# VLSI Implementation of A Novel Low Complexity Image Scalar Architecture

**Nitin Singh[1] and Shweta Agrawal[2]**
*[1]Research scholar,[2] Assistant Professor,*
*Dept of Electronics and Comm., SRCEM Banmore, Morena, India*

***Abstract*— The modern portable devices such as smart phones have been employing huge number of image/video applications in recent years. This results in exponential rise in the design complexity of these devices. Moreover, the advancement in the VLSI technology with nano-scale transistors dimensions further increases the complexity of these devices. The highly complex devices consume significant amount of energy while processing the signal. Further, the cost of the device increases with increasing complexity. Therefore low complexity architecture is the prime requirement. In several image processing applications image scaling is the commonly used operation to fit the image into the device display having different sizes. In several applications such as medical imaging, image scalar is commonly employed in the medical devices to see the image capture with different resolution device. This paper proposed low complexity image scalar and then design is implemented and evaluated. The simulation results show that the proposed image scalar reduces area by 12% and delay by 19.95% over the existing well-known architecture.**

***Keywords*— Digital Signal Processing (DSP), Image Scalar, Image Processing, Integrated Circuits, VLSI, Low Power Design.**

## I. INTRODUCTION

There is the exponential usage of the portable devices exhibiting multimedia application, in the recent years. Huge functionalities are being added on the same devices that become possible due to the advancement in the very large scale integration (VLSI) technology. The modern VLSI technology allows us to build an Integrated circuit (IC) that can have thousands of transistors into single chip. This integration of huge functionality is imposing several challenges to the VLSI designer as increasing functionality increases the complexity which in turn increases failure probability. Further as area, power and delay are main area of concern and at the same time it is very difficult to achieve optimal value of these parameters. Image processing application is the prime component of the multimedia applications. The huge power and energy requirement in these processing worsen the performance of these devices. Thus, an energy efficient image processing is required to achieve energy efficient design.

The primary design parameters (area, power and delay) form a tradeoff triangle i.e. improving one parameter damages the other. The conventional approach of VLSI design provides accurate results i.e. follow the given specification. But in real scenario it is not always required. There are many applications where minor error can be tolerated called as error tolerant applications. The multimedia applications such as image/video processing are error tolerant applications. In these applications, small error is tolerable as these applications produce output for human consumption [2] as human have limited visual perception. Along with the multimedia applications, several other applications such as that exhibit probabilistic computations and iterative computation also exhibits error tolerance. Thus, the accurate designs for these applications are the waste of power/area and performance. For these applications, accuracy can be seen as the new design parameter that can be traded to improve all design parameters.

Several image processing techniques such as image restoration, image enhancement, image scaling etc enable the user to tune the image according to his needs. Among these techniques, image scaling is one which has wide variety of applications ranging from medical imaging to mobile imaging. Image scaling is a process of enlarging or shrinking an image i.e. it is nothing but changing the image resolution. Higher the image resolution, higher the detail it will have. Image scaling is done by process called interpolation. Interpolation of pixels is done by the interpolator which is the heart of an image scalar. Conventional interpolation algorithm like nearest neighbor and bilinear interpolation provides scaled images very efficiently but fails to provide good quality. Although, other algorithm such as bicubic provides higher quality but consume more area and demand large power. Thus, there is a demand of a design that can provide scaled image energy efficiently with acceptable image quality.

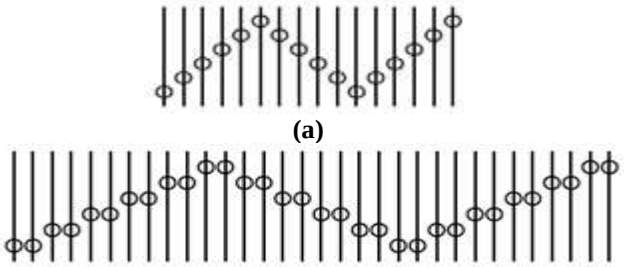
This paper proposes a novel low complexity image scalar. The existing and proposed designs are implemented and then evaluated using design and error metrics and compared with the well-known existing architectures. The simulation results shows that proposed image scalar provides significantly improved design metrics and simultaneously provides acceptable quality.

## II. RELATED WORK ON IMAGE SCALAR

As interpolation is the major part of an image scalar. Based on technique of computation of interpolation pixel, image scaling techniques can be divided into Non-polynomial and Polynomial based. The polynomial based techniques obtain the target pixel by solving the polynomial equation containing its neighboring pixels. Different methods like nearest neighbor [1], Bilinear [2], Bicubic [3] etc., are examples of Polynomial based techniques whereas Curvature interpolation [4], Blending Kernels [5], auto-regressive models [6] etc., are Non-Polynomial based techniques. Although these techniques are of high quality, they are hardware inefficient because of their computational complexity.

### 2.1 Nearest neighbour interpolation

Nearest neighbour interpolation [1] is the most basis type of interpolation. Its principle is to replicate the value which is nearest to the target pixel. Fig. 1 illustrated the technique of nearest neighbour.



**(a)**

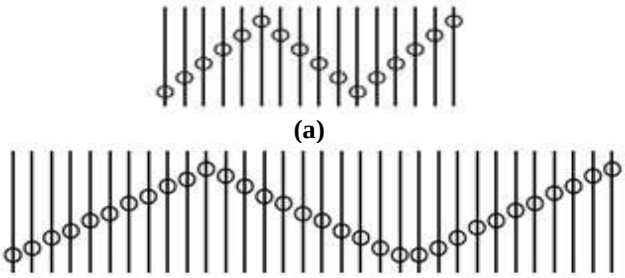
**(b)**

**Fig. 1: Illustration of nearest neighbor interpolation (a) Original (b) interpolated signal.**

Suppose that an image has to be enlarged two times. A simple way to do that is to assign any pixel with the value of intensity equal to the closest pixel in the input image. This replication method is termed as nearest neighbor interpolation because we are assigning the value of neighboring pixel. Though this approach is very simple, it tends to produce artifacts such as distorted edges, which are undesirable. So, this method is generally avoided. Images scaled using nearest neighbor interpolation produce strong aliasing and blocking effects. To reduce this, instead of replicating the neighbor, we use multiple neighbors to obtain the target pixel which is bilinear interpolation.

### 2.2 Bilinear interpolation

In this, four nearest neighbors are used to estimate the intensity of the target pixel. Each neighbor is assigned a weight depending on its distance from the target pixel i.e., nearest pixel will have more weight compared to the farthest [2].



**(a)**

**(b)**

**Fig. 2: Illustration of bilinear interpolation (a) Original (b) interpolated signal.**

In this method, temporary pixels are obtained first by performing interpolation in one direction. Then, interpolation is done in other direction using these pixels to obtain the target pixel. As shown in Fig. 3, to compute target pixel T, four neighboring pixels $N_{11}$, $N_{12}$, $N_{21}$, $N_{22}$ are considered. $x_d$ is the distance in horizontal direction from original pixel to the target pixel whereas yd is the distance in vertical direction. Original pixels are assumed to be at unit distance from each other. The intermediate pixels $I_1$ and $I_2$ are obtained by

$$I_1 = (1 - x_d) * N_{11} + x_d * N_{12} \quad (1)$$

$$I_2 = (1 - x_d) * N_{21} + x_d * N_{22} \quad (2)$$

Using these intermediate pixels, the target pixel T can be calculated as

$$T = (1 - y_d) * I_1 + y_d * I_2 \quad (3)$$

Using equations (1), (2) and (3), the target pixel can be calculated as

$$T = (1 - x_d)(1 - y_d) * N_{11} + (1 - x_d) * y_d * N_{12}$$
$$+x_d * (1 - y_d) * N_{21} + x_d * y_d * N_{22} \quad (4)$$

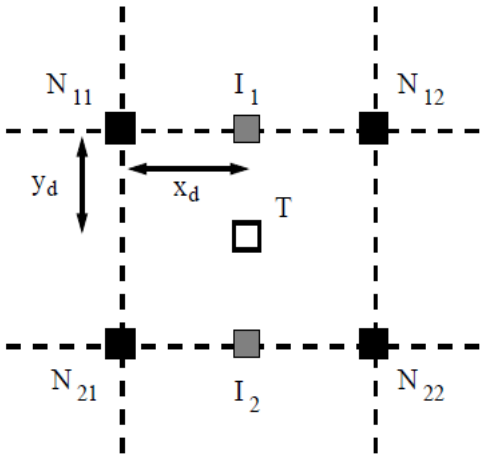



**Fig. 3: Determination of unknown pixel using bilinear interpolation**

Although this produces better results than nearest neighbor interpolation, it still exhibits blurring edges and aliasing. In order to improve the quality we can either use a better interpolation technique or add some filtering such that the post-interpolation effects are minimized. Next, subsection discusses interpolation technique that provides better scaled images.

### 2.3 Bicubic interpolation

This involves calculation of a pixel using 16 nearest neighbors [3]. The pixel is calculated by equation given below.

$$v(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \quad (5)$$

The coefficients are calculated in the same way as in bilinear interpolation. Fig. 4 illustrates bicubic interpolation method. The interpolated surface is smoother than bilinear and is chosen when speed is not the primary concern. It has high memory requirement and computational complexity, hence making it difficult for hardware implementation.

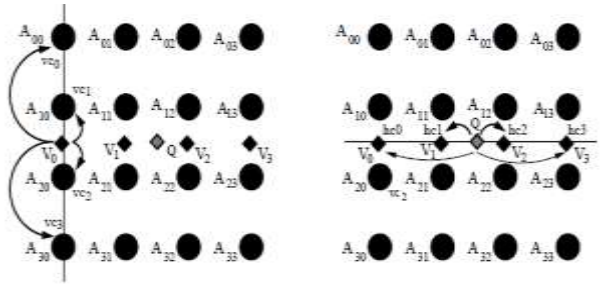


**Fig. 4: Illustration of bicubic interpolation**

### 2.4 Low cost high quality adaptive scalar

Chen et al. [7] proposed 2-D adaptive scaling architecture that uses bilinear interpolator along with sharpening and clamp filters as illustrated in Fig. 5.

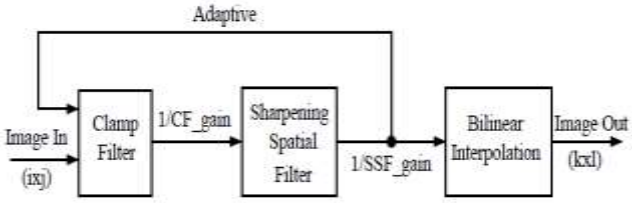



**Fig. 5: Block diagram of image scalar**

#### 2.4.1 Clamp filter

The clamp filter acts as low-pass filter. It reduces aliasing effects and also smoothen the discontinuous edges. The convolution kernel for the clamp filter is given below. In this algorithm, a Gaussian 3x3 filter is used which is shown below.

$$K_C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & C & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (6)$$

Where, C is a clamp parameter which is defined by user depending on the characteristics of the images. The output of clamp filter is then filtered using a sharpening spatial filter.

#### 2.4.2 Sharpening filter

The sharpening filter eliminates the low frequency noise therefore acts as a high-pass filter. It removes noise and also enhances the edges. The sharpening filter kernel commonly used in the scaling algorithm is shown below.

$$K_S = \begin{bmatrix} -1 & -1 & -1 \\ -1 & S & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (7)$$

Here, S is the sharpening parameter which is defined by user based on the image property.

#### 2.4.3 Combined filter

In order to reduce memory requirement and also to obtain hardware efficient design, the sharpening and clamp filters are convolved to produce a 5x5 filter whose kernel is shown below [8].

$$K_C * K_S = \begin{bmatrix} -1 & -2 & -3 & -2 & -1 \\ -2 & -2-C+S & -4-C+S & -2-C+S & -2 \\ -3 & -4-C+S & -8-C*S & -4-C+S & -3 \\ -2 & -2-C+S & -4-C+S & -2-C+S & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix}$$

The interpolated pixel can be calculated as follows

$$p'(i,j) = p(i,j) \begin{bmatrix} -1 & -2 & -3 & -2 & -1 \\ -2 & -2-C+S & -4-C+S & -2-C+S & -2 \\ -3 & -4-C+S & -8-C*S & -4-C+S & -3 \\ -2 & -2-C+S & -4-C+S & -2-C+S & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix} / [(C+8)(S-8)]$$

In order to further improve the performance a new low complexity image scalar is proposed which is discussed in the next section.

## III. PRAPOSED WORK

This section discusses the proposed image scalar architecture. The working principle and analysis on the complexity reduction of the proposed architecture is detailed in this section.

### 4.2 Proposed Image Scalar Architecture

It is observed that most of the existing image scaling algorithm/architecture utilizes sharpening filter to reduce the effect of the blurring which is the limitation of the bilinear interpolator. Consideration of improved interpolation increases the computation complexity significantly; therefore these algorithms are not feasible for hardware implementation. Thus, bilinear interpolator is the primary choice of interpolation technique used in the hardware implementation.

It is also observed that sharpening filter also have large computations and that consumes huge power and provides large delay. Reducing the complexity of the sharpening filter also provides significant reduction power and delay. Therefore, this work proposes a new sharpening filter kernel that has very small non-zero value. The architecture of the proposed image scalar is shown in Fig. 6. This architecture consists of proposed sharpening filter along with the bilinear interpolator. The input image data is applied to the sharpening unit that provides sharpened pixels which are given to the bilinear interpolator to achieve desired scaled image.

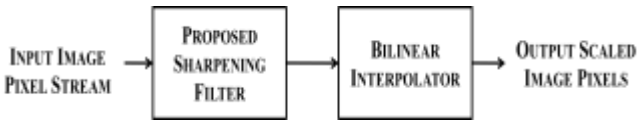



**Fig. 6: Block diagram of proposed image scalar**

#### 4.2.1 Proposed sharpening filter

In the conventional image sharpening kernel of 3x3, all the kernel coefficients value are -1 except center coefficient as given by Eq. 6. S is the sharpening parameter and is specified by the user depends of the required sharpening. In

order to reduce the complexity without significant reduction in the quality, outer coefficients are eliminated. The proposed sharpening kernel ($K_{ps}$) is given by Eq. 8.

$$K_{pS} = [-1 \quad S \quad -1] \tag{8}$$

The VLSI architecture of the conventional sharpening filter kernel is shown in Fig. 7. In the architecture, $P_{11}$, $P_{12}$, … $P_{33}$ are the input image pixels while Ps is the desired output sharpened pixel. In this architecture, $P_{22}$ is the reference pixel which is to be smoothened, and therefore will be multiplied with the sharpening parameter S.

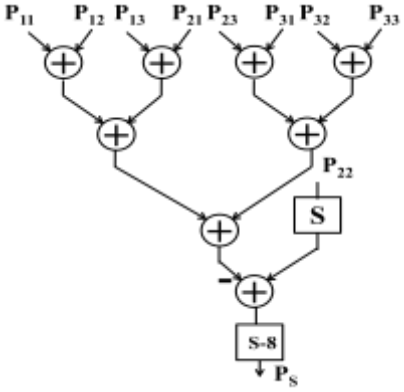



**Fig. 7: Circuit diagram of the conventional sharpening kernel**

It can be observed from the Fig. 7 that conventional kernel requires eight adders, one multiplier and a divider. In order to reduce the complexity of the divider, parameter S is selected such that the results term (S-8) will be in the form of power of two ($2^x$) which can be easy implemented by right shift operation. Therefore, it can be seen that the conventional design requires eight adders and a multiplier. The architecture of the proposed sharpened filter which implements the functionality of the Eq. 8 is shown in Fig. 8. Only three pixels will be the input to the sharpening kernel and $P_{22}$ is the reference pixel which will be replaced by the sharpened pixel $P_S$.

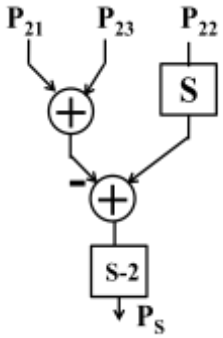



**Fig. 8: Circuit of the proposed sharpening kernel.**

The proposed sharpening kernel requires only single adder and a multiplier as the divider complexity can be reduced by selecting appropriate sharpening parameter. The proposed sharpening kernel reduces number of adder significantly over the conventional design thus requires very small implementation area and consumes less power over the conventional design. This proposed sharpening kernel is used in the proposed image scalar. Thus, the proposed image scalar architecture significantly reduces the complexity and return improved design metrics. The simulation results illustrated in the next section shows the effectiveness of the proposed image scalar of the existing.

## IV. EXPERIMENTAL RESULT & ANALYSIS

In order to evaluate the quality metrics [19, 20], MATLAB tool is to model the proposed and existing architectures of the image scalar. These implemented designs on MATLAB are then simulated with standard test images such as Lena and Baboon as shown in Fig. 9. The scaled images are extracted and compared. On the other hand, to evaluate the design metrics designs are implemented and Verilog HDL and simulated on ModelSim EDA tool. Further, the functionality of the proposed design is rigorously verified on ModelSim. Finally, the design metrics such as area, power and delay are extracted for the proposed and existing designs and compared.

**Fig. 9: Benchmark images**

### 4.1 Quality and Design Metrics

Various quality and design parameters are used to evaluate the design. This subsection introduces the different quality parameters which are used in our design.

#### 4.1.1 Mean Square Error (MSE)

In order to find out the error present in the output image, a most commonly used parameter called mean square error (MSE) is used. The MSE for an input image I and noisy output image K, is defined by the expression given below.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - k(i,j)]^2 \tag{9}$$

Where, variables m and n represent the number of row and column of the image.

#### 4.1.2 Peak Signal to Noise Ratio (PSNR)

The peak-signal-to-noise-ratio is the parameter used widely in image/video processing applications to quantify the amount of the noise present in the image and it is equal to the maximum signal power to the noise power. It is usually expressed in terms of decibel where signal and noise ratio is represented in logarithmic scale, as the signal can have wide dynamic range. For example in lossy image compression, the reconstructed images do not have the exact replica of the original image and therefore, there PSNR is generally used to represent amount of noise. The higher the value of the PSNR better the image is as it

reflects higher value of desired contents over the undesired (noise). The mathematical expression that computes the PSNR in decibel is given by the equation below.

$$PSNR_{db} = 10.\log_{10}(\frac{Sig_I^2}{MSE}) \quad (10)$$

Where, $Sig_I$ reflects the maximum signal value which for an image is 255.

### 4.2 Simulation results on MATLAB

To evaluate the efficacy of the proposed image scalar, all designs are implemented on MATLAB and simulated with benchmark image. The original image of size 256x256 pixels is scaled to 648x648 and then converted back to the original size using the same scalar. The error metrics are extracted for all these image scalar and the results are shown in Table 1.

**Table 1: Error metrics comparison**

| Metrics | Image scalar architecture | | | |
|---|---|---|---|---|
| | IS_WoSF | IS_WSForg | IS_WSFce | IS_prop |
| MSE | 10.16 | 2.78 | 5.49 | 7.26 |
| PSNR (dB) | 87.63 | 100.57 | 93.79 | 90.99 |

In the Table 1, IS_WoSF is the image scalar without sharpening filter whereas; IS_WSForg is the image scalar containing original sharpening filter. Similarly, IS_WSFec is the image scalar containing sharpening filter where corner non-zero terms are eliminated from the kernel.

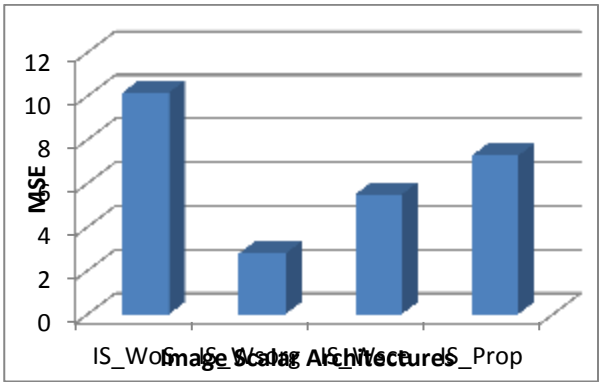



**Fig. 10: Comparison of mean square error.**


**Fig. 11: PSNR for various image scalars.**

The design is simulated with Lena image where the original image of size 256x256 pixels is scaled to 648x648 and then converted back to the original size using the same scalar as shown in Fig. 11.

**(a) Original Image**

**(b) Scaled image without SF**

**(c) Scaled image with accurate SF**

**(d) Scaled image with SF no corner coefficients**

**(e) Scaled image with proposed SF**

**Fig. 11: Original and scaled image via different image scalar**

### 4.3 Simulation Results on FPGA

To compute the hardware results of the proposed image scalar, all the designs are implemented in Verilog and processed through Xilinx ISE tool chain. The designs are synthesized and are implemented in FPGA. The simulation results for these designs are summarized in Table 3 which shows that proposed technique requires small area over accurate design.

**Table 3: FPGA result of image scalars**

| Parameter | | Image scalar | |
|---|---|---|---|
| | | Accurate | Prop. |
| Area | # LUTs | 5655 | 4967 |
| | # Logic | 5655 | 4966 |
| Performance | Delay | 3.626 | 3.12 |
| | Max. Frequency | 275.76 | 290.82 |

The Fig. 12 shows the comparison of the proposed and accurate image scalar architecture. It can be observed that proposed image scalar reduces 12.1%, 13.95% reduced

area and delay respectively over the accurate image scalar architecture.

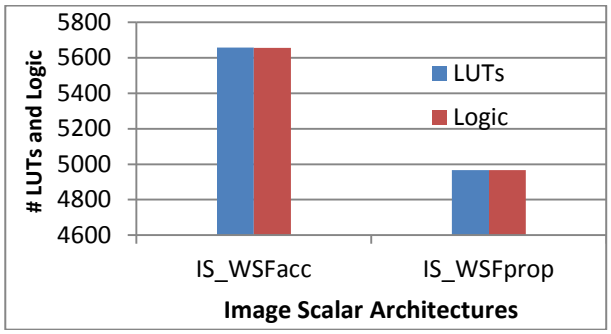



**Fig. 12: Area comparison of proposed and accurate image scalar**

## V. CONCLUSION

This paper presents a new energy efficient architecture for image scalar that provides higher performance without compromise in the quality. To evaluate the effectiveness of the proposed image scalar over the existing, all the existing image scalar architectures are implemented in MATLAB and Verilog. The designs on the MATLAB are simulated with benchmark input images and corresponding scaled image and quality metrics are extracted. The designs implemented on the Verilogy are synthesized with Xiling tool chain and simulated with benchmark inputs. The simulation results on FPGA shows that the proposed design reduces more than **12%** and **13.5%** area and delay respectively over the existing.

## REFERENCES

[1] V. Caselles, J.-M. Morel, and C. Sbert, "An axiomatic approach to image interpolation," in Image Processing, 1997. Proceedings, Int. Conference on, vol. 3, Oct 1997, pp. 376–379.

[2] K. Jensen and D. Anastassiou, "Subpixel edge localization and the interpolation of still images," Image Processing, IEEE Transactions on, vol. 4, no. 3, pp. 285–295, Mar 1995.

[3] R. Keys, "Cubic convolution interpolation for digital image processing," Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 29, no. 6, pp. 1153–1160, Dec 1981.

[4] H. Kim, Y. Cha, and S. Kim, "Curvature interpolation method for image zooming," Image Processing, IEEE Transactions on, vol. 20, no. 7, pp. 1895–1903, July 2011.

[5] L. Liang, "Image interpolation by blending kernels," Signal Processing Letters, IEEE, vol. 15, pp. 805–808, Dec 2008.

[6] X. Zhang and X. Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," Image Processing, IEEE Tran. on, vol. 17, no. 6, pp. 887–896, June 2008.

[7] S. L. Chen, H.-Y. Huang, and C.-H. Luo, "A low-cost high-quality adaptive scalar for real-time multimedia applications," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 21, no. 11, pp. 1600–1611, Nov 2011.

[8] S. L. Chen, "VLSI implementation of a low-cost high-quality image scaling processor," Circuits and Systems II: Express Briefs, IEEE Transactions on, vol. 60, no. 1, pp. 31–35, Jan 2013.

[9] "Standard test images from USC." [Online]. Available: http://sipi.usc.edu/database/database. php?volume=misc