TruncApp: A Truncation-based Approximate Divider for Energy Efficient DSP Applications

Shaghayegh Vahdat[1], Mehdi Kamal[1], Ali Afzali-Kusha[1], Massoud Pedram[2], and Zainalabedin Navabi[1]
[1]School of Electrical and Computer Engineering, University of Tehran, Iran
[2]Department of Electrical Engineering, University of Southern California, USA
{vahdat_s, m.kamal, afzali, navabi}@ut.ac.ir, pedram@usc.edu

Abstract— In this paper, we present a high speed yet energy efficient approximate divider where the division operation is performed by multiplying the dividend by the inverse of the divisor. In this structure, truncated value of the dividend is multiplied exactly (approximately) by the approximate inverse value of divisor. To assess the efficacy of the proposed divider, its design parameters are extracted and compared to those of a number of prior art dividers in a 45nm CMOS technology. Results reveal that this structure provides 66% and 52% improvements in the area and energy consumption, respectively, compared to the most advanced prior art approximate divider. In addition, delay and energy consumption of the division operation are reduced about 94.4% and 99.93%, respectively, compared to those of an exact SRT radix-4 divider. Finally, the efficacy of the proposed divider in image processing application is studied.

***Index Terms*—Approximate divider, Truncating, Low power, Energy efficient.**

I. Introduction

Approximate computing is one of the solutions to improve the speed and power efficiency of digital systems exploited in error-tolerant applications such as signal processing and data mining [1]. Arithmetic blocks consume a large portion of the total power consumption of digital processing systems and their speed has a great impact on the overall performance of the system [2]. Among the four basic arithmetic operations, the division operation is less frequently performed while its latency and energy consumption are larger than those of the other operations. Therefore, improving the speed and energy consumption of dividers may lead to improvement in the overall utility of applications (*e.g.*, graphic processing [3]).

There are many division algorithms. A group of them determine quotient and remainder by exploiting subtraction iteratively. As subtraction is a primary operation in this group of dividers, utilizing approximate subtractors in the divider architecture is a primary mechanism to improve the speed and reduce the power consumption [4]. In [5], exact division was performed on the truncated dividend and divisor to improve the performance of the division operation. There are some division algorithms which employ multiplication operation to perform the division. In these multiplicative division algorithms, the inverse (reciprocal) of the divisor is generated and then multiplied by the dividend. In [6], an approximate multiplicative divider, called SEERAD, has been proposed which performed multiplication operation by some add and shift operations. The divisor was truncated to a fixed length, and then, the approximate amount of its inverse value was extracted and stored in a lookup table.

In this paper, we propose a fast and energy efficient truncation-based approximate divider, called TruncApp divider. In this divider, the divisor's approximate inverse value is calculated and multiplied by the truncated dividend. Also, we propose using approximate multiplier to gain more performance with negligible accuracy loss.

The rest of the paper is organized as follows. The algorithm and hardware implementation of the proposed divider are presented in Section II. In Section III, the accuracy of the TruncApp divider is studied. Design parameters of the proposed divider in 45nm CMOS technology are extracted and compared with those of the other exact and approximate dividers in Section IV. Finally, Section V concludes the paper.

II. Proposed Approximate Divider

In this section, first, the algorithm of the proposed divider is explained and its corresponding mathematical equations are stated. Then, its hardware implementation is illustrated.

A. TruncApp Divider

Assume that N is an n-bit unsigned integer. It may be represented as

$$N = \sum_{i=0}^{i=k} 2^i \times x_i = 2^k \times X \quad (1)$$

where x_i is either 0 or 1 and k represents the position of the leading one bit. In (1), X is a fractional number between 1 and 2. According to (1), the value of A/B may be calculated as

$$\frac{A}{B} = 2^{k_A - k_B} \times \frac{X_A}{X_B}. \quad (2)$$

To improve the speed of the division operation, first, we propose to generate t-bit truncated form of X_A and X_B (which we shall denote by $(X_A)_t$ and $(X_B)_t$), and next, calculate the approximate value of $1/(X_B)_t$ (*i.e.*, $(1/(X_B)_t)_{app}$) and multiply it by $(X_A)_t$. Hence, the approximate amount of A/B (*i.e.*, $(A/B)_{app}$) is extracted by

$$\left(\frac{A}{B}\right)_{app} = 2^{k_A - k_B} \times (X_A)_t \times \left(\frac{1}{(X_B)_t}\right)_{app}. \quad (3)$$

Note that the value of $1/X_B$ is always between 0.5 and 1 and it is equal to 1 only in the case when X_B is equal to 1. The occurrence probability of this case is very low. Hence, we ignore this case and assume that the integer part of $1/X_B$ is always "0". Therefore, we propose to calculate the amount of $1/(X_B)_t$ as

$$\left(\frac{1}{(X_B)_t}\right)_{app} = \overline{\left(\frac{(X_B)_t + 1}{2}\right)} \quad (4)$$

where the bar sign indicates that the bits of the result should be inverted. $(1/(X_B)_t)_{app}$ can be produced by inverting the

fractional bits of $(X_B)_t$ and concatenating them with "1". As mentioned before, the value of $(1/(X_B)_t)_{app}$ is greater than or equal to 0.5. Thus, the most significant bit of $(1/(X_B)_t)_{app}$ is always "1" and this is the reason for the concatenation operation. The integer part of $(1/(X_B)_t)_{app}$ is always equal to "0". Hence, it is not required to allocate a bit to show its integer part and consider it in the multiplication step.

To exploit the proposed algorithm for signed dividers, the absolute value of the input operands are generated by using the algorithm proposed in [6]. In this approach, the bits of the input (output) should be inverted if input (output) is negative.

B. Hardware Implementation of TruncApp Divider

A block diagram of the signed TruncApp divider is shown in Fig. 1. First, the approximate absolute values of the input operands are generated by employing an Approximate Absolute unit. Then, the position of the absolute values leading one bits is determined by the Leading One Detector (LOD) block. This block generates k_A and k_B of input operands (see Eq.(2)). This is performed by generating signal K and exerting it to a lookup table. Signal K has $n-1$ bits and only one of them is "1" which corresponds to the leading one bit of the block input operand. Signal K is then exerted to a lookup table which has $\lceil log_2^n \rceil$-bit outputs. Its output represents the value of k_A and k_B. As an example assume that $K_A = (0001000)_2$ where in this case, $k_A = 3 = (011)_2$. Next, the truncated forms of the absolute values are generated using Truncation unit. The output of this unit (T), which has t bits, is generated by

$$T[i] = \bigvee_{j=0}^{j=n-1}(K[j] \wedge I[j+i-(t-1)]) \quad \text{for } i < t \,. \tag{5}$$

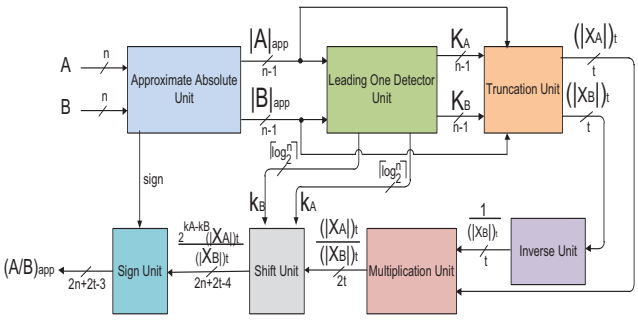

Fig. 1. Block Diagram of the proposed signed Divider.

In the next step, the divisor's approximate inverse value should be generated according to (4), which is performed in the Inverse unit. Based on (3), the result of this unit should be multiplied by the dividend truncated value which is performed in the Multiplication unit. We suggest two implementations for the multiplication unit, exact and approximate. Using approximate multiplier leads to lower delay and power consumption. The dot diagrams of the exact and the proposed approximate multiplication are shown in Fig. 2. The gray circles in Fig. 2(b) show the ignored partial products which are not considered in calculating the result. We call the proposed approximate divider, which employs approximate multiplier in Multiplication unit, as TruncApp_AM divider.

The output of the Multiplication unit should be shifted to the right by $k_B - k_A$ or be shifted to left by $k_A - k_B$ in the Shift unit. Finally, the Sign unit fixes the sign of the final output. If the divider input operands have the same sign, this unit passes its input to the output. In the other case, the bits of the input is inverted and passed to the output.

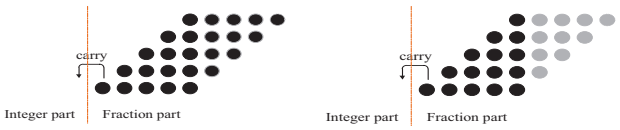

Fig. 2. Dot diagram of the a) exact and b) approximate multiplication step.

III. Accuracy of TruncApp Divider

To measure the accuracy of TruncApp divider, we use the Relative Error (RE) parameter defined by

$$RE = \frac{P_{app} - P}{P} = \frac{P_{app}}{P} - 1 \tag{6}$$

where P (P_{app}) represents the exact (approximate) value of the output. Assume that the division operation is performed on unsigned numbers; hence, based on (3) and (6), the RE of the proposed divider is expressed by

$$RE = \frac{P_{app}}{P} - 1 = \frac{\left(\frac{A}{B}\right)_{app}}{\frac{A}{B}} - 1 = \left(\frac{(X_A)_t}{X_A} \times \frac{\left(\frac{1}{(X_B)_t}\right)_{app}}{\left(\frac{1}{X_B}\right)}\right) - 1 \,. \tag{7}$$

According to (6), if $(A/B)_{app}$ is less (greater) than A/B, the RE value will be negative (positive). To calculate the RE value, the value of $(X_A)_t/X_A$ and $X_B \times (1/(X_B)_t)_{app}$ should be extracted. It is clear that $(X_A)_t$ is always less than or equal to X_A. Hence, the maximum value of $(X_A)_t/X_A$ is 1 and occurs when $X_A = (X_A)_t = 1.0$. On the other hand, when $(X_A)_t = 1.0$ and all of the truncated bits are equal to 1, $(X_A)_t/X_A$ has its minimum value. Thus, the minimum amount of $(X_A)_t/X_A$ is determined by

$$\left(\frac{(X_A)_t}{X_A}\right)_{min} = \frac{1}{1+2^{-(t-1)}-2^{-(n-1)}} \,. \tag{8}$$

To find the minimum and maximum values of $X_B \times (1/(X_B)_t)_{app}$, assume that

$$X_B = 1 + x_b \tag{9}$$

where x_b represents the fractional part of X_B. Hence,

$$(X_B)_t - 1 = (x_b)_{t-1} = \frac{\lfloor (X_B - 1) \times 2^{t-1} \rfloor}{2^{t-1}} \,. \tag{10}$$

Also, we have

$$\begin{aligned} &(x_b)_{t-1} + \overline{(x_b)_{t-1}} = 1 - 2^{-(t-1)} \\ &\Rightarrow \frac{(x_b)_{t-1}}{2} + \overline{\left(\frac{(x_b)_{t-1}}{2}\right)} = 1 - 2^{-t}. \end{aligned} \tag{11}$$

Now, based on (10) and (11), it can be concluded that

$$\overline{\left(\frac{(x_b)_{t-1}}{2}\right)} = 1 - \frac{(x_b)_{t-1}}{2} - 2^{-t}$$
$$= 1 - \frac{\lfloor (X_B - 1) \times 2^{t-1} \rfloor}{2^t} - 2^{-t} \quad (12)$$
$$\Rightarrow \overline{\left(\frac{(x_b)_{t-1}}{2}\right)} = 1 - \frac{1 + \lfloor (X_B - 1) \times 2^{t-1} \rfloor}{2^t}.$$

Finally, based on (4) and (12), we have

$$\left(\frac{1}{(X_B)_t}\right)_{app} = \overline{\left(\frac{(x_b)_{t-1}}{2}\right)} = 1 - \frac{1 + \lfloor (X_B - 1) \times 2^{t-1} \rfloor}{2^t} \quad (13)$$

Although equation (13) determines the value of $X_B \times (1/(X_B)_t)_{app}$, calculating the maximum and minimum of this term is complex. However, this task may be simplified by exploiting a table like Table I, which shows the limits of $X_B \times (1/(X_B)_t)_{app}$ according to X_B for the case where $t = 4$.

Based on (7), the maximum or minimum of *RE* (*MaxRE* or *MinRE*) occur when $(X_A)_t/X_A$ and $X_B \times (1/(X_B)_t)_{app}$ are the maximum or the minimum, respectively. For example, assume that $t = 4$ and $n = 32$. Based on (8), $((X_A)_4/X_A)_{min} \approx 1/(1 + 2^{-3})$ and by exploiting Table I, $(X_B \times (1/(x_B)_4)_{app})_{min} = 0.9375$ and $(X_B \times (1/(x_B)_4)_{app})_{max} = 1.125$. Hence, according to (6), $(RE)_{max} = 12.5\%$ and $(RE)_{min} = -16.67\%$.

Table I. Limits of $X_B \times (1/(x_B)_4)_{app}$ according to X_B values

X_B	X_B limits	$(1/(X_B)_4)_{app}$	$\alpha = X_B \times (1/(X_B)_4)_{app}$
1.000X…X	$1 \le X_B < 1.125$	$(0.1111)_2$=0.9375	$0.9375 \le \alpha < 1.055$
1.001X…X	$1.125 \le X_B < 1.25$	$(0.1110)_2$=0.875	$0.984 \le \alpha < 1.0937$
1.010X…X	$1.25 \le X_B < 1.375$	$(0.1101)_2$=0.8125	$1.0156 \le \alpha < 1.117$
1.011X…X	$1.375 \le X_B < 1.5$	$(0.1100)_2$=0.75	$1.0312 \le \alpha < 1.125$
1.100X…X	$1.5 \le X_B < 1.625$	$(0.1011)_2$=0.6875	$1.0312 \le \alpha < 1.117$
1.101X…X	$1.625 \le X_B < 1.75$	$(0.1010)_2$=0.625	$1.0156 \le \alpha < 1.0937$
1.110X…X	$1.75 \le X_B < 1.875$	$(0.1001)_2$=0.5625	$0.984 \le \alpha < 1.055$
1.111X…X	$1.875 \le X_B < 2$	$(0.1000)_2$=0.5	$0.9375 \le \alpha < 1$

Some error metrics of unsigned 32-bit TruncApp and TruncApp_AM dividers are compared according to their truncation lengths in Fig. 3. The reported error metrics include *MinRE*, the percentage of the outputs with positive *RE* value (*PosRE*), and finally, mean and variance of *RE* absolute values (*MeanARE* and *VarARE*). It should be noted that in all of the cases, *MaxRE* of the dividers is equal to 12.5% and hence it is not reported in Fig. 3. In the rest of the paper, the proposed dividers are indicated by TruncApp(t) and TruncApp_AM(t), where t denotes the truncation length.

As shown in Fig. 3(a), in both proposed structures, the amount of *MinRE* is improved by increasing t value. According to Fig. 3(b), in TruncApp structures with t values greater than 4, *PosRE* is more than 50%, which originates from the fact that the value of $X_B \times (1/(X_B)_t)_{app}$ is larger than 1 in most of the cases. Utilizing the multiplier proposed in Fig. 2(b) leads to smaller output values. Therefore, exploiting TruncApp_AM provides lower *MeanARE* for the cases when t is larger than 4. Based on Fig. 3(c), TruncApp(4) and TruncApp_AM(5) are the optimum choices among other TruncApp and TruncApp_AM dividers according to their least *MeanARE* values. The *VarARE* parameters of proposed dividers are almost the same based on Fig. 3 (d).

The *MeanAREs* and *VarAREs* of TruncApp divider and SEERAD structures under different bit lengths are reported in Table II. As the results show, by increasing the truncation length from 3 to 4, the accuracy of the TruncApp divider becomes higher than the SEERAD in 3rd accuracy level (SEERAD(3)). Among these structures, SEERAD(4) provides the highest accuracy. Note that in both divider structures, the accuracy of the dividers is nearly independent of the bit length of its input operands.

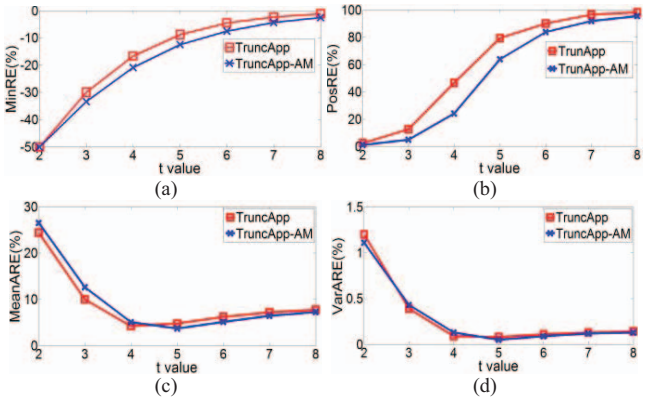

Fig. 3. (a) *MinRE*, (b) *PosRE* , (c) *MeanARE*, and (d) *VarARE* values for 32-bit unsigned TruncApp and TruncApp_AM dividers with different truncation lengths.

Table II. *MeanARE* and *VarARE* of TruncApp divider and SEERAD structures under different input operands bit length

Architecture	8-bit		16-bit		32-bit	
	MeanARE (%)	*VarARE (%)*	*MeanARE (%)*	*VarARE (%)*	*MeanARE (%)*	*VarARE (%)*
SEERAD(1)	16.55	1.1	16.25	1	16.25	1
TruncApp(3)	10.1	0.4	9.8	0.39	10	0.39
SEERAD(2)	9.15	0.41	8.77	0.35	8.77	0.36
SEERAD(3)	4.66	0.1	4.55	0.09	4.55	0.09
TruncApp(4)	4.3	0.09	4.22	0.09	4.2	0.09
TruncApp_AM(5)	3.8	0.06	3.76	0.06	3.67	0.05
SEERAD(4)	2.42	0.03	2.2	0.02	2.2	0.02

IV. Results and Discussion

In this section, first, the efficacy of the 32-bit signed and unsigned approximate dividers is studied, and compared with SEERAD [6] and two exact dividers. The proposed approximate dividers have been described in Verilog HDL and synthesized using Synopsis Design Compiler in NanGate 45nm CMOS Technology [7] and their results are compared to the ones reported in [6]. Next, the efficacy of TruncApp in image processing application is evaluated.

A. Design Parameters of TruncApp

Design parameters of the proposed unsigned and signed TruncApp divider compared to those of the SEERAD, exact Radix-2 SRT and exact Radix-4 SRT dividers are reported in Table III. The architectures are sorted from the highest to the lowest *MeanARE* values. The unsigned TruncApp_AM(5) improves area, power and energy consumption 66%, 64% and 52% compared to SEERAD(3) while it is more accurate. In most of the cases, power and energy consumption of TruncApp architectures is less than those of SEERAD with

almost the same accuracy level by occupying less area. Power, area, energy, EDP, and PDA of the proposed unsigned divider is almost, on average, 75.5%, 72.6%, 74.2%, 73.4% and 96.3% smaller than those of the SEERAD and 98.7%, 94.4%, 99.93%, 99.99%, and 99.99% smaller compared to the exact SRT Radix-4 divider. Also, the speed of the signed TruncApp_AM(5) divider is almost the same as SEERAD(4) while its area and energy consumption are improved by 84% and 85%, respectively.

Table III. Delay, power, area, energy, EDP, and PDA of the unsigned and signed TruncApp and SEERAD compared to the exact SRT structures

	Architecture	Delay (ns)	Power (mW)	Area (μm^2)	Energy (pJ)	EDP (pJ×ns)	PDA (pJ×μm^2)
Unsigned	SEERAD(1)	0.61	0.64	1343	0.39	0.24	524
	TruncApp(3)	0.84	0.56	1261	0.47	0.39	590
	SEERAD(2)	0.7	1.18	2445	0.83	0.58	2020
	SEERAD(3)	0.8	2.12	4378	1.69	1.35	7415
	TruncApp(4)	1.05	0.8	1483	0.84	0.88	1241
	TruncApp_AM(5)	1.08	0.75	1491	0.81	0.88	1211
	SEERAD(4)	1.07	7.53	12451	8.06	8.62	100319
Signed	SEERAD(1)	0.76	0.99	1537	0.75	0.57	1156
	TruncApp(3)	1.08	1.06	1695	1.14	1.24	1940
	SEERAD(2)	0.87	1.69	2684	1.47	1.28	3946
	SEERAD(4)	0.92	3.27	4868	3.01	2.77	14645
	TruncApp(4)	1.2	1.53	2099	1.84	2.20	3854
	TruncApp_AM(5)	1.26	1.42	2048	1.79	2.25	3664
	SEERAD(4)	1.27	9.72	12840	12.34	15.68	158502
	Radix-2 SRT	19.61	60.88	28691	1193	23411	34252945
	Radix-4 SRT	17.63	54.47	25051	960	16930	24056628

The impact of each fundamental blocks of the proposed divider in the total power and delay of the divider is reported in Table IV. As the results reveal, the Inverse unit (Shift unit) has almost the least (most) impact on the power and delay of the divider. As expected, by increasing the truncation length, the impact of Multiplication unit on delay and power consumption of the divider increases.

Table IV. Breakdown of the delay and power of the TruncApp divider structure

	Architecture	*Absolute*	*LOD*	*Truncate*	*Inverse*	*Multiplication*	*shift*	*Sign*
Delay	TruncApp(3)	10.6%	16%	16.8%	3%	17.5%	32%	3.8%
	TruncApp (4)	9.2%	14.5%	15.8%	0%	31.6%	25.7%	3.3%
	TruncApp_AM(5)	6%	22.1%	6.6%	1.2%	39.5%	20.9%	3.6%
Power	TruncApp (3)	10.4%	15.3%	4.9%	0.3%	5.9%	51.3%	7.8%
	TruncApp (4)	8.5%	11.9%	4.7%	0.4%	12.1%	49.8%	7.4%
	TruncApp_AM(5)	9.3%	10.9%	6.9%	0.3%	13.8%	47.4%	6.6%

B. Image Processing Application

To evaluate the effectiveness of the proposed divider in signal processing applications, we have exploited it in image division application [8]. This operation is performed by exploiting exact and approximate dividers and the quality of the approximate outputs are compared with those of the exact ones. The study has been performed on Walter Cronkite, Chemical Plant (close and far views), and Toy vehicle benchmark sequences [9]. Average *PSNR* and *MSSIM* [10] of the exact and approximate output images are measured by MATLAB simulations and the results are reported in Table V. As expected, output images of TruncApp(3) have the lowest quality compared to the other TruncApp structures. The results reveal that TruncApp architectures provide higher performance compared to that of SEERAD. For example, in all benchmarks, *PSNR* and *MSSIM* of TruncApp_AM(5) outputs are equal or higher than SEERAD(4) while *MeanARE* of TruncApp_AM(5) is greater.

Table V- Average *PSNR* and *MSSIM* of TruncApp application in different benchmarks

	Walter Cronkite		Chemical Plant (close view)		Chemical Plant (far view)		Toy vehicle	
Architecture	*PSNR*	*MSSIM*	*PSNR*	*MSSIM*	*PSNR*	*MSSIM*	*PSNR*	*MSSIM*
SEERAD(1)	28.1	0.75	21.6	0.49	18.6	0.51	31.7	0.94
TruncApp(3)	21.9	0.67	21.4	0.77	21.2	0.84	21.9	0.82
SEERAD(2)	29.1	0.75	22.7	0.49	19.9	0.54	31.7	0.94
SEERAD(3)	29.5	0.76	23.1	0.51	20.4	0.57	32.4	0.94
TruncApp(4)	29.8	0.81	29	0.90	28.9	0.94	32.5	0.89
TruncApp_AM(5)	35.7	0.94	35	0.96	34.5	0.98	34.1	0.96
SEERAD(4)	35.7	0.94	23.2	0.50	20.5	0.57	32.4	0.94

V. CONCLUSION

In this paper, a low-power and energy-efficient approximate divider was suggested. In the proposed divider, truncated form of the dividend is (approximately) multiplied by the approximate inverse value of the divisor. Comparing the design parameters of the proposed approximate divider with those of other relevant dividers revealed that this structure provides 66% and 52% improvements in the area and energy consumption, respectively. Also, delay, area, and energy consumption of the proposed unsigned divider were almost, on average, 94.4%, 94.4%, and 99.93% smaller compared to the exact SRT Radix-4 divider. Finally, employing the proposed divider in image division application showed its effectiveness in signal processing applications.

REFERENCES

[1] J. Miao, A. Gerstlauer, and M. Orshansky, "Multi-level approximate logic synthesis under general error constraints," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 504-510.

[2] S. Amanollahi and G. Jaberipur, "Energy efficient VLSI realization of binary-64 division with redundant number systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–8, 2016.

[3] K. Yoshida, T. Sakamoto, and T. Hase, "A 3D graphics library for 32-bit microprocessors for embedded systems," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 1107–1114, 1998.

[4] L. Chen, J. Han, W. Liu, and F. Lombardi, "Design of approximate unsigned integer non-restoring divider for inexact computing," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, 2015, pp. 51-56.

[5] S. Hashemi, R. I. Bahar, and S. Reda, "A low-power dynamic divider for approximate applications," in *Proceedings of the 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, USA, 2016, pp. 1-6.

[6] R. Zendegani, M. Kamal, A. Fayyazi, A. Afzali-Kusha, S. Safari, and M. Pedram, "SEERAD: A high speed yet energy-efficient rounding-based approximate divider," *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 1481-1484.

[7] Nangate 45nm Open Cell Library. http://www.nangate.com/.

[8] U. Qidwai and C.H. Chen, Digital Image Processing: An Algorithmic Approach with MATLAB, CRC Press, 2009.

[9] The USC-SIPI Image Database [Online]. Available: http://sipi.usc.edu/database

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004.