ATE-Accuracy Trade-Offs for Approximate Adders and Multipliers in Pipelined Processor Datapaths

M. Weißbrich*, A. Najafi†, A. García-Ortiz† and G. Payá-Vayá*
*Institute of Microelectronic Systems, Leibniz Universität Hannover, Appelstraße 4, 30167 Hannover, Germany
Email: {weissbrich, guipava}@ims.uni-hannover.de
†Institute of Electrodynamics and Microelectronics, Universität Bremen, Otto-Hahn-Allee 1, 28359 Bremen, Germany

***Abstract*—The energy efficiency of application-specific processors for high-performance embedded Computer Vision systems can be increased by applying Approximate Computing mechanisms. Due to prevalent error resilience in typical feature extraction or image classification algorithms, the requirement of precise additions and multiplications can be relaxed to obtain more area- and energy-efficient ALU architectures for real-time operation on a limited energy budget. However, state-of-the-art approximate adder and multiplier architectures do not consider the influence of pipelining in processor datapaths on the area-timing-energy (ATE) trade-offs. In this work, a pipelining-aware synthesis flow and ATE-accuracy trade-off exploration is presented, showing a reduction of up to 20% in silicon area and up to 11% in energy consumption when compared to unpipelined approximate units for the same target performance.**

I. Introduction

In the last years, *Approximate Computing* has become a promising design approach for the growing field of energy-efficient, high-performance portable embedded systems. Especially, in *Computer Vision (CV)* applications, often precise computations are not necessary for the functionality of, e.g., image feature extraction or neural network-based image classification algorithms [1]. When relaxing the requirement of fully precise operations, the system design space can be extended by the use of approximate arithmetic units to trade-off computational accuracy for improvements in processing performance, circuit area and energy efficiency [2].

A steadily growing CV application field is video-based *Advanced Driver Assistance Systems (ADAS)*. Image interpretation and scene understanding techniques under tight real-time conditions are commonly used in these systems, which requires specialized processor architectures. Horizontal and vertical vector architectures, i.e., microSIMD [3] and vector processors [4], exploit inherent data-level parallelism in CV algorithms to cope with the requirements on high computational performance at a low energy budget and programming flexibility for future software updates. Due to the error-resilience of the application, approximate additions and multiplications can be used in vector ALUs to further optimize area, timing and energy (ATE) metrics of the processor implementation.

A characteristic of data-parallel processor architectures is pipelined execution to obtain a high operation throughput. Therefore, pipelining of arithmetic units can be implemented without additional control overhead and may be used by the designer for ATE trade-offs of the specialized architecture. However, in state-of-the-art approximate adder and multiplier literature, the presence of pipelined datapaths is not considered and ATE trade-offs are only explored for single-cycle execution. The contribution of this work therefore focuses on:

- A generic implementation strategy for approximate adder and multiplier designs using optimized precise arithmetic sub-components and a configurable number of pipeline stages
- A two-phase synthesis flow with adaptive timing constraints for area-efficient gate-level implementations of pipelined arithmetic units
- ATE-Accuracy trade-off profiling for the implemented pipelined approximate arithmetic units

This paper is organized as follows: Section II presents an overview on frequently referenced approximate adder and multiplier designs. In Section III, the HDL implementation of these units and the synthesis flow for generating pipelined architecture implementations is described. Trade-off profiling results are given in Section IV, and Section V concludes the results and points out future work.

II. Related Work

Several approximate adder and multiplier architectures have been presented during the last decade [5]–[11]. These implementations attend to reduce the power consumption and critical path delay by generating imprecise results. However, no study comparing approximate units in pipelined processor datapaths has been presented so far.

A. Approximate Adders

Approximate adders reduce the critical path delay and power consumption of traditional accurate adder designs by breaking the carry propagation chain, thus generating imprecise addition results. A variety of approximate adders is reviewed and compared in [12]. The authors propose an architecture classification based on the applied approximation mechanisms for the summation, i.e., *carry speculation*, *segmentation* or *approximate full adder cells*. Table I lists the configuration parameters and the approximation classification for the implemented adders.

The most common *carry speculation* adder is the *Almost Correct Adder (ACA)* [5]. In this design, the maximum carry propagation chain for each sum bit is limited to K previous bits. If the speculation on this maximum carry propagation is

TABLE I
CLASSIFICATION OF APPROXIMATE ADDERS [12], [13]

Approximation Class	Error Class	Adder Name			Configuration Parameter
		short	long	Ref.	
Carry Speculation	ILM	ACA	Almost Correct Adder	[5]	Carry propagation chain length K
Segmentation	FSM	ESA	Equal Segmentation Adder	[6]	Sub-adder block width K
		ETA2	Error-Tolerant Adder Type 2	[7]	Sub-adder block width K
Approximate Full Adder Cell	FSM	LOA	Lower-Part OR Adder	[8]	Inaccurate less significant bits K

TABLE II
CLASSIFICATION OF APPROXIMATE MULTIPLIERS [14]

Approximation Class	Multiplier Name			Configuration Parameter
	short	long	Ref.	
Partial Product Generation	UDM	Underdesigned Multiplier	[9]	-
Partial Product Addition	BAM	Broken-Array Multiplier	[8]	Horizontal break level *HBL*, vertical break level *VBL*
	ETM	Error-Tolerant Multiplier	[10]	Inaccurate less significant bits K
	AWTM	Approximate Wallace Tree Multiplier	[11]	Accuracy mode

sufficient for specific operands, the sum output will be totally correct. In less probable cases of longer carry propagation, an erroneous sum bit may be generated in any position, so the error behavior of the ACA can also be classified as *Infrequent Large Magnitude (ILM)* errors [13]. The ACA implementation described in [5] consists of a K-bit precise sub-adder for the LSBs and a look-ahead tree structure to provide the limited carry propagation chain to each more significant sum bit.

For *segmentation*-based approximate adders, the design is split into sub-adders with a fixed length K, allowing no carry propagation across segment boundaries. Due to this property, there are distinct bit positions within the operands at which a carry bit is not propagated. Such adders produce *frequent small magnitude (FSM)* errors when segmentation is applied within the less significant part of the sum output [13]. The *Equal Segmentation Adder (ESA)* [6] and *Error-Tolerant Adder Type 2 (ETA2)* [7] belong to this approximation class. In case of the ESA, carry propagation is only applied within one adder segment, whereas the segment carry input in the ETA2 is coming from exactly one preceding less significant block.

The *Lower-Part OR Adder (LOA)* [8] approximates K sum LSBs by using simple OR gates as *approximate full adder cells* without any carry propagation. The MSB adder remains precise, so FSM error behavior is achieved. In addition to a shorter critical path and less power consumption, the circuit area of this adder is reduced compared to precise implementations because OR gates are significantly smaller than full adder cells.

B. Approximate Multipliers

In [14], approximate multipliers are compared and classified according to where approximation is applied in the multiplication process, i.e., either in the *partial product generation* or in the *partial product addition* step. The classification and configuration parameters for the implemented multipliers are listed in Table II. Because approximation errors accumulate during partial product addition, the reviewed multiplier designs produce imprecise results for virtually every operation. Therefore, a further ILM or FSM classification like for approximate adders is not reasonable here.

The *UnderDesigned Multiplier (UDM)* [9] uses approximate 2-by-2-bit basic blocks for *partial product generation*. By approximating $11_2 \cdot 11_2 = 1001_2$ with $11_2 \cdot 11_2 = 111_2$, the Karnaugh map of the basic block is simplified and only three instead of four bits are passed to a precise partial product addition tree, which reduces the overall power consumption.

Instead of generating approximate partial products, an imprecise *partial product addition* is performed in the *Broken-Array Multiplier (BAM)* [8], the *Error-Tolerant Multiplier (ETM)* [10] and *Approximate Wallace Tree Multiplier (AWTM)* [11]. In the BAM, the partial product array is truncated by some less significant rows or columns, parameterized by the *horizontal break level HBL* or the *vertical break level VBL*, respectively. The ETM truncates K LSBs from the input operands and applies precise multiplication on the input MSBs and simple operand ceiling on the LSBs to reduce the approximation error. In both cases, truncation leads to a significantly smaller circuit area and reduced power consumption.

In the *Approximate Wallace Tree Multiplier (AWTM)* [11], several inner columns of the partial product array are not accumulated, leading to less input bits to the carry-save reduction tree used for *partial product addition*. The authors of [11] propose an accuracy-configurable hierarchical architecture built out of smaller, either precise or approximate multiplication blocks. This way, the trade-off between approximation error and power consumption can be adapted to the application.

C. Approximation Error Metrics

To evaluate the accuracy of approximate circuits, error metrics have been proposed and applied to both adders and

multipliers in [12], [14], [15]. The *Error Rate (ER)* considers the amount of expected imprecise results and is defined as the probability of generating an inaccurate result. Error distance metrics like the *Relative Error Distance (RED)* are used to evaluate the deviation of the approximate arithmetic result to the precise one. The RED is defined as

$$\text{RED} = \frac{\|M' - M\|}{M} \tag{1}$$

where *M'* and *M* are the approximate and precise result, respectively. The RED can be specified for any input operand combination with a non-zero precise result. When averaged over all possible input operand combinations, the *Mean Relative Error Distance (MRED)* is obtained. Both ER and MRED are quantified by using either analytical model or Monte Carlo simulation approaches.

III. Generic Approximate Arithmetic Library

The approximate arithmetic units presented in Tables I and II are implemented using a generic VHDL description regarding bitwidth, the number of pipeline stages and the design-specific configuration parameters. To establish a fair comparison, precise sub-adder or sub-multiplier components within the approximate architectures are inferred using VHDL '+' and '*' operators, allowing the ASIC synthesis tool to select area- and delay-efficient architectures. Moreover, the Synopsys Design Compiler is used, which provides a DesignWare library including highly optimized arithmetic units. It is worth mentioning that, in this work, the *DWF_sum* function, which implements a carry-save summation tree [16], is applied in the UDM, BAM and AWTM multipliers for partial product addition.

In order to obtain a precise reference adder and multiplier, the VHDL operators '+' and '*' are also used, since this is the standard method to infer arithmetic units in processor architectures. The synthesis tool is parameterized to select a both area- and speed-aware implementation. For a typical operand bitwidth of 32 bit in general-purpose processors, this results in a parallel-prefix adder architecture and a Radix-4 Booth multiplier architecture. Due to this selection, the partial product array for the truncation-based BAM is implemented as a Radix-4 Booth-encoded representation [17] to obtain competitive circuit area and critical path delay results.

In order to efficiently balance the pipeline registers, a two-phase ASIC synthesis flow is implemented as depicted in Fig. 1:

1) In the first phase, an area-efficient architecture selection and mapping is performed by synthesizing the target arithmetic unit with a relaxed timing constraint $t_{c,initial}$
2) In the second phase, an incremental synthesis for the desired timing constraint $t_{c,retime}$ is performed with activated retiming/register balancing

This method leads to better area results than directly applying the desired timing constraint in the first phase, as it is commonly done in a standard synthesis flow. For the precise Radix-4 Booth multiplier, the resulting circuit area is found to be up

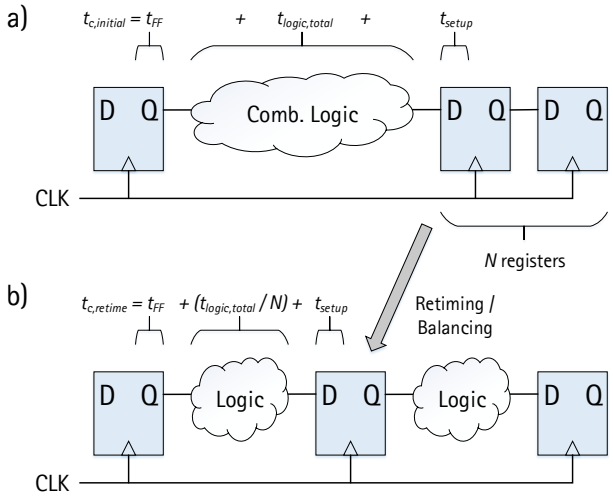

Fig. 1. Two-phase ASIC synthesis flow with a) the initial architecture selection and mapping with a relaxed timing constraint and b) the retiming and register balancing phase with the desired timing constraint.

to 10% smaller. The chosen relaxed timing constraint $t_{c,initial}$ for the first phase is derived from the idealized equations

$$t_{c,initial} = t_{logic,total} + t_{FF} + t_{setup} \tag{2}$$

and

$$t_{c,retime} = \frac{t_{logic,total}}{N} + t_{FF} + t_{setup} \tag{3}$$

for balancing $N - 1$ register stages in Fig. 1. By combining Eq. 2 and 3, the relaxed constraint

$$t_{c,initial} = N \cdot t_{c,retime} - (N-1) \cdot \left(t_{FF} + t_{setup}\right) \tag{4}$$

is obtained. $t_{logic,total}$ is the critical path of the total combinational logic, t_{FF} is the register output propagation delay and t_{setup} is the required register setup time. For a 40 nm low-power standard cell technology used in this work, $t_{FF} + t_{setup}$ is estimated to a worst-case value of 0.35 ns.

IV. Trade-Off Profiling for Pipelined Approximate Arithmetic Units

In this section, the implemented approximate adders and multipliers are profiled for a bitwidth of 32 bit commonly found in general-purpose processor datapaths. The explored design space includes the different adder and multiplier architectures shown in Table I and II. More specifically, the impact on the silicon area requirement and energy consumption per operation when increasing the number of pipeline registers or decreasing the desired clock period constraint is evaluated. Error rate (ER) and mean relative error distance (MRED) are the considered metrics for the approximation error.

Each adder and multiplier architecture is synthesized using Synopsys Design Compiler for a 40 nm low-power standard cell technology. The two-phase flow described in Section III is applied to generate pipelined designs. Both approximation error and energy per operation are extracted from netlist simulations and switching activity evaluation using Mentor Graphics ModelSim [18] and Synopsys PrimeTime [16] for

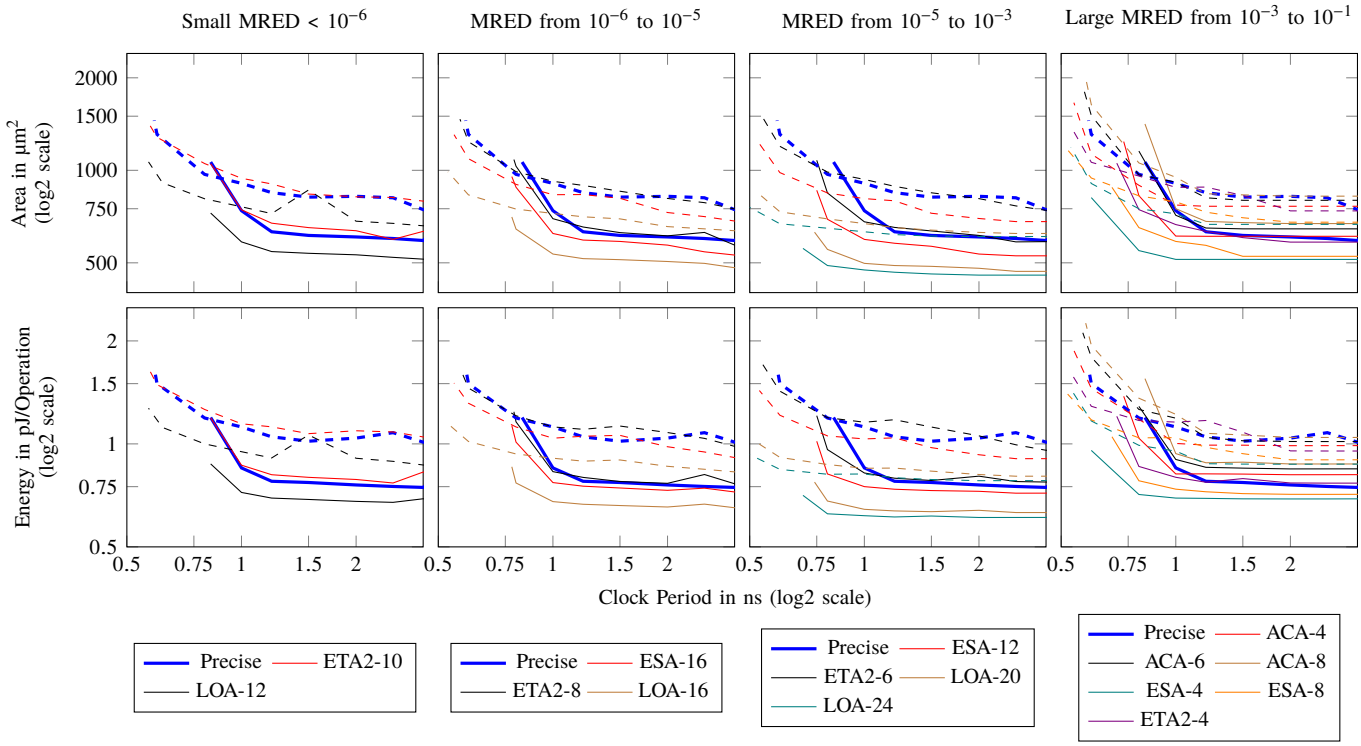

Fig. 2. Total area (top row) and energy consumption (bottom row) of 32 bit approximate adders as a function of the target clock period constraint. Solid lines denote designs with zero internal pipeline stages (input and output registration only), whereas the dashed lines denote designs with one additional internal pipeline stage. The precise reference adder is emphasized by thick lines within each plot.

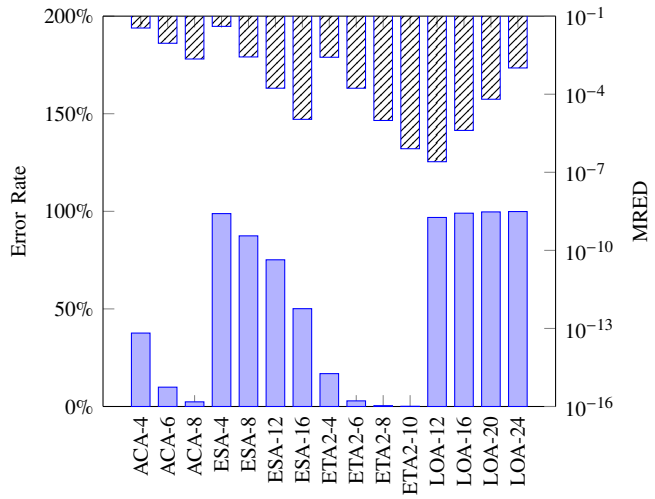

Fig. 3. Error rate (filled bars) and MRED (pattern bars) of 32 bit approximate adders. The value of an adder-specific configuration parameter is indicated as a suffix to the adder name.

10^5 uniformly-distributed pseudo-random operations with the full numeric range of 32 bit operands. Although the latency in clock cycles of a single arithmetic operation is increased by pipelining, the input operands are treated as being data-independent, like in vector processor architectures. Therefore, pipeline hazards are not considered and one operation finishes every clock cycle, resulting in equal time intervals for energy analysis of pipelined and unpipelined designs. ER and MRED are also independent from the number of pipeline stages because the circuit functionality is not influenced by register balancing.

A. Analysis of Pipelined Approximate Adders

Fig. 2 depicts the silicon area and energy per operation of the ACA with parameter $K = \{4, 6, 8\}$, the ESA with $K = \{4, 8, 12, 16\}$, the ETA2 with $K = \{4, 6, 8, 10\}$ and the LOA with $K = \{12, 16, 20, 24\}$ as functions of the target clock period constraint (0.5 to 3 ns), the amount of pipelining (zero to one internal register stages) and the desired MRED. ER and MRED are illustrated in Fig. 3. The following conclusions can be drawn:

- Throughout the complete MRED range, full adder approximation in the LOA is the most area- and energy-efficient approximation method. With the LOA-24, the area is reduced by up to 25% and the power is reduced by up to 20% in the clock period range of 1 to 3 ns without pipelining compared to the precise adder.
- Without internal pipeline stages, ESA and ETA2 designs are able to gain significant performance increases of up to 30% compared to the precise adder due to smaller possible clock periods down to 0.6 ns without violating the critical path. However, this is accompanied by errors larger than 10^{-5} in MRED.

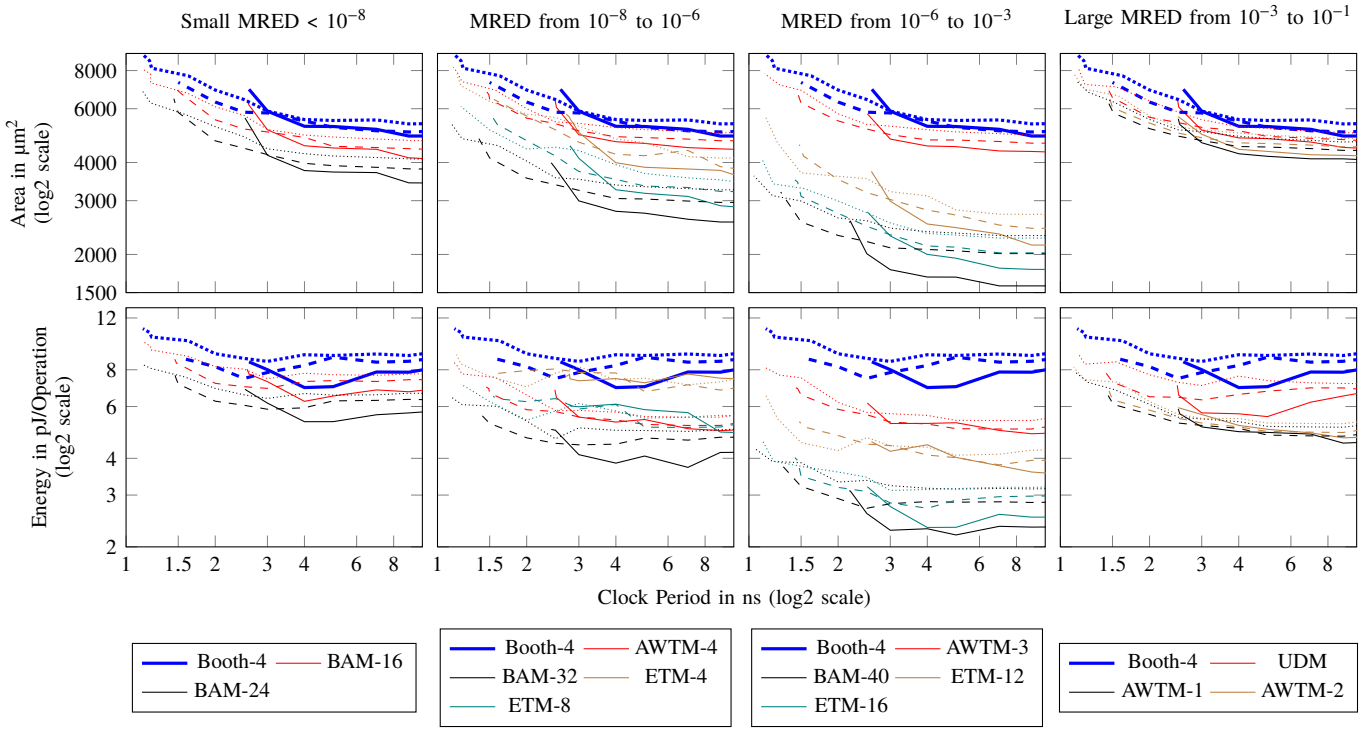

Fig. 4. Total area (top row) and energy consumption (bottom row) of 32 bit approximate multipliers as a function of the target clock period constraint. Solid lines denote designs with zero internal pipeline stages (input and output registration only), whereas the dashed lines denote designs with one additional internal pipeline stage and dotted lines denote designs with two additional internal pipeline stages. The precise Radix-4 Booth reference multiplier is emphasized by thick lines within each plot.

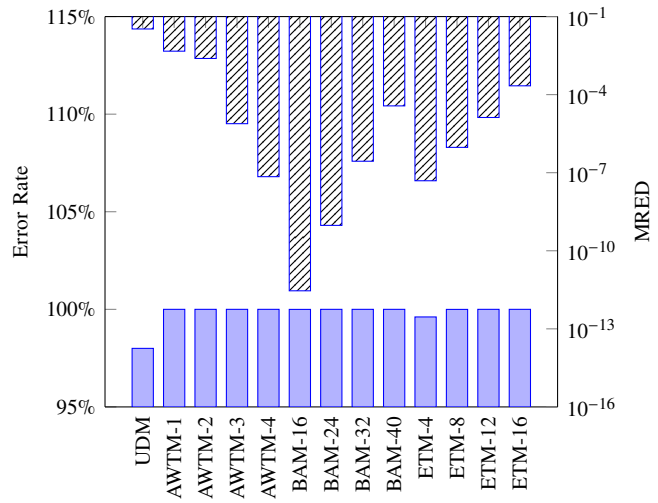

Fig. 5. Error rate (filled bars) and MRED (pattern bars) of 32 bit approximate multipliers. The value of a multiplier-specific configuration parameter is indicated as a suffix to the multiplier name.

- When applying one internal pipeline stage, high performance with clock periods of 0.6 ns is possible with the more precise LOA-12 and LOA-16 adders. Area and power of the pipelined LOA-12 are increased by 12% compared to the unpipelined ESA-4 at 0.6 ns, but with a MRED smaller by five orders of magnitude. Moreover, the pipelined LOA-12 has 30% less silicon area and energy consumption than the pipelined precise adder at 0.6 ns, including all pipeline register stages.
- Between 0.7 and 0.9 ns, the area of the precise adder, ETA2 and ACA shows an intersection between the pipelined and unpipelined designs. Therefore, a clock period margin exists where the insertion of a pipeline stage can also lead to a more area-efficient adder when not optimizing for maximum performance.

B. Analysis of Pipelined Approximate Multipliers

Fig. 4 depicts the silicon area and energy per operation of the UDM, the AWTM in different accuracy modes, the BAM with parameters $VBL = \{16, 24, 32, 40\}, HBL = 0$ and the ETM with $K = \{4, 8, 12, 16\}$ as functions of the target clock period constraint (1 to 10 ns), the amount of pipelining (zero to two internal register stages) and the desired MRED. ER and MRED are illustrated in Fig. 5. The following conclusions can be drawn:

- For all evaluated multipliers, the partial product truncation of the BAM is the most area- and energy-efficient approximation method. For the BAM-40, area and power are reduced by 70% in the clock period range of 3 to 10 ns without pipelining compared to the precise Radix-4 Booth multiplier.
- Without internal pipeline stages, the maximum performance increase of 16% compared to the precise multiplier

is achieved with the BAM-40 because of critical path truncation. However, this is accompanied by a MRED of $3 \cdot 10^{-5}$. Lower clock periods than 2.2 ns at smaller approximation errors require one or two internal pipeline stages at the expense of increased circuit area and energy consumption.

- Between 2.5 and 4 ns, area and energy show intersections between the same multiplier design with zero and one internal pipeline stage. When a second internal pipeline stage is added, BAM and ETM multipliers have further intersections between 1.5 and 2.5 ns. For example, the area of the BAM-24 with one internal pipeline stage is reduced by 20% and the energy requirement is reduced by 11% compared to the unpipelined multiplier at a clock period constraint of 2.5 ns. The ETM-12 with two internal pipeline stages achieves an area reduction of 12% and energy reduction of 10% when compared to one internal pipeline stage at 1.5 ns. For these units, early pipelining leads to more area- and energy-efficient multipliers when not optimizing for maximum performance.

V. Conclusion and Future Work

In this work, trade-offs between circuit area, energy requirements and accuracy of approximate adder and multiplier designs are explored for pipelined processor datapaths. A two-phase architecture mapping and register balancing synthesis with adaptive timing constraints is used to obtain area-efficient gate-level implementations for a parameterizable amount of pipeline stages. The trade-off analysis shows that besides to more performance, inherent datapath pipelining can also be used to implement more area- and energy-efficient approximate arithmetic units when the maximum performance is not required, obtaining area reductions of up to 20% and energy reductions of up to 11% for the same target clock period constraint. Furthermore, when higher performance is desired for a specific architecture, pipelining can be used instead of choosing another approximate unit with lower accuracy.

The applied register balancing to implement pipelined designs does not modify the functionality of the arithmetic architecture, so there is no accuracy variation with the number of pipeline stages. In general, state-of-the-art approximate adder and multiplier units are designed for single-cycle execution, however, pipelined datapaths can be further used to structurally exploit multi-cycle designs. By including the concept of pipelining directly in the architecture description instead of using register balancing, more area- and energy-efficient adder and multiplier implementations at given performance and accuracy requirements could be developed in future work.

Acknowledgment

This work was partly funded by the German Research Council (DFG) under project number PA 2762/1-1.

References

[1] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and Characterization of Inherent Application Resilience for Approximate Computing," in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC '13, 2013, pp. 113:1–113:9.

[2] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm for Energy-Efficient Design," in *2013 18th IEEE European Test Symposium (ETS)*, 2013, pp. 1–6.

[3] J. Sankaran, C.-Y. Hung, and B. Kisačanin, "EVE: A Flexible SIMD Coprocessor for Embedded Vision Applications," *Journal of Signal Processing Systems*, vol. 75, no. 2, pp. 95–107, 2014.

[4] S. Nolting, F. Giesemann, J. Hartig, A. Schmider, and G. Paya-Vaya, "Application-Specific Soft-Core Vector Processor for Advanced Driver Assistance Systems," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–2.

[5] A. K. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '08, 2008, pp. 1250–1255.

[6] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of Voltage-Scalable Meta-Functions for Approximate Computing," in *2011 Design, Automation Test in Europe*, 2011, pp. 1–6.

[7] N. Zhu, W. L. Goh, and K. S. Yeo, "An Enhanced Low-Power High-Speed Adder for Error-Tolerant Application," in *Proceedings of the 2009 12th International Symposium on Integrated Circuits*, 2009, pp. 69–72.

[8] 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 Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2010.

[9] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," in *2011 24th Internatioal Conference on VLSI Design*, 2011, pp. 346–351.

[10] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-Power High-Speed Multiplier for Error-Tolerant Application," in *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, 2010, pp. 1–4.

[11] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and Area-Efficient Approximate Wallace Tree Multiplier for Error-Resilient Systems," in *Fifteenth International Symposium on Quality Electronic Design*, 2014, pp. 263–269.

[12] H. Jiang, J. Han, and F. Lombardi, "A Comparative Review and Evaluation of Approximate Adders," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '15, 2015, pp. 343–348.

[13] J. Huang and J. Lach, "Exploring the Fidelity-Efficiency Design Space Using Imprecise Arithmetic," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '11, 2011, pp. 579–584.

[14] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A Comparative Evaluation of Approximate Multipliers," in *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, 2016, pp. 191–196.

[15] J. Liang, J. Han, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.

[16] "Synopsys Design Compiler, PrimeTime and DesignWare IP Website," https://www.synopsys.com/, accessed: 2018-03-29.

[17] W. J. Dally and T. M. Aamodt, *Digital Design Using VHDL*. Cambridge University Press, 2015.

[18] "Mentor Graphics ModelSim Website," https://www.mentor.com/, accessed: 2018-03-29.