[11] A. J. Viterbi, "An intuitive justification of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[12] T. C. Denk and K. K. Parhi, "Exhaustive scheduling and retiming of digital signal processing systems," *IEEE Trans. Circuits Syst., Part II: Analog Dig. Signal Process.*, vol. 45, no. 7, pp. 821–838, Jul. 1998.

[13] W. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electron. Lett.*, vol. 34, no. 16, pp. 1577–1578, Aug. 1998.

[14] Y. Wu, B. D. Woerner, and T. K. Blankenship, "Data width requirement in SISO decoding with module normalization," *IEEE Trans. Commun.*, vol. 49, no. 11, pp. 1861–1868, Nov. 2001.

A Flexible Architecture for Precise Gamma Correction

Dong-U Lee, Ray C. C. Cheung, and John D. Villasenor

***Abstract*—We present a flexible hardware architecture for precise gamma correction via piece-wise linear polynomial approximations. Arbitrary gamma values, input bit widths, and output bit widths are supported. The gamma correction curve is segmented via a combination of uniform segments and segments whose sizes vary by powers of two. This segmentation method minimizes the number of segments required, while providing an efficient way for indexing the polynomial coefficients. The outputs are guaranteed to be accurate to one unit in the last place through an analytical bit-width analysis methodology. Hardware realizations of various gamma correction designs are demonstrated on a Xilinx Virtex-4 field-programmable gate array (FPGA). A pipelined 12-bit input/8-bit output design on an XC4VLX100-12 FPGA occupies 146 slices and one digital signal processing slice. It is capable of performing 378 million gamma correction operations per second.**

***Index Terms*—Displays, field programmable gate arrays (FPGAs), fixed-point arithmetic, video signal processing.**

I. Introduction

The term "gamma" originates from the nonlinear responses of cathode ray tubes (CRTs) caused by electrostatic effects in the electron gun. The luminance produced by CRTs is not linearly proportional to the input voltage. Instead, the produced luminance L is proportional to the input voltage V raised by a power gamma γ

$$L = V^{\gamma} \tag{1}$$

where V is normalized over zero and one. In order to compensate this nonlinearity, gamma correction is performed on the input signal to achieve correct production of the luminance on the display [1]. Fig. 1 illustrates the signal flow in a typical display system. The most commonly used value for gamma correction is $\gamma = 1/0.45$ which is the basis of various standards including the ITU Rec. 709 specification [1].

Manuscript received April 27, 2006; revised November 21, 2006. This work was supported in part by the National Science Foundation under Grant CCR-0120778 and Grant CCF-0541453, by the Office of Naval Research under Contract N00014-06-1-0253, and by the Croucher Foundation.

D. Lee and J. D. Villasenor are with the Electrical Engineering Department, University of California, Los Angeles, CA 90095 USA (e-mail: dongu@icsl.ucla.edu; villa@icsl.ucla.edu).

R. C. C. Cheung is with the Department of Computing, Imperial College London, London, SW7 2BZ U.K. (e-mail: r.cheung@imperial.ac.uk).

Digital Object Identifier 10.1109/TVLSI.2007.893671

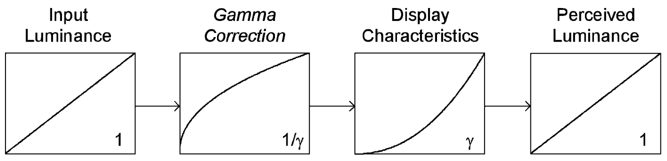

Fig. 1. Signal flow in a typical display system.

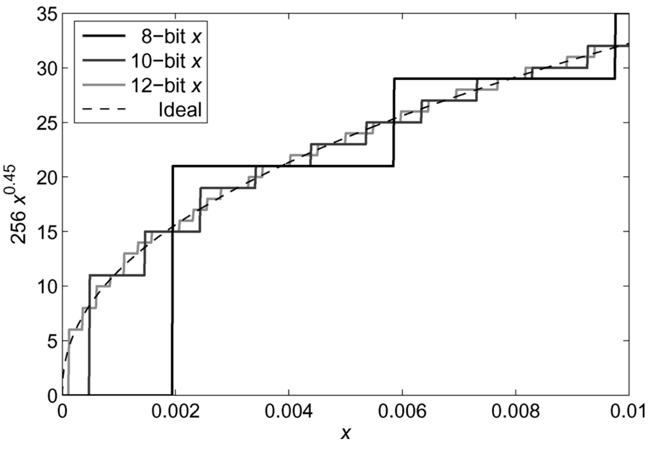

Fig. 2. Quantization effects of gamma correction in low luminance regions for different input x bit widths with the output fixed at eight bits. A gamma of 1/0.45 is assumed.

Although modern display devices such as liquid crystal displays (LCDs) and plasma display panels (PDPs) are inherently linear, gamma correction is still performed for perceptual coding purposes [2]. This is related to the fact that inverse of the CRT's transfer function is remarkably similar to the perceptual uniformity of human vision given by the Weber–Fechner law [3].

The most straightforward realization of gamma correction involves direct table lookups [4], [5]. For a typical display system that supports 8 bits per component, an 8-bit input/8-bit output table is often placed before or after the frame buffer. This approach is simple to implement and requires a table size of just $2^8 \times 8 = 2048$ bits. However, using eight bits to select the table entry can cause significant banding (or contouring) in the low luminance regions. This is because in these regions the gamma correction curve has a slope greater than one, and thus maps quantization step sizes at the input to larger step sizes at the output. When these transformed step sizes are overlayed on the inherent step sizes available at the output, the ability to generate certain luminance levels at the output can be lost. To fully exploit the output resolution in the presence of gamma correction, a higher input resolution is needed. This is illustrated in Fig. 2 which shows gamma correction in the low luminance regions of 8-, 10-, and 12-bit inputs with the output fixed at eight bits. The lower input resolutions create large luminance jumps in the low luminance regions at the display device, causing banding artifacts.

While the severity of these artifacts can be lessened by using more bits at the input, this leads to an exponential increase in storage if a direct lookup table is used. For instance, a 12-bit input/8-bit output table has a memory requirement of 32 768 bits, which can be problematic for resource constrained platforms. For example, many medical imaging displays support 12 bits per component and a recent paper by Kim *et al.* [6] describes an LCD display with ten bits per component. Such high-end displays impose even more stringent requirements on input resolution, making direct table lookup costly even for systems with better memory resources.

The use of multiple tables for different parts of the input interval has been considered in [7]. This leads to reduction in overall table size relative to a single table approach, but still suffers relatively large memory requirements. In order to overcome the large memory space drawbacks of traditional lookup table-only-based approaches, several implementations involving piece-wise linear polynomial approximations and interpolations have been proposed [8], [9]. Under this approach, the input interval is partitioned into nonuniform segments and linear approximation or interpolation is performed for each segment. Although the memory burden is significantly reduced, the precision of the gamma corrected samples is significantly lower than that available through direct table lookup methods. Such loss of precision can potentially lead to misrepresented luminance values on the display. Moreover, the nonuniform segmentation strategies, while quite effective in reducing memory, have typically been performed manually and are not guaranteed to be optimal.

In contrast to previous piece-wise linear methods, the approach described in this paper partitions the input interval in an automated systematic manner that results in the minimal number of segments for a given precision requirement. In addition, the gamma corrected samples are analytically guaranteed to be accurate to one unit in the last place (ulp). This means that if the gamma corrected samples are eight bits over [0,1), the error is at most 2^{-8} compared to infinitely precise gamma samples.

II. Segmentation

For hardware-based piece-wise polynomial approximations, uniform segmentation in which all segments are of equal size has been commonly employed. The most significant bits of the input serve as the index to the coefficient table. Although uniform segmentation has the advantage of being simple, for functions with high absolute first-order or higher order derivatives (referred to as nonlinear functions in the following), the number of segments can be impractically large [10]. To minimize segment count, the sizes of the segments should be adapted to the local nonlinearities of the functions.

We utilize hierarchical segmentation described in [10], which consists of a two-level hierarchy of uniform segments and segments whose sizes vary by powers of two. Since the gamma correction curve has a high and rapidly changing first derivative at the beginning of the interval, the segmentation scheme denoted $P2S_L$ and having segment sizes that increase by powers of two is chosen for the outer segmentation. Uniform segmentation US is used for the inner segmentation.

The B_x bits of the input x are split into three partitions: x_0, x_1, and x_2. x_0 and x_1 are used to index the outer and inner segmentation, respectively, while x_2 is used for the polynomial arithmetic. The number of addressable segments s_i of the partition i is constrained as follows:

$$s_i = 2^{B_{x_i}}, \qquad \text{if } \Lambda_i = US \tag{2}$$

$$s_i \le B_{x_i} + 1, \qquad \text{if } \Lambda_i = P2S_L \tag{3}$$

where B_{x_i} denotes the bit width and Λ_i denotes the segmentation of the partition i.

For the $P2S_L$ case, it is not intuitive why up to $B_{x_i} + 1$ segments can be formed. Consider the case when $B_x = 12$, the outer segmentation is $P2S_L$, and $B_{x_0} = 8$. As illustrated in Table I, it is possible to construct a maximum of nine segments. With the exception of the initial segments, the segment lengths increase by powers of two.

The $P2S_L$ segment address for a given x_0 can be computed by

$$P2S_L\_\text{addr} = \begin{cases} B_{x_0} - \text{LZD}(x_0), & \text{if } \text{MSB}(x_0) = 0 \\ B_{x_0}, & \text{if } \text{MSB}(x_0) = 1 \end{cases} \tag{4}$$

TABLE I
Segment Ranges in Binary Representation for $B_x = 12$, $P2S_L$ Outer Segmentation, and $B_{x_0} = 8$. The Eight Bits Corresponding to x_0 Are Highlighted in Bold. The Bits to the Left of the Vertical Partition Lines Correspond to $\bar{x}_0$. The Italic Bits Correspond to x_1 if B_{x_1} Is Kept Constant at Two

Index j	Segment Range
0	$\mathbf{0\,0\,0\,0\,0\,0\,0\,0} \mid \mathit{0\,0}\,0\,0 \sim \mathbf{0\,0\,0\,0\,0\,0\,0\,0} \mid \mathit{1\,1}\,1\,1$
1	$\mathbf{0\,0\,0\,0\,0\,0\,0\,1} \mid \mathit{0\,0}\,0\,0 \sim \mathbf{0\,0\,0\,0\,0\,0\,0\,1} \mid \mathit{1\,1}\,1\,1$
2	$\mathbf{0\,0\,0\,0\,0\,0\,1} \mid \boldsymbol{\mathit{0\,0}}\,0\,0\,0 \sim \mathbf{0\,0\,0\,0\,0\,0\,1} \mid \boldsymbol{\mathit{1\,1}}\,1\,1\,1$
3	$\mathbf{0\,0\,0\,0\,0\,1} \mid \boldsymbol{\mathit{0\,0}}\,0\,0\,0\,0 \sim \mathbf{0\,0\,0\,0\,0\,1} \mid \boldsymbol{\mathit{1\,1}}\,1\,1\,1\,1$
...	...
7	$\mathbf{0\,1} \mid \boldsymbol{\mathit{0\,0}}\,\mathbf{0\,0\,0\,0}\,0\,0\,0\,0 \sim \mathbf{0\,1} \mid \boldsymbol{\mathit{1\,1}}\,\mathbf{1\,1\,1\,1}\,1\,1\,1\,1$
$8 = s_0 - 1$	$\mathbf{1} \mid \boldsymbol{\mathit{0\,0}}\,\mathbf{0\,0\,0\,0\,0}\,0\,0\,0\,0 \sim \mathbf{1} \mid \boldsymbol{\mathit{1\,1}}\,\mathbf{1\,1\,1\,1\,1}\,1\,1\,1\,1$

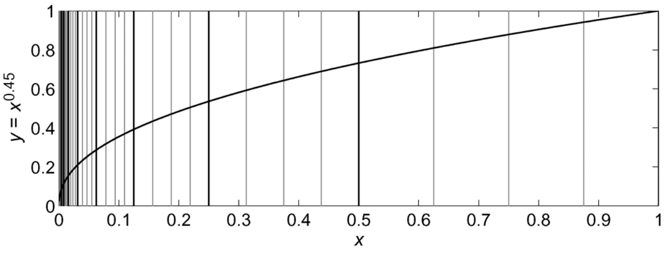

Fig. 3. Segmentation of the $y = x^{0.45}$ gamma correction curve for piece-wise linear approximations to 12-bit input x and 8-bit output y. The error requirement is set to 0.3×2^{-8}. The black and grey vertical lines indicate the boundaries for the outer and inner segmentations, respectively.

where $\text{LZD}(x_0)$ and $\text{MSB}(x_0)$ return the number of leading zeros and the most significant bit of x_0, respectively.

Let $\bar{x}_i$ denote the set of bits that remains constant within an outer segment (bits left to the vertical partition lines in Table I). For instance in Table I, when $j = 3$, then $\bar{x}_0 = 000001$. The inner segmentation uses B_{x_1} bits immediately right of $\bar{x}_0$ (italic bits in Table I). For the case when the outer segmentation is $P2S_L$, the number of bits corresponding to $\bar{x}_0$, $B_{\bar{x}_0}$ can be computed as follows:

$$B_{\bar{x}_0} = \begin{cases} B_{x_0}, & \text{if } P2S_L\_addr = 0 \\ B_{x_0} - P2S_L\_addr + 1, & \text{otherwise.} \end{cases} \tag{5}$$

The original hierarchical segmentation method allows variable numbers of uniform segments to each outer segment, i.e., B_{x_1} can be variable. Two barrel shifters are required due to the variable natures of $B_{\bar{x}_0}$ and B_{x_1}. Barrel shifters are rather costly in hardware in terms of both area and delay. Since display devices require fast response times, stringent requirements are placed on the gamma correction circuity. In $P2S_L$, $B_{\bar{x}_0}$ is inherently variable, hence, the first barrel shifter is mandatory. The second barrel shifter, however, can be omitted by keeping B_{x_1} constant, meaning that the number uniform segments to all outer segments are kept to be the same. We found that for the designs considered in this work, eliminating the second barrel shifter lead to approximately 30% increase in segment count and coefficient table size, but reduced the overall area and latency by approximately 5% and 10%, respectively.

Fig. 3 illustrates the segmentation of the $y = x^{0.45}$ gamma correction curve for piece-wise linear approximations for the case of a 12-bit input x and an 8-bit output y. The polynomial error requirement is set to 0.3×2^{-8} in order to reserve a minimum of 0.7×2^{-8} for finite precision effects (since the total error budget is 2^{-8}). The automated segmentation tool then computes $B_{x_0} = 8$ and $B_{x_1} = 2$, which results

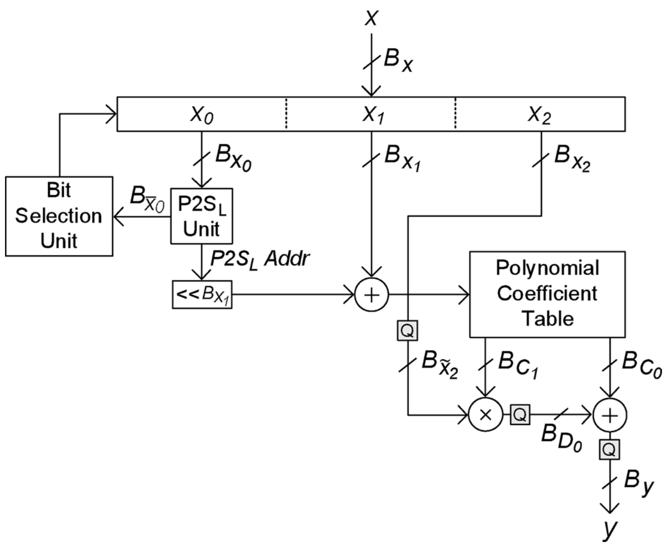

Fig. 4. Proposed gamma correction architecture. The grey "Q" squares perform quantization.

in the minimal number of segments. These parameters mean that there are a total of 9 $P2S_L$ outer segments with four US segments in each, resulting in a total of 36 segments. Chebyshev coefficients are used for the polynomials. Although the results discussed in this paper are based on $\gamma = 1/0.45$ and linear approximations, arbitrary γ functions (such as the ITU Rec. 709 specification [1]) and polynomial degrees are supported by the framework.

III. Architecture and Bit-Width Determination

A. Architecture

Fig. 4 depicts the hardware architecture for piece-wise linear approximation-based gamma correction. The $P2S_L$ unit performs the $P2S_L$ address and $B_{\bar{x}_0}$ computation [(4) and (5)]. The bit selection unit selects the appropriate bits for x_0, x_1, and x_2 from the input x in the following manner:

$$x_0 = x\,[B_x - 1 : B_x - B_{x_0}] \tag{6}$$

$$x_1 = x\,[B_x - B_{\bar{x}_0} - 1 : B_x - B_{\bar{x}_0} - B_{x_1}] \tag{7}$$

$$x_2 = x\,[B_x - B_{\bar{x}_0} - B_{x_1} - 1 : 0]\,. \tag{8}$$

A barrel shifter is present inside this unit due to the variable nature of $B_{\bar{x}_0}$.

The appropriate coefficient table address for a given x is generated by shifting the $P2S_L$ address by B_{x_1} bits to the left and adding x_1 to it. Since x_0 and x_1 are implicitly known for a given segment, x_2 is used instead of x for the polynomial arithmetic to reduce the size of the operators. x_2 is scaled to occupy the range [0,1), which in turn requires appropriate transformations on the Chebyshev coefficients. x_2 is quantized to $B_{\tilde{x}_2}$ bits before it is fed to the multiplier.

Some contemporary gamma correction hardware, such as that found in recent ATI Radeon graphics cards [11], support "programmable gamma correction," where the gamma value can be altered on-the-fly by the user. This feature can be added to the architecture in Fig. 4 by employing a writable polynomial coefficient table, i.e., a RAM. The coefficients corresponding to a set of predefined gamma values can be stored in a separate ROM or in software. Alternatively, full control can be given to the user to supply custom coefficients.

TABLE II
Bit Widths Obtained After Bit-Width Optimization for the 12-bit x/8-bit y Example

Signal	$\tilde{x}_2$	C_1	C_0	D_0
B	8	9	13	14
(IB, FB)	(0,8)	(-2,11)	(1,12)	(-2,16)

B. Bit-Width Determination

In the multiply-and-add step of Fig. 4, signals are quantized between each operation to reduce the size of the operators and coefficients. For the subsequent discussions, the integer bit-width (IB) and the fractional bit-width (FB) of a signal z are denoted by IB_z and FB_z, respectively, i.e., $B_z = \mathrm{IB}_z + \mathrm{FB}_z$. IB governs the range, while FB governs the precision of a signal. Two's complement fixed-point arithmetic is assumed.

A technique based on computing the roots of the derivative of the signal is used to determine the required $\mathrm{IB}s$ for each signal. This range analysis approach allows the computation of the exact range for every signal. For FB determination, an improved version of the MiniBit approach [12] is used to determine the bit widths. For the quantization of a signal z into FB_z fractional bits, the original MiniBit approach assumed a fixed worse case error bound of $2^{-\mathrm{FB}_z}$ for truncation and $2^{-\mathrm{FB}_z-1}$ for round-to-nearest.

By contrast, here, we obtain tighter error bounds by comparing the full precision of the original value against the desired precision of its quantized version. The following modified quantization errors ε_z can be established for truncation and round-to-nearest for a signal z:

$$\text{Truncation}: \varepsilon_z = \max(0, 2^{-\mathrm{FB}_z} - 2^{-\mathrm{FB}_{z'}}) \tag{9}$$

$$\text{Round-to-nearest}: \varepsilon_z = \begin{cases} 0, & \text{if } \mathrm{FB}_z \geq \mathrm{FB}_{z'} \\ 2^{-\mathrm{FB}_z - 1}, & \text{otherwise} \end{cases} \tag{10}$$

where $\mathrm{FB}_{z'}$ is the full precision of the unquantized z. For the addition $z = x + y$ and the multiplication $z = x \times y$, $\mathrm{FB}_{z'}$ is defined as follows:

$$z = x + y: \quad \mathrm{FB}_{z'} = \max(\mathrm{FB}_x, \mathrm{FB}_y) \tag{11}$$

$$z = x \times y: \quad \mathrm{FB}_{z'} = \mathrm{FB}_x + \mathrm{FB}_y. \tag{12}$$

Using the previous equations, an analytical error expression that is a function of the internal signal bit widths can be constructed at the output y in Fig. 4. Simulated annealing is applied to the error expression in conjunction with a hardware area estimation function to determine the optimal $\mathrm{FB}s$ [12] to each signal.

Table II shows the bit widths determined for the 12-bit x/8-bit y example in Section II. Truncation is assumed for $\tilde{x}_2$ and D_0, while round-to-nearest is assumed for C_1, C_0, and y. Note that IB can be negative as in the third column of Table II. In this example, $\mathrm{IB} = -2$ means that the first two fractional bits of C_1 will always be zero. This fact can be exploited in the hardware implementation. The width of the two coefficients is $9 + 13 = 22$ bits. Since there are a total of 36 segments, the polynomial coefficient table size is $22 \times 36 = 792$ bits, which is a factor of 41.4 reduction compared to the 32 768 bits required for direct table lookup.

Fig. 5 shows an error plot for all possible input values for the 12-bit x/8-bit y case with the bit-widths in Table II incorporated. Values computed in IEEE double-precision floating point are used as the reference for error computation. As anticipated, the ulp error of all values are found to be less than 1 ulp. In addition, 94% of the values have an error of less than 1/2 ulp (i.e., exactly rounded) meaning that in most

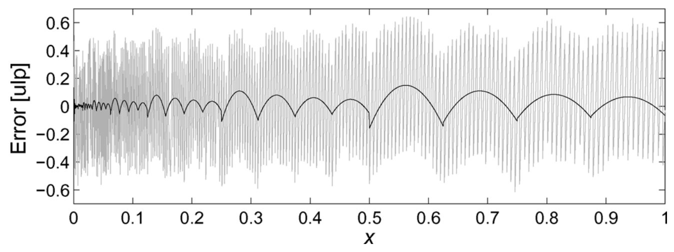

Fig. 5. Error plot for all possible input values for the 12-bit x/8-bit y case. The black curve indicates the inherent approximation error, while the grey curve indicates the error with finite precision effects. 94% of the samples have an error of less than 1/2 ulp.

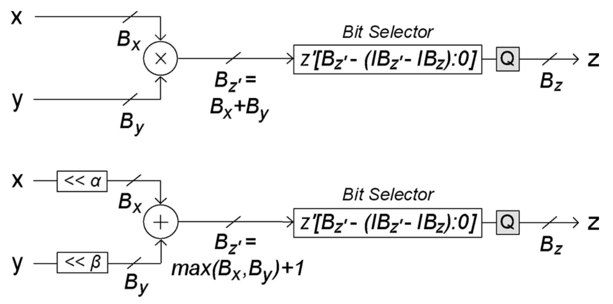

Fig. 6. Emulating fixed-point multiplication and addition using integer arithmetic. The grey "Q" squares perform quantization.

cases the precision obtained is equivalent to a direct table lookup. The mean-squared error is found to be 1.38×10^{-6}, which is marginally higher than the 1.27×10^{-6} of a direct table lookup. Similar trends are observed for other input and output bit-width combinations.

Although hardware design tools facilitate fixed-point arithmetic, support for negative $\mathrm{IB}s$ are not provided. This can be addressed by performing integer arithmetic with an implicit binary point. Fig. 6 illustrates how the fixed-point multiplication $z = x \times y$ and the fixed-point addition $z = x + y$ are performed via integer arithmetic. The multiplier generates the result z' which is $B_x + B_y$ bits wide, where $\mathrm{IB}_{z'} = \mathrm{IB}_x + \mathrm{IB}_y$ and $\mathrm{FB}_{z'} = \mathrm{FB}_x + \mathrm{FB}_y$. The most significant bits (MSBs) and least significant bits (LSBs) of z' that are not to be used in z need now to be eliminated. A bit selector is used to trim off the unwanted $\mathrm{IB}_{z'} - \mathrm{IB}_z$ MSBs. A quantizer is then applied to the resulting signal to eliminate the $\mathrm{FB}_{z'} - \mathrm{FB}_z$ LSBs. The final signal is the desired z that is B_z bits wide.

In addition, it is important to ensure that the implicit binary point of the operands are aligned. If $\mathrm{FB}_x > \mathrm{FB}_y$, α in Fig. 6 is set to 0 and $\beta = \mathrm{FB}_x - \mathrm{FB}_y$. Similarly, if $\mathrm{FB}_x < \mathrm{FB}_y$, then $\alpha = \mathrm{FB}_y - \mathrm{FB}_x$ and $\beta = 0$. For the case when $\mathrm{FB}_x = \mathrm{FB}_y$, both α and β are set to zero. Note that these are constant shifts, requiring just wires and no extra hardware resources. The adder generates the result z' which is $\max(B_x, B_y) + 1$ bits wide, where $\mathrm{IB}_{z'} = \max(\mathrm{IB}_x, \mathrm{IB}_y) + 1$ and $\mathrm{FB}_{z'} = (\mathrm{FB}_x, \mathrm{FB}_y)$. As in multiplication, the bit selector and the quantizer are used to generate the desired output z.

IV. Implementation Results

For hardware implementation, a Xilinx Virtex-4 XC4VLX100-12 field-programmable gate array (FPGA) was used. Designs were written in VHDL, synthesized with Synplicity Synplify Pro 8.4, and placed and routed using Xilinx ISE 8.1.02i.

Table III compares the memory and hardware area requirements for direct table lookup against the proposed piece-wise linear approach with input bit widths ranging from 12 to 14 bits at various output bit widths. Though current graphics formats are typically limited to 12 bits per color, high dynamic range (HDR) graphics which are receiving increasing attention, often require depths beyond 12 bits. For instance, the Leaf Aptus cameras by Kodak capture HDR images with 16 bits per color [13]. For comparison purposes, the designs in Table III use slices (logic elements inside Xilinx FPGAs) only and are fully combinatorial. Embedded RAMs and multiply-and-add units are not used. The table shows that while the memory size of direct table lookup increases exponentially with B_x, only a small linear increase is observed in the piece-wise linear approach. Depending on B_x and B_y, table size reduction factors ranging from approximately 20 to approximately 140 are obtained. Another observation of interest is the rapid increase in table size with B_y of the proposed approach. This is because a single increment in B_y causes the error tolerance to be halved, resulting in significantly more segments. With respect to the area comparisons, hardware area reduction factors ranging from approximately 5 to approximately 20 are obtained using the proposed over direct table lookup. The area reduction is less pronounced memory reduction due to the extra computational hardware involved in the proposed piece-wise linear approach.

TABLE III
Memory and Area Comparisons for Direct Table Lookup Versus the Piecewise Linear Approach

Input B_x	Output B_y	Memory [bits]		Area [slices]	
		Direct Table Lookup	Proposed	Direct Table Lookup	Proposed
12	8	32,768	792	1,035	177
	9	36,864	1,008	1,168	216
	10	40,960	1,728	1,292	233
13	8	65,536	864	2,059	180
	9	73,728	1,100	2,311	220
	10	81,920	1,920	2,573	242
14	8	131,072	920	4,110	191
	9	147,456	1,100	4,625	235
	10	163,840	2,080	5,135	249

The combinatorial delay averages 5 ns for the direct table lookup designs and 38 ns for the proposed designs. Though 38 ns may seem rather large, since we are dealing with feed-forward systems, pipelining can be applied in a straightforward manner to reduce the critical path. We have implemented a heavily pipelined design for 12-bit x/8-bit y on the Virtex-4 XC4VLX100-12 FPGA and found that it occupies 146 slices and one DSP slice (which can do an 18-bit by 18-bit multiplication followed by a 48-bit addition). Embedded 18-Kb RAMs are not utilized since the coefficient table size of this design is only 792 bits. The pipelined design runs at 378 MHz (critical path of 2.6 ns) and has a latency of 13 clock cycles ($\sim$34 ns). The same design has also been mapped onto a low-cost Spartan-3 XC3S5000-5 FPGA, in which the pipelined design is able to run at 180 MHz. Similar clock speeds are obtained for pipelined direct table lookup designs (with registered inputs and outputs), which have a latency of two clock cycles. Although the piecewise linear designs exhibit initial delays of 11 cycles ($\sim$29 ns) more than the direct table lookups, which is a small compromise given the considerable area advantage of the piece-wise linear approach.

We have also performed comparison with respect to the 10-bit x/10-bit y design described by Lin *et al.* [8], which performs linear interpolation with the gamma correction curve partitioned nonlinearly into eight segments. When mapped onto the Virtex-4 XC4VLX100-12 FPGA combinatorially using slices only, the design in [8] occupies 72 slices and has a delay of 11 ns. A 10-bit x/10-bit y implementation using our approach leads to an area of 221 slices and a delay of 33 ns, which is larger by a factor of three in both area and delay. However, the Lin *et al.* design exhibits a maximum error of 85 ulp and a mean squared error of 2.33×10^{-4} which is two orders of magnitude larger than the maximum errors of 0.68 ulp and three orders of magnitude larger than the mean-squared error of 8.50×10^{-8} provided by the design methodology we describe.

V. Conclusion

A flexible and highly efficient hardware architecture for precise gamma correction via piece-wise linear polynomial approximations has been presented. The flexibility of the architecture allows the support of arbitrary gamma values, input bit widths, and output bit widths. The gamma correction curve is segmented in a nonuniform manner, resulting in low segment count while also allowing hardware-efficient polynomial coefficient indexing. Analytical bit-width analysis has been described for deriving the minimal integer and fractional bit widths to each signal in the data path. The analysis allows the outputs to exhibit comparable precision to that of a direct table lookup approach. Experimental results for combinatorial and pipelined implementations on a Xilinx Virtex-4 FPGA have been presented showing significant reductions in memory sizes over direct table lookups.

References

[1] C. Poynton, "Gamma and its disguises: The nonlinear mappings of intensity in perception, CRTs, film and video," *SMPTE J.*, vol. 102, no. 12, pp. 1099–1108, Dec. 1993.

[2] S. Kang, H. Do, B. Cho, S. Chien, and H. Tae, "Improvement of low gray-level linearity using perceived luminance of human visual system in PDP-TV," *IEEE Trans. Consum. Electron.*, vol. 51, no. 1, pp. 204–209, Feb. 2005.

[3] S. Hecht, "A theory of visual intensity discrimination," *J. General Physiol.*, vol. 18, no. 5, pp. 767–789, 1935.

[4] K. Akeley, "Reality engine graphics," in *Proc. ACM Int. Conf. Comput. Graph. Interactive Techn.*, 1993, pp. 109–116.

[5] B. Lucas, "Method and apparatus for converting floating-point pixel values to byte pixel values by table lookup," U.S. Patent 5 528 741, Jun. 18, 1996.

[6] J. Kim, B. Choi, and O. Kwon, "1-billion-color TFT-LCD TV with full HD format," *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1042–1050, Nov. 2005.

[7] D. Warren, A. Bowen, and D. Dignam, "Floating point gamma correction method and system," U.S. Patent 6 304 300, Oct. 16, 2001.

[8] T. Lin, H. Cheng, and C. Kung, "Adaptive piece-wise approximation method for gamma correction," U.S. Patent 6 292 165, Sep. 18, 2001.

[9] E. Kim, S. Jang, S. Lee, T. Jung, and K. Sohng, "Optimal piece linear segments of gamma correction for CMOS image sensors," *IEICE Trans. Electron.*, vol. E88-C, no. 11, pp. 2090–2093, Nov. 2005.

[10] D. Lee, W. Luk, J. Villasenor, and P. Cheung, "Hierarchical segmentation schemes for function evaluation," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, 2003, pp. 92–99.

[11] ATI Technologies Inc., Markham, ON, Canada, "Radeon X1900 graphics technology—GPU specifications," (2006). [Online]. Available: http://www.ati.com/products/RadeonX1900/specs.html

[12] D. Lee, A. Abdul Gaffar, R. Cheung, O. Mencer, W. Luk, and G. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 10, pp. 1990–2000, Oct. 2006.

[13] Kodak, Rochester, NY, "Leaf Aptus 75 specifications," (2005). [Online]. Available: http://www.leaf-photography.com

A Processor-In-Memory Architecture for Multimedia Compression

Brandon J. Jasionowski, Michelle K. Lay, and Martin Margala

***Abstract*—This paper presents the design and development of a novel, low-complexity processor-in-memory (PIM) architecture for image and video compression. By integrating a novel-processing element with SRAM, bandwidth is improved and latency is greatly reduced. This paper also presents PIM design techniques for reduced power, area, and complexity for rapid deployment and reduced cost. A design methodology is presented and followed by an analysis of the processing element performance and capabilities. The proposed datapath solution delivers between 2 to 40 times higher performance compared to other presented solutions. The architecture executes a discrete cosine and wavelet transforms achieving up to 40% higher throughput per watt and occupying as little as 0.9% area compared to a commercial digital signal processing and other application-specified integrated circuit implementations while maintaining precision. A comprehensive comparative analysis is also provided. The proposed processor-in-memory is implemented in 1.8-V 0.18-μm CMOS technology and operates with a 300-MHz clock.**

***Index Terms*—DCT, DWT, image and video compression, low-power architecture, processor-in-memory (PIM), VLSI.**

I. Introduction

There is a growing demand for mobile communication and portable computing with cellular technology and digital photography permeating the mainstream. Therefore, the need for high-speed and low-power signal processing is apparent, especially concerning portable devices, which require extended operation, high speed, and small area.

The proposed architecture utilizes a multiplier-based application-specific processor (ASP). Specifically, the ASP is constructed to compute key algorithms and operations essential to image and video processing in order to minimize the complexity and the power consumption, delivering high throughput at very low cost. The focus in this paper is mainly on the datapath design. The system level design has been previously described in [1] and the memory architecture has been previously described in [2].

This paper is organized as follows. Section II presents the design tradeoffs, optimizations, and the proposed processor-in-memory (PIM) architecture along with simulation and synthesis results in Section III. Section IV describes the comparative study and the discussion of the results. Concluding remarks are made in Section V.

II. PIM Architecture

In order to fully take advantage of the bandwidth available in a PIM design, it is vital to maximize parallelism. Cavalli *et al.* performed an analysis on various MPEG-4 algorithms and found that motion estimation/compensation and DCT coding require the largest amount of

Manuscript received January 7, 2005; revised September 6, 2005 and July 14, 2006.

B. J. Jasionowski is with the SET Corporation, Vienna, VA 22180 USA.

M. K. Lay is with the U.S. Patent and Trademark Office, Alexandria, VA 22314 USA.

M. Margala is with theDepartment of Electrical and Computer Engineering, University of Massachusetts at Lowell, Lowell, MA 01854 USA (e-mail: martin_margala@uml.edu).

Digital Object Identifier 10.1109/TVLSI.2007.893672