On the Use of Approximate Adders in Carry-Save Multiplier-Accumulators

Darjn Esposito, Davide De Caro, Ettore Napoli, Nicola Petra, Antonio G. M. Strollo
Dept. of Electrical Engineering and Information Technology
University of Napoli "Federico II", Italy
astrollo@unina.it

***Abstract*— Approximate computing improves digital circuit performance by relaxing the requirement of performing exact calculations. In this paper, we investigate the use of approximate adders in the final stage of a carry save multiplier-accumulator (MAC), designed for image filtering application. We propose a design flow based on synthesis tools, starting from HDL description. After a first step in which an exact carry-propagate adder is used, the synthesized netlist is simulated to extract the statistics of the terms summed in the carry-propagate adder. We then design the approximate adder, to meet the required error characteristics given the inputs statistics. The netlist is finally modified by substituting the exact adder with the approximate one, and a final synthesis and optimization is performed. The presented design example in 28nm CMOS shows that a 14% power gain can be obtained, with a limited image quality degradation.**

Keywords—Approximate computing; Approximate adder; Multiplier-Accumulator; Arithmetic circuits; Imprecise hardware.

I. INTRODUCTION

Approximate computing has become a promising approach to improve digital circuit performance by relaxing the requirement of performing exact calculations [1]. This technique is fruitfully applicable to applications that can tolerate some errors in the computed results, such as multimedia processing, data mining and recognition.

Among approximate datapath operators, adders have received the largest interest [2]-[12] being a commonly used subsystem in error-tolerant applications. In an n-bit adder, the worst-case condition in which the carry propagates through all (or most) of the n–bits rarely occurs when inputs are uniformly distributed. Approximate adders exploit this observation and compute their output by assuming that each carry traverses no more that p bits, where $p<n$ is a design parameter. By reducing p, the approximate adder becomes faster but the error rate increases. Approximate adders are segmented by using multiple smaller sub-adders operating in parallel [2]. The Fig.1 shows an example. Following the notation introduced in [6], the i-th sub-adder produces r_i sum bits that contribute to the final result and employs p_i bits used to predict the carry (please note that the sub-adder #0 is an exception, contributing with all its output bits to the final result). The size of each sub-adder is

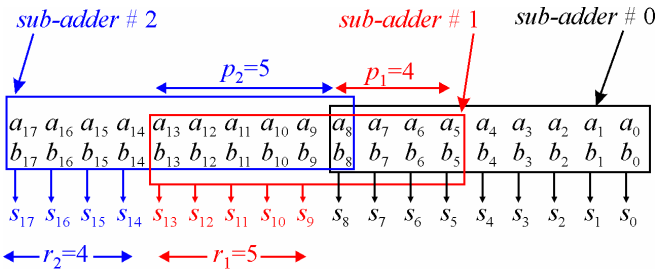

Fig. 1. Speculative adder segmentation. A n=18 bit adder is segmented using three sub-adders. Each sub-adder produces r sum bits that contribute to the final result and employs p bits to predict the carry.

r_i+p_i. The adders proposed in [7]-[10],[12] belong to this general scheme.

In this paper, we investigate the use of approximate adders in the final stage of a carry save multiplier-accumulator (MAC). This is an important application since MACs are common building blocks to perform convolution, which is a basic operation in multimedia signal processing. In particular, we will focus on a MAC designed for image filtering application. Other papers investigate the performance of approximate adders for image filtering application, but perform several simplifying assumptions that are hardly found in practical applications. For example, [11] uses approximate adders for image processing, but assumes that subtractions and multiplications are performed with exact functional units. A similar assumption is used in [9].

In this paper, we consider a realistic MAC architecture composed by a carry-save Wallace tree for partial products compression, followed by a final (approximated) carry-propagated adder. The circuit is designed with the help of automatic synthesis tools, starting from HDL description. In a first step, an exact carry-propagate adder is used. The synthesized netlist is then simulated to extract the statistics of the terms added. We then design the approximate adder, to meet suitable error characteristics given the inputs statistics. The synthesized netlist is finally modified by substituting the exact adder with the approximate one, and a final synthesis and optimization is performed.

In addition to the new design flow, the other main contributions of this paper can be summarized as follows.

978-1-4673-6853-7/17/$31.00 ©2017 IEEE

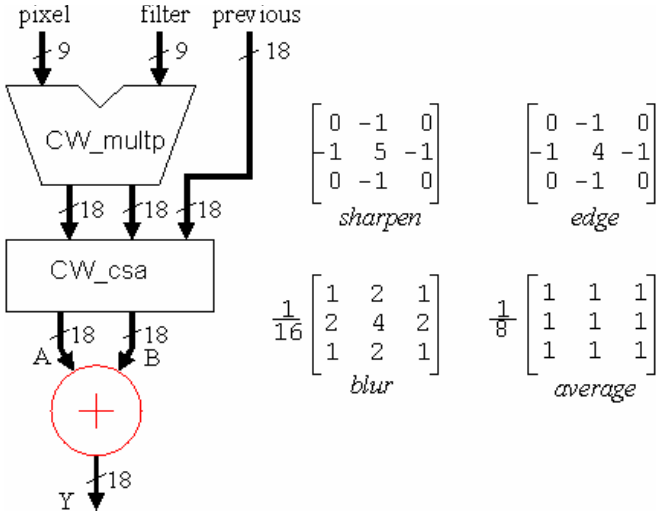

Fig. 2. MAC structure. `CW_multp` is the partial product multiplier, while `CW_csa` is the carry save adder. The final carry propagate adder is highlighted in red. On the right some of the considered 3x3 kernels are reported

Firstly, we show that in a typical image processing application the inputs of the carry-propagate adder are far from being uniformly or Gaussian distributed. As a consequence, formulas presented in literature for predicting error probability [5],[11], [12], are inadequate to judge the actual performance of the MAC. Secondly, we observe that for some kernel filters the errors produced by the approximate adder result in sensible image noise, and we propose a simple approach to reduce this phenomenon. Thirdly, we show that the particular distribution of the arrival times of the inputs of the carry-propagate adder limits the performance improvements related to the use of the approximate adder.

The paper is organized as follows. In Section II we present the proposed design flow and the MAC used as a test example. In Section III we report the error performance for the selected approximate adder configuration and for different filter kernels. Finally, in section IV we present and discuss hardware implementation results in 28nm technology.

II. Design Flow

The structure of the investigated multiplier accumulator is shown in Fig.2. We start from an HDL description of the circuit, which is then elaborated by the synthesizer. The synthesizer exploits datapath extraction to transform arithmetic operators (in our case addition and multiplication) into optimized blocks, using carry save arithmetic to improve performance. A carry-propagate adder is exploited to sum the intermediate outputs of the carry-save stage, giving the final result. Unfortunately, when synthesizing a high-level construct like `y<=A+B*C` in VHDL, the designer has no access to the intermediate carry-save signals. Therefore, to describe the MAC we resort to the use of the arithmetic IP Components available with the synthesis tool. In particular, we used ChipWare IP Components of Cadence Encounter RTL Compiler [13] (similar IPs are also available in other synthesis tools, like DesignWare Building Block IP in Synopsys Design Compiler [14]).

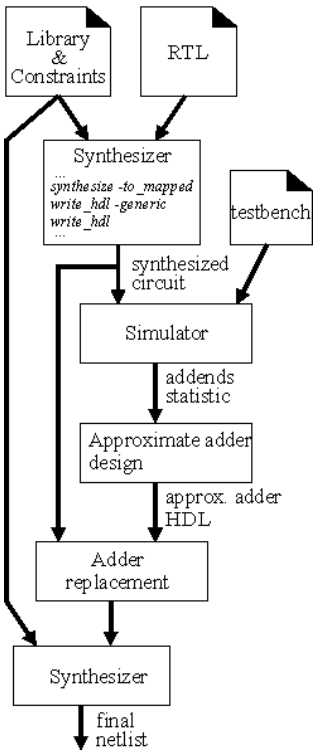

Fig. 3. Proposed design flow.

The MAC is optimized to handle 8bit images. We used a signed multiplier, so the pixel input in Fig. 2 is represented as a 9bit value (its MSB is zero). The filter coefficients are signed and use 4 fractional bits (LSB=2^{-4}), to easily represent commonly used 3x3 kernels (some of the considered kernels are also reported in Fig.2). The block `CW_multp` in Fig.2 is the partial product multiplier, while `CW_csa` is the carry save adder, used to sum an 18-bit addend to the carry-save multiplier outputs. Please note that the datapath is dimensioned to avoid overflow while performing image filtering. The carry-propagate adder highlighted in red in Fig. 2 produces the MAC output Y.

The proposed design flow is shown in Fig. 3. We start with a simple HDL structural description of the MAC. Then we apply constraints and synthesize the circuit. We now simulate the synthesized netlist to extract the statistics of the two addends *A* and *B* summed in the final carry-propagate adder. Please note that this simulation cannot be done with the initial HDL description. The synthesizer, in fact, selects the appropriate architecture for the partial product multiplier depending on the applied constraints. The values of *A* and *B* may hence differ from the values obtained with the simulation models [12] (although their sum remains the same).

The Fig. 4 reports the distributions of *A* and *B* when performing an edge filtering on the Lena test image. As it can be observed, the distribution is neither uniform nor Gaussian; consequently, formulas presented in literature for predicting error probability, while giving an important insight on approximate adders operation, are inadequate to judge the actual performance of the MAC.

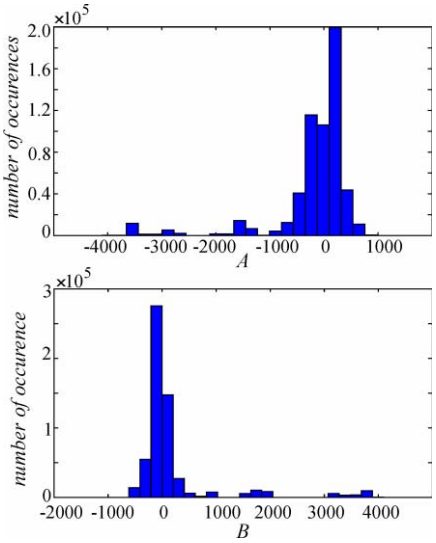

Fig. 4. Distributions of the inputs A and B of the carry-propagate adder while performing an edge filtering on the Lena test image.

In order to design the approximate adder, we numerically extract two set of values from simulated data: the probability of having a carry propagation from bit i to bit j (indicated as $p_{i,j}$) and the probability of having a carry generation from bit i to bit j (indicated as $g_{i,j}$). From these values, we can compute numerically the error probability of the approximate adder. Let us consider the adder shown in Fig. 1. The sub-adder #0 (computing the 9 LSBs) is exact. In the sub-adder #1 an error occurs if the carry-out of sub-adder #0 is high, while the carry c_9 computed by sub-adder #1 is low. When this happens, there is a carry-out coming from bit position #4 that propagates across bits #5,6,7,8. This condition has a probability $P(E_1)=g_{0,4}\ p_{5,8}$. In the sub-adder #2 an error occurs if the carry-out of sub-adder #1 is high, while the carry c_{14} computed by sub-adder #2 is low. Thus, we have an error when there is a carry-out coming from bit position #7 that propagates across bits #8 through #13, and this occurs with a probability $P(E_2)=g_{0,7}\,p_{8,13}$.

Following [6], the overall error probability is given by: $P(E)=P(E_1 \cup E_2)=P(E_1)+P(E_2)-P(E_1 \cap E_2)$. Simple calculations yield, in this example, $P(E)=g_{0,4}\,p_{5,8}+g_{5,7}\,p_{8,13}$. By generalizing this approach, we are able to quickly investigate different approximate adder configurations, after characterizing the distributions of A and B signals. In the following, we consider convolution with 3×3 kernel, performing 9 multiplications for every pixel of the filtered image; hence the probability of computing an erroneous pixel can be estimated as $9P(E)$. It is worth noting that, after convolution, each pixel value is limited to an 8-bit integer value, i.e. any negative value is limited to zero while any positive value is saturated to 255. Therefore, not every error in the convolution results in an erroneous pixel in the output image.

The Fig. 5 displays some results obtained by performing image filtering using the approximate adder of Fig.1 as the MAC carry propagate adder. The approximate adder performs well with the edge filter. The image quality can be quantified with the Structural Similarity Index, SSIM [15]. A value of SSIM=1 means perfect similarity between two images and SSIM=0.98 for the two images in Fig. 5(b) and 5(c). With the blur filter, the use of approximate adder results instead in sensible image noise in the form of spurious black pixels, with SSIM=0.81 for the two images in Fig. 5(d) and 5(e). We propose a simple yet effective approach to reduce this artifact. We augment the approximate adder with a simple circuitry to detect error condition, by checking the carry-out of sub-adders #1 and #2 and the carries c_9 and c_{14} (as previously

(a) Original image (b) edge filter, exact adder (c) edge filter, approximate adder

(d) blur filter, exact adder (e) blur filter, approximate adder (f) blur filter, approximate adder, skipped pixels

Fig. 5. Image filtering results.

TABLE 1. VLSI IMPLEMENTATION RESULTS

design	V_{DD} (V)	Minimum period (ns)	Area (μm^2)	Norm. Power ($\mu W/MHz$)
MAC with Exact adder	1.0	1.38	324	1.46
MAC with Approximate adder	1.0	1.11 (-19%)	341 (+5%)	1.52 (+4%)
MAC with Approximate adder Voltage scaled	0.91	1.38	341	1.26 (-14%)

described). When an error is detected, the convolution is skipped and the previous filtered pixel is outputted. As shown in Fig. 5(f) this simple technique reveals effective, with SSIM=0.98 for the two images in Fig. 5(d) and 5(f).

After completing the design of the approximate adder, as shown in the flow of Fig.3, we modify the netlist by substituting the exact adder with the approximate one. Another synthesis and optimization step yields the final netlist of the MAC.

III. VLSI IMPLEMENTATION RESULTS

We have synthesized the MAC with either exact or approximate adder with STM 28nm technology, standard V_T, typical corner. We have imposed constraints aimed to obtain a minimum area, low-power design.

The Table 1 shows synthesis results. As it can be observed, there is a 19% increase in speed, payed with an 4-5% increase in area and power. We can trade speed for power by voltage scaling the design, as displayed in the last row of Tab. 1. By reducing the supply voltage to 0.91V, the design using the approximate adder has the same speed as the MAC with exact adder, with a 14% power gain. This improvement in performance, while noticeable, is probably less than one would expect.

To better investigate this behavior, the Fig. 6 shows the arrival times for the inputs of the carry propagate adder. As it can be observed, the signals corresponding to the middle bits of the addends arrive later than the others [16]. This partly overcomes the speed advantages related to sub-adder decomposition of approximate adders.

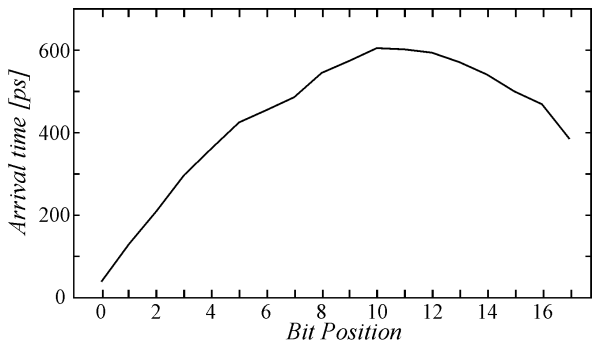

Fig. 6. Arrival times of carry propagate adder inputs.

IV. CONCLUSIONS

We have investigated the use of approximate adders in the final stage of a carry save MAC designed for image filtering application. A 14% power gain is obtained in a voltage scaled design for the small MAC considered in this paper, with a limited image quality degradation. Further investigation is needed to obtain the maximum performance improvement of the approximate adder taking into account the distribution of inputs arrival times.

REFERENCES

[1] Shih-Lien Lu, "Speeding up processing with approximation circuits," Computer , vol.37, no.3, pp.67,73, Mar 2004.

[2] H. Jiang, J. Han, F. Lombardi, "A comparative review and evaluation of approximate adders", Proc. of Great Lakes Symposium on VLSI, pp. 343-348, Pittsburgh, 2015.

[3] D. Esposito, D. De Caro, E. Napoli, N.Petra, A.G.M. Strollo, "Variable Latency Speculative Han-Carlson Adder", IEEE Trans. on Circuits and Systems I: Regular Papers, vol. 62, n. 5, pp. 1353-1361, May 2015.

[4] D. Esposito, D. De Caro, A.G.M. Strollo, "Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands", IEEE Trans. on Circuits and Systems I: Regular Papers, vol. 63, n. 8, Aug. 2016.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders", IEEE Trans. on CAD, vol. 32, no. 1, pp. 124-137, jan. 2013

[6] M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, "A low-latency generic accuracy configurable adder", proc. of Design Automation Conf., DAC'15, pp. 86:1-86:6, June 2015, San Francisco.

[7] A. K. Verma, P. Brisk, P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," proc. of Design, Automation and Test in Europe, DATE '08, pp.1250-1255, March 2008.

[8] N. Zhu, W. L. Goh, G. Wang, K. S. Yeo, "An enhanced low-power high-speed adder for error tolerant applications", proc. of International Symp. on Integrated Circuits, (ISIC), pp. 67-69, 2009.

[9] A. B. Kahng, S. Kang, "Accuracy configurable adder for approximate arithmetic design", proc. of Design Automation Conf., DAC'12, pp.820-825, June 2012, San Francisco.

[10] K. Du, P. Varman, K. Mohanram, "High performance reliable variable latency carry select addition," Design, Automation and Test in Europe DATE'12, pp.1257-1262, March 2012.

[11] C. Liu, J. Han, F. Lombardi, "An analytical framework for evaluating the error characteristics of approximate adders", IEEE Trans. on Computer, vol. 64, n. 5, pp.1268-1281, May 2015.

[12] D. Esposito, G. Castellano, D. De Caro, E. Napoli, N. Petra and A. G. M. Strollo, "Approximate adder with output correction for error tolerant applications and Gaussian distributed inputs," 2016 IEEE ISCAS Conf., Montreal, QC, 2016, pp. 1970-1973.

[13] ChipWare IP Components in Encounter® RTL Compiler, Cadence, Product Version 14.2, August 2015.

[14] Synopsys DesignWare Building Block IP User Guide, 2009.

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

[16] V. G. Oklobdzija, D. Villeger, S. S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach", IEEE Transactions on Computers, Vol. 45 n. 3, March 1996, pp.294-306

.