Real-time Dynamically Reconfigurable 2-D Filterbanks

Daniel Llamocca and Marios Pattichis
Electrical and Computer Engineering Department
The University of New Mexico
Albuquerque, NM, USA
dllamocca@ieee.org, pattichis@ece.unm.edu

***Abstract*— We introduce a novel dynamically reconfigurable 2D filterbank that is based on separable, one-dimensional filters. At the lowest level, each 2D filter is implemented using dynamic reconfiguration between two one-dimensional filters. Then, at a higher level, filterbanks are implemented using dynamic partial reconfiguration of efficient 1D filter blocks (based on distributed arithmetic). We have evaluated the system's overall performance as a function of the reconfiguration rate and we show that real-time video processing speeds for 2D video filtering can be attained for several image sizes.**

Keywords: FPGA, dynamic partial reconfiguration, filterbank hardware implementation

I. INTRODUCTION

A common filterbank implementation often requires to have all its 2D filters present in the hardware and to select them via multiplexers. A filterbank may be implemented using internal multiplier and adder circuits (where coefficients can be modified) or by a fixed-coefficients circuitry. We will refer to these approaches as static implementations, since the hardware remains unchanged. Static implementations exhibit several limitations. As the number of coefficients or the number of filters grows, there may not be enough hardware resources for efficient implementation. Moreover, energy consumption is often a function of the implemented static hardware regardless of whether the hardware resources are actually needed.

Dynamic Partial Run-Time Reconfiguration (DPR) addresses the aforementioned problems by time-multiplexing FPGA resources [1]. This technique does not require the device to be turned off prior to reconfiguration. DPR allows for a portion of the device to be reconfigured, providing significant time and power savings over static reconfiguration approaches.

Embedded hardware image processing systems have been widely reported (e.g.: [2, 3, and 4]). FPGA implementations of multiplier-based 2-D filters have also been reported in [5] and [6]. These realizations are static implementations. The use of DPR has been reported for DSP applications in [7, 8, and 9]. However, embedded image processing systems that employ DPR are yet to be found.

Dynamically reconfigurable 1D filters were shown in [7, 10]. In [10], a multiply and add dynamic filter was implemented. In [7], we showed the highly efficient implementation using distributed arithmetic. In this paper, we present a dynamically reconfigurable system for implementing 2-D separable filterbanks.

Our approach only requires that the device has sufficient resources to implement a single 1-D filter at any given point in time. A portable VHDL description allows us to pre-compute 1D filter blocks that use different number of coefficients, different coefficients, and filtering structure. This is an extension of our earlier version of the 1D filter that only allowed changes in the coefficients [7].

The filter reconfigurations are handled inside the FPGA, allowing the system to run automatically with no external control, unlike full static reconfiguration. The reconfiguration time overhead can be reduced significantly using newly developed reconfiguration controllers [11, 12].

The rest of the paper is organized as follows: Section II explain the methodology followed to design the filterbank. Section III provides results in terms of resource utilization and throughput as a function of the reconfiguration rate. Section IV summarizes the paper and discusses further work.

II. METHODOLOGY

A. System Block Diagram

Fig. 1 depicts the system block diagram. The PowerPC (PPC) processor and the dynamic Filter core are linked the high speed Fast Simplex Bus [13]. The dynamic portion of the filter core is called the Partial Reconfiguration Region (PRR) and holds a specific filter realization. The PRR can be dynamically reconfigured via the internal configuration access port (ICAP) driven by the ICAP controller core [14]. The Double Data Rate Random Access Memory (DDRRAM) stores volatile data at run-time, e.g.: input

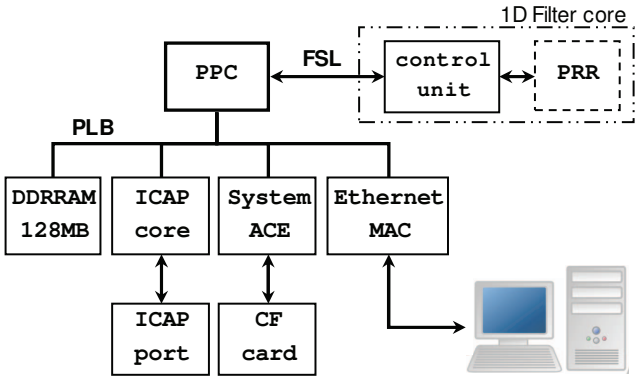

Figure 1. System Block Diagram

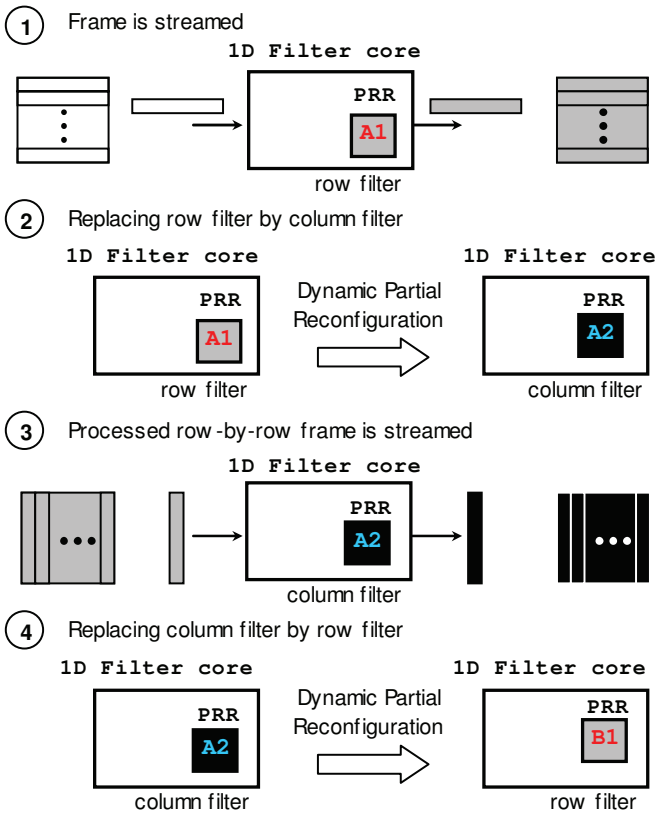

Figure 2. Dynamically Reconfigurable 2-D Filterbank using partial reconfiguration of 1-D modules.

frames, processed frames, and partial bitstreams. SystemACE [15] is used for reading a Compact Flash (CF) card that stores the partial bitstreams at power up. The Ethernet core provides reliable communication with a PC, and allows us to read new partial bitstreams or new input frames, and write processed frames back to the PC.

B. 2-D Filterbank implementation with DPR

Since we work with separable 2-D filters, we implement a 2-D filter by working with one 1-D filter at a time, as Fig. 2 depicts. The modification of the one-dimensional Filter is carried out via DPR, which modifies only the 1-D filter circuit. This 1D circuit is an efficient and highly parameterized filter. The one-dimensional filter implementation is based on distributed arithmetic and requires very few hardware resources [7]. Dynamic reconfiguration allows for an efficient implementation of any separable filterbank while requiring few hardware resources for a single 1-D filter. Furthermore, dynamic reconfiguration allows us to modify the input and/or output bitwidth as well as the number of coefficients and its values.

Before explaining the 2-D filterbank procedure, it is important to remark the 1-D filtering process in the context of Fig. 1: The PPC reads data from DDRRAM, then streams data to the 1D filter core, and finally grabs the output of the 1D filter core and writes them back to DDRRAM.

The 2-D Filterbank processing scheme is shown in Fig. 2. The process is as follows:

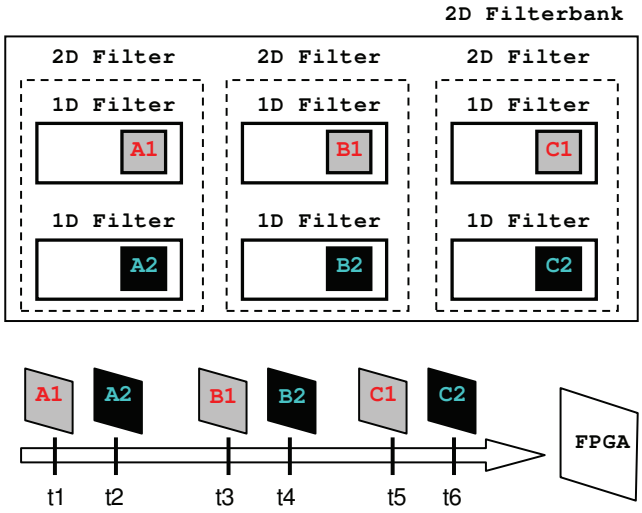

Figure 3. 2D filterbank implementation by time-multiplexing 1D filters.

- Step 1: We start by filtering all the rows and storing the processed pixels back in memory.
- Step 2: We reconfigure the 1D filter core with the corresponding column filter.
- Step 3: We filter the processed pixels from Step 1 through the 1D column filter. Here we get the final filtered frame in DDRRAM.
- Step 4: We reconfigure the filter again with the row filter of the next 2-D filter of the filterbank. Then proceed to Step 1. If all the filters have been applied, we reconfigure the filter with the row filter of the first 2-D filter of the filterbank, and we proceed with a new frame at Step 1.

As we can see, we perform two reconfigurations per 2D filter. This specific implementation was chosen because of its very low hardware and power requirements.

In terms of memory accesses, we note that column filtering is slower than row filtering. Row filtering processed and stores the results in a sequential row-by-row fashion. Column filtering requires that we fetch a single pixel from each row, i.e. memory accesses are not sequential.

Fig. 3 depicts how a filterbank with 3 filters can be realized using 1D filters. We start by filtering a frame with the 2-D filter defined by the set {A1, A2}, then with the filter {B1, B2} (B1 inserted at Step 4 and B2 at Step 2 in Fig. 2) and finally with the filter {C1, C2}. After the frame has been filtered by the three filters, we start over with a new frame. We can see that the filterbank is implemented as a collection of 1D filters that are multiplexed in time.

III. RESULTS

A. Platform testing scheme

We use the ML405 Development board that contains a XC4VFX20-11FF672 Xilinx® Virtex-4 device. The PPC is clocked at 300 MHz and the peripherals run at 100 MHz. In order to improve performance, the DDRRAM memory space is cached. We employ several sets of grayscale 40-frames videos, each set with a different frame size.

TABLE 1. HARDWARE UTILIZATION ON VIRTEX-4 XC4VFX20-11FF672

Module	FF	(%)	Slice	(%)	LUT	%
PRR (columns)	1852	11%	1759	21%	1759	10%
Static Region	4019	24%	5521	65%	8078	47%
Overall	5871	34%	7280	85%	9837	58%

TABLE 2. RECONFIGURATION TIME FOR A 124KB BITSTREAM

Scenario	Reconfiguration Speed	Reconfiguration Time
1. Current	3.23 MB/s	38.39 ms
2. Custom [11]	180 MB/s	0.69 ms
3. Custom [12]	295.4 MB/s	0.42 ms
4. Ideal	400 MB/s	0.31 ms

Results are shown under the following conditions: Both 1D filters have 16 coefficients with 16-bit width. The row-filter has an 8-bit input and a 16-bit output. The column-filter has a 16-bit input and a 16-bit output width. The PRR occupies a tightly packed area of 20x96 = 1920 slices (124000 bytes bitstream). This PRR holds the column-filter (the larger one), so that any 1D filter with the previously mentioned parameters can fit. We consider four 2-D filters.

B. *Hardware resource utilization.*

Table 1 shows the hardware resource utilization of the static region, dynamic region, and the whole system. The static region includes the static portion of the FIR filter and all the peripheral controllers shown in Fig. 1. As it can be seen, the static region takes most of the FPGA fabric. The reason for this is the large size of the DDRRAM controller. And the PRR (our 1D filter) takes about 20% of the device.

C. *Reconfiguration Time*

Table 2 shows the reconfiguration time for 4 scenarios. In the basic setup [16], called Scenario 1, we used the Xilinx® ICAP core and obtained a reconfiguration time of 38 ms yielding a reconfiguration speed of 3.23 MB/s. We also consider improved reconfiguration rates based on a custom embedded controller [11] (Scenario 2). Similar results are reported in [12] (Scenario 3). The dramatic improvement in reconfiguration speed lies on the use of a custom ICAP controller, Direct Memory Access, and burst transfers. Scenario 4 is the maximum theoretical throughput, which for Virtex-4 is 400 MB/s [16].

D. *Throughput measurements.*

The operation of the circuit for throughput measurement purposes is as follows: We first place the partial bitstreams and input frames on DDRRAM. An input frame is streamed through the row-filter core, and the resulting frame is stored in DDRRRAM. This frame is then streamed through the column-filter core, and the final output frame is written back to the DDRRAM. This process is continuously repeated with different partial bitstreams loaded after each frame is processed (either row-wise or column-wise), following the guidelines in Section III.B, so as to get different 2D filters.

Note that the number of 2D filters of the filterbank is irrelevant to the output throughput measurements. The only difference more filters make is that more space is needed in DDRRAM to store the partial bitstreams. On the other hand,

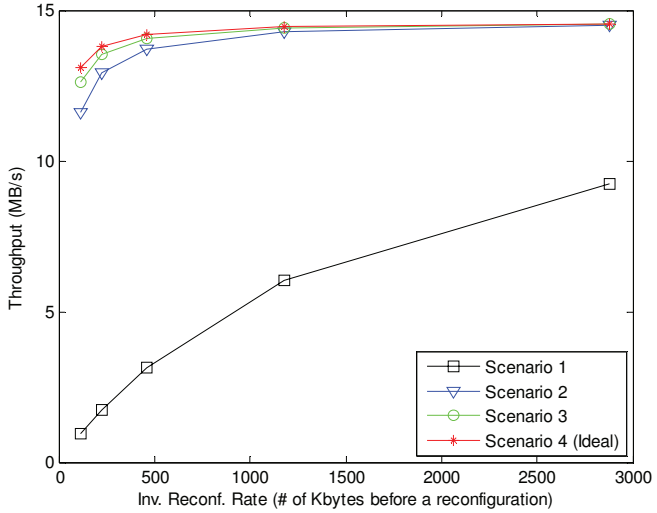

Figure 4. Output throughput against inverse reconfiguration rate

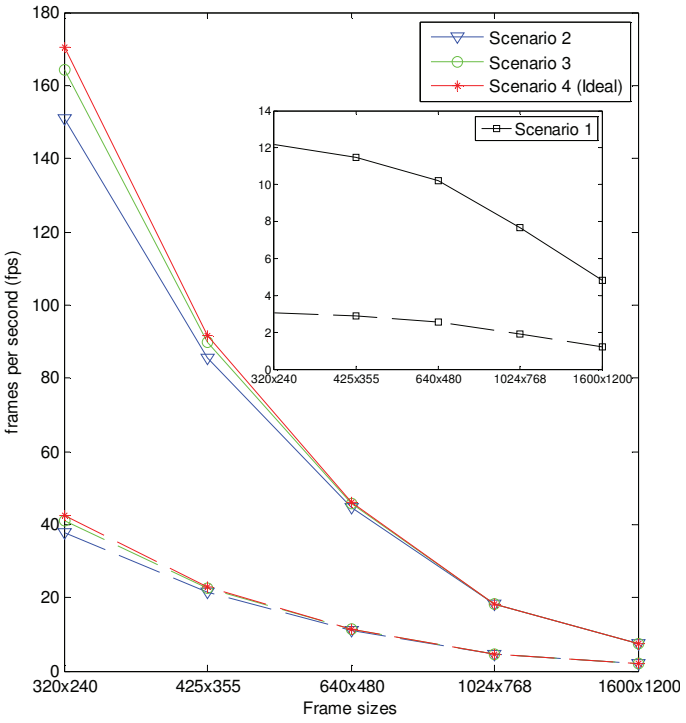

Figure 5. fps for different frame sizes.
Solid lines: one 2-D filter; Dashed lines: 2-D filterbank with four 2-D filters

the number of filters will linearly scale with performance, in terms of processed frames per second. In other words, the use of N filters will reduce the number of processed frames per second by a factor of N.

By using timer functions in the PPC software routine, the processing time for one frame was measured from the moment the input frame is streamed from the DDRRAM until the processed frame is written back in the DDRRAM.

We measure the reconfiguration rate in terms of number of processed bytes before a new reconfiguration is performed. For one filter, we reconfigure twice per frame. Note that if the input frame for the row-wise filtering has SIZE bytes, the processed frame ready for the column-wise filtering has 2*SIZE bytes (recall that input pixels are 8-bit wide and processed pixels are 16-bit wide). As a result, we can say that on average we process 1.5*SIZE bytes before a new reconfiguration.

Note that the only way of adjusting the reconfiguration rate is by modifying the frame size. We define the dynamic reconfiguration rate in terms of the inverse of the number of bytes that are being processed prior to a reconfiguration.

Fig. 4 reports the output throughput attained in all the reconfiguration scenarios for several frame sizes (shown in Fig. 5). The five points plotted for each Scenario corresponds to a particular frame size. However, we found it more meaningful to present the results in terms of the number of processed bytes prior to reconfiguration, which corresponds to the inverse of the reconfiguration rate.

In Fig. 5, we report results in terms of the number of frames per second attained in all reconfiguration scenarios for several frame sizes. Clearly, as the frame size increases, the number of processed frames per second goes down. Two set of graphs are shown. In the first case (group of solid lines), we only consider one 2-D filter. In the second case (group of dashed lines), we consider four 2-D filters, and as expected the fps gets reduced by a factor of four.

Here, we note that for Scenario 2, 3, and 4, we report synthetic results based on our measured static performance and reconfiguration speeds of Table 1.

From Figure 5, we see that for the 2D filters, we have performance levels that exceed 30 fps for most of the cases (Scenarios 2, 3, or 4). This means that real-time video filtering is possible with Dynamic Partial Reconfiguration provided we use a better ICAP controller. In the case of the 2D filterbank, real-time video processing was attained for frame sizes up to 320x240.

IV. CONCLUSIONS

An efficient implementation of a generic 2D filterbank has been presented that employs dynamic partial reconfiguration. The architecture has shown promising results in the sense that it is can process real-time video for frames sizes up to 320x240.

The dynamic implementation was evaluated in terms of performance and resource consumption. It was shown that the dynamic performance depends on the input stream size. In general, the dynamic performance is less sensitive to the effects of reconfiguration time overhead as the stream size increases. Regarding the frames per second, as expected, the performance gets worse as the frame size increases, no matter how negligible the reconfiguration time overhead is.

Further work will focus on: i) reducing the reconfiguration time overhead by using a faster ICAP controller, as exemplified by Scenarios 2 and 3, ii) assessing the power savings attained compared to a multiply-and-add static implementation, and iii) implement and image analysis system. In general, the implementation has shown that real time video processing is attainable for both 2-D filtering and 2-D filterbank processing. The real-time image analysis system will be used to implement AM-FM feature extraction and classification.

ACKNOWLEDGMENT

The research presented in this paper has been funded by the Air Force Research Laboratory under grant number QA9453-060C-0211.

REFERENCES

[1] J. Becker, M. Hubner, G. G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, "Dynamic and Partial FPGA Exploitation", Proc. IEEE, vol. 95, no. 2, pp. 438-452, 2007.

[2] D.G. Bariamis, D.K. Iakovidis, D.E. Maroulis, and S.A. Karkanis, "An FPGA-based architecture for real time image feature extraction", Proceeding of IPRC 2004, Cambridge, United Kingdom, Aug. 2004, pp. 801-804.

[3] J. Diaz, E. Ros, F. Pelayo, E. M. Ortigosa, S. Mota, "FPGA—based real-time optical-flow system", IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 2, pp. 274-279, 2006.

[4] S.O. Memik, A.K. Katsaggelos, M. Sarrafzadeh, "Analysis and FPGA Implementation of Image Restoration under resource constraints", IEEE Transactions on Computers, vol. 52, no. 3, pp. 390-399, 2003.

[5] C.-S. Bouganis, G.A. Constatinides, P.Y.K. Cheung, "A novel 2D filter design methodology for heterogeneous devices", Proceedings of FCCM 2005, Napa, CA, April 2005, pp. 13-22.

[6] A. Madanayake, L. Bruton, C. Comis, "FPGA Architectures for real-time 2D/3D FIR/IIR plane wave filters", Proceedings of ISCAS 2004, Vancouver, Canada, May 2004, vol. 3, pp. 613-616.

[7] D. Llamocca, M. Pattichis, and G. A. Vera, "A Dynamically Reconfigurable Platform for Fixed-Point FIR Filter", Proceedings of ReConFig'09, Cancun, Mexico, Nov. 2009, pp. 332-337.

[8] T. Rissa, R. Uusikartano, and J. Niitylahti, "Adaptive FIR filter architectures for run-time reconfigurable FPGAs", in Proceedings of 2002 IEEE International Conference on Field-Programmable Technology, Hong Kong, China, Dec. 2002, pp. 52-59.

[9] C. Choi, and H. Lee, "A partial self-reconfigurable adaptive FIR filter system", in Proceeding of 2007 IEEE Workshop on Signal Processing Systems, Shanghai, China, Oct. 2007, pp. 204-209.

[10] Y. Oh, H. Lee, C. Lee, "A reconfigurable FIR filter design using dynamic partial reconfiguration", Proceedings of ISCAS 2006, Island of Kos, Greece, May 2006, pp. 4851-4854.

[11] John C. Hoffman, "High-Speed Dynamic Partial Reconfiguration for Field Programmable Gate Arrays", M.S. Thesis, University of New Mexico, Albuquerque, NM, USA, July 2009.

[12] C. Claus, B. Zhang, W. Stechele, L. Braun, M. Hubner, and J. Becker, "A Multi-Platform Controller allowing for maximum dynamic partial reconfiguration throughput", Proceedings of FPL'2008, Heidelberg, Germany, Sept. 2008, pp. 535-538.

[13] "Fast Simplex Link (FSL) Bus Product Specification (DS449)", v2.11a ed., Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124, Jun. 2007.

[14] "XPS HWICAP (DS586)", v1.00a ed., Xilinx Inc., 2100 Logic Drive, San Jose, CA, 95124, Nov. 2007.

[15] "XPS SYSACE (System ACE) Interface Controller (DS583)", v1.00a ed., Xilinx Inc., 2100 Logic Drive, San Jose, CA, 95124, Oct. 2007.

[16] Guillermo A. Vera, "A Dynamic Arithmetic Architecture: Precision, Power, and Performance Considerations", Ph.D. Dissertation, University of New Mexico, Albuquerque, NM, USA, May 2008.