HAL
open science

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

Ali Isavudeen, Nicolas Ngan, Eva Dokladalova, Mohamed Akil

To cite this version:

Ali Isavudeen, Nicolas Ngan, Eva Dokladalova, Mohamed Akil. Highly Scalable Monitoring System on Chip for Multi-Stream Auto-Adaptable Vision System. Research in Adaptive and Convergent Systems 2017, RACS 2017, ACM SIGAPP, Sep 2017, Krakow, Poland. 10.1145/3129676.3129721 . hal-01576878

HAL Id: hal-01576878

https://enpc.hal.science/hal-01576878

Submitted on 24 Aug 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

A. Isavudeen
University Paris-Est
LIGM, A3SI, ESIEE
Noisy le Grand, France
a.isavudeen@esiee.fr

N. Ngan
Safran Defense
Eragny, France
nicolas.ngan@sagem.com

E. Dokladalova, M. Akil
University Paris-Est
LIGM, A3SI, ESIEE
Noisy le Grand, France
e.dokladalova@esiee.fr

ABSTRACT

The integration of multiple and technologically heterogeneous sensors (infrared, color, etc) in vision systems tend to democratize. The objective is to benefit from the multi-modal perception allowing to improve the quality and robustness of challenging applications such as the advanced driver assistance, 3-D vision, inspection systems or military observation equipment.

However, the multiplication of heterogeneous processing pipelines makes the design of efficient computing resources for the multi-sensor systems very arduous task. In addition to the context of latency critical application and limited power budget, the designer has often to consider the parameters of sensors varying dynamically as well as the number of active sensors used at the moment.

To optimize the computing resource management, we inspire from the self-aware architectures. 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. And finally, we show that the proposed solution guarantees a processing of 1080p resolution frames at more than 60 fps.

CCS Concepts

•**Hardware** → *On-chip resource management;* **Application specific integrated circuits;**

Keywords

on-chip monitoring; self-aware computing; auto-adaptive architecture; router; vision sensors; multi-stream; FPGA

1. INTRODUCTION

More and more embedded vision systems involve multiple 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 to enable a new industrial usage. To illustrate, we can cite the 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].

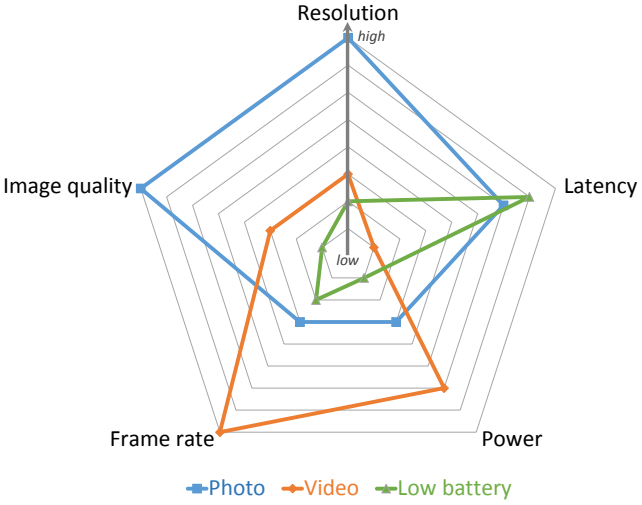

Figure 1: Illustration of the use-cases and required performances.

At the same time, it is commonly expected that the modern multi-sensor vision system provides computing capabilities supporting the numerous functionalities as photo capture, face detection, image fusion or moving object tracking [7]. However, it imposes to deal with the different performance requirements in terms of frame rate, frame resolution or processing latency (Fig. 1). Also, the parameters of each sensor can dynamically vary and the number of sensors used at the moment can dynamically change according to the luminosity conditions or the applicative requirements.

This makes the design of computing resources very arduous task, especially considering the context of latency critical application with a limited power budget and space occupation constraints.

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

RACS '17 , September 20–23, 2017, Krakow, Poland

© 2017 Association for Computing Machinery.

ACM ISBN 978-1-4503-5027-3/17/09· · · $15.00
https://doi.org/10.1145/3129676.3129721

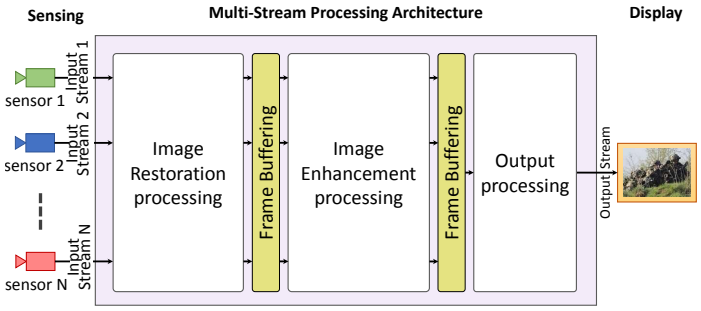

Figure 2: Multi-sensor embedded vision system.

1.1 State of the art

The computationally efficient hardware design for multi-sensor systems (Fig. 2) is a hot research topic that one can illustrate by the numerous publications.

For instance, in [8] the authors develop and implement an FPGA-based, scalable and resource-efficient multi-camera IP core for image reconstruction. In [9] the authors demonstrate an approach to the minimization of the implementation cost of multi-modal sensing systems. Also, a runtime reconfigurable system on chip is proposed in [10, 11], but the authors consider only the utilization of two sensors. And in [12, 13] the authors present an architecture based on network on chip. The main limitations of these solutions remain in the decreasing performance with respect to the sensor number. It is the consequence of the limited scalability of the cited solutions.

Also, they suppose the fixed working parameters set, known in advance. The computing system is then designed for some given trade-off, or, frequently even for the worst-case configuration. Obviously, it results in over-sized computing pipelines (Fig. 3). And, if all the sensors are not used, such solutions become very costly and inefficient.

To overcome these issues, some authors propose to use a principle of the system monitor. It allows observing system state, collecting the application constraints and to decide in the real-time how to use the computing resources.

In the past, some interesting monitoring approaches have been proposed. For instance, in [14, 15], authors present a Multiprocessor System-on-Chip monitoring for frequency scaling. Another monitoring method for Partial and Dynamic Reconfiguration application is presented in [16].

In [17], authors propose a network on chip monitoring based on programmable probes.

However, these solutions are not directly applicable to the

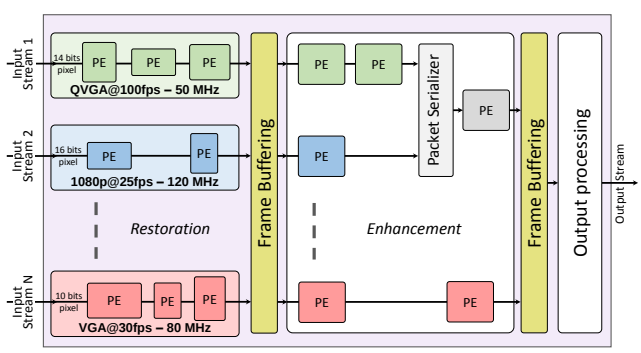

Figure 3: Pipelined static multi-stream architecture

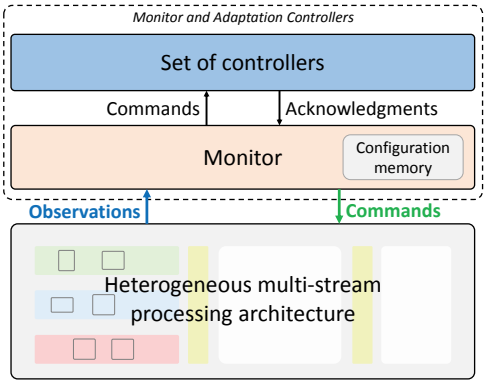

Figure 4: On-Chip monitoring principle

heterogeneous multi-stream architectures and their 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. The key element is a proposal of the network on chip dedicated to system observation and routing of adaptation command.

The paper is organized as follows. Section II presents the proposed Monitoring solution. Performance evaluation of hardware implementation is given in Section III and IV, the Conclusions resume the main contributions of the paper.

2. ON-CHIP MONITORING SYSTEM

The role of Monitor is used to collect runtime status (*Observations* - OBS) of the architecture (processing resources and hardware controllers) (Fig. 4).

The Observations serve to collect the information required for the decision of an on-the-fly computing resources adaptation. The Monitor compares the observed system status with the required performances and manages the architecture reorganization through adaptation *Commands* (CMDs). During this operation, we have to guarantee the data coherency. Hence, the attention is paid to the stream management and synchronization during the adaptations. Notice that the proposed on-chip monitoring solution withstands multi-pipeline architecture with multiple clocking domains.

According to the considered adaptation, the Monitor may have to load configuration data from *Configuration memory*. Then it generates the *Adaptation commands* destined to the processing pipelines or to the hardware controllers of the architecture. Also, the Monitor supports the partial dynamic reconfiguration, the Monitor only needs to check the bit-stream memory address in the *Configuration memory*.

The Monitor communicates with processing pipelines via an original Network on Chip dedicated to the system monitoring and adaptation (Fig. 5). Each processing element (PE) of is bound to a router. 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 the OBS channel while the Monitor sends commands to the pipelines through CMD channels. OBS and CMD channels of R_M

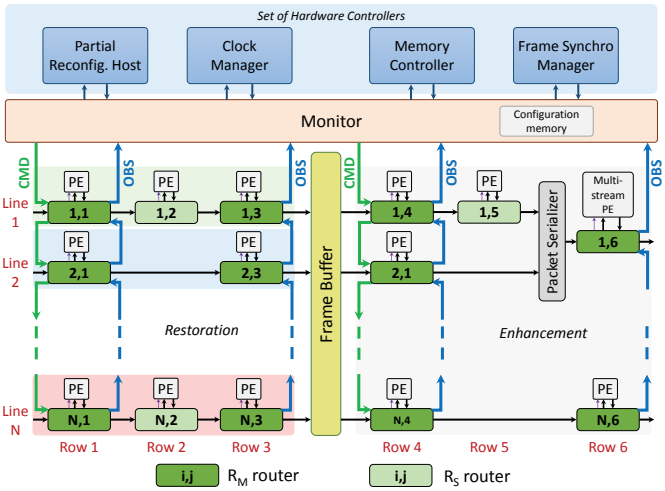

Figure 5: Monitoring system based on dedicated network on chip.

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 neighbor 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 neighbor router until reaching the target PE.

The number of PEs and pipelines in figure 5 is 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 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 where the adaptation commands are encapsulated into the data stream header [13]. 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, quite similar to ALTERA Avalon or XILINX AXI4 stream interface.

2.1 Monitor communication protocol

A Start and Stop signals indicate respectively the beginning and the end 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. The Ready signal is used as a back pressure signal to prevent data loss. By the way, using a back pressure signal reduces buffer memory footprint.

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

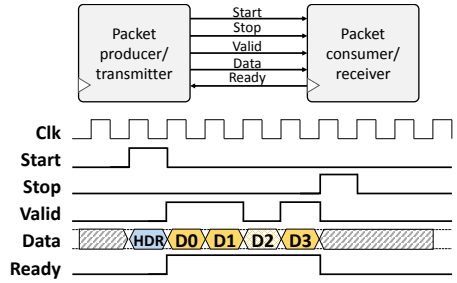

Figure 6: Communication interface and protocol

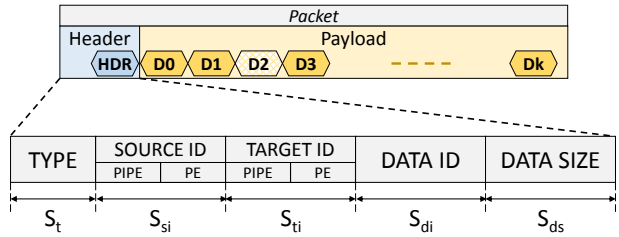

Figure 7: Packet header description

(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.

2.2 Routers

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 the 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 supplementary configurations are possible: FWD_{CMD} and FWD_{OBS}. They are used to

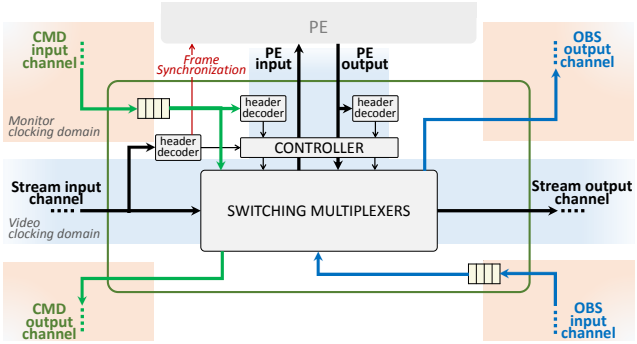

Figure 8: R_M router internal structure

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.

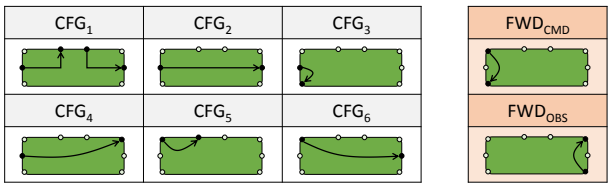

Figure 9: Set of routers configurations

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. The router's configuration will depend on the *Type* of the packet. If the type is PIX, the router is configured as CFG_1.

If it is an OBS packet, the router checks whether the OBS Output channel is 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 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.

Notice that for multi-stream Processing Element, such as color-infrared streams fusion, a Packet Serializer is used to buffer and interlace both streams (Fig. 5). 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 general router.

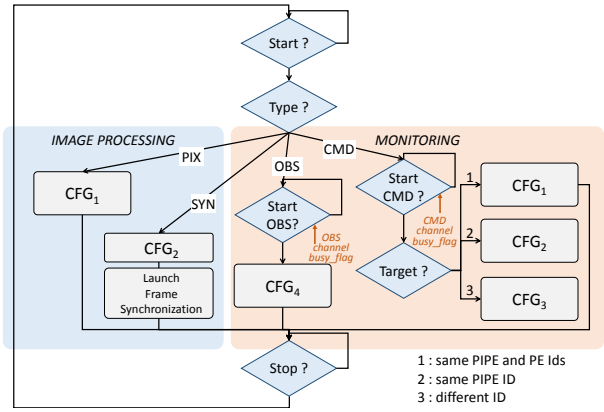

Figure 10: R_M routing mechanism

3. PROTOTYPING AND EVALUATION

We implemented the proposed solution in an ALTERA Cyclone V FPGA (5CGXFC7D6F). Thereafter, we evaluated the performance of this solution through the two use-cases taken from Fig. 1.

1: Runtime frame characteristics adaptation

Context: The application requires the sensor frame rate or resolution adaptation.
Observation: Present sensor characteristics.
Adaptation: Monitor decides and initiates an on-the-fly pixel clock frequency adaptation.
Controller: Frame synchronization manager for PLL[3] reconfiguration.

When the environment context changes (i.e. luminosity condition), the application could change the set of the used sensors and/or their appropriated parameters. Consequently, the characteristics of the input stream, especially the frame rate and the resolution need to change on-the-fly.

Instead of scaling the architecture's Processing Element with the highest worst-case frequency, this solution allows scaling dynamically 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 required pixel clock frequency of a given pipeline of the architecture. Then, the Monitor drives the fine-tuning of the current clock frequency if its value does not fit with the required one.

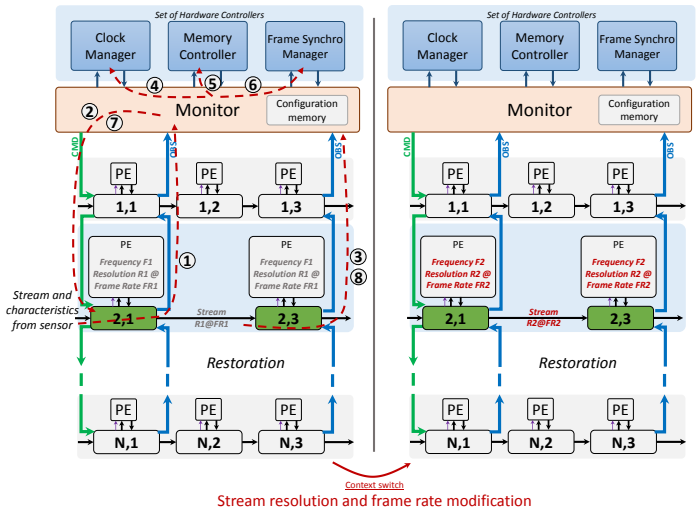

Figure 11: Stream frame rate and resolution adaptation principle

The adaptation of the clock frequency consists in reconfiguring the PLL corresponding to the concerned clock. The Monitor also manages the frame synchronization and before any frequency adaptation, it sends a command to the concerned pipeline to freeze the communication interfaces of the PEs.

1. Observation of the sensor characteristics from the input stream and evaluation of the adaptation requirements.
2. Sending the freezing command toward PEs of the concerned pipeline.
3. Acknowledgement of Freezing operation success.
4. New required frequency computation and frequency adaptation command toward the Clock Manager (then PLL Reconfiguration).

[3]PLL - Phase-Lock Loop

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 acknowledgement.

2: Runtime sensor type switching

Context : New sensor connected to the system, i.e. switching between sensor types in the application.
Observation : Processing pipeline characteristics, sensor specific informations.
Adaptation : Monitor initiates dynamic re-allocation of computing resources.
Controller : Partial Reconfiguration Host of FPGA.

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 nightly. 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 reusable for the other active sensor. In this 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.

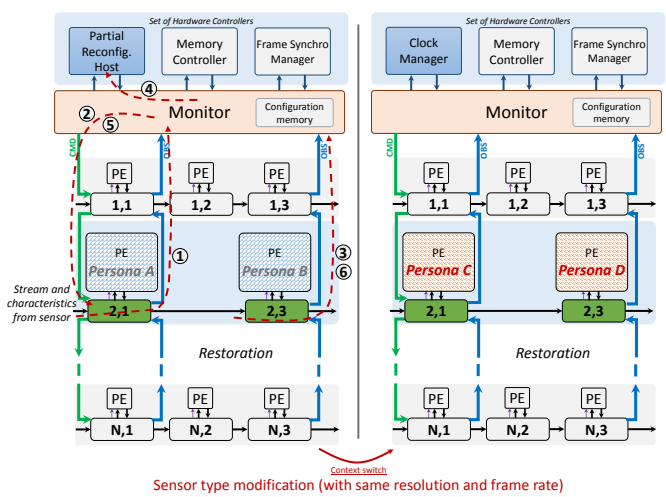

Figure 12: Use-case 2 : stream type modification

① Observation of the characteristics of the sensor from the input stream.

② Freezing command toward PEs of the concerned pipeline in case of any characteristic modification.

③ Acknowledgement of Freezing operation success.

④ Monitor decides of the new required processing context and sends the 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 acknowledgement.

For evaluation purpose, we simulated luminosity condition switching scenarios (day, evening, night). When the luminosity condition changes, the Monitor checks the current active sensors. If the required sensor is not active, it shifts the sensor and adapts the sensor-specific image pre-processing pipeline by means of partial dynamic reconfiguration.

4. LATENCY COST EVALUATION

The proposed solution has been described in HDL and evaluated with an 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.

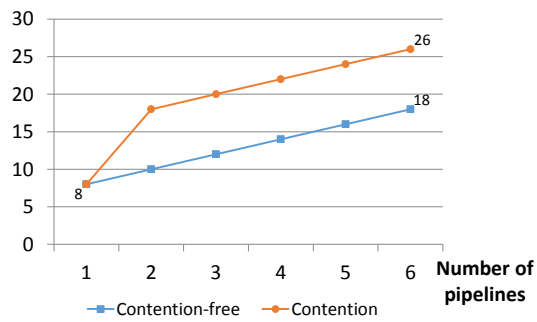

Figure 13: Monitoring packet routing latency

Any packet crosses a R_M or R_S router with a minimal 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 is presented. We evaluated the latencies in the 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. For 3 pipelines-based multi-stream architecture, we have a worst-case latency of 20 cycles. That is to say, a CMD or OBS packet from/to the Monitor takes at most 20 cycles to reach the farthest PE.

Besides, in addition to the interlacing operation latency, the Packet Serializer adds an extra two cycles latency.

4.1 Synthesis results

The synthesis results are based on Altera Cyclone V FPGA (5CGXFC7D6F) implementation with a 32 bit data size. The header fields sizes of this implementation are given in table 1. The area overhead of the presented monitoring solution is given in table 2.

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

Table 1: 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. The 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 (%)	6.7%	2.9%	0.9%

Table 2: Monitoring solution area overhead

The architecture of the proposed solution is directly scalable in the terms of the sensor number as well as in the terms of the number of processing stages. It is obvious that the surface overhead due to the routers will increase linearly with the size of the monitoring network. Notice that the monitor router has the capability to manage simultaneous command and observation packets.

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 bit 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 3: Frequency performance (MHz)

Frequency performance of the proposed solution is presented in table 3. Typical multi-stream architectures have 3 or 4 pipelines. Results in table 3 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 frames per second.

5. CONCLUSION

In this paper, we introduced an original and scalable 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 clock 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 data path flexibility.

6. REFERENCES

[1] Y. Yuan, H. Xu, Z. Miao, F. Liu *et al.*, "Real-time infrared and visible image fusion system and fusion image evaluation," in *Photonics and Optoelectronics, 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, International Congress on*, 2012.

[3] K. Hisatomi, M. Kano, K. Ikeya, M. Katayama *et al.*, "Depth estimation using an infrared dot projector and an infrared color stereo camera," *IEEE Trans. Circuits Syst. Video Technol*, 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 *IEEE Intelligent Vehicles Symposium*, June 2008, pp. 132–137.

[6] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit *et al.*, "Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems," in *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 Int. Conf. on*, 2011.

[8] O. W. Ibraheem, A. Irwansyah, J. Hagemeyer, M. Porrmann *et al.*, "A resource-efficient multi-camera gige vision ip core for embedded vision processing platforms," in *International Conference on ReConFigurable Computing and FPGAs*, Dec 2015.

[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] L. Chen and Y. Jia, "A parallel reconfigurable architecture for real-time stereo vision," in *2009 International Conference on Embedded Software and Systems*, May 2009, pp. 32–39.

[11] 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*, France, 2006, p. 10pp.

[12] C. Ttofis, C. Kyrkou, and T. Theocharides, "A low-cost real-time embedded stereo vision system for accurate disparity estimation based on guided image filtering," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2678–2693, Sep. 2016.

[13] N. Ngan, E. Dokladalova, and M. Akil, "Dynamically adaptable noc router architecture for multiple pixel streams applications," in *IEEE Int. Symp. on Circuits and Systems*, 2012, pp. 1006–1009.

[14] D. Goehringer, M. Chemaou, and M. Huebner, "On-chip monitoring for adaptive heterogeneous multicore systems," *Reconfigurable and Communication-centric Systems-on-Chip*, 2012.

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

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

[17] L. Fiorin, G. Palermo, and C. Silvano, "A monitoring system for nocs," in *Proc. of the Third Int. Workshop on Network on Chip Architectures.* ACM, 2010, pp. 25–30.