A Fine-Granularity Image Pyramid Accelerator for Embedded Processors

Chun-Jen Tsai[1] and Chiang-Yi Wang [1]

[1] Dept. of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan
cjtsai@cs.nctu.edu.tw

Abstract. Image-pyramid generator is an important component for many computer vision applications. Due to the complexity of image scaling operations, embedded application processors usually generate three- or four-level image pyramids for vision applications. In this paper, we present an image pyramid accelerator for embedded processors that generates image pyramids of up to 24-levels of down-sampled resolutions for the input image. Since image pyramids are crucial for the coarse-to-fine analysis of many computer vision problems, more resolution levels in the image pyramid can improve the parameter-estimation accuracy. Furthermore, the down-sampling filters used in the proposed design is based on a long-tap Sine-windowed Sinc function filter. Therefore, it preserves more image details than other low-complexity filters such as the bilinear or the bicubic interpolation filter. The proposed circuit is verified on an FPGA development board with a Xilinx Kintex-7 device and will be made open-source. The experimental results show that the design is very promising for real-time computer vision applications.

Keywords: image pyramid, down-sampling filter, computer vision, coarse-to-fine processing, hardware-software codesign.

1 Introduction

Coarse-to-fine processing is a typical technique for computer vision and image recognition applications [1][2][3]. For example, for optical flow estimations, one of the most popular methods is to solve the optical flow constraint equation proposed by Horn and Schunck [4]. However, the optical flow equation is a first-order approximation to the spatiotemporal differential change of the image intensity of a moving pixel. The approximation is valid when the motion magnitudes of all objects between two video frames are less than a pixel. However, in practice, this is only true if the frame rate of the video sequence is extremely high, or if the motions are extremely slow. Since most videos are acquired at a frame rate of 30 or 60, to validate the optical flow equation, image pyramids are used to diagonally down-scale an image in the spatial domain, which effectively down-scales the temporal differences between two consecutive frames. Therefore, the motions at low-resolution levels become very small, which can be used as the initial guesses for estimating the motions at higher resolution levels. Most state-of-the-art optical flow estimation techniques are based on the non-linear

estimation of motion vectors under the optical flow constraint in the image pyramid data [5]. The optimization complexity and accuracy of such algorithms depend on the resolution levels and the quality of the image pyramid used. Using more levels of the image pyramids and better down-scalers can simplify the non-linear estimation problem.

Another example of the image pyramid application is for visual recognition. The most successful approaches today for visual object recognition are based on the deep learning convolutional neural networks (CNN). However, when one trains the CNN to recognize the features of some objects, the training data are usually set around a fixed range of resolution scales. Although one can increase the range to train a CNN to capture scale-independent features, for the same amount of training data, increasing the resolution scales of the training data usually means decreasing the training data for other image features. With an image pyramid structure, a CNN can perform better in locating the trained objects with drastic scale differences in an image [6].

In this paper, we present the design and implementation of a HW-SW codesigned image pyramid generator. The main contribution of this paper is that this is the first open-source[1] circuit implementation of a 24-level image pyramid generator that is based on the Sine-windowed Sinc-function (SWS) filters [7]. The SWS filter was adopted in the scalable extension of the H.264/AVC video standard since it preserves image details better than other filter kernels.

The top-level controller of the pyramid generator is a piece of software executed by an embedded processor. A hardwired fine-granularity down-sampling AXI-bus master circuit is responsible for the image scaling task. The processor core is responsible for specifying the input image address, the output image address, and the decimation level(s) to the scaling circuit for image pyramid generation. The proposed system can be flexibly used for various multi-resolution vision applications. The scaling circuit can generate 24 different levels of down-scaled resolutions from an input image. The paper is organized as follows. In section 2, we describe the problem of image pyramids generation and its challenges for hardware architecture design. Section 3 presents the proposed HW-SW codesign architecture. Implementation results and the analysis of the proposed design are presented in section 4. Finally, some conclusions and future work are described in section 5.

2 Problem Formulation

An image pyramid is illustrated in Fig. 1. The input to a pyramid generator is an image with the highest resolution. A down-sampler is used to scale the first level of the image to smaller resolutions. To avoid propagation of sub-sampling errors, low resolution images are always subsampled directly from the original input image. Therefore, multiple down-sampling filters with different decimation rates must be used for a high-quality pyramid generator.

[1] See https://github.com/eisl-nctu/pyramid.

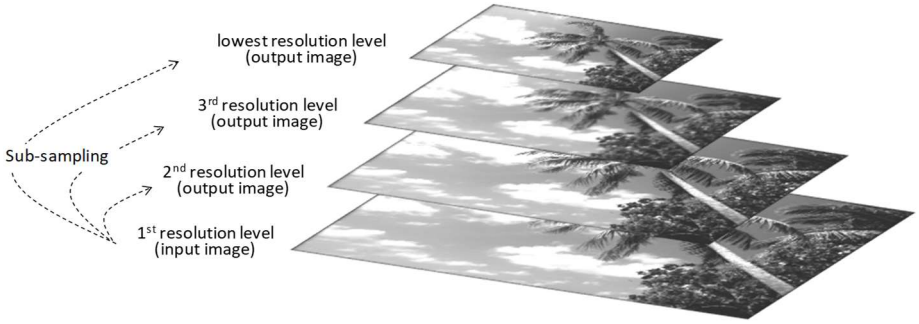

Fig. 1. Image pyramid generation.

There are two main design decisions to make for an image pyramid generator. The first one is about the image filter used for scaling. The second one is about the memory architecture (the on-chip buffering mechanism) for the scaler circuit. We will present brief discussions on each issue in the following two subsections.

2.1 Mathematical Model of the Down-Scaler

An image pyramid allows us to manipulate a digital image at multiple scales. Ideally, given a digital input image, it can be mathematically modelled as a continuous 2-D waveform using a 2-D Sinc filter, and then resampled to different scales. In practice, such operations are often replaced by a 2D convolutional kernel that combines low-pass filtering and down-sampling operations. Popular filters used for this purpose are short-tap Gaussian, bilinear, or bi-cubic filters [9]. However, when the filter tap is short, it can result in loss of detail (high frequency) information. Therefore, more complex filters have been proposed to construct an image pyramid [8].

In this paper, we use a long-tap 2D filtering kernel to design the down-sampling filter. We have chosen to use the SWS filter as the kernel of the proposed image pyramid. The SWS filter was adopted in the scalable extension of the H.264/AVC video coding standard to support spatial scalability of coded video sequences [7]. The mathematical model of the filter kernel is defined in equation (1):

$$f(x) = \begin{cases} \frac{\sin\frac{\pi x}{D}}{\pi x/D} \times \sin\left(\frac{\pi}{2}\left(1 + \frac{x}{N \times D}\right)\right), & |x| < N \times D \\ 0, & otherwise \end{cases} \quad (1)$$

where N is the number of Sinc filter lobes on each side (fixed to 3 here) and D is a decaying parameter that produces different frequency responses of the filter. D is dependent on the decimation rate of the down-scaler. The one-dimensional digital FIR filter derived from equation (1) is a set of 12-tap filters. For 2D image scaling operations, this set of 1D filters can be applied as a set of separable 2D filters. That is, two passes of 1D filtering operations, one in horizontal direction and one in vertical direction, can be used to scale a 2D image.

The SWS filter is an adaptive filter such that the filter weights are adaptive to different scales. The original SWS filter used in the H.264/AVC scalable coding standard only requires three to four levels of kernel coefficients for scalable video coding applications. In this paper, we have designed 24 levels of kernel coefficients based on equation (1) for fine-granularity image pyramid generation.

2.2 Related Work

There have been a few papers on the design of image pyramid generators for FPGAs. Lee [3] proposes an SoC architecture for image pyramid generation that is similar to the one presented in this paper. However, the design was for a low-complexity Gaussian pyramid generator using a 3-tap filter. The implementation was based on a simple AMBA AHB bus for on-chip communications. Agrawwal and Chowdhury [8] present a Laplacian pyramid circuit for image fusion applications, a 5-tap Gaussian filter is used for scaling operations. The design constrains the number of image levels to 4 to serve the needs of the target application. Nuno-Maganda and Arias-Estrada [9] propose a 4-tap bicubic hardware scaler for FPGAs. However, bicubic interpolator tends to overshoot around the sharp edges of an image. Such filters may not be suitable for vision applications that require the preservation of fine image features at different scales. A high-level synthesis (HLS) approach for automatic generation of image pyramid circuits for FPGAs is presented in Schmid et al.[14]. It is known that HLS can generate high performance circuits for image and vision applications at the cost of using more hardware resources. It is difficult to make a direct comparison between the circuit in [14] and the one presented in this paper because different filters are implemented. Non-linear bilateral filters are used in [14], while we have implemented 12-tap SWS filters for 24 different scales. For 3×3 bilateral filters, the proposed circuit in [14] requires 56,978 LUTs while the scaler in this paper used 4,557 LUTs, both on Xilinx 7th generation FPGAs.

2.3 Buffering and Pipeline Considerations of the Pyramid Generators

The main challenges of an image pyramid generation circuit come from the fact that the image data are usually too big to fit in the on-chip memory. Therefore, an efficient buffering scheme must be used to increase the performance of 2-D filtering, otherwise, the data transfer overhead between the main memory (DRAM) and the on-chip memory can be much higher than the processing speed of the circuit. If short-tap filters are used, on-the-fly data transfer and processing can be achieved without using large on-chip buffers [9]. However, the resulting down-scaled images may lose some high frequency details. There are some approaches trying to adopt a short-tap filter (e.g., the bilinear filter) to perform the scaling operations while using the edge detectors and the sharpening filters to recover some of the details [12]. Such scalers may be appropriate for multimedia applications, but not suitable for computer vision applications.

On the other hand, if long-tap filters (e.g., the SWS filters), or adaptive filers (e.g., the Newton filter [11]) are applied in the system, two passes of separable 1-D filtering operations can be used to implement the 2-D filters to maintain burst transfer of data from DRAM to the on-chip buffer [10]. In the proposed design, we chose to use the

long-tap SWS filters since they have been verified thoroughly by the MPEG organization.

Another memory-related decision to make is about the generation order of the image pixels in the pyramid. There are two types of pyramid generation pipelines, the segment pipelines and the linear pipelines [13]. For the segment pipeline, one complete level of down-scaled image is generated after another (as the illustration in Fig. 1 shows). For the linear pipeline, the generator works on the down-scaling operations for all levels of images in parallel (or in a time-sharing manner, as proposed in [13]). Theoretically, the linear pipeline design is more memory bandwidth-efficient than the segment pipeline since an area of the input image can be cached in an on-chip buffer and used to generate the co-located areas in all levels of scaled images before the circuit needs to read in another area. However, the proposed design in this paper chooses the segment buffer approach due to its application flexibility. The proposed design allows for 24 levels of output resolution. Nevertheless, not all vision applications require all 24 levels of scales for processing. In addition, many coarse-to-fine analysis algorithms are iterative across different resolution scales. With the segment pipeline design, it would be possible to adaptively determine the resolution scale for the next level of analysis based on the outcome of the current level. As a result, a segment pipeline may be more memory- and computation-efficient than a linear pipeline in the end.

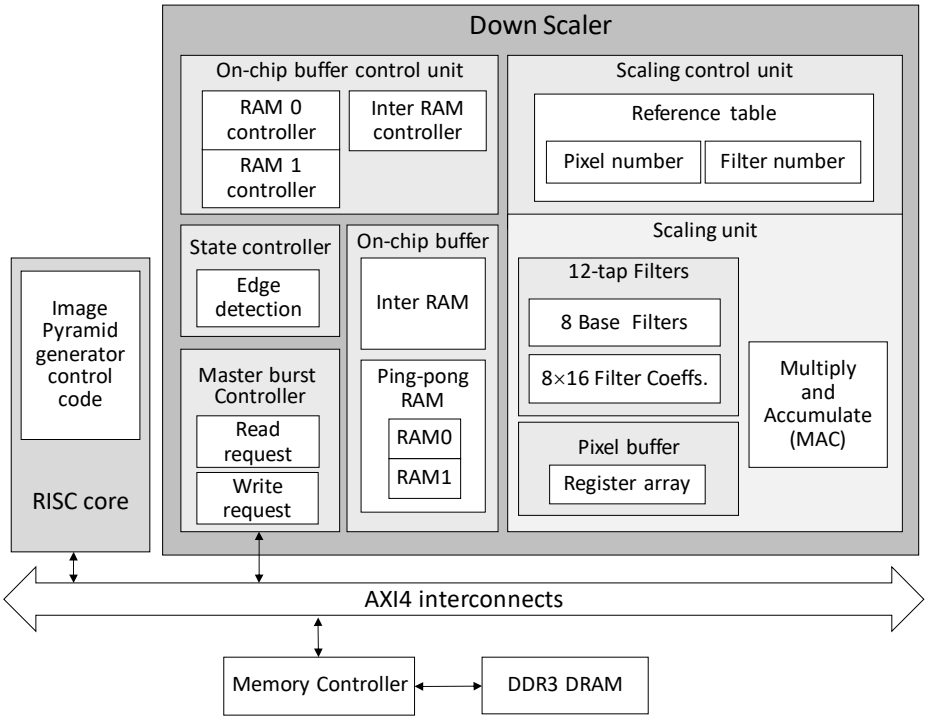

Fig. 2. The overall system architecture of the image pyramid generator.

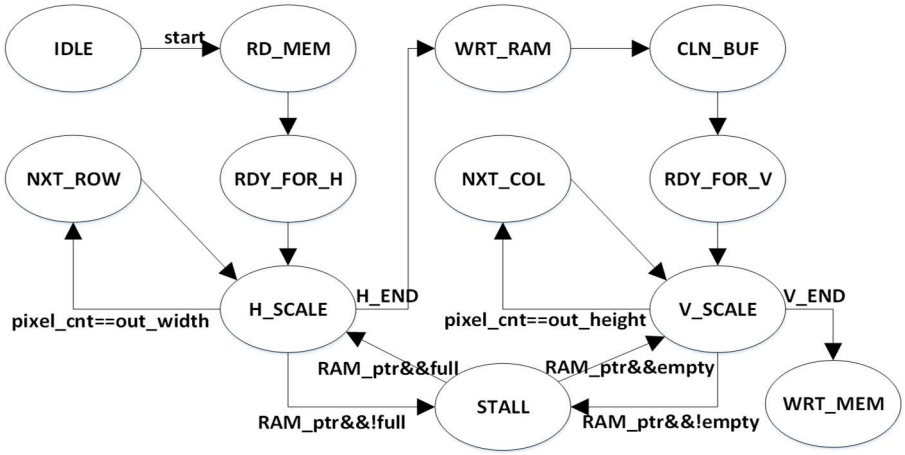

Fig. 3. The main controller of the down-scaler circuit architecture.

3 Architecture Design of the Pyramid Circuit

To generate a fine-granularity image pyramid, the key component would be a set of down-scaler filters with controllable scaling factors that preserve image features as much as possible. In this paper, we present a hardware design of the SWS filters with 24 resolution scales. The filter can generate output images of scaling ratios of $k/32$, where $k = 8, 9, \ldots, 31$. The complete pyramid generator architecture is composed of a RISC core, a down-scaler circuit, and a DDR3 DRAM shared between the RISC core and the down-scaler through an ARM AXI4 bus interconnect, as shown in Fig. 2.

The RISC core is responsible for executing the top-level control code for pyramid generation. It specifies the input/output DRAM addresses of the image pyramid and triggers the execution of the down-scaler with different resolution scales. The down-scaler circuit is a bus-master IP that, once triggered, reads an input image from the DRAM memory and output the down-scaled image back to the DRAM memory. Although it is easy to replace the RISC core with a hardwired controller to create a fully hardwired image pyramid circuit, there will be no performance advantage for such design. We opt for a small RISC core in the proposed design due to its application flexibility. As mentioned in section 2.2, some coarse-to-fine vision algorithms may adaptively determine the resolution scales it needs for analysis. It would be much easier to implement such algorithms if the pyramid generation controller is done in software.

3.1 Controller Design of the Down-Scaler

The finite-state machine of the main controller is shown in Fig. 3. Two separable 1-D horizontal and vertical filters are used to construct the 2-D down-sampler. The horizontal scaling is performed before the vertical scaling. The image pixels stored in DRAM will be fetched into an on-chip buffer for the filtering operations. As soon as there are enough data in the buffer for horizontal scaling, the filtering operations will begin. The output from the horizontal scaling filters will be stored in a second on-chip buffer waiting for vertical scaling. The output from the vertical scaler will be written back to the first buffer for burst output to DRAM.

To save computational resources, the horizontal and the vertical scaling operations share the same processing elements. Thus, the 1-D horizontal and vertical scalers do not operate concurrently. On the other hand, the scaling operations do overlap with the DRAM read/write operations to save time. Since the DRAM bandwidth is often the bottleneck for many embedded systems, there is no point wasting logic resources on concurrent horizontal/vertical scaling operations if the DRAM bandwidth is limited. If one opts for the concurrent operations of the horizontal and vertical filters, the on-chip buffers as well as the processing elements must be increased to maintain efficient operations, which may not be cost effective. The timing diagram of various tasks of the proposed design is shown in Fig. 4. From this diagram, one can see that the memory read/write operations take about the same time as the scaling operations. The overlapping of the memory and the scaling operations can efficiently increase the performance of the circuit. Note that we used a 32-bit memory controller to accesses the DRAM data. There are methods that can improve the DRAM performance, which will be discussed in section 4.

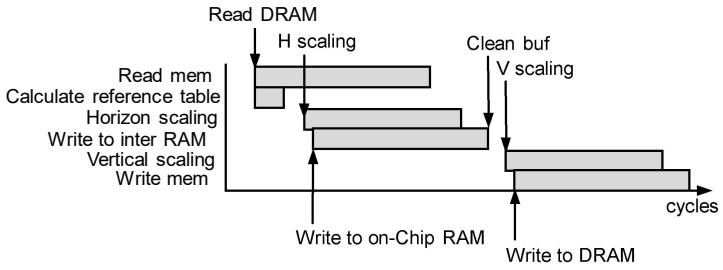

Fig. 4. The timing diagram of the scaler tasks.

3.2 The Memory Architecture of the Down-Sampler

The down-sampling architecture is composed of two parts: the on-chip memory architecture and the datapath of the down-scaling filters. Fig. 5 illustrates the on-chip memory architecture while Fig. 6 shows the datapath of the scaler.

There are two SRAM buffers in the proposed architecture. The first one contains two SRAM blocks that form a ping-pong SRAM pair. This buffer, referred to as the ping-

pong RAM, is used as the input pixel source for the horizon scaling operation. Since it takes a long time to read data from the DRAM, the scaling operation will begin as soon as one of the ping-pong SRAM blocks is filled with data. When the circuit is scaling the pixels in the first SRAM block, the DRAM reading circuit will be filling the second SRAM block of the ping-pong buffer simultaneously. The other SRAM buffer, called the Inter RAM, is used to store the output data for the horizontal scaler as well as the input data for the vertical scaler. Note that the two scalers share the same datapath circuit and do not operate concurrently.

At the vertical scaling stage, the ping-pong RAM is also used to store the output pixels and to overlap the scaling operations with the DRAM write-back operations. In Fig. 5, there is a 32×9-bit reference table that is implemented as a register array. The table is used to select the pixels and the filter coefficients for computing an output pixel. The input to the table is the "down_level" parameter (ranges from 8 to 31, representing the down-scaling rates of 8/32 to 31/32, respectively) and the "pixel_counter" parameter. The outputs are the "filter_number" and the "pixel_number" parameters (to be explained in section 3.3) for the filtering operations. The initialization of the reference table requires 16-bit integer divisions. It is initialized once before the first DRAM read operation and does not impose extra overhead to the scaling operations.

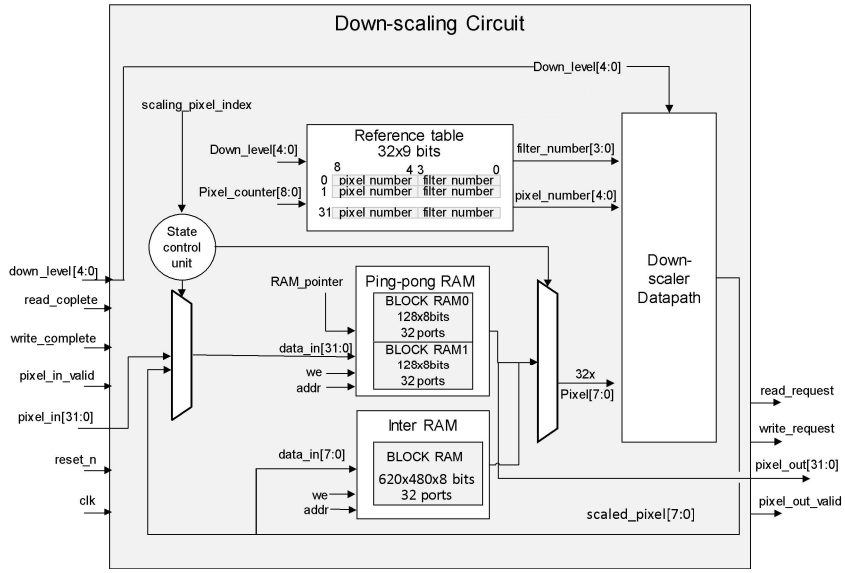

Fig. 5. The on-chip memory architecture of the down-scaler.

3.3 The Datapath of the Down-Sampler

The proposed 2D down-scaler circuit implements a two-pass separable 1D SWS filter based on equation (1). However, only integer arithmetic is used to approximate the SWS filter. As discussed in section 2, each of the resulting 1D filters has a 12-tap

convolutional kernel. The center of the filter is located at the filter coefficient position number 6 (the position index starting at 0). The input image is divided into pixel groups. Each pixel group contains 32 pixels. Once the decimation rate is determined, some pixels in each pixel group will be selected as the center for each filtering operation. For example, if the decimation rate is 11/32, then 11 pixels in each pixel group will be selected as the center of 11 filtering operations. Each operation computes the normalized convolution between the 12 filter coefficients and the 12 pixels around the center pixel (six to the left and five to the right). The filter weights are computed based on equation (1), the decimation rate, and the position of the center pixels.

In the proposed design, we do not compute the filter weights on-the-fly. Instead, the filter weights are pre-computed and stored in on-chip ROMs. The ROM contains eight filter set (for eight different D values) and each filter set contains 16 different 12-tap filter coefficients. Each coefficient is stored as a 9-bit signed integer number. A filter selection logic is used to select the correct filter to use. After each convolution operation, the output must be divided by 128 (7-bit right shift) for normalization purpose.

Fig. 6 is the datapath of the down-scaler architecture. Both the horizontal and the vertical 1-D scalers share the same datapath. The down-level parameter (i.e., the decimation rate) and the filter number parameter are used to select the base filter when the datapath is activated. During the scaling process, the filter_number parameter from the reference table is used to select and adjust the filter coefficients. The source pixel buffer stores several pixel groups (each one contains 32 pixels) of the original image. Since a 12-tap filter is used, only 12 pixels in the buffer centered on the pixel_number position will be used for filtering. If the pixels fall outside the image boundary, the pixel at the image boundary will be used to extend the image. The output pixel will be written to the Inter RAM during horizontal scaling and to the DRAM during vertical scaling.

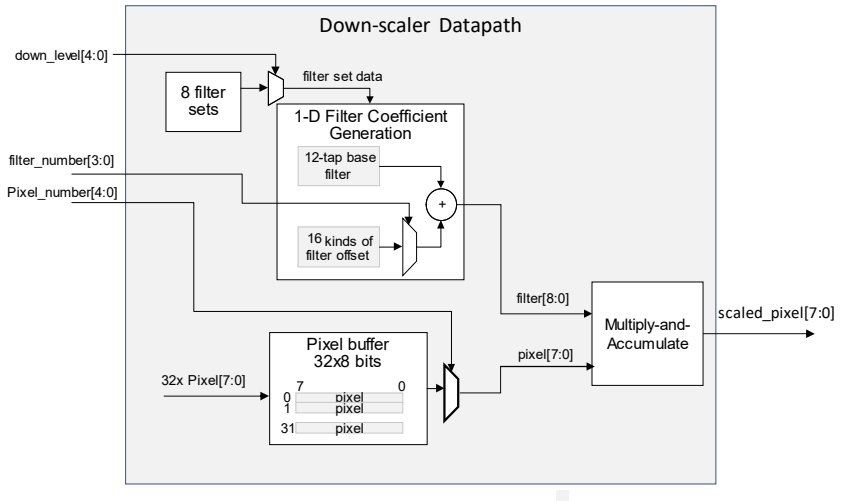

Fig. 6. The datapath of the down-scaler.

4 Implementation and Experimental Results on FPGA

The proposed architecture has been implemented on a Xilinx KC705 platform with a Kintex-7 XC7K325T-2FFG900C FPGA device. The RTL model of the down-scaler circuit is written using Verilog and synthesized at 100MHz using the Xilinx Vivado 2018.2 EDA tool. AXI-4 bus with a 32-word burst length is used for DRAM accesses. The logic utilization on the Kintex-7 FPGA is shown in Table 1. Table 2 presents a list of the on-chip memory blocks used in the design. Finally, Fig. 8 shows the number of execution cycles of the proposed down-scaler to generate output images at each resolution scale when the input image is 640×480.

Table 1. Logic usage on a Xilinx Kintex-7 FPGA DEVICE. The full system contains a Microblaze RISC core, the down-scaler, and all AXI4 infrastructure logics in Fig. 2.

Logic	#LUT6	# flip-flops	#BRAMs (4KB)
Down-scaler	4557	2490	128
Full system	27814 (13.65%)	27173 (6.67%)	136 (30.56%)

Table 2. On-chip memory usage (for 640x480 input images).

Memory type	Size	Function
Ping-pong RAM	2×128×8 bits	Composed of two SRAM units. Each unit contains 128×8-bit memory. That is, each SRAM block can be filled up by one burst of AXI4 DRAM access.
Inter RAM	465×680×8 bits	Stores the output pixels. This buffer is not optimized. The size can be further reduced to 12×640×8 bits in the future.
Pixel buffer	32×8 bits	The buffer is synthesized using registers.
12-tap base filters	8×12×9 bits	Stores the base filter coefficients in each filter set (per each D value).
All filter coefficients	8×16×12×9 bits	Stores the coefficient differences between all other filters and the base filter of each D value.
Reference table	31×9 bits	Stores the mapping between the decimation rate and the filter set number and the center pixel position.

Current design of the hardware-software system adopts a 32-bit memory controller with the logic running at 100 Mhz. With these parameters, the image pyramid generator can generate eleven 24-level pyramids per second. This number can be derived from the performance chart shown in Fig. 7, which was measured using a hardware counter register on the real platform. The main bottleneck of the design is in the DRAM bandwidth. The data path design is fine-tuned to match the DRAM bandwidth such that the memory read/write time and the logic computation time have maximal overlaps. As illustrated

in Fig. 4, the scaler pipeline does strike a good balance between the computation time and the memory access time.

For the KC705 platform used in this paper, the DRAM chips are connected to the FPGA using 64 bits wires. The DDR3 DRAM clock can run at 800MHz. Therefore, the FPGA platform can be theoretically used to emulate a 512-bit DRAM bus at 100MHz (or a 256-bit DRAM bus at 200MHz). In this case, we can hardwire the pyramid generator to pump out thirty 24-level pyramids per second for real-time applications. However, this also means that we need bus wrappers to adapt the 32-bit logics (e.g., the RISC core) to the 512-bit DRAM bus. We will leave this work for future research.

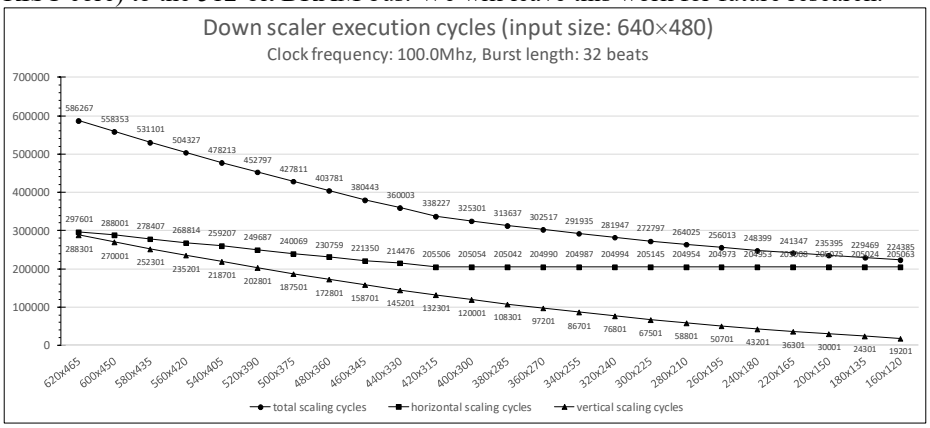

Fig. 7. The number of cycles it takes for the proposed down-scaler to generate images at different scales on the Xilinx KC705 platform.

5 Conclusions and Future Work

In this paper, we have presented the design of a fine-granularity down-sampler for image pyramid generation. The circuit has been implemented and verified on a Xilinx Kintex-7 FPGA. The proposed design adopts a sophisticated 12-tap Sine-windowed Sinc filter to produce high quality down-scaled images. The current design uses only a single scaling pipeline to reduce the quantity of circuit resources while maintaining high performance.

Based on the measurements on a real FPGA platform, the proposed architecture can generate eleven 24-level pyramids per second from a 640×480 input image at 100Mhz with a 32-bit DDR3 DRAM bus. It is possible to increase the performance further by introducing a wider DRAM bus and extra parallel pipelines, as discussed in section 4. For future work, we will look into the optimization of the architecture to support 30-fps real-time applications. We will also investigate the reduction of on-chip memory requirement by optimizing the buffering architecture.

In addition to the refinement of the image pyramid generator, we will also apply the system to construct a dense optical flow estimator. The state-of-the-art in optical flow estimators use coarse-to-fine approaches with complex non-linear iterative optimizers

that make real-time application impractical. With a 24-level image pyramid generator, it is possible to adopt a fast total least-squares estimator to iterate through each level of the image pyramid to obtain accurate optical flow estimates. In summary, the proposed HW-SW system is very promising for computer vision applications.

References

1. Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., Ogden, J. M.: Pyramid methods in image processing. RCA Engineer, 29-6, pp.33-41 (1984).
2. Nuno-Maganda, M. A., Arias-Estrada, M. O., Feregrino-Uribe, C.: Three video applications using an FPGA-based pyramid implementation: tracking, mosaics and stabilization. In Proc. of IEEE Int. Conf. on Field Programmable Technology (FPT). Tokyo, Japan (2003).
3. Lee, B.: Design and implementation of image-pyramid. Journal of Korea Multimedia Society 19-7 (2016).
4. Horn, B., Schunck, B.: Determining optical flow. Artificial Intelligence, 17, pp. 185-203 (1981).
5. Deqing, S., Roth, S., Black, M. J.: Secrets of optical flow estimation and their principles. In IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'10). 13-18, pp.2432-2439. San Francisco (2010).
6. Farabet, C., Poulet, C., Han, J. Y., LeCun, Y.: CNP: An FPGA-based processor for convolutional networks. In Proc. of 2009 Int. Conf. on Field Programmable Logic and Applications, pp. 32-37. Prague (2009).
7. Sun, S., Reichel, J.: AHG Report on Spatial Scalability Resampling, JVT-R006. The 18th Meeting of the Joint Video Team. Bangkok, Thailand (2006).
8. Agrawal, K., Chowdhury, S. R.: Real-time multisensor Laplacian fusion on FPGA. In Proc. of the 19th Int. Symp. on VLSI Design and Test, (2015).
9. Nuno-Maganda, M. A., Arias-Estrada, M. O.: Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling. In Proc. of Int. Conf. on Reconfigurable Computing and FPGAs (ReConFig'05), pp. 28-30. Puebla City, Mexico (2005).
10. Ramachandran, S., Srinivasan, S.: Design and FPGA implementation of an MPEG based video scalar with reduced on-chip memory utilization. Journal of Systems Architecture 51, pp. 435–450. Elsevier (2005).
11. Xiao, J., Zou, X., Liu, Z., Guo, X.: Adaptive interpolation algorithm for real-time image resizing. In Proc. of the 1st Int. Conf. on Innovative Computing, Information and Control - Volume I (ICICIC'06). Beijin, China (2006).
12. Suryavanshi, R. A., Sheelvant, S.: Review of implementation of image scaling algorithm on FPGA. Int. Journal of Innovations & Advancement in Computer Science. 4-7, pp. 96-99. Academic Science (2015).
13. Zhu, Q., Garg, N., Tsai, Y.-T., Pulli, K.: An energy efficient time-sharing pyramid pipeline for multi-resolution computer vision. In Proc. of IFIP/IEEE 21st Int. Conf. on Very Large Scale Integration (VLSI-SoC), pp. 278-281. Istanbul, Turkey (2013).
14. Schmid, M., Reiche, O., Schmitt, C., Hanning, F., Teich, J.: Code Generation for High-Level Synthesis of Multiresolution Applications on FPGAs, *arXiv:1408.4721*, (2014).