International Workshop on Network on Chip Architectures*, ser. NoCArc '10. ACM, 2010, pp. 25–30.

[16] A. Isavudeen, N. Ngan, E. Dokladalova, and M. Akil, "Auto-adaptive multi-sensor architecture," in *IEEE International Symposium on Circuits and Systems, ISCAS*, 2016, pp. 2198–2201.

[17] P. Possa, N. Harb, E. Dokladalova, and C. Valderrama, "P2IP: A novel low-latency Programmable Pipeline Image Processor," *Microprocessors and Microsystems: Embedded Hardware Design (MICPRO)*, p. In press, Jun. 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01171651

[18] J. Bartovsky, P. Dokládal, M. Faessel, E. Dokladalova, and M. Bilodeau, "Morphological CoProcessing Unit for Embedded Devices," *Journal of Real-Time Image Processing*, 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01251331