HAL
open science

Dynamically adaptable NoC router architecture for multiple pixel streams applications

Nicolas Ngan, Eva Dokladalova, Mohamed Akil

To cite this version:

Nicolas Ngan, Eva Dokladalova, Mohamed Akil. Dynamically adaptable NoC router architecture for multiple pixel streams applications. Circuits and Systems (ISCAS), 2012 IEEE International Symposium on, May 2012, South Korea. pp.1006 - 1009. hal-00789363

HAL Id: hal-00789363

https://hal.science/hal-00789363

Submitted on 18 Feb 2013

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.

Dynamically adaptable NoC router architecture for multiple pixel streams applications

Nicolas Ngan[1,2], Eva Dokladalova[1] and Mohamed Akil[1]
[1]Laboratoire Informatique Gaspard Monge, Equipe A3SI, Unite Mixte CNRS-UMLV-ESIEE (UMR 8049), France
[2]Sagem, Groupe SAFRAN, Argenteuil, France
Emails: nicolas.ngan@sagem.com; {e.dokladalova, akilm}@esiee.fr

***Abstract*—Modern computing systems for vision have to support advanced image applications. They involve several heterogeneous pixel streams and they have to respect hard timing and area constraints. To face those challenges, an adaptable ring-based interconnection network-on-chip (NoC) has been recently proposed. This NoC is based on a new router architecture, with a dynamically adaptable internal datapath, which allows handling of multiple parallel pixel streams. An original datapath adaptation control is proposed by combining instructions and pixel data to be processed in a single packet. Timing performance and area occupation are evaluated on an FPGA prototype.**

***Index Terms*—network-on-chip, router, FPGA, adaptable, image processing**

I. Introduction

Modern embedded vision systems need a performant computing system enabling more and more advanced functionalities and complex applications. Those applications usually involve multiple pixel streams from several heterogeneous image sensors such as low-light, infrared or colour sensors. We can cite for instance panoramic view, picture-in-picture, stereovision [1] or image fusion applications [2]. The cited applications require both spatial and temporal pixel streams management. They also have to meet variable real-time constraints from 25 to 50 frames per second, with a high definition image pixel resolution up to 1080p (1920×1080 pixels).

Despite numerous dedicated architectures for different image processing problems [3], [4], there are few systems considering efficient processing and adaptability solutions for multiple pixel streams applications [5], [6]. In particular, performance bottlenecks remain on interconnections to manage multiple streams and efficient parallel data accesses.

In the last decade, several NoC-based solutions [7] have been proposed for image processing applications [8]. The NoC has been studied both at the functional and architectural level [9] in order to bring more flexibility between processing elements (PEs) communication. However, those solutions remain very complex in the context of multiple pixel streams management and application switching adaptations.

Recently, we have proposed a new adaptable ring-based interconnection network-on-chip for a multi-sensors embedded vision system, called *Multi Data Flow Ring* (MDFR) [10].

To fully exploit that NoC, the router proposed in [11] had to be improved in datapath control transmission. The aim is to avoid a centralized controller, limiting the number of PEs, with dedicated command links to all routers in the NoC.

In this paper, we present a new NoC router architecture which features an original router datapath adaptation control. Datapath adaptations are specified by instructions combined with pixel data in a single packet.

The paper is organized as follows. Section II is a presentation of the NoC principle. Then, the dynamic datapath control is described in Section III. The router architecture proposition is presented in Section IV. Finally, timing and area performance are evaluated in Section V for an FPGA prototype.

II. Dynamically adaptable NoC for a multiple image sensors system

We present in figure 1, an exemple of using our proposed NoC architecture with three different image sensors at the input and two image displays at the output of the processing system.

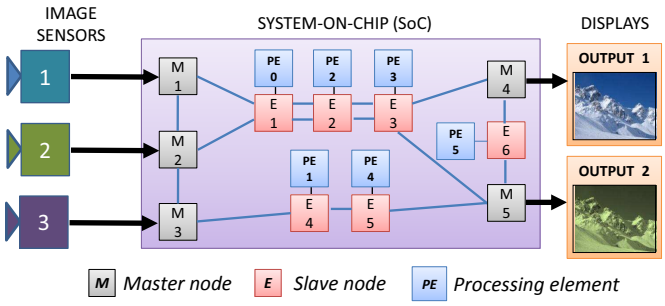


Fig. 1. Principle of using the proposed NoC architecture

This figure shows input image sensors and output displays connected to the NoC through master nodes (M), which represent interfaces to the outside of the embedded computing system. Then, a master can acquire an input pixel stream, or send a pixel stream to a connected display. Inside the network, pixel streams are processed by PEs, which are integrated in the network through slave routers (E) positioned between master nodes. Let us consider that each PE is dataflow-oriented and can process pixel streams with internal pipelined computing structures.

The main role of a slave router (E) is to analyse input pixel data packets and to redirect those packets towards its PE, depending on running image processing applications.

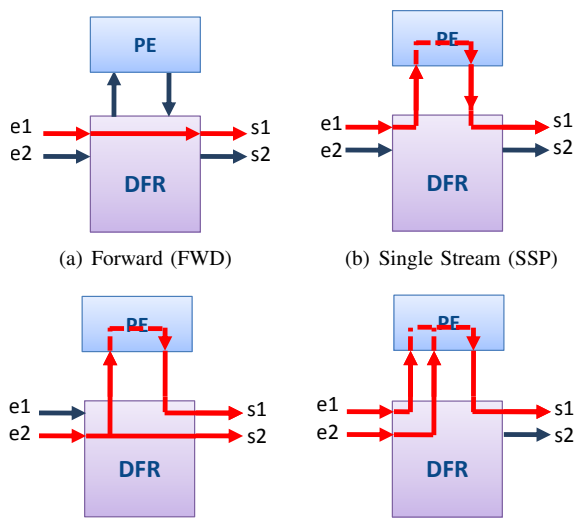
This paper is focused on an architecture proposal for the slave router (E) called *Data Flow Router* (DFR).

A. Data Flow Router operating modes

In order to switch applications involving multiple pixels streams, a DFR has to dynamically adapt its internal datapaths between input and output ports, while avoiding pixel data collisions.

Thus, we define four different predefined operating modes in a DFR: (a) *Forward* (FWD), (b) *Single Stream* (SSP), (c) *Single Stream & Forward* (SSF) and (d) *Multi Stream* (MS).

Figure 2 illustrates exemples for a DFR equipped with two unidirectional input ports $e1$, $e2$ and two unidirectional output ports $s1$, $s2$. This figure highlights, in red colour, the datapath defined for each operating modes.



(a) Forward (FWD) (b) Single Stream (SSP)

(c) SSP & Forward (SSF) (d) Multi Stream (MS)

Fig. 2. DFR operating modes

Forward (FWD) mode, illustrated in figure 2(a), consists in redirecting an input data packet towards an output port without any modification. This mode is activated in the case of the PE is busy (FWD auto), or not able to compute datas with an operation required by the application (FWD check).

Single Stream (SSP) mode, illustrated in figure 2(b), is used when the input packet can be processed by the PE. In this mode, input packet datas are transferred to the PE before being sent out of the DFR.

In *Single Stream & Forward* (SSF) mode, illustrated in figure 2(c), the input packet is duplicated. One copy is directly sent out of the DFR without any modification, and the second one is transferred to the PE. A second output port is selected for sending out the resulting output datas from the PE.

In *Multi Stream* (MS) mode, illustrated in figure 2(d), the PE is able to compute N input pixel streams in parallel ($N = 2$ in our example) to produce one processed pixel stream in a DFR output port.

Those operating modes allow to process several pixel streams sequentially or in parallel, along several DFRs, between two master nodes. Note that MS mode is particularly useful for specific PEs able to combine in parallel pixel streams such as an image fusion or a picture-in-picture operation. Thus, DFR proposition is more optimized for pipelined PE processing compared to generic NoC router solution such as Hermes [12] for instance.

III. Dynamic datapath adaptation

In our NoC, DFR datapath adaptation control is specified as a group of instructions. Those instructions are defined from image applications to be implemented by setting a sequence of operations on pixel datas. According to this sequence, DFRs dynamically adapt internal datapaths in different operating modes.

We propose an original DFR datapath control transmission by combining in a single packet, datapath instructions and associated pixel datas to be processed, as illustrated in figure 3.

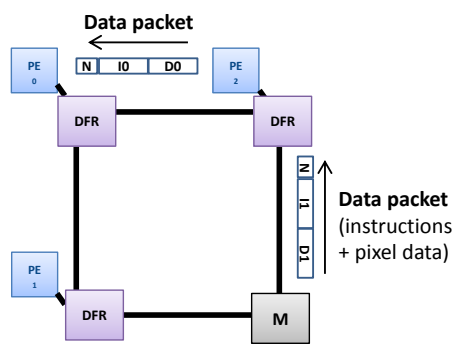


Fig. 3. Communication of packets containing instructions and pixel datas

In this figure, a master node (M) transmits two data packets, containing network adresses `[N]`, instructions `[I]` and datas `[D]`, to be decoded in DFRs. In our NoC, `[H]` and `[C]` are specified in packet headers, associated to pixel datas `[D]` to be processed.

A. Packet header structure

As illustrated in figure 4, the packet header structure is divided in three main parts: a first part of S_p-bits wide, indicating the *packet size*; a second part of S_i-bits wide, reserved to instructions and a last part of S_a-bits wide, describing image attributes. The header is delimited by a *start* and an *end* flit of S_t-bits wide.

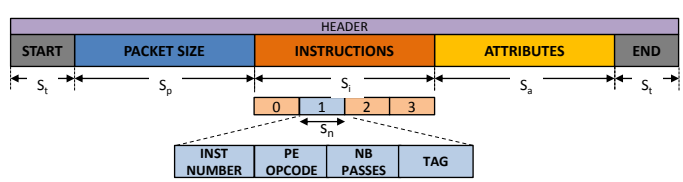


Fig. 4. Packet header structure description

Thus, the complete header of S_H-bit wide, is the sum of all parts as defined in equation 1.

$$S_H = (2 \times S_t) + S_p + S_i + S_a \quad (1)$$

The *packet size* part specifies the number of pixels in the packet by giving the image block resolution value in pixel width X and pixel height Y. In a multiple sensors system, it is critical to be able to differentiate images in the NoC. Thus, the part describing attributes contains image characteristics such as image sensor source id value, an image timestamp ts value and the last image operation pe applied on pixel datas. It also contains network adresses `[N]` of master node source and destination.

B. Instructions

As detailed in figure 4, an instruction is organized in four fields. (1) `[INST NUMBER]` identifies the instruction number in order to ensure consistency between operations on pixels. (2) `[PE OPCODE]` gives the image operation, identified by a defined number, to process pixel data in packet. (3) `[NB PASSES]` can specify the number of iterations required for the image operation given in `[PE OPCODE]`. (4) `[TAG]` indicates the computing sequence of image operations in a group of instructions (sequential or parallel execution).

The combination of several instructions in a same header infer DFR operating modes, in order to build different PE pipeline structures in the NoC.

IV. Data Flow Router architecture

The global DFR architecture is presented in figure 5.

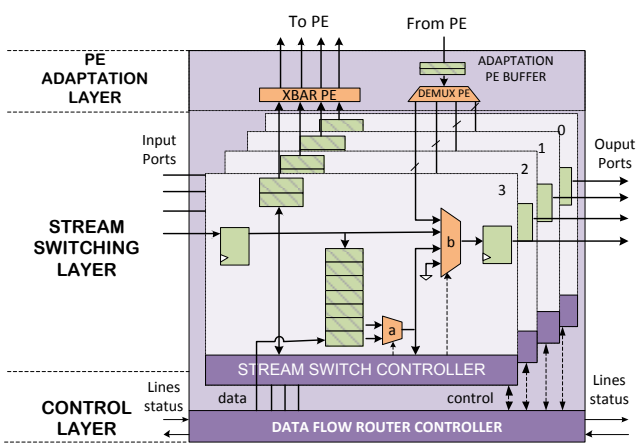


Fig. 5. Global DFR architecture with four input ports

The DFR architecture is organized in three layers: a *stream switching layer* which is a central layer dedicated to switching input packets; a *PE adaptation layer*, representing the interface between the DFR and the PE; and a *control layer*, containing the global DFR control.

The stream switching layer is constituted of several independent units, called *Stream Switches*, which adapt datapaths of input pixel streams to the connected processing unit or to neighbour routers. There are as many *Stream Switches* units as the number of input pixel streams to manage in a DFR.

This layer interacts with the control layer containing the global controller, called *DFR Controller*. The main function of this unit is to control the whole internal DFR datapath adaptation and to arbitrate requests from *Stream Switches* to access the PE.

Depending on arbitration decision and activated operating modes in *Stream Switches*, the controller commands the PE adaptation layer to change the input pixel streams direction. The PE adaptation layer contains selection units implemented as a crossbar (XBAR PE) and a demux units (DEMUX PE).

V. Hardware Prototyping

To validate the architecture proposition and evaluate the performance in time and area, the DFR has been implemented in a FPGA (Altera Stratix III EP3SL150).

A. Header implementation

Table I presents a header size partition for each part. We define a 6-flits header equivalent to 192 bits, with 2 flits dedicated to a group of four 16-bits instructions.

	S_t	S_p	S_i	S_a	S_H (total)
Number of flits	1	1	2	1	6
Size (bits)	32	32	64	32	192

TABLE I
Header size partition

As described in previous section, a 16-bits instruction is divided in four fields presented in table II.

Field	Size	Position	Description
INST NUMBER	4 bits	[15..12]	Instruction line number
PE OPCODE	6 bits	[11..6]	Image operation identification
NB PASS	4 bits	[5..2]	Number of iterations
TAG	2 bits	[1..0]	Sequential/Parallel execution

TABLE II
16-bits instruction structure

With this partitioning, an application can be described by 16 instruction lines with a maximum of 64 distinct types of image operation. Each image operation can be applied on data with a maximum of 16 iterations.

B. DFR area evaluation

Table III presents the FPGA area occupation in a Stratix III EP3SL150 for a DFR with a 6-flits packet header structure.

	LEs	Registers	Mem (bits)	(% FPGA)
Stream Switch	294	246	100	0.25
DFR Controller	226	130	0	0.2
PE Adapt	21	10	224	0.01
DFR (total)	1154	1738	624	1.5

TABLE III
Area : Data Flow Router (EP3SL150)

It shows the number of Logic Elements (LEs), the number of registers, the on-chip-memory (Mem) size required and the area occupation ratio % in targeted FPGA.

For this prototype, the number of *Stream Switches* is set to k=4, with a bus size set to n=32 bits. A 32-bits bus size is sufficient for transmitting any pixel granularity from an 8-bit greyscale to a 24-bit RGB image.

In this configuration, a complete DFR occupies less than 2 % of FPGA with a maximum frequency at 234 MHz.

C. DFR adaptation latency evaluation

This section evaluates the total latency L_R in cycles between the first flit input and output time in a DFR. Depending on DFR operating modes and the PE of L_{PE} latency, L_R value cannot be deterministic. Let us consider δ_h latency in cycles between the last header flit and the first data flit, and δ_T latency between two input packets to be combined in a PE.

Figure 6 shows a timing diagram describing a transmission of two input packets, with a 2-flits H0-H1 header, in a DFR. Assuming that the PE is not busy, the Stream Switch 0 (SS0) requests to process its input packet in a Single Stream (SSP) operating mode and transmits the packet with a latency of $L_R(SSP)$ cycles. During that packet processing by the PE, any packet in the other Streams Switches cannot be processed and are directly transfered outside the DFR such as the packet in SS1 in this example, with a latency of $L_R(FWD)$ without any header analysis.

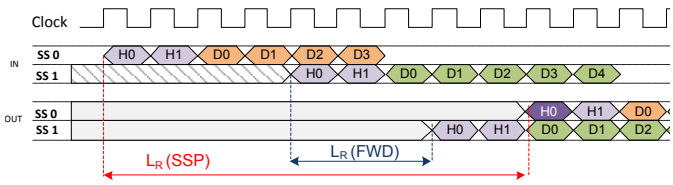


Fig. 6. Timing diagram for *Single Stream* and *Forward* DFR operating modes

Table IV presents the measured latencies for outputting a packet in each operating modes without the arbitration time equivalent to one clock cycle in this implementation. The table details the datapath adaptation latency L_a and the global interface latency L_i in cycles. The global interface latency L_i depends on registers placed at input and output ports in DFR.

Mode	L_a (cycles)	L_i (cycles)	L_R (cycles)
Forward (auto)	0	2	2
Forward (check)	5	2	7
Single Stream	6	6	12+L_{PE}+δ_h
SSF (packet 1)	5	2	7
SSF (packet 2)	6	6	12+L_{PE}+δ_h
MS	6+δ_T	6	12+L_{PE}+δ_T

TABLE IV
DFR PACKET PROCESSING LATENCIES

In this table, there is a distinction in *Forward* mode depending on packet analysis requirement. In *Forward* (auto) mode, packets are bypassing the analysis and the DFR traversal latency is minimal and equivalent to 2 clock cycles. If the PE is available, 5 additional clock cycles are necessary to analyse packet header, in *Forward* (check) mode.

In *Single Stream* mode, L_a is 6 clock cycles including 1 cycle for requesting datapath modification and 5 cycles to decode header. L_i is more important because of PE access and L_R is given with an additional latency δ_h.

The *Single Stream & Forward* mode generates two packets. The first one, described as packet 1 in table IV, is transmitted without any modification and the second one, as packet 2, is processed by the PE. Note that, for packet 1, L_R latency is equal to *Forward* mode with header analysis.

Finally, the *Multi-Stream* mode is obviously dependent on the timing difference δ_T, in cycles, between the arrival of all necessary packets to be combined in parallel by the PE.

From those values, we can conclude that the proposed DFR architecture can output a packet with a L_R latency value from 2 to 12 clock cycles without considering the PE latency. As our NoC is data-flow oriented with multiple PE pipelines, DFR latencies become negligible compared to the data size in packets representing frame blocks or complete frames. Those pipelines allow to respect real-time image application constraints with a high bandwidth between PEs.

VI. Conclusion

In this paper, we have presented a dynamically adaptable NoC router, called Data Flow Router, which is able to manage several pixel streams in parallel. Depending on an original instructions integration in packet headers, the DFR dynamically adapts its internal datapath in order to transmit efficiently pixel streams by building pipelines between PEs in the NoC. FPGA prototype evaluations have been presented for a DFR architecture proposition and show fair performance in timing and area for implementation. Future works will focus on DFR performance evaluation in a complete NoC for different image applications.

References

[1] L. Chen and Y. Jia, "A parallel reconfigurable architecture for real-time stereo vision," may. 2009, pp. 32 –39.

[2] R. S. Blum and Z. Liu, *Multi-Sensor Image Fusion and Its Applications*. CRC, 2005.

[3] A. C. Bovik, *Handbook of image and video processing*. Academic Press, 2005.

[4] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. Morgan Kaufmann, 2007.

[5] S. Chikamatsu, T. Nakaya, M. Kouda, N. Kuroki, T. Hirose, and M. Numa, "Super-resolution technique for thermography with dual-camera system," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 30 2010-june 2 2010, pp. 1895 – 1898.

[6] T.-Y. Cheng, T.-H. Chen, J. Chen, and S.-Y. Chien, "Coarse-grained reconfigurable image stream processor architecture for high-definition cameras and camcorders," in *SoC Design Conference (ISOCC), 2010 International*, nov. 2010, pp. 95 –98.

[7] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684 – 689.

[8] V. Fresse, J. Tan, and F. Rousseau, "Exploration of an adaptive noc architecture on fpga dedicated to multi and hysperspectral algorithm for art authentication," jul. 2010, pp. 529 –534.

[9] F. Clermidy, R. Lemaire, Y. Thonnart, and P. Vivet, "A communication and configuration controller for noc based reconfigurable data flow architecture," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, May 2009, pp. 153 –162.

[10] N. Ngan, E. Dokladalova, M. Akil, and F. Contou-carrere, "Dynamically adaptable architecture for real-time video processing," in *IEEE International Symposium on Circuits and Systems (ISCAS'10)*, 2010.

[11] N. Ngan, G. Marpeaux, E. Dokladalova, M. Akil, and F. Contou-carrere, "Memory system for a dynamically adaptable pixel stream architecture," in *IEEE International Conference on Field Programmable Logic and Applications (FPL'10)*, 2010.

[12] F. Moraes, N. Calazans, A. Mello, L. Mller, and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69 – 93, 2004.