On the Approximation of Accuracy-configurable Sequential Multipliers via Segmented Carry Chains

Jorge Echavarria, Stefan Wildermann, Oliver Keszocze, Faramarz Khosravi, Andreas Becher, Jürgen Teich
Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany
Email:{jorge.echavarria, stefan.wildermann, oliver.keszoecze, faramarz.khosravi, andreas.becher, juergen.teich}@fau.de

Abstract—In this paper, we present a multiplier based on a sequence of approximated accumulations. According to a given splitting point of the carry chains, the technique herein introduced allows varying the quality of the accumulations and, consequently, the overall product. Our approximate multiplier trades-off accuracy for a reduced latency—with respect to an accurate sequential multiplier—and exploits the inherent area savings of sequential over combinatorial approaches. We implemented multiple versions with different bit-width and accuracy configurations, targeting an FPGA and a 45nm ASIC to estimate resources, power consumption, and latency. We also present two error analyses of the proposed design based on closed-form analysis and simulations.

I. INTRODUCTION

The recent growing interest on APPROXIMATE COMPUTING (AC) is partially due to insufficient resources for handling the ever-increasing amount of data created every single day. One of the most prominent domains being multimedia applications. In digital image processing, for example, AC trades-off imperceptibility quality degradation to the human eye with non-functional objectives. That is, rather than computing fully accurate results, AC allows an algorithm to operate to some degree inaccurately in order to satisfy user-defined goals like reduction of latency, decrease of silicon area usage, or reduction of power dissipation.

From this perspective, and in hand with the fact that arithmetic operations are the backbone of multimedia processing, we propose in this paper a sequential approximate multiplier which operates with reduced carry chains. This allows to shorten the critical path within the combinatorial part and increase the operational frequency. As discussed in the following, we are aware of only little-to-no work in the literature tackling approximate multiplication following such a sequential approach.

II. RELATED WORK

Multiple works have proposed the use of approximate building blocks to create larger combinatorial multipliers either by, for example, using approximated compressors or reduced circuitry by manipulating their truth tables [1], [2]. Some other works have exploited the capability of Booth encoders for reducing the number of partial products in approximate signed multiplication [3]. Such techniques focus on underdesigining combinatorial approaches aiming to reduce area, and therefore, power consumption.

Little effort has been made in the area of sequential approximate multiplication, overlooking the inherent area savings of such approaches, and possible timing reductions. Certainly, the most relevant work experimenting with sequential approaches is the one presented by Chandrasekharan et al. in [4]. The authors propose an architecture based on multiple 8-bit sequential multipliers made of adders taken from a collection of approximate arithmetic units found in the literature. These approximate adders were implemented with different configurations, and thus different degrees of inaccuracy. The authors of [4] did not present any formal analysis of the errors introduced by such architectures. They did, however, obtain and report what they call the worst-case error and the maximum bit-flip by using a miter. Unfortunately, this error estimation approach requires simulation of many input patterns which, in order to obtain reliable results, demands large input sets.

As seen, little consideration has been given to sequential multipliers and to formal techniques for the analysis of errors of such architectures.

III. PRELIMINARIES

The so-called grade-school multiplication algorithm multiplies each bit of the multiplicand by the multiplier. Representing a multiplication by the radix, each partial product is shifted to the left by the amount equal to the power of the corresponding bit of the multiplicand. Consider the example shown in Table Ia. Line 5 corresponds to the sum of the multiplier multiplied by the 0th and the 1st bits—shifted 0 and 1 positions, respectively—of the multiplicand. Due to the addend in the partial addition being shifted to the left, the least significant bit (LSB) of the augend—shown in black—does not take part in the addition. This extra bit is simply concatenated as the least significant bit (LSB) of the product. Table Ib shows the product of the example shown in Table Ia. Line 15 contains the product, with the last added bit being blue and in bold.

| Multiplier | 1 0 1 1 | Multiplier | 1 0 1 1 |
|------------|--------|------------|--------|
| Multiplicand | 1 1 0 1 | Multiplicand | 1 1 0 1 |
| 1 0 1 1 | 3 |
| 1 0 1 1 | 6 |
| 0 0 0 0 | 4 |
| 0 0 1 0 1 1 | 5 |
| 1 0 1 1 | 7 |
| 1 0 0 0 0 1 | 8 |
| 0 0 1 0 1 1 | 9 |
| 1 0 0 1 1 1 1 | 10 |
| 1 0 0 0 1 1 1 1 | 11 |
| 1 0 0 0 1 1 1 1 | 12 |
| 1 0 0 0 1 1 1 | 13 |
| 1 0 0 0 1 1 1 | 14 |
| 1 0 0 0 1 1 1 | 15 |

Table I: Accurate (a) combinatorial and (b) sequential multiplication.

Due to this approach, the combinational part of the multiplication is partially due to insufficient resources for handling the ever-increasing amount of data created every single day. One of the most prominent domains being multimedia applications. In digital image processing, for example, AC trades-off imperceptibility quality degradation to the human eye with non-functional objectives. That is, rather than computing fully accurate results, AC allows an algorithm to operate to some degree inaccurately in order to satisfy user-defined goals like reduction of latency, decrease of silicon area usage, or reduction of power dissipation.
up by a single \((n + [2^k - 2 + 0.5]) \times n\)-bit adder, in total \(k = \log_2 n\) additions are required. Such a design would need \(\sum_{i=1}^{\log_2 n} n^i = n - 1\) adders, scaling linearly with the input bit-width \(n\).

On the other hand, a sequential multiplication such as shown in Table Ib, and a circuit shown in Figure 1a, requires fewer resources. Some power savings are expected in comparison with a combinatorial approach because there is less interconnect, which in turn means that the capacitive load on the signals in the design decreases. With a lower load, dynamic power consumption can be reduced. As seen in the figure, only a single \(n \times n\)-bit adder, one D flip-flop, and two \(n\)-bit shift registers are needed. Consider the example shown in Table Ib, where Line 6 (resp. Line 7) corresponds to the augend (resp. addend) shown in Line 3 (resp. Line 4) of Table Ia. The result, however, is shifted once to the right—clearly, shifting the augend to the right is equivalent to shifting the addend to the left. By doing so, the next partial product may be added directly to the result, as shown in Line 10. The bits emphasized in blue represent those bits where the actual addition takes place. Observe that, as opposed to the \(n - 1\) additions performed by a combinatorial approach, a sequential architecture performs only a single addition per clock cycle.

A. Boolean representation

Formally, let \(p : \mathbb{B}^{2n} \to \mathbb{B}^{2n}\) represent the accurate product of two \(n\)-bit binary numbers \(a\) and \(b\) with \(a, b \in \mathbb{B}^n\) and \(p(a_n-1, \ldots, a_0, b_n-1, \ldots, b_0) = (p_2n-1, \ldots, p_0)\). More specifically, \(p(a, b) = \text{dec}(a) \cdot \text{dec}(b)\), where \(\text{dec}(x)\) corresponds to the decimal representation of a binary number \(x\). Let \(a_i\) be the \(i\)-th bit of multiplier \(a\), \(b_j\) be the \(j\)-th bit of multiplicand \(b\), \(S^i_j\) be the \(i\)-th bit in the \(j\)-th accumulated sum, and \(C^i_j\) the \(i\)-th carry-bit in the \(j\)-th carry chain. The values of \(S^i_j\) and \(C^i_j\) are then determined by:

\[
S^i_j = \begin{cases} 
0, & \text{if } j = 0 \text{ and } i \in \{0, n\} \\
S^i_{j-1} \land (a_0 \land b_j), & \text{if } j = 0 \text{ and } i = n \\
S^i_{j+1} \lor S^i_{j-1} \land (a_i \land b_j), & \text{if } j > 0 \text{ and } i = 0 \\
S^i_{j+1} \lor C^i_{j-1} \land (a_i \land b_j), & \text{if } j > 0 \text{ and } i \in \{0, n\} \\
C^i_{j-1}, & \text{if } j > 0 \text{ and } i = n,
\end{cases}
\]

\[
C^i_j = \begin{cases} 
0, & \text{if } j = 0 \\
S^i_{j-1} \land (a_0 \land b_j), & \text{if } j > 0 \text{ and } i = 0 \\
S^i_{j+1} \lor S^i_{j-1} \land (a_i \land b_j), & \text{if } j > 0 \text{ and } i \in \{0, n\}
\end{cases}
\]

where \(\land, \lor, \text{ and } \oplus\) represent the logical conjuction, disjunction, and exclusive disjunction, respectively. Consequently, the product \(p\) is constructed as follows:

\[
p_r = \begin{cases} 
S^r_{0}, & \text{if } r \in [0, n-1] \\
S^{n-1}_{r-n+1}, & \text{if } r \in [n-1, 2n-1],
\end{cases}
\]

At clock cycle \(j = 0\), the multiplicand \(b\) is stored in the shift register \(B\), and all bits in shift register \(A\) are set to zero. During each shift to the right the LSB of register \(B\), i.e., \(b_{j-1}\), is disposed to \(B_{t-1}\), the LSB of register \(A\), i.e., \(A'_0\), is introduced from the left to register \(A\), and the D flip-flop storing the carry-out \(C_{t-1}\) from the adder in the previous addition, i.e., \(C'_{n-1}\), is introduced from the left to register \(A\), see the right angled arrows in Tables Ib and Ia. The inputs of the adder are 1) the result of \(\text{ANDing bit } B_{t-1}\) with multiplier \(a_0\), and 2) the previous addition—located in register \(A\), right-shifted once. At clock cycle \(j = n\), shift register \(A\) contains the \(n\) most significant bits (MSBs) and shift register \(B\) contains the \(n\) LSBs of the \(2n\)-bit product \(p\), see Equation (1). Before introducing a new approximate sequential multiplier, we introduce established error metrics for approximate computations.

B. Error metrics

The following paragraphs summarize error metrics used throughout this paper. A formal definition is provided for each metric considering an accurate product \(p\) and an approximate product \(\hat{p}\). These definitions will later be used to derive the formulation of metrics for the error evaluation of the proposed accuracy-configurable sequential multiplier via segmented carry chains.

When neither a well-accepted error model nor additional knowledge on the design are available, the likelihood of computing an erroneous result is a metric typically used:
the arithmetic error across arithmetic units with different bit-widths:  

impact the target application greatly:  

error metric  

Multiplier  

| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

Arithmetic circuits are generally evaluated based on an arithmetic error metric, since the magnitude of its lexicographic deviation can impact the target application greatly:

- **Arithmetic error rate (ER)** is the likelihood of producing an erroneous result in at least a single bit:

\[ \text{ER}(p, \hat{p}) = \frac{1}{|\mathbb{B}^n|} \cdot \sum_{a,b \in \mathbb{B}^n} p^i(a,b) \oplus \hat{p}^i(a,b). \]  (2)

Arithmetic circuits are generally evaluated based on an arithmetic error metric, since the magnitude of its lexicographic deviation can impact the target application greatly:

- **Arithmetic error rate (ER)** is the likelihood of producing an erroneous result in at least a single bit:

\[ \text{ER}(p, \hat{p}) = \frac{1}{|\mathbb{B}^n|} \cdot \sum_{a,b \in \mathbb{B}^n} \text{dec}(p(a,b)) - \text{dec}(\hat{p}(a,b)). \]  (3)

- **Error distance (ED)** is the arithmetic deviation of the approximate output from the accurate result:

\[ \text{ED}(p(a,b), \hat{p}(a,b)) = \text{dec}(p(a,b)) - \text{dec}(\hat{p}(a,b)). \]  (4)

- **Maximum absolute error (MAE)** is the largest absolute ED:

\[ \text{MAE}(p, \hat{p}) = \max_{a,b \in \mathbb{B}^n} \{ |\text{ED}(p(a,b), \hat{p}(a,b))| \}. \]  (5)

- **Mean error distance (MED)** is the average ED under a given input distribution:

\[ \text{MED}(p, \hat{p}) = \frac{1}{|\mathbb{B}^n|} \cdot \sum_{a,b \in \mathbb{B}^n} \text{ED}(p(a,b), \hat{p}(a,b)). \]  (6)

Finally, normalized and relative error metrics allow us to evaluate the arithmetic error across arithmetic units with different bit-widths:

- **Normalized mean error distance (NMED)** is the normalized MED by the maximum accurate output (cf. [3]):

\[ \text{NMED}(p, \hat{p}) = \frac{\text{MED}(p, \hat{p})}{\max_{a,b \in \mathbb{B}^n} \{ 1, \text{dec}(p(a,b)) \}}. \]  (7)

- **Mean relative error distance (MRED)** is the averaged absolute relative ED over the accurate result (cf. [3]) under a given input distribution:

\[ \text{MRED}(p, \hat{p}) = \frac{1}{|\mathbb{B}^n|} \cdot \sum_{a,b \in \mathbb{B}^n} \frac{\text{ED}(p(a,b), \hat{p}(a,b))}{\max_{a,b \in \mathbb{B}^n} \{ 1, \text{dec}(p(a,b)) \}}. \]  (8)

### IV. APPROXIMATE SEQUENTIAL MULTIPLIER

In accordance with the advantages of a sequential over a combinatorial design, we propose a circuit applying the concepts of AC into the former approach. As pointed out in [4], the authors of Almost Correct Adder (ACA) showed that a carry propagates only a small fraction of the carry chain in average. Based on this observation, approximate addition via segmented carry chains has become a well-investigated methodology [5], resulting in multiple works on its error analysis: most notably works proposing closed form error analysis [6], and algorithmic approaches [7]. Consequently, the adder within our approximate sequential multiplier is based on carry chain segmentation.

**A. Proposed design**

Figure 1b shows the schematic of the circuit implementing the approximate algorithm proposed in this paper. As seen, it is made of two fully accurate adders. Segmenting the carry chain of the partial product accumulation with a least significant part (LSP) t-bit adder and a most significant part (MSP) (n-t)-bit adder makes it possible to reduce the latency to max{lat(MSP), lat(LSP)}. Note that such arithmetic units are not constrained to any approach of addition.

Observe in Figure 1b the carry-out of the LSP adder connected to the D flip-flop driving the carry-in of the MSP adder. This causes a delay of the carry propagation by one clock cycle. This situation is described with the example shown in Table III. The bits marked by the orange rectangle lay on the right side of the splitting point. In other words, they represent the MSBs of the LSP adder. Examining the resultant accumulation in Line 11, one can observe that a carry generates at this location does not immediately propagate to the MSP adder. Instead, it becomes the carry-in in the next clock cycle at the same location of the bits marked by the blue rectangle. When such a carry is 1 during the last clock cycle, i.e., $C_{t-1}^n=1$, a zero detect
signal enables the multiplexers to set all \( n + t \) LSBs to 1, representing the decimal value \( 2^{n+t} - 1 \), that is, the closest decimal value to the disregarded overflow, thus, considerably reducing the magnitude of the absolute error distance \( |ED| \). Such a fix-to-1 instrumentation reduces the MED when considering absolute EDs. However, for error compensation in, for example, cascaded approximate multipliers, it may be disabled to allow for negative EDs, and hence, reduce the global MED.

Our approximate multiplier is formally defined as follows:

\[
\hat{S}^j_l = \begin{cases} 
    a_i \land b_i, & \text{if } j = 0 \text{ and } i \in [0, n) \\
    0, & \text{if } j = 0 \text{ and } i = n \\
    \hat{S}^j_{l+1} \oplus (a_i \land b_i), & \text{if } j > 0 \text{ and } i = 0 \\
    \hat{S}^j_{l+1} \oplus b_i \land \hat{C}^j_{l-1}, & \text{if } j > 0 \text{ and } i = t \\
    \hat{S}^j_{l+1} \oplus \hat{C}^j_{l-1} \oplus (a_i \land b_i), & \text{if } j > 0 \text{ and } i \in (0, t) \cup (t, n) \\
    \hat{C}^j_{n-1}, & \text{if } j > 0 \text{ and } i = n, 
\end{cases}
\]

Consequently, the approximate product \( \hat{p} \) is constructed as follows:

\[
\hat{p}_r = \begin{cases} 
    \hat{S}^0_r, & \text{if } r \in [0, n-1) \text{ and } \hat{C}^{n-1}_{t-1} = 0 \\
    1, & \text{if } r \in [0, n-1) \text{ and } \hat{C}^{n-1}_{t-1} = 1 \\
    \hat{S}^{n-1}_{r+1}, & \text{if } r \in [n-1, t+n) \text{ and } \hat{C}^{n-1}_{t-1} = 0 \\
    1, & \text{if } r \in [n-1, t+n) \text{ and } \hat{C}^{n-1}_{t-1} = 1 \\
    \hat{S}^{n-1}_{t+n+1}, & \text{if } r \in [t+n, 2n-1). 
\end{cases}
\]

B. Error analysis

The likelihood of an error being observable at the MSP of the approximate product \( \hat{p} \), that is, at any output bit \( \hat{p}_{2n-1}, \ldots, \hat{p}_n \equiv \hat{S}^{n-1}_{n-1}, \ldots, \hat{S}^{n-1}_{1}, 1 \), can be computed as follows.

Let \( \hat{S}^j \) be the \( j \)-th accumulated sum, then:

\[
\text{ER}(\hat{S}^j, \hat{S}^t) = \rho \left( \bigcup_{i=0}^{t-2} \left( \hat{S}^j_{i+1} \land (a_i \land b_i) \right) \right) \bigcup \left( \bigcup_{i=t+1}^{t-1} \left( \hat{S}^j_{i+1} \land (a_i \land b_i) \right) \right),
\]

when \( j > 0 \), where \( \rho(\chi) \) represents the probability of an event \( \chi \) to be true. The remaining accumulation \( j = 0 \) does not introduce errors, \( \text{ER}(\hat{S}^0, \hat{S}^0) = 0 \). The term on the right side of the disjunction in Equation (9) represents the event of a carry being generated anywhere before the MSB and being propagated until and by the MSB of the LSP of the approximate accumulated sum \( \hat{S}^j \), and the term on the left side of the disjunction represents the event of a carry being generated directly at the MSB of the LSP.

The remaining product bits \( \hat{p}_{n-1}, \ldots, \hat{p}_0 \equiv \hat{S}^{n-1}_{0}, \ldots, \hat{S}^{n-1}_{0} \) shall then be computed independently. Note that the ER of the LSP of \( \hat{p} \) and the BER of each remaining output bit consider mutually exclusive events. The general disjunction rule shall be applied to avoid computing manifold the occurrence of multiple events. According to this rule for \( n - t \) number of events, it can be shown that:

\[
\text{ER}(p, \hat{p}) = \sum_{k=1}^{n-t} \left( -1 \right)^{k-1} \cdot \sum_{i \in C_{k,n-t}} \rho \left( \text{ER} \left( S^{n-1}, \bar{S}^{n-1} \right) \right)_{t_k} \neq 0
\]

\[
\cap \text{BER} \left( S_0^{n-t-2}, \bar{S}_0^{n-t-2} \right) \neq 0 \cap \ldots
\]

\[
\text{BER} \left( S_0^{n-1}, \bar{S}_0^{n-1} \right) \neq 0,
\]

for \( n > 4 \) and \( t \leq \frac{n}{2} \), where \( C_{k,n-t} \) represents the set of all ordered \( k \)-tuples \( l_i < \ldots < l_k \) of \( \{1, \ldots, n-t\} \). When a carry out is generated within the LSP adder at the last clock cycle \( n \), it cannot be propagated to the MSP adder as the carry would only be available in the D flipflop in the next clock cycle. As a remedy, Shift Registers \( A_{\text{LSP}} \) and \( B \) are fixed to 1 right before the last shift-to-right operation which results in fixing the approximated product bits \( \hat{p}_{n+t-1}, \ldots, 1 \) to 1. The goal is to reduce the overall error.

In addition to the fact that every error introduced in the accumulations during clock cycles \( j < n-1 \) are shifted altogether to the right, the MAE only occurs when there is a carry propagated at bit position \( t-1 \) in the second last partial accumulation \( S^{n-2} \) and no carry at all at the same position in the last partial accumulation \( S^{n-1} \). Therefore, the probability of the approximated product \( \hat{p}(a,b) \) evaluating to an erroneous result with an ED as large as the MAE is:

\[
\rho \left( \text{ED} \left( p(a,b), \hat{p}(a,b) \right) = \text{MAE} \left( p, \hat{p} \right) \right) = \rho \left( \hat{C}^{n-2}_{t-1} \land \bar{C}^{n-1}_{t-1} \right).
\]

Moreover, as the MAE occurs when \( \hat{C}^{n-2}_{t-1} = 1 \) and \( \bar{C}^{n-1}_{t-1} = 0 \), an error is introduced at bit position \( t \) in the last partial accumulation, that is, \( S^{n-1} \), with a magnitude of \( 2^t \). Furthermore, during the calculation of \( S^{n-1} \), Shift Register \( B \) carries the \( n-1 \) LSBS of \( \hat{p} \), incrementing the magnitude error introduced in \( S^{n-1} \) by \( 2^{n-1} \). Finally, the \( t+1 \) LSBs are fully accurate whenever there is not a fix-to-1 operation, reducing the MAE by \( 2^{t+1} \):

\[
\text{MAE} \left( p, \hat{p} \right) = 2^{n-t-1} - 2^{t+1}.
\]

The MED considers all the erroneous results to calculate the average error:

\[
\text{MED} \left( p, \hat{p} \right) = \sum_{\delta \in \text{BER}(p, \hat{p})} \left( \sum_{a,b} \left( \Pr(a) \cdot \Pr(b) \right) \right),
\]

where \( \Pr \) represents the measured probability density function \( \text{PDF} \) of a binary number. Note that \( \delta \) iterates over all possible EDs, with the ED defined as follows:

\[
\text{ED} \left( p(a,b), \hat{p}(a,b) \right) = \sum_{i=0}^{2n-1} 2^i \cdot \text{dec} \left( \hat{p}(a,b) \oplus \hat{p}(a,b) \right) \cdot \text{sgn} \left( \text{dec} \left( \hat{p}(a,b) \right) - \text{dec} \left( \hat{p}(a,b) \right) \right),
\]

where \( \text{sgn}(x) = \frac{x}{|x|} \) when \( x \in \mathbb{Z}^0 \), 0 otherwise.

V. RESULTS

A. Error complexity

**Theorem 1.** Computing the BER is \#P-complete.

**Proof:** We will show this by reduction to and from the computation of ER, which is known to be \#P-complete [8].
The overall ER can be constructed by computing $2n$ different BER values by carefully watching out not to count errors multiple times. We make use of the identity $\text{BER}(p, \hat{p}) = \text{BER}(p \oplus \hat{p}, 0)$ to reformulate the BER computation. The ER can then be determined as:

$$\text{ER}(p, \hat{p}) = \sum_{i=1}^{m} \text{BER}\left((p_i \oplus \hat{p}_i) \land \bigwedge_{j=1}^{i-1} p_j \leftrightarrow \hat{p}_j, 0\right),$$

showing that $\text{ER} \leq \text{BER}$.

This concludes the proof that computing BER is $\#P$-complete.

\textbf{Theorem 2.} Computing the MED and MRED is $\#P$-complete.

\textit{Proof:} The proof for the average case error in [8, Theorem 6] can be applied with only the minor change of not taking the absolute value of the difference of $p$ and $\hat{p}$ to show that computing the MED is in $\#P$.

The authors of [8] further present a framework for easily determining the complexity of computing error metrics having a certain structure. The idea is to employ so-called rating functions $\hat{\phi}$ that describe the error metric to be used in generalized error metrics. The generalized mean error distance (GMED) is defined as:

$$\text{GMED}_{\phi, \zeta} := \zeta \cdot \sum_{x \in \mathcal{B}^n} \phi(f(x), \hat{f}(x)),$$

with a scaling factor $\zeta$.

If the function $\phi$ is in the class NC$^d$—roughly speaking: a circuit realizing the function can be created polynomially; see [9] for details—computing the generalized metric has the complexity of $\#P$.

Using the rating function:

$$\hat{\phi}(a, b) = \frac{\text{dec}(a) - \text{dec}(b)}{\max\{1, |\text{dec}(a)|\}},$$

we can write MRED as:

$$\text{MRED}(p, \hat{p}) := \text{GMED}(p, \hat{p}).$$

As $\phi$ is in NC$^d$, this proves that determining the MRED has the complexity of $\#P$.

\textbf{B. Error estimation}

We have empirically found that the general cases of $\hat{S}_i^j$ and $\hat{C}_i^j$, that is:

$$\hat{S}_i^j = \hat{S}_{i+1}^{j-1} \oplus \hat{C}_{i-1}^{j-1} \oplus (a_i \land b_j),$$

$$\hat{C}_i^j = \left(\left(\hat{S}_{i+1}^{j-1} \oplus (a_i \land b_j)\right) \land \hat{C}_{i-1}^{j-1}\right) \lor \left(\hat{S}_{i+1}^{j-1} \land a_i \land b_j\right),$$

have well-conditioned controllabilities regardless of their high fanout-reconvergence. Simply put, small changes in input variables $\hat{S}_{i+1}^{j-1}$, $\hat{C}_{i-1}^{j-1}$, $a_i$, and $b_j$ only slightly affect the probabilities $\rho(\hat{S}_i^j)$ and $\rho(\hat{C}_i^j)$, which are required for the calculation of the error metrics previously introduced. In an attempt to tackle the $\#P$-completeness of the ER, MED, NMED, and MRED metrics, we propose an approximation of the probabilities $\rho(\hat{S}_i^j)$ and $\rho(\hat{C}_i^j)$. Consider Equations (12) and (13) in their disjunctive normal forms:

$$\hat{S}_i^j = \left(\hat{S}_{i+1}^{j-1} \land \hat{C}_{i-1}^{j-1}\right) \lor \left(\hat{S}_{i+1}^{j-1} \land \hat{C}_{i-1}^{j-1}\right) \lor \left(\hat{S}_{i+1}^{j-1} \land \hat{C}_{i-1}^{j-1}\right) \lor \left(\hat{S}_{i+1}^{j-1} \land \hat{C}_{i-1}^{j-1}\right),$$

$$\hat{C}_i^j = \left(\hat{C}_{i-1}^{j-1} \land a_i \land b_j\right) \lor \left(\hat{S}_{i+1}^{j-1} \land a_i \land b_j\right) \lor \left(\hat{C}_{i-1}^{j-1} \land a_i \land b_j\right).$$

Observe that we may define:
\[ \hat{\rho} \left( S^i_{t+1} \right) = \hat{\rho} \left( S^i_{t+1 \mid (\pi_i, \nu)} \right) \cdot \hat{\rho} \left( \tilde{C}^i_{t+1 \mid (\pi_i, \nu)} \right) \cdot \hat{\rho} \left( \tilde{S}^i_{t+1 \mid (\pi_i, \nu)} \right) \]
\[ + \hat{\rho} \left( S^i_{t+1 \mid (\pi_i, \nu)} \right) \cdot \hat{\rho} \left( C^i_{t+1 \mid (\pi_i, \nu)} \right) \cdot \hat{\rho} \left( a_i \mid \nu \right) \cdot \hat{\rho} \left( b_j \right) \]
\[ + \hat{\rho} \left( S^i_{t+1 \mid (a_i, \nu)} \right) \cdot \hat{\rho} \left( C^i_{t+1 \mid (a_i, \nu)} \right) \cdot \hat{\rho} \left( a_i \mid \nu \right) \cdot \hat{\rho} \left( b_j \right) \]
\[ + \hat{\rho} \left( S^i_{t+1 \mid (\nu)} \right) \cdot \hat{\rho} \left( \tilde{C}^i_{t+1 \mid (\nu)} \right) \cdot \hat{\rho} \left( \tilde{S}^i_{t+1 \mid (\nu)} \right) \]

where \( F^i_{\nu} \) represents the cofactor of \( F \) w.r.t. every \( a_i \in \nu \). Note that cofactor \( C^i_{t+1 \mid \nu} = 1 \) may never occur and \( C^i_{t+1 \mid \nu} = 1 \) always holds true, thus making unnecessary considering correlations with \( b_j \). As the reader may observe, correlations between \( S^i_t \) and \( C^i_t \) are disregarded as we only consider cofactors w.r.t. \( a_i \), and not among themselves.

C. Error evaluation

Unfortunately, we may not evaluate the ER, MED, NMED, and MRED metrics using the formulae presented in Sections III-B and IV-B due to their \#P-completeness. Therefore, in the sake of fairness when comparing our empirical findings with those from the related work, we performed exhaustive simulations for approximate multipliers with bitwidths \( n \leq 16 \). We opted for MC simulations for larger designs using input sets with \( 2^{32} \) uniformly distributed input patterns. Figure 2 shows the findings of such evaluations. Note that the results shown in ?? were obtained using the closed-form formula from Equation (11). The abscissae of Figure 2 were scaled logarithmically, similarly for the ordinates of ???????. We compare our findings with those found in the literature [1]–[3], [10]–[12]. As can be seen, not many attempts have been made to evaluate the ER and MED of multipliers with bitwidths larger than 16. However, Liu et al. do report their findings after simulating their 32-bit approximate combinatorial multiplier for MAE, NMED, and MRED in [3]. On the one hand, one may notice that our designs do not dominate every design found in the literature. However, we may also observe that our approach is within the same range of accuracy as the state-of-the-art methodologies. On the other hand, our design takes advantage of the inherent resource savings of sequential over combinatorial multipliers, as we will discuss in the following.

D. Evaluation of Latency and Power Tradeoffs

Our design is platform-agnostic, however, due to the massive parallel processing and reconfiguration capabilities of FPGA technology delivering a great price-performance ratio for the algorithms commonly used in multimedia processing, we report results for ASIC as well as for FPGA implementations. In the latter, apart from the available dedicated digital signal processing (DSP) slices, one can place arithmetic processing elements based on look-up tables (LUTs) wherever they are most effective without structural restrictions.

When there are more multipliers in the design than available DSP slices, the critical path on the FPGA may traverse one of

\[ \text{Area} \left[ \mu\text{m}^2 \right] \]
\[ \text{Latency} \left[ \text{ns} \right] \]
\[ \text{Power} \left[ \text{mW} \right] \]

the multipliers implemented on the LUTs, thus making the regular FPGA’s fabric responsible of the overall latency. Therefore, part of our experimentation was performed in LUTs.

We show in Figure 3a the number of LUTs, latency, and power consumption estimated after implementing an Accurate Sequential Multiplier and our Approximate Sequential Multiplier on an FPGA. In average, we observed an area overhead of our proposed approximate sequential multiplier w.r.t. a fully accurate sequential multiplier, and consequently, a slight power dissipation overhead of only 3.6%.

Yet, when comparing the accurate and the approximate sequential multiplier designs, one may also observe in Figure 3a that even though our approximate design has a slight area overhead w.r.t. the accurate sequential multiplier for each value of \( n \), it always has a shorter latency by 19.15% in average and up to 29%—for \( n = 256 \). Interestingly, we observed—not shown in the figure—that fully combinational multipliers with bitwidths smaller than 8 consume fewer resources than sequential approaches. This is due to the area overhead of the sequential architectures for such small arithmetic
units. This, however, amortizes for higher bitwidths with up to 99%—i.e., the 256-bit architecture—of area savings and reduced dynamic power consumption of the sequential w.r.t. a combinatorial design.

Finally, we show in Figure 3b the latency, area, and power consumption after implementing a Sequential Accurate Multiplier and our Sequential Approximate Multiplier on a Nangate 45nm Open Cell Library from Silicon Integration Initiative (Si2). As can be seen, for the ASIC target, the slightly noticeable resource overhead for the 4- and 8-bit designs vanishes for architectures with greater bitwidths. Similarly to our findings after implementing our design on an FPGA, the latency is always reduced due to the shortened carry chains by 16.1% in average and up to 34.14%—for \( n = 8 \). Coincidentally, the power dissipation overhead for ASICs is fairly similar to the one observed in Figure 3a, only 3.6%. The area overhead, on the other hand, is under 3%.

VI. CONCLUSION

In this paper, we introduced an approximate sequential multiplier with segmented carry chain and variable accuracy. In order to better model and understand the accuracy degradation of our design we formally defined closed-form formulae for the most common error metrics. We further discussed the complexity of these error metrics presenting proofs for the \#P-completeness of the BER, MED, and MRED. Finally, in order to ease further comparisons, we implemented multiple ASIC as well as FPGA designs with different bitwidths and splitting points using Verilog and VHDL. By means of exhaustive simulations we found that the approach herein presented has an accuracy degradation well within the same ranges as those proposed in the literature. At the same time, estimations using well known synthesizers showed that our approximate-multiplication technique has significant latency improvements—of up to 29 and 34.14% when targeting FPGAs and ASICs, respectively—due to the shortened critical paths achieved by the segmentation of the carry chains with negligible power and area overheads.

REFERENCES

[1] W. Liu, T. Zhang, E. McLarnon, M. O’Neill, P. Montuschi, and F. Lombardi, “Design and analysis of majority logic based approximate adders and multipliers,” *IEEE Transactions on Emerging Topics in Computing*, 2019.

[2] N. Van Toan and J. Lee, “FPGA-based Multi-Level Approximate Multipliers for High-Performance Error-Resilient Applications,” *IEEE Access*, 2020.

[3] W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. S. Jr., and F. Lombardi, “Design and analysis of approximate redundant binary multipliers,” *IEEE Trans. Computers*, vol. 68, no. 6, pp. 804–819, 2019. [Online]. Available: https://doi.org/10.1109/TC.2018.2890222

[4] A. Chandrasekharan, D. Große, and R. Drechsler, *Design Automation Techniques for Approximation Circuits: Verification, Synthesis and Test*. Springer International Publishing, 01 2019.

[5] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, “A low latency generic accuracy configurable adder,” in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, 2015, pp. 86.1-86.6. [Online]. Available: https://doi.org/10.1145/2744769.2744778

[6] J. Echavarria, S. Wildermann, A. Becher, J. Teich, and D. Zienert, “FAU: Fast and error-optimized approximate adder units on LUT-Based FPGAs,” in *2016 International Conference on Field-Programmable Technology, FPT 2016, Xi’an, China, December 7-9, 2016*, 2016, pp. 213–216. [Online]. Available: https://doi.org/10.1109/FPT.2016.7929536

[7] J. Echavarria, S. Wildermann, E. Potwigin, and J. Teich, “Efficient arithmetic error rate calculus for visibility reduced approximate adders,” *Embedded Systems Letters*, vol. 10, no. 2, pp. 37–40, 2018. [Online]. Available: https://doi.org/10.1109/LES.2017.2760922

[8] O. Keszocze, M. Soeken, and R. Drechsler, “The complexity of error metrics,” *Information Processing Letters*, vol. 139, pp. 1–7, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019018301352

[9] H. Vollmer, *Introduction to Circuit Complexity: A Uniform Approach*, 2007th ed., ser. Texts in Theoretical Computer Science. Springer, 1999.

[10] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, “Design and evaluation of approximate logarithmetic multipliers for low power error-tolerant applications,” *IEEE Trans. on Circuits and Systems*, vol. 65-I, no. 9, pp. 2856–2868, 2018. [Online]. Available: https://doi.org/10.1109/TCSI.2018.2792902

[11] Y. Guo, H. Sun, and S. Kimura, “Small-Area and Low-Power FPGA-Based Multipliers using Approximate Elementary Modules,” in *25th Asia and South Pacific Design Automation Conference, ASP-DAC 2020, Beijing, China, January 13-16, 2020*, 2020.

[12] Z. Ebrahimi, S. Ullah, and A. Kumar, “LeAp: Leading-one Detection-based Softcore Approximate Multipliers with Tunable Accuracy,” in *25th Asia and South Pacific Design Automation Conference, ASP-DAC 2020, Beijing, China, January 13-16, 2020*, 2020.