Trading Accuracy for Power with an Underdesigned Multiplier Architecture

Parag Kulkarni(paragk@ucla.edu)*, Puneet Gupta(puneet@ee.ucla.edu)*, Milos Ercegovac(milos@cs.ulca.edu)†
*Department of Electrical Engineering, University of California, Los Angeles
†Department of Computer Science, University of California, Los Angeles

***Abstract*—We propose a novel multiplier architecture with tunable error characteristics, that leverages a modified inaccurate 2x2 building block. Our inaccurate multipliers achieve an average power saving of $31.78\% - 45.4\%$ over corresponding accurate multiplier designs, for an average error of $1.39\% - 3.32\%$. Using image filtering and JPEG compression as sample applications we show that our architecture can achieve 2X - 8X better Signal-Noise-Ratio (SNR) for the same power savings when compared to recent voltage over-scaling based power-error tradeoff methods. We project the multiplier power savings to bigger designs highlighting the fact that the benefits are strongly design-dependent. We compare this circuit-centric approach to power-quality tradeoffs with a pure software adaptation approach for a JPEG example. We also enhance the design to allow for correct operation of the multiplier using a *residual* adder, for non error-resilient applications.**

I. Introduction

ENERGY consumption is a critical design criterion for today's embedded and mobile systems. Significant effort has already been devoted to improve energy efficiency at various levels, from software, to architecture all the way down to circuit and device levels.

Techniques which trade energy for quality of final solution are typically at the algorithmic level and work with parameters such as quantization levels and precision of coefficients [1]–[3], but they utilize accurate building blocks. Any application which can withstand bounded and relatively small errors from their constituent components stands to gain from inaccurate but low-power building blocks. For instance, [4] uses color interpolation filtering to demonstrate graceful degradation of SNR during voltage-scaling by ensuring that important computations are least affected. [5] demonstrated how correctness of arithmetic primitives themselves can be traded for energy consumed. In [6], the authors scale the voltage below the minimum voltage supply value needed, so as to trade accuracy for power. The authors in [7] improved on this by using the observation that errors in bits of higher value affect the quality of the solution more as compared to lower bits, hence they operate adders at more significant bits with a higher voltage and over-scale the voltages for lower bits. [8] introduced the first methodology for a voltage scaling based inaccurate multiplier, their Monte-Carlo simulation based approach achieves a 50% reduction in power using four voltage domains, but they do not report a SNR for the filtering application they use. Such techniques which require multiple voltage domains within a single arithmetic units are likely to be impractical for a realistic design flow due to layout, voltage level conversion etc., overheads which are ignored by [8] and others.

Though, most existing work [5], [7], [8] introduces errors by over-scaling voltage supplies, there has been some work in the direction of introducing error into a system via manipulation of its logic-function, for adders [9]–[11] as well as combinational logic [12]. The focus of the optimization is not power in either case and the latter paper acknowledges poor results for multi-output logic such as arithmetic units. [13] uses inaccurate $4:2$ counters to build adders with fewer stages of logic with power savings of $\sim 3\% - 8\%$. [14] reports power savings of up to 66% without affecting accuracy of programs that manipulate low resolution data, by reducing the bitwidth of floating point multipliers. None of these works provide any way to correct the incorrect output if needed. This may be especially important for general purpose computing hardware which runs a variety of applications.

Majority of the work in probabilistic or inaccurate low-power design has focused on adders and their derivative systems. Multipliers on the other hand are one of the primary sources of power consumption in DSP applications such as Finite-Impulse-Response (FIR) filters [15]. This work focuses on low-power approximate multiplier architectures. Our contributions are as follows.

- We present a 2x2 underdesigned multiplier block and show how it can be used to build arbitrarily large power efficient inaccurate multipliers. The architecture lends itself to easy tunability of error and we present methods to correct error (at a power cost) if needed.
- We evaluate the operation of this multiplier for image filtering and JPEG applications and compare it with voltage scaling based method.
- For a complete study, we also project power savings from different software configurations and compare with our approach.

Rest of this paper is organized as follows. The inaccurate multiplier is described in section II, section III overviews our experimental setup and results, section IV details the impact on real applications and section V introduces a correction mechanism and we conclude in section VI.

II. Inaccurate Multiplier

In this section we introduce the building block for our inaccurate multipliers and show how larger multipliers can be efficiently built from it. We also discuss the associated errors.

A. Building Block

Our objective is to introduce error into the multiplier by manipulating its logic function, using the 2x2 multiplier as

1063-9667/11 $26.00 © 2011 IEEE
DOI 10.1109/VLSID.2011.51

a building block. The modified Karnaugh Map (K-Map) is shown in Fig. 1, with the changed entry highlighted. The motivation behind this change was the observation that by representing the output of $3 * 3$ using three bits (111) instead of the usual four (1001), we are able to significantly reduce the complexity of the circuit. The resulting simpler circuit is shown in Fig. 2a, and has an output that is correct for fifteen out of the sixteen possible inputs. Error occurs with a magnitude of $(9 - 7) = 2$, with a probability of $\frac{1}{16}$. The inaccurate version has close to half the area of the accurate (Fig. 2b) version (see Section III.A); a shorter and faster critical path and less wires. Since the inaccurate version of the 2x2 multiplier has smaller switching capacitance than its accurate counterpart, it offers the potential for significant dynamic power reduction for the same frequency of operation.

B. Building Larger Multipliers

Larger multipliers are built by using the inaccurate 2x2 block to produce partial products and then adding the shifted partial products [16]. Fig. 3, shows an example of a single 4x4 multiplier built out of four 2x2 blocks, where A_H, X_H and A_L, X_L are the upper and lower two bits of inputs A, X respectively. This can sometimes be restrictive for optimization of the adder tree [16]. But since the inaccurate 2x2 building block has no adder or XOR gates, it does not suffer from this restriction. Hence larger multiplier blocks can be built out of the 2x2 building block, and still perform better in terms of power and area as compared to accurate architectures. The results presented in section III will reflect this. Note that our baseline architectures are not built using 2x2 components, but are regular power optimized multipliers which are optimized by a commercial synthesis tool RTL-Compiler (RC) [17]. The optimization of the adder tree is also left to RC, for both the accurate and inaccurate cases. When building larger multipliers, we introduce inaccuracy via the partial products, the adder tree remains accurate. This makes our error rates easily computable and is the topic of the following sub-section.

A_1A_0 \ B_1B_0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	111	110
10	000	010	110	100

Figure 1. K-Map for the inaccurate 2x2 multiplier

C. Error Rates

The 2x2 multiplier introduced in Section II.A, has a small and easily computable error probability of $\frac{1}{16}$ with a max error magnitude of 22.22%. But building multipliers of higher bit widths using the inaccurate multiplier as a building block, leads to slightly more complicated relationships for their error rates. We wrote simulation models in C++ to compute the error probabilities and mean error for higher bit widths. The results in Table I show that while the max-possible error percentage remains constant at 22.22% the probability of error rises with increasing bit-width. But the mean-error increases slowly and

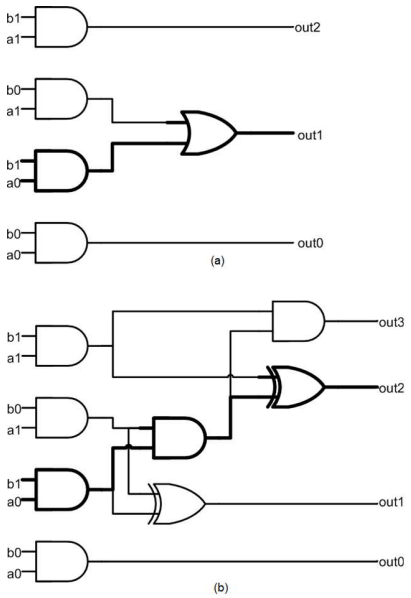

Figure 2. The accurate (b) and inaccurate (a) 2x2 multipliers, with the critical paths highlighted

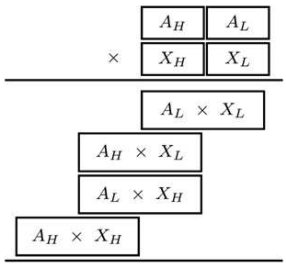

Figure 3. Building larger multipliers from smaller blocks

then almost saturates between $3.3\%-3.35\%$. This is important, as the mean-error is a good indicator of the SNR, as we will see in Section IV. The graph in Fig. 4 gives a clearer understanding of why the mean-error saturates - for higher bit widths the less significant errors are dominant and the larger errors are more unlikely. This results in an almost static mean-error. We will also see in later sections, that the $\sim 3.3\%$ mean-error compares well with the mean-error achieved via other methods.

D. Tunable Error

The inaccurate multiplier we introduced has a fixed mean-error and error-probability for a given bit-width (Table I), but a designer may want to exploit other points on the accuracy-power curve. Since our multiplier is built using 2x2 components, it is possible to replace these individual components with accurate versions to reduce the error rate and mean-error. Such a replacement results in smaller power savings, but provides a means to achieve different points on the error vs. power savings trade-off curve. The resulting power vs. accuracy curve for our inaccurate multiplier is shown in Fig. 7 (dotted line). As expected, increasing the mean-error results in greater power savings.

Table I
ERROR PROBABILITY AND MEAN-ERROR FOR VARYING BIT-WIDTHS

Bit-Width	Error-Prob	Mean-Error	Max-Error
2	0.0625	1.39%	22.22%
4	0.19	2.60%	22.22%
8	0.46	3.25%	22.22%
12	0.675	3.31%	22.22%
16	0.81	3.32%	22.22%

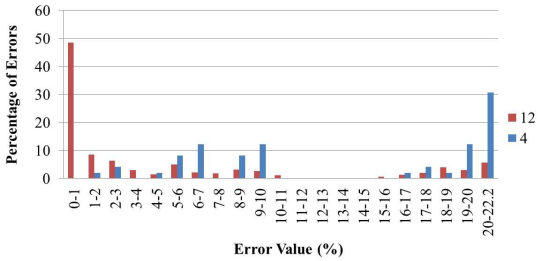

Figure 4. Error percentage distribution for 4-bit and 12-bit inaccurate multipliers

III. EXPERIMENTAL SETUP AND CIRCUIT LEVEL RESULTS

A. Experimental Setup

In this section we present a brief overview of our power measurement methodology and experimental setup. All architectures were written in Verilog, and synthesized by RC [17] to meet the target frequencies. The inaccurate multipliers were built using the 2x2 inaccurate building blocks to generate the partial products; but the adder network to generate the final product was generated and optimized by the tool and was completely accurate. The accurate versions were implemented in two different ways:

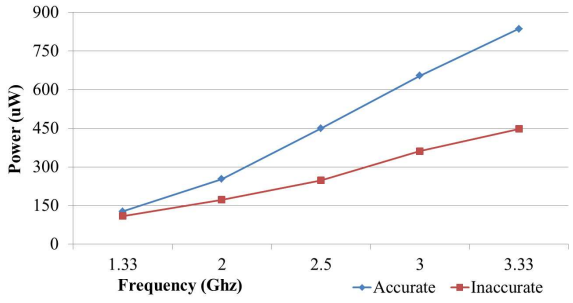

Figure 5. Dynamic power vs. frequency for accurate and inaccurate multipliers

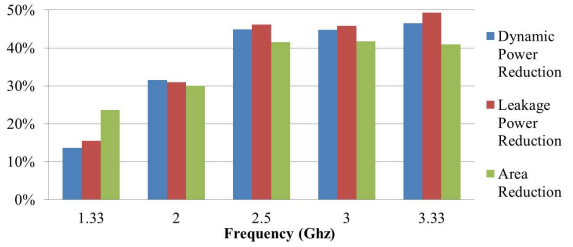

Figure 6. Dynamic power, leakage power and area savings for a 4-bit inaccurate multiplier

Table II
DYNAMIC POWER REDUCTION WITH INCREASING BIT-WIDTHS FOR VARIOUS FREQUENCIES

Bit	F (%)	1.25F (%)	1.5F (%)	1.75F (%)	2F (%)	Avg. (%)
2	44.9	42.1	42.1	48.9	48.9	45.4
4	13.7	31.6	44.8	44.7	46.5	36.3
8	33.1	40.4	26.3	48.8	58.9	41.5
16	25.6	29.6	32.4	33.8	37.4	31.8

1) By generating partial products and adding shifted versions of them as was done for the inaccurate case;
2) The entire architecture selection and optimization of the multiplier is left to synthesis tool.

The best result of the two was used for comparison for each case. To obtain accurate power numbers as well as error characteristics, the synthesized netlists were simulated in NCSIM [18], using all possible input vectors with back-annotated delays [19]. Resulting switching activity information is extracted using a Value-Change-Dump (VCD) file [20], and fed to RC for dynamic power computation. The designs are synthesized using the $45nm$ Nangate open cell library [21]. For voltage scaled versions of the multipliers, the library was recharacterized at different voltage points.

B. Power and Area Results

Fig. 6 shows the reduction in dynamic/leakage power as well as area for a 4-bit inaccurate multiplier. Table II shows the dynamic power reduction ($31.8\% - 45.4\%$) at higher bit-widths and varying frequencies. We take measurements at five different frequency values between F and $2F$, where $2F$ is the maximum possible achievable frequency of the accurate multiplier. We observe that the power benefits of the 2x2 multiplier are carried forward to higher bit-widths. Also increasing the frequency of operation results in greater benefits (Fig. 5). This is because the inaccurate version is inherently faster, and needs less aggressive gate sizing to meet increasing frequency constraints. Less gate sizing results in smaller switching capacitance.

C. Design Level Power Savings

To confirm power savings in a larger design that instantiates it, we used the inaccurate multiplier in a variety of designs from [22]. The results are presented in Table III. As expected the power savings are best in multiplier intensive designs such as the FIR filter, and far less pronounced on other designs such as the mini RISC processor. These results highlight that approximate arithmetic approaches may not be useful for all designs.

Table III
DESIGN LEVEL POWER SAVINGS

Design	# Multi-pliers	# Gates (~)	Mult. Power Reduction	Total Power Reduction
FFT	32	158K	25.166%	13.98%
FIR	4	1.1K	31.09%	18.30%
RISC	1	10K	28.04%	1.51%

Table IV
ERROR RATES AND POWER SAVINGS FOR INACCURATE ADDER BASED MULTIPLIER

Error Prob.	Mean Error	Max Error	Power Reduction
0.29	10.07%	62.22%	37.89%
0.23	6.01%	57.14%	32.35%
0.20	4.40%	57.14%	28.87%
0.15	3.40%	57.14%	20.16%
0.16	3.04%	57.14%	16.83%
0.19	2.92%	44.44%	19.81%
0.12	2.18%	44.44%	12.14%

D. Partial Products vs. Adder Tree

Our design introduces errors via the partial products. Alternatively it is also possible to introduce the inaccuracy via the adder-tree, using an inaccurate adder like the one introduced in [9]. One of the issues with this is that it is hard to analyze the errors, as noted in [8], making it difficult to build a correction unit. For a comparison of the power-accuracy trade-off for such a system, we used the inaccurate adder introduced in [9], to build inaccurate multipliers. Using accurate partial products and by placing these inaccurate adders (best possible locations were exhaustively searched) at different points in the adder tree we were able to obtain the error-power tradeoff. It can be seen from Table IV that the mean and max error from this technique is relatively large. Moreover, the power savings are roughly in the same range as what we encountered before. The accuracy-power tradeoff (Fig. 7) for the partial product technique is better than the inaccurate adder technique.

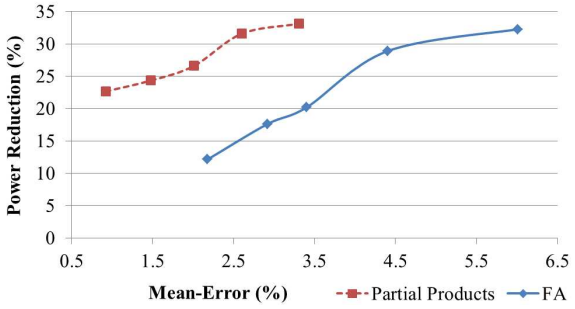

Figure 7. Accuracy vs. power tradeoff comparison for partial product and adder based approaches. The proposed partial product based approach give a much better tradeoff.

IV. IMPACT ON REAL APPLICATIONS

In this section we test our inaccurate multiplier on two image processing applications and then compare software based power-quality tradeoff to our hardware based technique on the JPEG image compression algorithm.

A. Image Filtering

The first application we use is a Gaussian smoothing based image sharpening filter, modeled in MATLAB, similar to the one used in [8]. This is done by convolving the image with a matrix identical to the one presented in [8]. For the inaccurate filter, the 8-bit multiplication in the convolution is performed by an inaccurate multiplier, using its corresponding MATLAB model. Fig. 8 shows the results for accurate as well as various inaccurate multiplier approaches. Our underdesigned multiplier has an average power saving of 41.48% with a SNR of 20.36dB. In comparison, the authors in [5] report a SNR of $19.63dB$ for 21.7% power saving (though for a different technology) over baseline, using four different voltage domains. Fig. 8 (e) and 8 (d) show that our approach results in 2X - 8X better SNR when compared to simple voltage over-scaling [6]. This suggests that image processing/filtering applications could employ the presented inaccurate multiplier with significant power savings and minimal loss in image quality. Note that the SNR for the filtering application is defined between the accurately filtered image and inaccurately filtered image, this was done for sake of uniform comparison with [5], who use this notation. For the JPEG application we revert back to the more common definition, where SNR is defined between the original noise-less image and the filtered result.

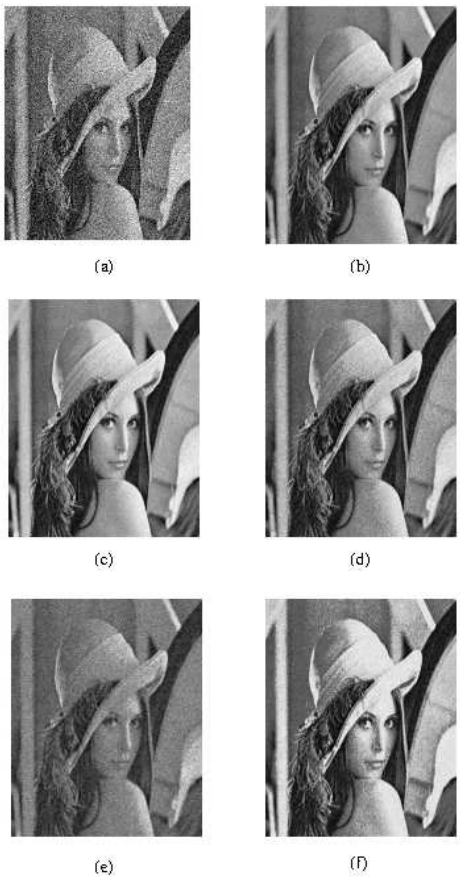

Figure 8. Image sharpening (a) original blurred image; (b) enhanced using accurate multiplier; (c) by inaccurate multiplier, power reduction 41.5%, SNR : $20.365dB$; (d) voltage over-scaling for 30% power reduction, SNR : $9.16dB$; (e) voltage over-scaling for 50% power reduction, SNR : $2.64dB$; (f) by introducing errors via the adder-tree, SNR : $7.3dB$

B. Comparison with Software-level Power-Quality Tradeoff

As a second application we use a JPEG compression algorithm to observe the effects of our inaccurate multiplier on a more complex application and to compare software and hardware based quality tradeoffs. As before, we replace the multiplication in the JPEG algorithm with the model of the inaccurate multiplier. Table V compares compression quality for four benchmark images. The average SNR reduction is found to be roughly in the same range as the mean-error introduced (Table I).

The JPEG algorithm can trade accuracy for runtime by reducing the number of coefficients used for compression, allowing for a software based tradeoff. To compare with the software approach, we synthesized the inaccurate multiplier again to consume the same power as the accurate one but operate at a greater frequency. This would result in speed up of the JPEG application assuming that the multiplier constitutes the critical path of the implementation. We first run the baseline JPEG increasing the number of coefficients (runtime) used, resulting in the SNR vs. runtime curve shown in Fig. 9. We use the same coefficients, but with the inaccurate multiplier, giving us a different (lower) set of SNR points. Using the frequency scaling factor from our synthesis results, we derive a SNR vs. runtime curve for when the multiplier is on the critical path (scaled inaccurate case in Fig. 9). Fig. 9 shows that for the JPEG application, hardware based approach has limited benefits and the software based approach yields a better tradeoff, especially at higher SNR values.

In section II we showed that the inaccurate multiplier can be built to have different values of mean-error and power consumption. Using that resulting accuracy-power curve (Fig. 7), the frequency power table previously presented (Table II) and the SNR vs. runtime curve derived above (Fig. 9), we are able to compare the accuracy vs. power curves of the hardware and software based approaches. In our experiments the total runtime for the JPEG compression is kept constant. We use various configurations of the inaccurate multiplier, each with a different mean-error and power consumption (Fig. 7), yielding a power vs. SNR curve for the hardware based approach. From the runtime vs. SNR curve in Fig. 9 we know the amount of runtime (hence number of coefficients) the software approach would need to achieve the same SNR. Since we keep the runtime constant, we scale the frequency of operation appropriately and use our power-frequency tables to derive a SNR vs. power relationship for the software approach. The comparison of the two in Fig. 10 shows that the hardware based approach still consumes less power than the software one, to achieve the same SNR in a fixed amount of runtime. Though the difference in power consumption is significantly smaller than that of the stand-alone inaccurate multiplier over the baseline. These experiments hold under the assumption that the multiplier determines the frequency of the operation and consumes the bulk of the power.

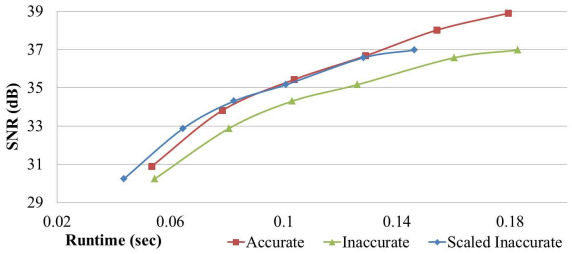

Figure 9. Run-time-accuracy tradeoff for software and hardware based approaches for JPEG compression. Accurate : accurate hardware, purely software based tradeoff. Inaccurate : inaccurate multiplier, when not in critical path. Scaled Inaccurate : inaccurate multiplier and in critical path.

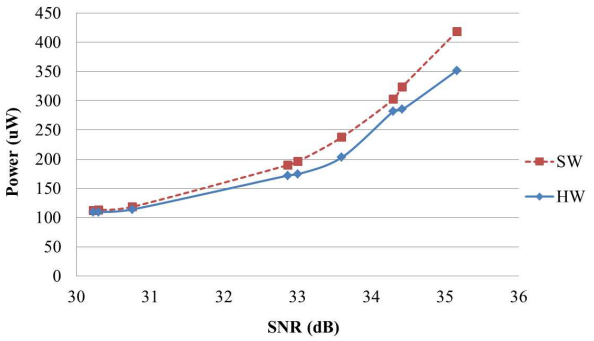

Figure 10. Power-accuracy tradeoff for software and hardware based approaches for JPEG compression

V. Accurate Mode of Operation

One of the advantages of our approach, is that simple decoder logic can be used to detect the magnitude of error for any input vector. This error amount can then be added to the inaccurate product to yield the accurate result when needed. Fig. 12 shows the example of the error-detection and correction unit for the 2x2 case. The AND gate acts as a simple decoder, detecting the $3 * 3$ input vector and the correcting adder adds the required amount (2) when the error triggering input pattern is detected. Such a correction mechanism involves an overhead, and will be less efficient in terms of area than the baseline architecture. Therefore we envision a system (Fig. 11) with two modes of operation - a regular, non-critical and inaccurate mode, and a mission-critical and hence accurate mode. In the non-critical mode, the correction unit will be either completely switched off or power gated, resulting in the basic inaccurate operation, with its significant power savings. In the critical mode of operation, the system produces an accurate result at the cost of greater power in the critical mode of operation, and works at a slightly slower frequency in this mode. We re-ran our initial experiments to evaluate this overhead. Synthesizing the new architecture to work at 0.85 times the original frequency in the accurate-mode, and at the same frequency as the baseline for the inaccurate

Table V
JPEG compression using the inaccurate multiplier

Image	Inaccurate SNR (dB)	Accurate SNR (dB)	SNR Reduction
Lenna	25.19	25.56	1.44%
Coins	19.31	19.55	1.22%
Sand-Dunes	36.5	37.94	3.79%
Fireman	30.23	30.89	2.13%
Average	-	-	2.15%

mode. As before, we synthesized this modified design for multiple frequencies and observed an average area overhead of $4.6\% - 10.5\%$ and in the inaccurate mode an average power overhead of $4.8\% - 8.56\%$ (Table VI).

Table VI
ACCURATE OPERATION AREA AND POWER OVERHEAD

Bit Width	Average Area Overhead	Max Area Overhead	Average Power Overhead	Max Power Overhead
2	4.6%	8.14%	4.8%	8.32%
4	5.87%	7.67%	7.5%	16.14%
8	10.5%	13.44%	8.56%	12.88%
16	9.875%	13.85%	8.22%	13.67%

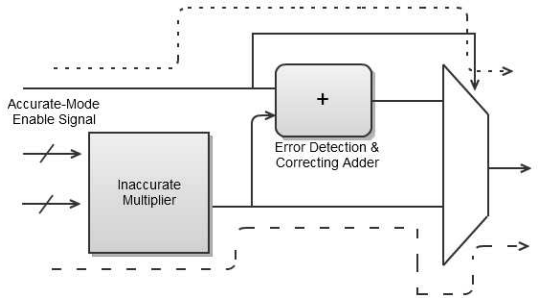

Figure 11. Accurate mode extension, the upper path is for accurate operation and the lower path is for inaccurate operation

VI. CONCLUSION

With a mean error of $1.39\% - 3.35\%$ and power savings between $30\% - 50\%$, the underdesigned multiplier architecture presented allows for trading of accuracy for power. It achieves 2X - 8X better SNR than simple voltage over-scaling techniques, and does not suffer from overheads associated with the multiple voltage domains of advanced over-scaling techniques. A simple correction mechanism is proposed for usage in a critical mode. We also show that introducing errors via partial products is more promising than via the adder tree. The results suggest that design-for-error based techniques have significant potential for power savings, and can be easily integrated into today's automated ASIC design flow. Future work includes extending the approach to other arithmetic components and an algorithm for finding the point of maximum power benefit for a given error rate.

ACKNOWLEDGEMENT

The authors would like to thank Pratyush Aditya from Cadence Design Systems, for his valuable feedback and contributions towards setting up power optimization for multiplier components generated by RC.

REFERENCES

[1] M. Lamoureux, "The poorman's transform: approximating the fourier transform without multiplication," *Signal Processing, IEEE Transactions on*, vol. 41, no. 3, pp. 1413 –1415, mar 1993.

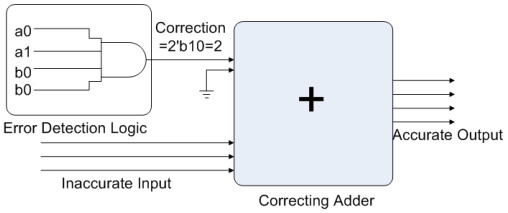

Figure 12. The error detection and correction logic for the 2x2 case

[2] I. Reed, D. Tufts, X. Yu, T. Truong, M.-T. Shih, and X. Yin, "Fourier analysis and signal processing by use of the mobius inversion formula," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 3, pp. 458 –470, mar 1990.
[3] G. Boudreaux-Bartels and T. Parks, "Discrete fourier transform using summation by parts," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '87.*, vol. 12, apr 1987, pp. 1827 – 1830.
[4] N. Banerjee, G. Karakonstantis, J. H. Choi, C. Chakrabarti, and K. Roy, "Design methodology for low power and parametric robustness through output-quality modulation: application to color-interpolation filtering," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 28, no. 8, pp. 1127–1137, 2009.
[5] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*. New York, NY, USA: ACM, 2006, pp. 158–168.
[6] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *ISLPED '99: Proceedings of the 1999 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 1999, pp. 30–35.
[7] K. V. Palem, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Trans. Comput.*, vol. 54, no. 9, pp. 1123–1137, 2005.
[8] M. S. Lau, K.-V. Ling, and Y.-C. Chu, "Energy-aware probabilistic multiplier: design and analysis," in *CASES '09: Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems*. New York, NY, USA: ACM, 2009, pp. 281–290.
[9] D. Shin and S. K. Gupta, "A re-design technique for datapath modules in error tolerant applications," *Asian Test Symposium*, vol. 0, pp. 431–437, 2008.
[10] D. Kelly and B. Phillips, "Arithmetic data value speculation," in *Asia-Pacific Computer Systems Architecture Conference*, 2005, pp. 353–366.
[11] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, 2004.
[12] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *in Proc. 13th IEEE Design, Automation and Test in Europe*, 2010.
[13] B. J. Phillips, D. R. Kelly, and B. W. Ng, "Estimating adders for a low density parity check decoder," F. T. Luk, Ed., vol. 6313, no. 1. SPIE, 2006, p. 631302. [Online]. Available: http://link.aip.org/link/?PSI/6313/631302/1
[14] J. Tong, D. Nagle, and R. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 8, no. 3, pp. 273 –286, jun. 2000.
[15] S. K. Sangjin, S. Hong, M. C. Papaefthymiou, and W. E. Stark, "Low power parallel multiplier design for dsp applications through coefficient optimization," in *in Proc. 12th IEEE International ASIC/SOC Conference*, pp. 286–290.
[16] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. A.K. Peters, 2002.
[17] "Cadence rtl-compiler," http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx.
[18] "Cadence incisive simulator," http://www.cadence.com/products/ld/design_team_simulator/pages/default.%aspx.
[19] "Standard delay format," http://www.vhdl.org/sdf/sdf_3.0.pdf.
[20] "Value change dump file," http://en.wikipedia.org/wiki/Value_change_dump.
[21] "Nangate open cell library," http://www.nangate.com/index.php?option=com_content&task=view&id=137&It%emid=137.
[22] "Opencores," http://www.opencores.org/.