Evaluation of Approximate Computing Techniques for Power Reduction on FPGAs

Jorge Echavarria, Katja Schütz, Andreas Becher, Stefan Wildermann, Jürgen Teich
Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany
Email: {jorge.a.echavarria, katja.schuetz, andreas.becher, stefan.wildermann, juergen.teich}@fau.de

***Abstract*—Approximate computing allows to tackle conflicting objectives, such as power and accuracy of computations. In this paper we first describe how knowledge of stimuli's specific features can help in quantifying and improving power savings by means of approximate computing. We investigate FPGA implementations of several approximate circuits and compare their power consumption with non-approximating versions. In particular, we study approximate arithmetics and a clock-gate based technique called memoization. Moreover, we compare the accuracy of estimation techniques for power consumption evaluation versus real measurements under controlled environments. We also experimentally quantify the relationship between switching activity and power consumption. Two important results are concluded from our investigations: (1) Approximate arithmetics do not necessarily consume less power than conventional circuits, whereas memoization techniques can in fact reduce power consumption. (2) Simulation-based power evaluation for approximate FPGA implementations can reach fidelity values up to about 90% in input-dependent power characteristics. Yet, to evaluate absolute savings, measurements are needed.**

***Keywords*-Switching Activity; Power Consumption; Approximate Computing.**

I. Introduction

Approximate Computing (AC) makes possible to improve non-functional objectives like latency, throughput, and power consumption for the sake of accuracy. AC is of big advantage when not only one but multiple objectives are considered at once. Usually, such objectives are conflicting. E.g., improving latency or performance resulting in higher power consumption.

In recent years, many approximate designs targeting ASICs and FPGAs have been proposed. Most of them concentrate on how AC can help to improve latency/throughput. In this paper, we concentrate on FPGA circuit implementations and the question whether AC techniques in general do allow for power savings as claimed in literature and in quantifying the impact of AC in terms of dynamic power savings. We show that not all of them are able to obtain savings compared to non-approximating circuit implementations. We validate our findings by power estimations and concrete power measurements.

A. Preliminaries

For studying AC techniques for their potential for power consumption reduction, it is first required to understand the factors influencing power. Eq. (1) describes the power consumed by a CMOS integrated circuit [1]. Static power consumption P_{st} is composed by device and design quiescent power. Device quiescent power represents the power the FPGA consumes independent of its programming; the main contributors are subthreshold leakage, gate leakage, junction temperature and contention current, all of them function independent. Design quiescent power corresponds to the power consumed by the logic, without toggling, but still consuming power regardless of switching activity. Dynamic power consumption P_{dyn}, on the other hand, depends directly on the stimuli fed to the circuit. The dynamic charging and discharging of capacitances in CMOS circuits is the main contributor in P_{dyn}:

$$P = P_{st} + P_{dyn} = P_{st} + \sum_i \alpha_i C_i f V_{DD}^2, \qquad (1)$$

where α_i and C_i correspond to the switching activity and capacitance of signal i, respectively, and the circuit is operated at a frequency f and a supply voltage V_{DD}. While the capacitance, frequency and voltage are expected to be constant during operation, the switching activity is variable and dependent on the circuit's inputs. Clearly, the switching activity –as the rate (percentage) of which a signal (net or logic element) switches its logic value (either from 1 to 0 or from 0 to 1)– scales linearly in P_{dyn}. One premise in this work states that maintaining the value α low should maintain the dynamic power dissipation low as well, and thus, the total power consumption.

B. Related Work

Prior work on approximate arithmetics has introduced low-error high-performance arithmetic units mostly tackling addition and multiplications [2], [3]. The importance of switching activity, on the other hand, can be appraised by the recent attention drawn towards its estimation [4]–[6] and its minimization by means of accurate [7] and approximate computing [8]–[11]. Apart from circuits that use the above well-known approximate computing arithmetic functions, we alternatively study a technique called memoization [12] as an approximation technique based on *accurate* switching activity minimization methodologies, such as [7], to reduce power consumption.

C. Outline

The remaining of the paper is organized as follows. In Section II we describe the applications and approximation techniques considered in this work. In Section III we analyze

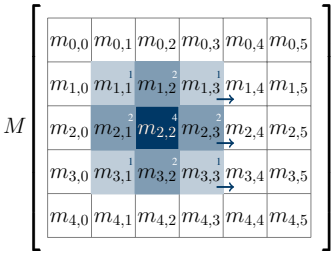


Fig. 1. Convolution on an image matrix M where exemplary the output at pixel $m_{2,2}$ is computed acc. to Eq. (2). The kernel coefficients given in the upper right of the highlighted cells are used for Gaussian filtering.

the capabilities of these techniques for power reduction on FPGAs based on estimations and real power measurements. Finally in Section IV, we conclude with the two major observations that (1) unlike memoization approaches, approximate arithmetic techniques proposed in literature do not per se help to reduce power consumption on FPGA targets, and that (2) estimations provide a high fidelity for correctly comparing different designs for power, but measurements are indispensable in order to quantify absolute power numbers.

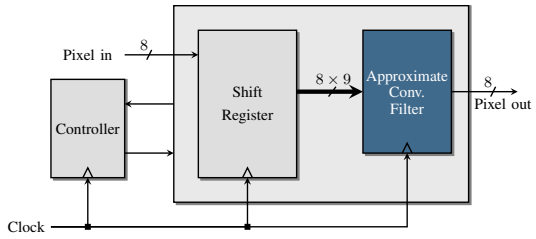
II. Analysis of AC Techniques for Power Reduction

In this paper, we empirically study the impact of approximate computing on power consumption reduction on FPGAs. As use case, we analyze different implementations of convolution filters for digital image processing of grayscale images. Digital image/signal processing is a major application domain for approximate computing as algorithms are computationally expensive, and thus, provide a high potential of gaining benefits (e.g., latency, throughput, power consumption) when applying approximation techniques. Furthermore, they are inherently robust with respect to noise, and thus, can also cope with errors to some degree. For the use case, we implemented a shift register for pixel-wise storing and processing of image frames as well as filter modules for applying 3×3 convolution kernels:

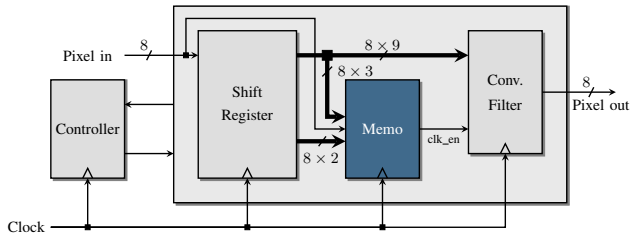
$$out(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3} img(x+i-2, y+j-2) \cdot k_{i,j}, \quad (2)$$

where $k_{i,j}$, $i,j \in \{1,2,3\}$ represent the coefficients of the kernel. Fig. 1 illustrates how a kernel is applied for calculating $out(2,2)$ at pixel $m_{2,2}$.

Fig. 2 illustrates two design variants highlighting different approximation techniques that are analyzed in this paper. The first approach is to use *approximate arithmetics* to implement the convolution filter module. We will analyze two versions of this first architecture, each distinguished by using different approximate arithmetic units from literature. The second approach proposes to use *memoization* by storing and reusing results of previous calculations. The former technique usually has the goal to accelerate the computation, whilst the latter aims to reduce power consumption. In this paper, we want to analyze how both approximation techniques affect the power consumption on an FPGA target.

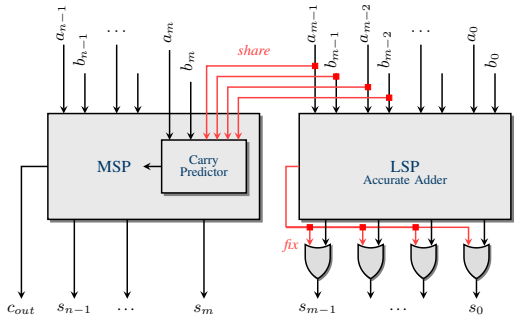


(a) Use case with approximate arithmetics

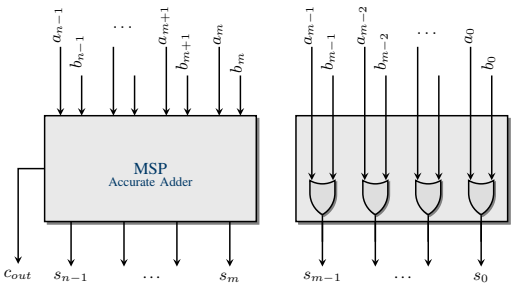


(b) Use case with memoization

Fig. 2. Use case with approximation techniques. Per clock, one pixel of the input image is stored in the shift register and one output is calculated by applying the convolution filter on available pixels.



(a) FAU [2]



(b) LOA [13]

Fig. 3. Approximate adders used within Karatsuba. Note that for our experiments m equals half the bit-width of the addends, that is $\frac{n}{2}$.

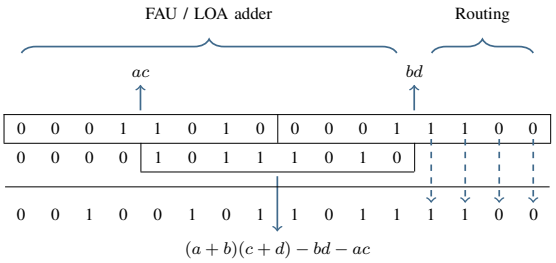
A. Approximate Arithmetics

Adders and multipliers are the basic building blocks in arithmetic circuits, and also required for implementing convolution filtering (see Eq. (2)). In the following, we describe the investigated techniques.

1) Approximate Adders: We consider and compare two alternative approximate adder implementations in this paper. The first represents a sophisticated design utilizing error reduction techniques and the second a simplified implementation of an adder.

TABLE I

APPROXIMATE VERSION OF KARATSUBA ALGORITHM. THE TWO APPROXIMATE VERSIONS OF THE ADDERS ARE SHOWN IN DETAIL IN FIG. 3. NOTE THAT THE SUB-PRODUCT ac AND THE NEW MULTIPLIER $(a+b)(c+d)-bd-ac$ ARE SHIFTED n AND $\frac{n}{2}$ BITS TO THE LEFT, RESPECTIVELY, AS DESCRIBED IN EQ. (3).



The main goal of the *Fast and Error-Optimized Approximate Adder Unit (FAU)* [2] shown in Fig. 3(a) is performance. This approximate adder reduces the critical carry chain of any addition by halving the addends. Each half is still calculated accurately, however, the most significant part (MSP) uses a predicted carry. The prediction of such carry only considers a fixed amount of shared input bits from the least significant part (LSP). However, if the LSP of the adder *observes* a carry that should be propagated to the MSP generated at a position lower than those shared bits (which cannot be predicted due to the limited observability of carry prediction), every bit from the LSP is set to 1. This *fixing* technique allows FAU to reduce the error magnitude as well as the error rate, as shown in [2].

Fig. 3(b) shows the *Lower-part-Or Adder (LOA)* [13]. There are two main differences between LOA and FAU. Firstly, in LOA the LSP is no longer accurately calculated. Instead, the authors propose to **or** the input bits, as shown on the right side of Fig. 3(b). Secondly, there is no error reduction mechanism. Clearly, the power-consumption/accuracy trade-off should be more pronounced when using this approach. Hence, our interest in investigating this adder in more detail.

2) Approximate Multiplier: For the calculation of Eq. (2), we finally need to implement multiplications. Here, we propose approximate multipliers that are built from approximate adders. The design proposed is based on the Karatsuba algorithm [14], which is known as an efficient way to calculate the multiplication of two integer numbers. Let a (c) and b (d) be the most and least significant halves of multiplicand x (multiplier y), respectively, that is:

$$x = a \cdot B^{\frac{n}{2}} + b, \qquad y = c \cdot B^{\frac{n}{2}} + d,$$

Karatsuba and Ofman showed that the product xy can be calculated as follows:

$$xy = ac \cdot B^{n} + bd + ((a+b)(c+d) - bd - ac) \cdot B^{\frac{n}{2}}, \quad (3)$$

where B is the radix of the numeral system. Considering our focus on binary numbers, we can reorder the multiplicand x and multiplier y as shown in Table I following Eq. (3). Note that the $\frac{n}{2}$ least significant bits do not require any extra calculation but simply being correctly routed. Moreover, it can be seen that the product has been reduced to smaller multiplications and a few small additions. We created two approximate versions of Karatsuba by using either FAU or LOA adders only. Of great interest to us is the addition with the longest carry chain in Karatsuba, as shown in Table I (labeled as **FAU / LOA adder**).

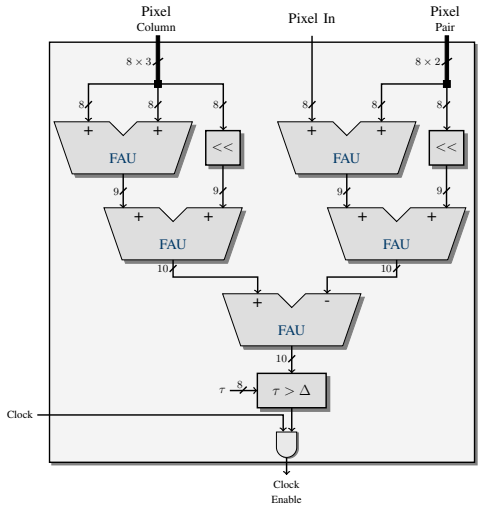


Fig. 4. Component *Memo* from Fig. 2(b). The approximate adder details are shown in Fig. 3(a).

B. Memoization

The second approximation technique considered in this work is called memoization. This approach has been previously investigated by Sinha and Zhang in [12]. The idea consists of reusing previously calculated results as long as the current set of inputs are similar *enough* to the previous ones. Fig. 1 illustrates how this concept can be applied for processing convolution filters. In the image, the background entries represent the image being processed by the filter. The shadowed cells represent the pixels considered by the convolution kernel, where the kernel values of a Gaussian filter are exemplary written in the upper right corner of each shadowed cell. The convolution kernel replaces during each clock cycle each column by its immediate right neighbor for calculating the output for the next pixel: The leftmost column $(m_{1,1}, m_{2,1}, m_{3,1})$ will be replaced by an inner column, whilst the rightmost column is replaced by the outer right column $(m_{1,4}, m_{2,4}, m_{3,4})$. Our memoization approach compares the similarity between these columns:

$$\Delta = |m_{1,1} + 2 \times m_{2,1} + m_{3,1} - m_{1,4} - 2 \times m_{2,4} - m_{3,4}|.$$

If the difference Δ is greater than a threshold τ the new output at pixel $m_{2,3}$ must be calculated acc. to Eq. (2), else, it takes the filtered value of the output calculated for $m_{2,2}$.

Fig. 2(b) shows the implementation of memoization by a component named *Memo*. As seen, *Memo* clock gates the filter. The upper input of the module represents the leftmost column from the kernel $(m_{1,1}, m_{2,1}, m_{3,1})$. The middle signal is the incoming pixel $(m_{3,4})$. And the bottom signal carries the remaining two pixels $(m_{1,4}, m_{2,4})$.

Fig. 4 shows *Memo* in detail. The rightmost signal carries the upper two pixels from the incoming column $(m_{1,4}, m_{2,4})$.

TABLE II
VECTORLESS ESTIMATED DYNAMIC POWER CONSUMPTION P_{dyn} [mW]. EACH INPUT TOGGLE RATE IS FIXED TO 12.5%.

Source	FAU	LOA	Memoization	Accurate
I/O	< 1	< 1	< 1	< 1
Clocks	6	7	6	6
Signals	14	10	12	12
Logic	20	13	14	13
Overall	**41**	**31**	**34**	**32**

The middle signal is the incoming new pixel ($m_{3,4}$). First pixels $m_{3,4}$ and $m_{1,4}$ are summed up whilst pixel $m_{2,4}$ is multiplied by two (see Fig. 1) with a shift operation (for *Memo* we reject the product to avoid overheads). The sum of the resultant values are subtracted from the sum of the kernel's leftmost column ($m_{1,1}, m_{2,1}, m_{3,1}$). As seen in Fig. 4, also in this implementation the additions are performed using an approximate adder to improve performance. Finally, the clock will be gated depending on the difference Δ and a threshold τ which controls the degree of approximation.

III. EXPERIMENTAL RESULTS

In the following experiments, we analyze the power consumption of the proposed FPGA implementations of the two image filter versions distinguished by the type of approximate adders used internally (FAU, LOA), the memoization architecture, and an accurate implementation of the filter.

A. Setup

We use Vivado 2017.1 power estimation and analysis tools. The following power estimations as well as all physical measurements were performed on an Artix-7 35T Arty FPGA Evaluation Kit as target. To reduce artifacts from environmental sources during power measurements, we used a temperature test chamber to ensure a constant ambient temperature. The test chamber is a VT 4002 model from Vötsch. In order to avoid discrepancies, we constrained the placement of our design. The clock period was always set at 15 ns. For simplicity, we will call from now on FAU (LOA) the Gaussian filter implemented with the Karatsuba algorithm which in turn instantiates FAU (LOA) instances internally. Accurate and approximate designs (FAU, LOA, Memoization) were tested with each of the 1,000 stimuli files for vector-based power estimations. For the physical on-board measurements, we took the average over 5,000 runs for each stimuli.

B. Power Estimations

Technically, two different types of power estimations can be distinguished, namely *vectorless* and post routing *vector-based*. A goal of using both estimation techniques is to obtain a more solid conclusion regarding the importance of switching activity and power consumption.

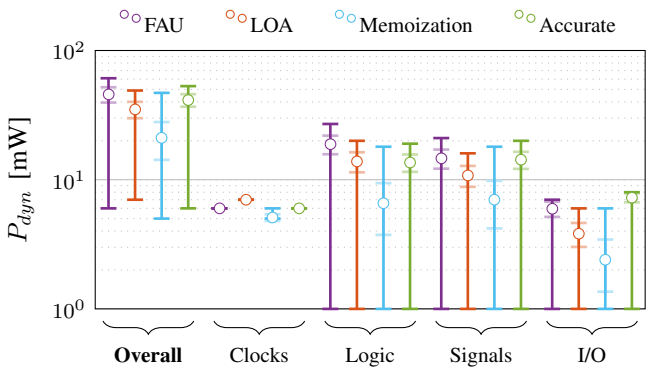


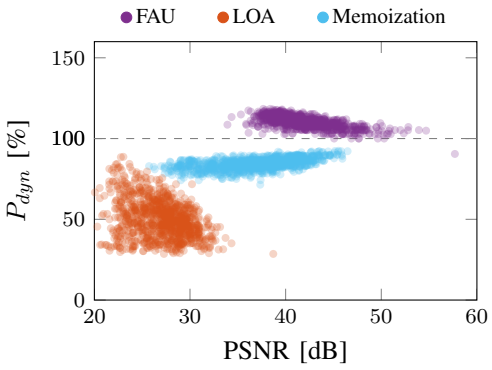
Fig. 5. Min-max bars for the vector-based estimated dynamic power consumption in log scale. The round mark shows the average value. The inner horizontal lines represent the standard deviation.

1) Vectorless: This technique does not require any activity provided either by the user or a simulation log. Instead, vectorless power estimation algorithms fix each node within the net with initial probabilities. Such activity probabilities are then propagated through the entire net until reaching the primary outputs. Finally, the power consumption is estimated using Eq. (1).

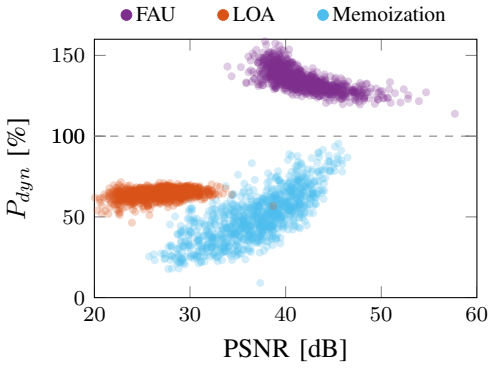
Table II shows the results based on vectorless estimation of power consumption of each design. Interestingly, the three approximate designs show no power reductions ($>$ 1mW) over the accurate filter design. As can also be seen, FAU is the design with the highest power consumption whilst LOA has the lowest. This result shows us that high performance approximate arithmetic units like FAU are not necessarily expected to reduce power consumption. Specially due to possible resource overheads when including error reduction mechanisms, as seen in row *Logic*. However, approximate units like LOA with aggressive trade-offs are in fact expected to reduce power consumption. Another interesting point to observe is that even the implementation using memoization is expected to draw more dynamic power than to the accurate design. Clearly, the reason for this misprediction is that vectorless power estimators expect an overhead introduced by the additional memoization circuitry, as shown in Fig. 2(b).

2) Vector-based: This simulation-based estimation technique requires a constrained design to be placed and routed. Power consumption is evaluated based on simulated traces. For this purpose, we created a benchmark with 1,000 artificial stimuli files. Post routing vector-based power estimation uses the activity log obtained from the simulation to populate the switching activity of each node in the net.

As vector-based estimations require switching activity information obtained from simulation to increase the accuracy of such estimations, the results reported in Fig. 5 show the averaged power consumption of each design (including the accurate version as reference) separated in source of the power dissipation. As seen, the overall average, minimum and maximum possible value for memoization is lower than for all other implementations. The standard deviation shows us that the design memoization obtains more dispersed results. Even more, the positive standard deviation of the memoization



(a) Power estimations



(b) Power measurements

Fig. 6. Trade-off between quality and dynamic power consumption.

design remains below the negative standard deviation of the other approaches.

We may draw two conclusions from these intermediate observations: (1) Approximate arithmetic circuits do not naturally provide any reduction in dynamic power consumption. (2) Approximate computing must address the problem of switching activity minimization in order to obtain any power reductions. Take for example Fig. 6(a) showing the trade-off between quality and power consumption. As can be seen, the approximate design based on LOA allows for power reductions ranging from 10% to 70% w.r.t. the accurate baseline. However, the highest possible PSNR LOA can reach is around 39 dB, while FAU's PSNR ranges from 34 dB to 58 dB, clearly a great quality improvement. In contrast with FAU, the design using the memoization approach expects always power improvements ranging from 8% to 29%. It shows, however, a better quality than LOA. Clearly, memoization seems to be a promising approximation candidate.

C. Power Measurements

In order to get confidence on how good these vector-based simulative estimations reflect reality, we performed power measurements. In the following results, the power consumption values measured were averaged over 5,000 runs per stimuli file. As due to physical constraints, only the overall power P acc. to Eq. (1) could be measured, we estimated the dynamic power P_{dyn} consumption by subtracting the static power P_{st} from it. To obtain P_{st}, we performed the same experiment as

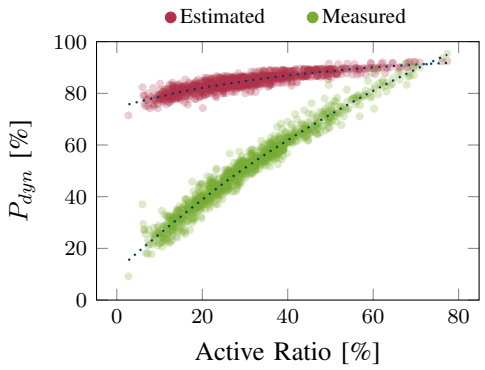


Fig. 7. Influence of the switching activity over P_{dyn} using the memoization design shown in Fig. 2(b). *Active Ratio* represents the time rate signal `clk_en` is active-high.

before but excluding the design from the implementation (i.e., design in Fig. 2(a) without the filter module and design in Fig. 2(b) also without component *Memo.*) The same stimuli files used for the post routing vector-based power estimation were also used here.

Similar to Fig. 6(a), Fig. 6(b) shows the quality/power consumption trade-off obtained from real on-board measurements. The real measurements show that LOA's power consumption is not as widely spread as expected with improvements ranging from 28% to 54%. FAU showed a power overhead higher than the estimated making clear that high performance approximate adders tend to increase the power consumption. Memoization on the other hand showed an even wider range of power consumption improvements than expected. As can be seen, its improvements range from 5% to 91%. This graph also verifies that the extra control component (see Fig. 4) to clock gate the original design does not introduce any effective power overhead.

Fig. 7 shows how the linear dependency expected between switching activity and power consumption holds true, specially for real measurements. Clearly, the higher the amount of time the signals are active, the higher the power consumption. Moreover, as shown in Fig. 8, the rate at which the signals are active clearly affects the output quality. As seen, the more the active ratio drops, the worst the quality gets. In contrast, with a lower difference threshold the quality increases. Our benchmark showed that a threshold of 22 would obtain accurate results (with an infinite PSNR), of course this also means the filter must be (almost) always active.

D. Fidelity

Fig. 9 shows the overall dynamic power consumption of the vectorless and post routing vector-based estimators, and those obtained from real on-board measurements. When looking at the two implementations FAU and Accurate, one could believe that both vectorless and vector-based estimators are underestimating. But LOA and Memoization show that this is not always true. Clearly, estimations and measurements attained deviant results. To adequately evaluate the quality of each estimation technique, we calculate the *fidelity* using a metric based on the Kendall rank correlation coefficient [15].

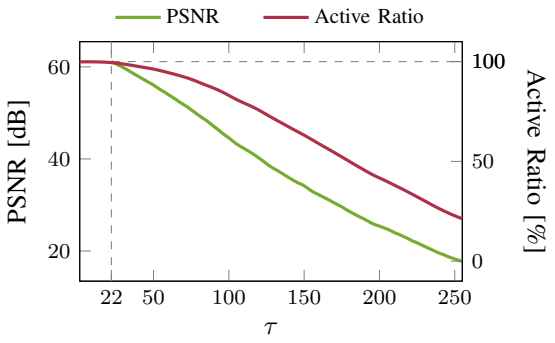


Fig. 8. Relation between signals activity and quality for design memoization. The horizontal axis represents the threshold τ introduced in Section II-B. *Active Ratio* represents the time rate signal `clk_en` (as shown in Fig. 2(b)) is active-high.

Let $E(\Psi)$ be the estimated and $M(\Psi)$ the measured power consumption of an implementation. Let $\Psi = \{\psi_1, \ldots, \psi_n\}$ be the set of stimuli a design is evaluated for, then Eq. (4) estimates the fidelity between $E(\Psi)$ and $M(\Psi)$ [16]:

$$\mathcal{F} = 100 \cdot \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \delta_{i,j}, \tag{4}$$

$$\delta_{i,j} = \begin{cases} 1 & \text{if } (E(\psi_i) \leq E(\psi_j) \wedge M(\psi_i) \leq M(\psi_j)) \vee \\ & \quad (E(\psi_i) > E(\psi_j) \wedge M(\psi_i) > M(\psi_j)) \\ 0 & \text{otherwise.} \end{cases}$$

For the $n = 1,000$ input stimuli simulated, resp. measured, we obtain a value of $\mathcal{F}_{vb} = 89.14\%$ for the vector-based method as average over the four designs. From this, we can conclude that vector-based evaluations have similar tendencies in relation to measured. Whereas, for the vectorless estimator, an average value of $\mathcal{F}_{vl} = 0.0036\%$ is obtained. This evaluator therefore cannot be used to adequately reflect input-dependent power characteristics like those drawn from memoization-based designs.

IV. Conclusions

In this paper, we evaluated approximate circuit designs for their potential to reduce dynamic power consumption in FPGA implementations. Two major results were drawn from our evaluations of different techniques of approximation: (1) Approximate arithmetics as proposed in literature do not necessarily consume less dynamic power than their accurate implementations, whereas switching activity aware techniques as memoization-based designs do. (2) In order to evaluate dynamic power savings, vector-based simulation has a high fidelity of up to 90% to reproduce measured input-dependent power characteristics. Yet, for the evaluation of absolute power margins, board-level measurements are recommended.

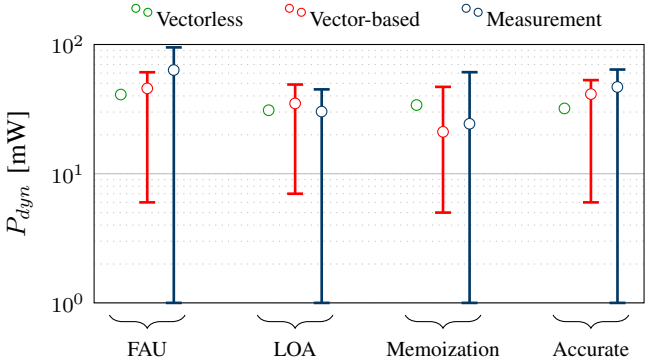


Fig. 9. Min-max bars comparing vectorless and vector-based estimation, and measured dynamic power consumption (log scale). The round marks show the average value. The difference between minimum values of the vector-based estimations and real measurements are believed to be a consequence derived from clock off-sets (note that clocks must remain enabled to obtain the baseline power consumption as explained in Section III-C).

References

[1] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Pearson Education, 2003.

[2] J. Echavarria, S. Wildermann, A. Becher, J. Teich, and D. Ziener, "FAU: Fast and Error-Optimized Approximate Adder Units on LUT-Based FPGAs," in *2016 International Conference on Field-Programmable Technology (FPT)*, Dec 2016, pp. 213–216.

[3] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, 2014, pp. 1–4.

[4] E. Hung, J. J. Davis, J. M. Levine, E. A. Stott, P. Y. K. Cheung, and G. A. Constantinides, "Kapow: A system identification approach to online per-module power estimation in FPGA designs," in *24th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2016, Washington, DC, USA, May 1-3, 2016*, 2016, pp. 56–63.

[5] L. Ren, S. Sun, M. Deo, J. Jaffari, P. Anmula, P. Boyle, J. L. Drewniak, and D. G. Beetner, "A vectorless approach for predicting switching activity in a digital circuit," *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 3, pp. 828–835, June 2016.

[6] Z. Lin, W. Zhang, and S. Sinha, "Decision tree based hardware power monitoring for run time dynamic power management in FPGA," in *27th International Conference on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, September 4-8, 2017*, 2017, pp. 1–8.

[7] P. R. Panda, B. V. N. Silpa, A. Shrivastava, and K. Gummidipudi, *Power-efficient System Design*, 1st ed. Springer US, 2010.

[8] S. Wang, Y.-W. Wu, O. T. C. Chen, and R.-L. Ma, "Low-power multipliers by minimizing inter-data switching activities," in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Cat.No.CH37144)*, vol. 1, 2000, pp. 88–92 vol.1.

[9] A. Sathish, M. M. Latha, and K. L. Kishore, "Efficient switching activity reduction technique for fault tolerant data bus," *International Journal of Computer Applications*, vol. 36, no. 12, pp. 7–12, 2011.

[10] G. Theodoridis, S. Theoharis, D. Soudris, and C. E. Goutis, "Method for minimising the switching activity of two-level logic circuits," *IEE Proceedings - Computers and Digital Techniques*, vol. 145, no. 5, pp. 357–363, Sep 1998.

[11] M. C. Molina, R. Ruiz-Sautua, A. A. D. Barrio, and J. M. Mendias, "Subword switching activity minimization to optimize dynamic power consumption," *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 68–77, 2009.

[12] S. Sinha and W. Zhang, "Low-power FPGA design using memoization-based approximate computing," *IEEE Trans. VLSI Syst.*, vol. 24, no. 8, pp. 2665–2678, 2016.

[13] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. on Circuits and Systems*, vol. 57-I, no. 4, pp. 850–862, 2010.

[14] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," in *Doklady Akad. Nauk SSSR*, vol. 145, no. 293-294, 1962, p. 85.

[15] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[16] J. Teich and C. Haubelt, *Digitale Hardware/Software-Systeme: Synthese und Optimierung*, 2nd ed., ser. eXamen.press. Springer Berlin Heidelberg, 2007.