Memory Efficient VLSI Architecture for QCIF to VGA Resolution Conversion

Asmar A. Khan and Shahid Masud

Department of Computer Science and Engineering
Lahore University of Management Sciences
Opp. Sector-U, D.H.A. Lahore 54792, Pakistan
{asmara,smasud}@lums.edu.pk

Abstract. This paper presents the design of an FPGA based real time video display size resolution conversion for QCIF to VGA. The architecture is based on a pre-computed memory mapping that facilitates reduction in memory size and latency. The scheme has been realized for real time resolution conversion of a QCIF video at 30 fps. The memory requirement has been reduced to 400 KB which is significantly lower than an earlier hardware based scheme [2] where memory used was nearly 5 MB. The results have been validated on Xilinx Spartan-2E FPGA running at 100MHz. The area of complete design is around 66K gates including input and output memory.

Keywords: Display resolution conversion, Image-scaling, VLSI architecture, FPGA, QCIF, VGA.

1 Introduction

In recent years, many hardware based designs have been proposed for different image resolution and resizing operations. Due to advancements in network and communication technologies, more and more multimedia applications and compatible devices are frequently coming in use. As a consequence, image scaling has become an important research problem. Growing demands on interoperability of emerging devices necessitate the use of image scaling and resolution conversion operations as well. Many devices connected to CDMA or GPRS network have different spatial resolutions. The data-broadcast by mobile switching centre (MSC) implies that each receiving device has its own transcoder making the image compatible to its spatial display resolution. The transcoder's operations include spatial resolution conversion for which image scaling is an important component. This paper focuses on the image scaling part of a video transcoding procedure. Most image resolution conversion techniques found in literature are software based [5] and [6] and meant for off-line processing. There is thus a need for dedicated hardware architecture that can achieve real time performance. Recently, some hardware based image scaling designs have been proposed in [2] and [3]. These schemes target only a fixed image size for which the ratio of size conversion is either an integer or a fraction close to a whole number. A large

T. Wada, F. Huang, and S. Lin (Eds.): PSIVT 2009, LNCS 5414, pp. 829–838, 2009.

memory would be needed in these existing schemes to support a non-integer image scaling ratio. Important objective of the work presented in this paper is to develop memory efficient techniques for image resizing in non-integer conversion ratios. The design presented here achieves QCIF (176x144) to VGA (640x480) resolution conversion while requiring far less memory than the previous architecture in [2] and it is capable of achieving real time performance. The design is modular and scalable and can be conveniently converted for other size resolutions. The rest of the paper is organized as follows. Section II presents a background on image scaling operation as well as interpolation techniques with specific examples of QCIF to VGA. Section III describes the proposed Controller Based design and its memory requirements. Results and analysis are included in section IV followed by the conclusions.

2 Image Scaling and Interpolation

Image scaling is the process of resizing an image. The focus in this paper is on upsizing operation that involves (a) signal processing operations to maintain subjective quality and (b) interpolation operations to construct the additional (missing) data. An image looses information when reduced in size and requires smoothing operation in order to maintain the subjective quality. When an image is increased in size, extra data (missing pixels) is inserted through interpolation to form the new image. Nearest Neighbor, Bi-cubic, Quadratic and Spline are some of the well known interpolation techniques [4],[5]. An important issue in image interpolation is that it is not possible to discover any more information in the image than what already exists and the image quality inevitably suffers. The methods that are used in improving the perceptual quality of scaled image are intensive in terms of computations and memory requirements. It is because of these reasons that the image scaling operation has traditionally been performed in software. Some recent works [9], [10] have proposed image scaling hardware but this QCIF to VGA conversion has not been targeted yet. Most software based implementations are serial in nature and less parallelism can be exploited; whereas the proposed dedicated hardware design can target resolution conversions more efficiently. A typical process of image scaling is shown in figure 1.

Some important issues encountered in peculiar QCIF to VGA size conversion are summarized below:

2.1 Image Interpolation Procedure and Techniques

The complete operation of image resizing is illustrated in figure 1. The process comprises three main steps, namely (i) Up-sampling, (ii) Interpolation and finally (iii) down-sampling. Up-sampling step introduces blank pixels interspersed between existing pixels depending on the resizing desired. This inevitably leads to blocking artifacts and blurring in the image. Block and edge distortions usually occur when an image is up-sampled to an extent where the pixels become visible enough and discrete nature of image becomes more evident. To improve

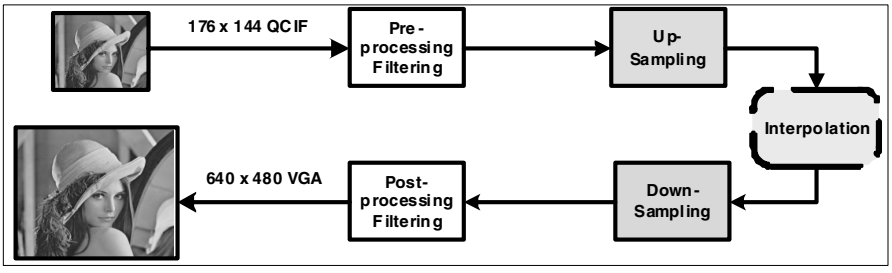

Fig. 1. Image scaling procedure for QCIF to VGA conversion

the perceptual quality, a post-processing operation is necessary. Here, the high frequency aberrations due to edges or scene changes are removed through the application of a low pass filter. Interpolation techniques approximate the blank pixels that have been introduced by up-sampling. An ideal interpolator has a frequency response which passes all frequency components in the original image and stops the remaining [1]. This is non-trivial operation in terms of computational complexity. Some advanced and complex interpolation techniques like Bi-cubic [4], Quadratic and Spline [5] are commonly used in software based transcoders [6], [8]. Hardware approaches discussed in [2], [3] and [4] require large amount of memory. The Bi-cubic interpolation proposed in [4] uses zoom processors to zoom a VGA resolution image. Although the proposed HABI design targets a real time scenario however, it consumes a large amount of Block-RAMs which is 44 in case of 8 zoom processors. The design is not scalable, as with high speed processing the required memory increases tremendously. A new memory mapped interpolation approach has been proposed in this paper that reduces not only the computational cost but also reduces the required memory for QCIF to VGA conversion. Any image which is down-sampled also suffers from aliasing. To avoid this artifact, the image is filtered through a low pass filter and then interpolated accordingly. This is shown as 'Pre-processing Filtering' in figure 1.

2.2 QCIF to VGA Memory Requirement

When a QCIF (176x144) image is converted to VGA (640x480), the conversion ratio for horizontal and vertical pixels is 40/11 and 10/3 respectively. Therefore, while converting 176 pixels to 640, an up-sampling by 40 is required followed by down-sampling by 11. Similarly, for converting 144 pixels to 480, an up-sampling by 10 and down-sampling by 3 is required. The intermediate storage of up-sampled rows and columns (by factors of 40 and 10) necessitate a huge memory requirement. Figure 2 compares this memory demand for some of the common conversions used in multimedia applications using up-sampling and down-sampling approach. Major problem in QCIF to VGA conversion is its non integer conversion factor. Most of the work in the past has been done on evenly divisible images [2], [3]. Although the schemes proposed in [4], [9] and [10] present scaling for non integer factors but estimated memory requirement

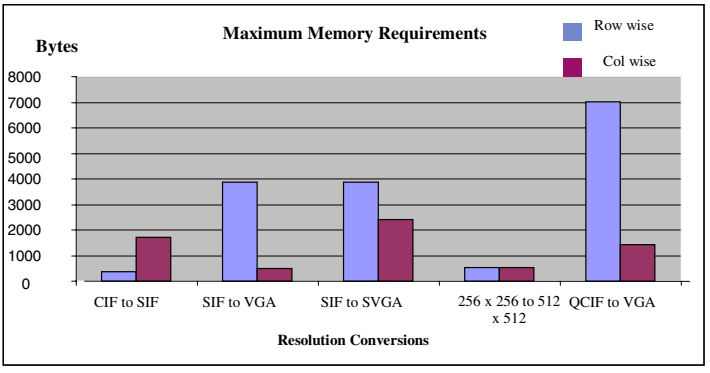

Fig. 2. Memory Requirements for Different Conversions

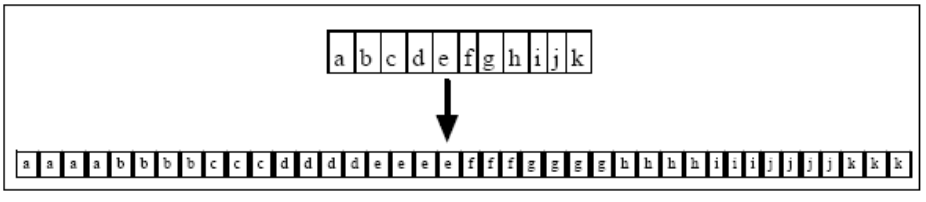

Fig. 3. Memory Mapping of 11 to 40 Samples

for QCIF to VGA is far greater than the ones proposed earlier. Furthermore, in case of complex schemes like Quadratic and Spline interpolation, the results are based only on software simulations using MATLAB or C-language and results have not been validated on any hardware platform [3].

A pre-computed memory mapping has been developed in this work that directly maps each pixel to its respective position in up-scaled image based on the calculations previously done off-line. Figure 3 shows the mapping of 11 samples to 40 samples. This scheme has been derived from the process of up-sampling 176 pixels to 7040 and then down-sampling them to 640. A routine in MATLAB was written to calculate this mapping. A QCIF image was up-sampled to 7040 in horizontal direction and then down-sampled to 640 to actually calculate the position of each pixel. The scheme maps a pixel at every 0.275 index value which is in fact the factor 11/40. As perceptual quality is usually measured in PSNR, it is assumed that PSNR best reflects the perceptual quality. Different QCIF images like 'cameraman' and 'lena', when scaled to VGA using the traditional software based up-sampling and down-sampling operations, were compared to the proposed technique with comparable PSNR values. This shows that the quality of images is not affected while memory mapping approach is applied. The technique is based on the Nearest Neighbor kernel provided in equation 1. The use of nearest neighbor interpolation has been used for being computationally cheapest. Another advantage of using nearest neighbor is to preserve edges [9].

$$h(x) = \begin{cases} 1 & 0 < |x| < 0.5 \\ 0 & \text{elsewhere} \end{cases} \quad (1)$$

3 Proposed Design

The proposed design is based on a controller based state machine which uses the memory map presented earlier and reduces the required memory by having a shared memory architecture. The architectural design of memory module is the key to meeting stringent timing requirements, memory latencies and delays. The proposed technique and its VLSI design is elaborated below. A comparison with slice based approach presented in [2] has also been made.

3.1 Controller Based Approach

A controller based approach is proposed in this work which reduces the memory required for QCIF to VGA scaling by utilizing the pre-computed memory map scheme. The design is based on a state machine which reads the data from external memory and then maps it to the memory with pre-calculated mapping. In this paper, the hardware has been designed for a specific conversion; however a generic formulation of mapping is possible. In this scheme, the input image is scaled in two stages; first stage is horizontal scaling factor calculation where rows to be interpolated are calculated and the map is used by state machine. This map is used to repeat (being the nearest neighbor) the columns which are interpolated in stage II. Second stage is the vertical scaling where each column is interpolated and scaled to the desired level. As a result, a complete image is scaled to the desired resolution. The proposed memory mapping technique obviates the need for pre-processing step as aliasing cannot occur for the sizes involved in this particular conversion. A state machine based controller reads the contents of a pre-computed memory map shown in figure 3. This mapping is used to repeat each column in order to write the data to new locations. This processing is done column-wise which is then repeated by pre-computed memory map. This process will interpolate the rows to the desired factor. The column-wise interpolation converts the 144 pixels to 480 using a similar map shown in figure 3. Completion of column-wise conversion implies that each column of 144 pixels will now be repeated by its count as per figure 3 and ultimately 176 rows will be scaled to 640 pixels. The scaling operation of a processing element is explained by the state diagram shown in figure 4. An input memory of 144 bytes and output memory of 480 bytes is needed to convert each column of 144 to 480 pixels. A register counts 3 input samples from the memory and then maps them to 10 locations, thus mapping every sample at 0.3 index value similar to figure 3. These 10 samples are convolved to the post-processing seven-tap filter. The filter coefficients, shown in equation 2 , are same as previously reported for SIF to CCIR-601 conversion [7].

$$[\text{-12 } 0 \text{ } 140 \text{ } 256 \text{ } 140 \text{ } 0 \text{ -12}] \times \frac{1}{256} \quad (2)$$

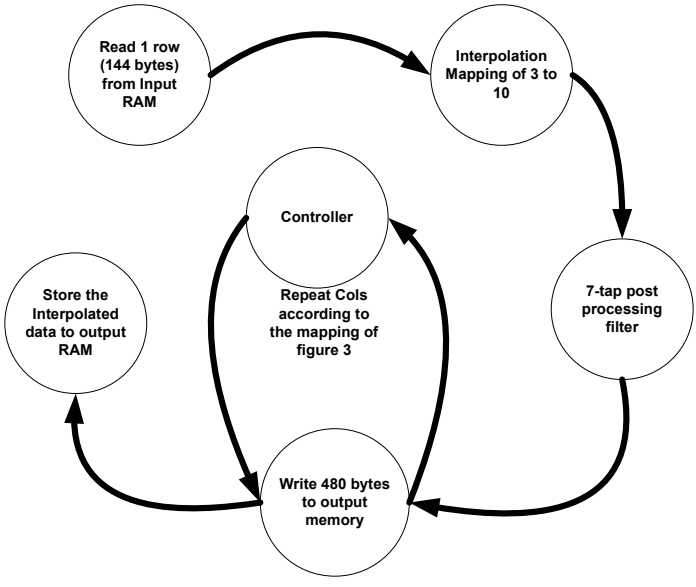

Fig. 4. State Diagram for Processing Element

These samples are then stored to the output memory which is 480 bytes wide. For an image with 144 columns, the requirement is of 144 parallel processing elements (PEs). Where each PE performs this particular operation on one complete column of 144 pixels. It must be mentioned that these 144 operations could be spread over multiple cycles if smaller segments from a column are processed in one go. However, this would slow down the processing accordingly and the conversion may not complete within the real-time constraints. The distributed memory architecture described later mitigates the timing delays encountered while performing this operation in serial fashion. The horizontal and vertical scaling is performed separately to reduce the computational complexity. After the scaling of column to 480 pixels, each column is repeated for horizontal scaling. For example, the first column must be repeated four times like pixel 'a' in figure 3. Similarly the index of each succeeding column will be repeated according to the map that is pre-calculated by the controller itself. The controller monitors the count for column index and repeats the pixel value of each column accordingly. The output of this operation is the desired VGA size image. This controller can be provided with desired scaling factor and can work as a generic image scaler as well. The controller circuitry is equipped to automatically read column index.

3.2 Distributed Memory Architecture

A distributed memory architecture has been developed in which each PE has its own memory module. The original image is distributed to these memory

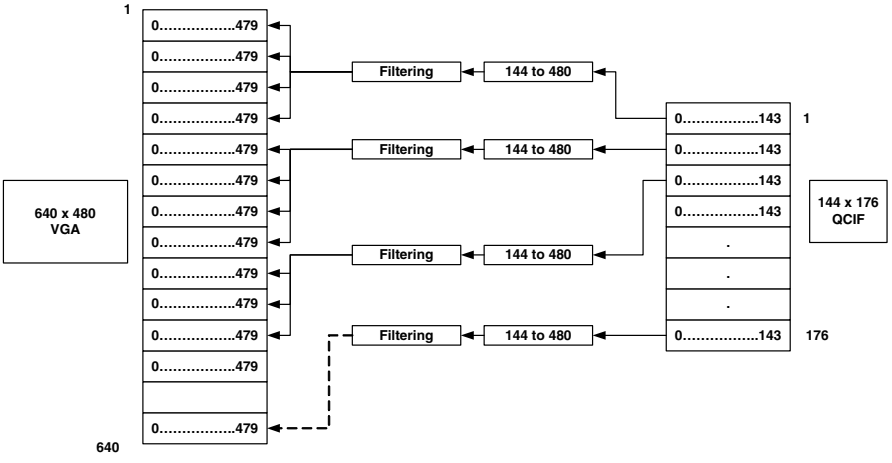

Fig. 5. Distributed Architecture for Controller Based Approach

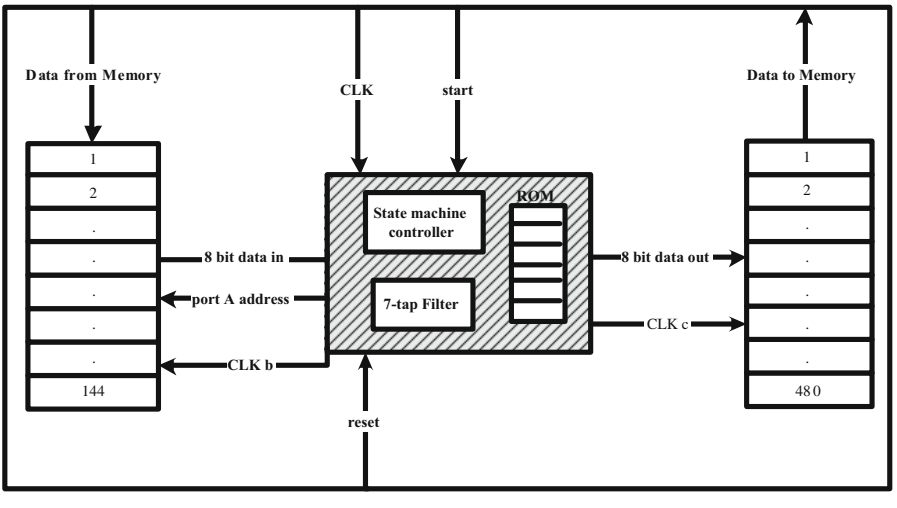

Fig. 6. Block Diagram of Processing Element

modules as column-wise input of 144x1 pixels. The pixels in the block boundaries have been processed by zero-padding the affected pixels. The processing elements described above scale one column to the required resolution. A complete image conversion requires all columns and rows to be scaled; therefore, a parallel hardware with distributed memory architecture has been developed to meet stringent delay constraints. Figure 5 shows the block diagram of system organization. The image is read column-wise and fed into 144 parallel processing units which produce the complete VGA image. Figure 6 depicts the architecture of a processing element and its operation. Each PE needs to have memory modules of 144 bytes and 480 bytes. The shaded area represents the state machine

based controller with post processing filter and registers for temporary storage required in memory mapping.

3.3 Memory Count

The system's memory requirement compared to the slice based approach presented in [2] is considerably reduced. Table 1 describes the proposed controller based system's memory requirements. The memory was calculated as per method presented in [2].

Table 1. Memory required for controller based approach

Proposed Approach	Memory in (bytes)	Memory out (bytes)
Col wise	144 x 144	144 x 480
Row wise	-	640 x 480

The design was simulated on Modelsim and was mapped on Xilinx Spartan-II FPGA running at 100 MHz clock frequency. The processing time required for a single PE is 4.5 clock cycles which is sufficient to support a frame rate of 30 fps for QCIF frame size.

4 Analysis and Discussion

This section presents the analysis of the proposed design in comparison with the slice based approach presented in [2]. Although the design presented in [2] did not address this specific conversion (QCIF to VGA) but the technique claimed to be effective for evenly divisible images. The image is divided into equal size slices and all the slices are parallel scaled to the desired level. It uses the Nearest Neighbor interpolation method. Table 2 describes the memory requirement for slice based approach where 768 slices (each of 11x3 bytes of image) were up-scaled to 768 slices (each of 40x10 bytes). The actual calculations for this particular conversion were made using Table 2 provided in [2]. Some hardware based architectures like [4] proposed interpolation hardware which uses dual port Block-RAMs to store the image. In our proposed design, no extra memory is required to store the intermediate resultant image. In [4], there are 44 dual port Block-RAMs of 16Kb each required to achieve real time video processing frame rate. However, the memory requirement can not be reduced by using faster memory. Secondly the design proposed in [4] is valid for interpolation purposes only. The complete resolution conversion procedure was not presented.

Our proposed design is for luminance component only and chrominance has not been considered. The design is a proof of concept which can easily be extended to the chrominance as well. Nevertheless, this would influence the cost of design in terms of memory and time.

Table 2. Memory required for slice based approach

Slice Based Approach	Memory in (bytes)	Memory out (bytes)
Col wise	768 x 3	768 x 10
Row wise	768 x 176 x 3	768 x 640 x 10

4.1 Memory Requirement for Controller Based Approach

The QCIF to VGA conversion requires non-integer scaling. The memory requirements exceed tremendously while performing up-sampling and down-sampling of pixels. Our work has reduced the required memory and hardware to a significant level. There are 144 + 480 units of memory for vertical scaling requiring 90KB of memory. The resultant image requires a memory of 640 x 480 which makes the total required memory to be 397KB. This is more than 10 times less than the design proposed in [2] where the memory requirement for this specific conversion is estimated to be 5.33 MB.

4.2 Gate Count

The gate count for a single PE unit is 460 gates. There are 144 units. Thus the total gate count for the complete resolution conversion hardware is 144 x 460 = 66K gates. This design is fully parallel however gate count could be further reduced by using LUTs instead of ROM. As discussed earlier, our mapping uses ROM and that is a big reason of large gate count [10].

4.3 Timing Constraints

A single PE unit, which converts a row of 144 to 480 pixels, takes 4.5 clock cycles at 100 MHz clock rate. The non integer clock cycle is due to the presence of different clocks inside the state processing element. The controller takes another 4 clock cycles. This corresponds a time of 0.045 msec. The total time required for conversion of one frame from QCIF to VGA consumes 144 x (45+40) = 12240 nsec = 12.24 msec. This corresponds to a frame rate of 80fps, which is far greater than the one proposed in [10] for 16VGA to SXGA. The proposed architecture is scalable and modular where the controller can be provided with scaling factor and can be used as a generic converter for real time video streaming. The area occupied could be reduced through parallelism exploiting the redundancy in multimedia data. Use of pipelining architecture can also contribute to further reduction in area. However this will increase the computational complexity and put stringent constraints on processing time. The throughput of the design can be improved using high speed FPGA like Virtex-4 or Virtex-5 running at more than 500 MHz.

5 Conclusion

The work proposes a specific resolution conversion with reduced resources yet it targets a real time application [8]. This scheme is valid for decoded data and

does not require any compatibility for encoding scheme. The hardware can be used in small devices like mobile phone and PDAs due to its low complexity and reduced memory. The PE is building block of the design which interpolates and decimates the pixels by using a memory map causing the required memory to be reduced. The distributed memory architecture enables the design to meet the stringent real time processing requirements. The proposed design is a scalable and modular and capable of performing generic image scaling operations for any given conversion ratio.

Acknowledgements

The authors acknowledge the support of Higher Education Commission Pakistan and Computer Science Department at Lahore University of Management Sciences, Pakistan.

References

1. Lehmann, T.M.: Survey: Interpolation Methods in Medical Image Processing. IEEE transactions on medical imaging 18(11) (November 1999)
2. Aho, E., Vanne, J., Hämäläinen, T.D., Kuusilinna, K.: Block-Level Parallel Processing for Scaling Evenly Divisible Images. IEEE Transactions on circuits and systems 52(12), 2717–2725 (2005)
3. Ramachanran, S., Srinivasan, S.: Design and FPGA implementation of an MPEG based video scalar with reduced on-chip memory utilization. Journal of Systems Architecture 51, 435–450 (2005)
4. Aurelio, M., Arias-Estrada, M.O.: Real Time FPGA Based Architecture for Bicubic Interpolation: An Application for Digital Image Scaling. In: Proceedings of International Conference on Reconfigurable Computing and FPGAs, September 28-30 (2005)
5. Lin, T.-C., Truong, T.-K.: DCT-Based Image Codec Embedded Cubic Spline Interpolation with Optimal Quantization. In: Proceedings of IEEE international Symposium on Multimedia, pp. 2746–2749 (September 2006)
6. Wang, L., Wang, Q.: A fast Intra Mode Decision Algorithm for MPEG-2 to H.264 Video Transcoding. In: Proceedings of IEEE 10th International Symposium on Consumer Electronic, pp. 1–5 (December 2006)
7. Standards documents MPEG-1: Coding of moving pictures and associated audio for digital storage media at up to 1.5 Mbps. ISO/IEC 11172-2: video (November 1991)
8. Wanrong, L., Bushmitch, D.: Design and implementation of a high quality DV50-MPEG2 software transcoder. In: International Conference on Consumer Electronics, pp. 142–143 (June 2002)
9. Kim, C.-H., Seong, S.-M., Lee, J.-A., Kim, L.-S.: Winscale: An Image-Scaling Algorithm Using an Area Pixel Model. IEEE Transactions on Circuits and Systems for Video Technology 13(6), 549–553 (2003)
10. Aho, E., Vanne, J., Hämäläinen, T.D., Kuusilinna, K.: Configurable Implementation of Parallel Memory Based Real-time Video Downscaler. Microprocessors and Microsystems 31(5), 283–292 (2007)