P4 Approximate Circuits Analysis

System on Chip Lab Report

Approxies group

Institute of Computer Technology (ICT)
TU Wien
March 2022

Contents

1 Organisation

Our group consists of five members (see table 1). There is no hierarchy, except that Peter Traunmüller (e1326142@student.tuwien.ac.at) is the designated contact person.

Name	**Student ID**	**eMail** (+"@student.tuwien.ac.at")
Fabian Garber	01425023	e1425023
Simon Howind	01607387	e1607387
Kagan Özten	01028901	e1028901
Martin Resetarits	01425143	e1425143
Peter Traunmüller	01326142	e1326142

Table 1: Group members

2 Introduction

Approximate computing has shown to be a promising approach to realize improvements in efficiency and all parts of the design of circuits [VCRR15]. To be more energy-efficient, approximate computing tolerates some loss of quality in the computed result [HO13][VCRR15]. It has been called one of the most promising energy-efficient paradigms and has therefore reached a lot of research attention [XMK16]. A crucial point in analyzing the performance of an approximate circuit is the improvement in area and power consumption as well as identifying the maximum error compared to its exact counterpart.

In this project we will implement two versions of a Full-Adder (FA); one exact and one approximated. The approximated part is based on the Inexact Full Adder 1 (INXA1) as proposed in [PK19, p. 63]. We will analyze its error, performance and resource consumption.

3 Approximation Algorithm

In Figure 1 an exact FA is shown. The approximation we are using is located at the gate level. We use the INXA1 model from [PK19]. The design of this adder is based on an accurate sum signal and an approximated carry signal, as shown in Figure 2.

For further explanation the truth table is shown in Table 2. In the truth table, hooks show that the approximate FA is equal to the exact one and x's show deviations between the models.

As shown in Figure 2 INXA1 only approximates the carry signal, whereas the sum signal is calculated accurately. Nevertheless, the carry signal produces the right output in 6 out of 8 cases (see Table 2).

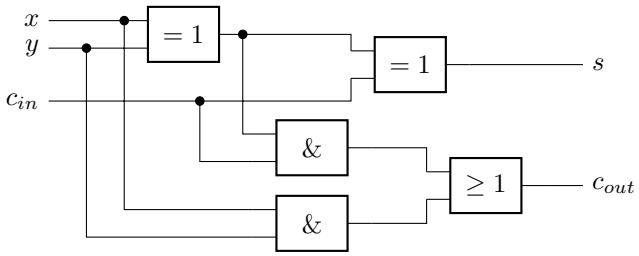

Figure 1: Circuit diagram of an accurate Full Adder.

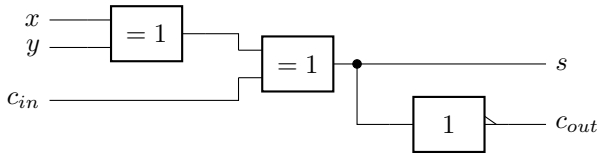

Figure 2: Circuit diagram of the INXA1 (based on [PK19]).

Input			**Full Adder**		**INXA1**	
x	y	c_{in}	c_{out}	s	c_{out}	s
0	0	0	0	0	0 ✓	0 ✓
0	0	1	0	1	1 ✗	1 ✓
0	1	0	0	1	0 ✓	1 ✓
0	1	1	1	0	1 ✓	0 ✓
1	0	0	0	1	0 ✓	1 ✓
1	0	1	1	0	1 ✓	0 ✓
1	1	0	1	0	0 ✗	0 ✓
1	1	1	1	1	1 ✓	1 ✓

Table 2: Comparison of accurate Full Adder and INXA1 (based on [PK19]).

2

4 Error Analysis

The correct Error Analysis is one of the most important characteristics of our approximate circuit. For multi bit inputs, the error is not the Hamming distance, therefore it's crucial to calculate the absolute error, to quantify the quality of the approximate circuit. In other words: the error must be implemented as a number.

As seen in Figure 4 we take the results of the INXA1 and the exact adder. Then we subtract and build the absolute to calculate our final error. For less complex circuits it is feasible to do a truth table analysis.

4.1 Binary Decision Diagram

As the system grows exponentially a truth table analysis is not recommended anymore. To solve that problem we are using a Binary Decision Diagram (BDD). A BDD is a graph that represents a Boolean function. This graphs enhances computer analysis.

To generate our BDD we use the python library PyEDA. PyEDA is a Python library for electronic design automation. In addition to PyEDA we use GVmagic. GVmagic is a package to use the external GraphViz library inside python. According to the documentation Graphviz is an open source graph visualization software, which represents structural information as diagrams of abstract graphs. Our steps are as follows:

1. Import necessary libraries
2. Build logic in python (e.g.: s_{out}, c_{out})
3. Generate a n-bit truth table
4. Convert truth table to BDD

In Table 3 we see the truth table of the BDD in Figure 3. Using the BDD we can see that after the inputs X[0] and X[1] are '0', we can instantly go to the final result '0'. Therefore we are already "saving" one row in the truth table.

Input			Output
X[0]	X[1]	X[2]	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 3: Truth table to the BDD

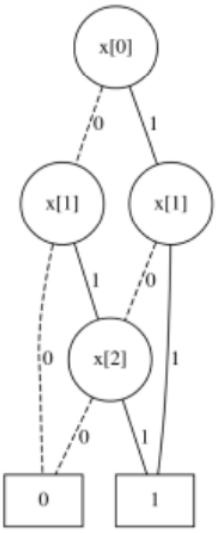

Figure 3: Reduced BDD corresponding to Table 3

To generate a BDD for a 2 bit adder we first have to generate outputs for exact and approximate adder. Then we can calculate the error by subtracting the exact result from the approximate result (see Figure 4).

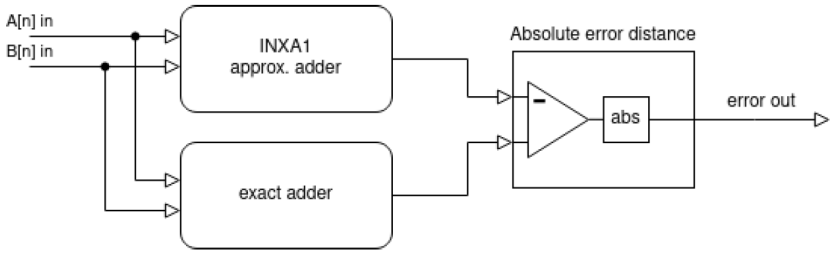

Figure 4: Calculation of the absolute Error

Furthermore we extract the truth tables and then generate a BDD for each bit of both the exact and approximate adder. A visualization is shown in Figure 5 and in Figure 6.

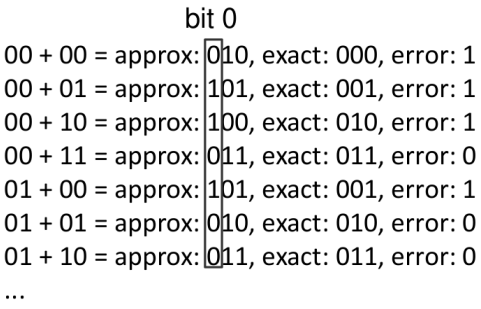

Figure 5: Calculation of bit 0 of the approximate adder

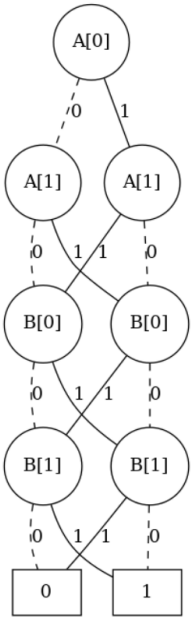

Figure 6: BDD of bit 0 of the approximate adder

To see the difference in complexity of the approximate adder we created the Figure 7 of the approximate adder and Figure 8 of the exact adder. This example is showing a BDD for the output bit '1' of an 8 bit full adder with 65536 input combinations. While the BDD of the exact adder has 49 nodes, the BDD of the approximate adder only has 31 nodes.

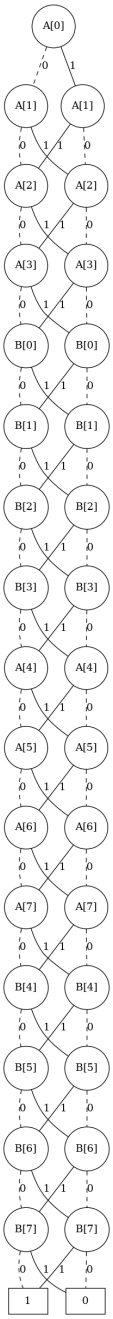

Figure 7: BDD of bit 1 of the approximate adder

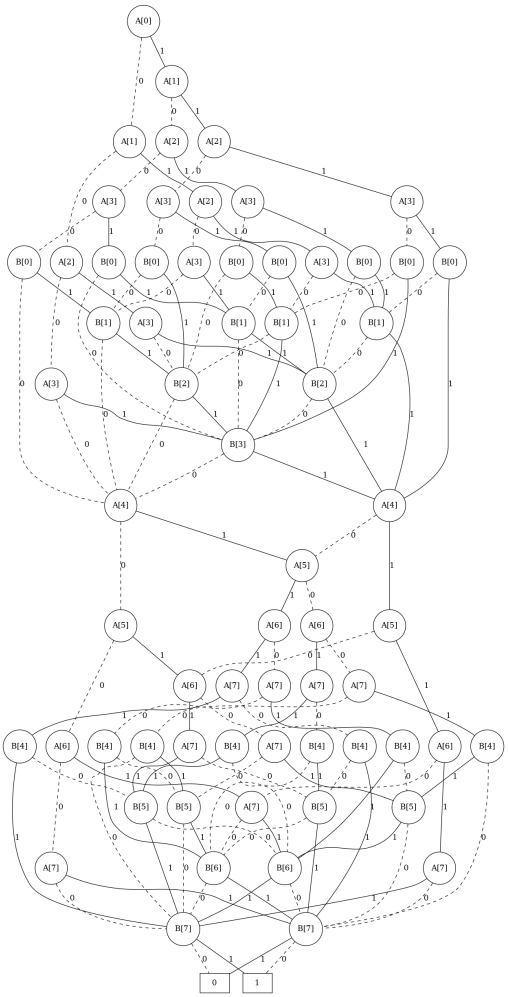

Figure 8: BDD of bit 1 of the exact adder

Another example is shown in Figure 9. Here we can see the Error BDD for

8 bit adders. This BDD has more than 100 nodes. So it's easy to see that those BDDs can get unwieldy quite quickly.

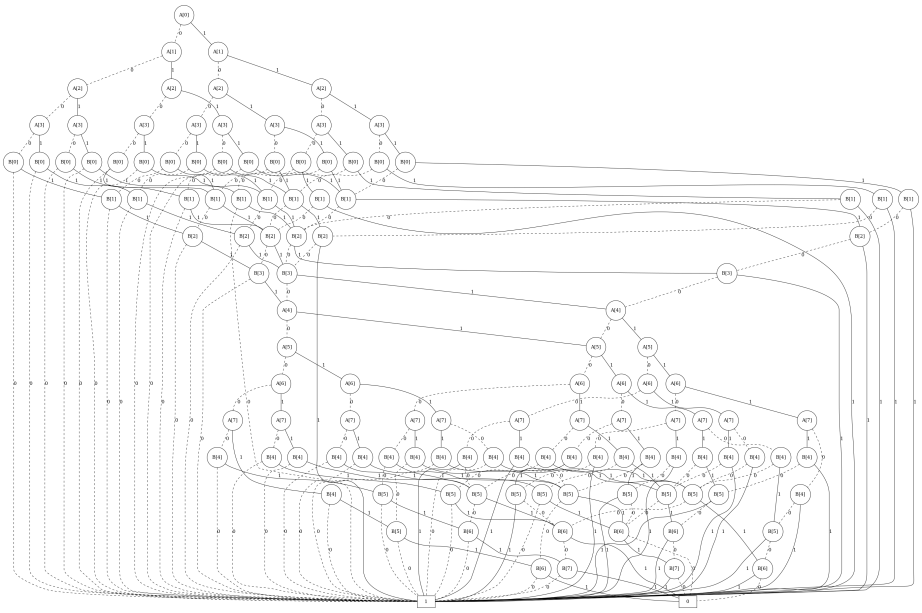

Figure 9: Error BDD for an 8-bit approximate adder

5 Results

5.1 Calculation Time

One of the most important aspect of the approximate circuit approach is to have a faster circuitry and thus to have less calculation time in a process compare to traditional exact circuit design.

In theory, having fewer transistor in a circuit directly affects the general switching activity in that circuit and therefore leading to a less delay time, which results in higher signal rate. Table 4 shows timing analysis of Vivado with virtual clock settings in different full adder designs of both approaches. The signal rate is automatically calculated and estimated by Vivado itself and the inputs of full adders A, B, C (Carry in) are given a fixed signal rate. By comparing the output signal rates of both designs, we can say with certainty that the approximate design is capable of producing much faster data rate at its output.

Type	A	B	C	S	C_{out}
4 Bit Exact	2,5	2,5	2,5	6,235	3,814
4 Bit Approx.	2,5	2,5	2,5	7,830	9,24
16 Bit Exact	2,5	2,5	2,5	6,460	3,847
16 Bit Approx.	2,5	2,5	2,5	9,39	9,999

Table 4: Timing Analysis – Signal Rate in Mega transactions (Mtr) per second

It is worth mentioning that by using default settings in Vivado there is huge difference in results regarding signal rates and power consumption in both approaches. Since our priority is to focus on analyzing power consumption in relation with signal rate of approximate design, we decided to create a virtual clock with 50 ns of period and appoint I/O of these designs to specific pins of the Field Programmable Gate Array (FPGA) board by generating a constraint file in Vivado, so that we can observe and examine more accurate analysis results. In the next subsection, we are going to cover power analysis of these signal rates with the same constraint settings.

5.2 Power Consumption and Area Analysis

One of the other advantages of using approximate design is to save up space on Printed Circuit Board (PCB) or chip area. One method to calculate the used area on a PCB is to simply count up the basic gates used in that design. The area used (measured in m^2) can vary depending on which nanometer technology we are using, so just by counting the basic gates in different designs we can roughly estimate more or less how much space we can spare compared to each other. Table 5 shows the number of basic gates used in multiple designs.

Type	XOR	OR	AND	NOT	Basic Gates in Total
1 Bit Exact	2	1	2	X	11
4 Bit Exact	8	4	8	X	44
8 Bit Exact	16	8	16	X	88
16 Bit Exact	32	16	32	X	176
1 Bit Approx.	2	X	X	1	9 (-3)
4 Bit Approx.	8	X	X	4	36 (-8)
8 Bit Approx.	16	X	X	8	72 (-16)
16 Bit Approx.	32	X	X	16	144 (-32)
					= ca. 18.2 % less

Table 5: Area Analysis

XOR gate is limited to 2-input basic gates (AND, OR and NOT), it requires 2 AND gates, 1 OR gate and 1 NOT gate, which basically means that an XOR

gate consists of 4 basic gates in total. After counting all the basic gates in a 16 bit approximate full adder (INXA1), we can see that compared to its exact full adder counterpart, we can save up to 18.2 % of the given area.

As we mentioned before, one of the important advantages of using approximate designs is to acquire more efficiency in power consumption. Here we have focused on examining 4 different full adder designs for more detailed power analysis, namely: 4 bit and 16 bit full adder designs which were synthesized in exact and approximate method. The analysis in Figure 10 is done again by Vivado with the same constraints and timing settings which were used in the Calculation Time subsection.

Overall power consumption (Dynamic and static combined) is estimated at 0.105 W for 4 bit both exact and approximate designs, for the 16 bit exact FA 0.107 W and lastly for 16 bit approximate INXA1 design 0.108 W.

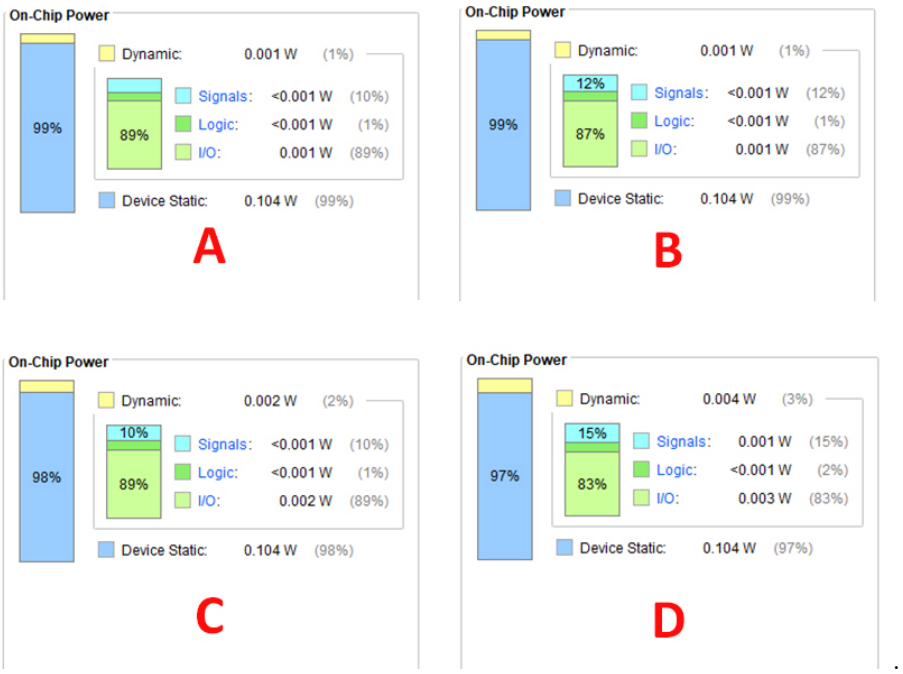

Figure 10: Power consumption of 4 bit exact (A), 4 bit appr. (B), 16 bit exact (C) and 16 bit appr. (D) FA

As we can see from the Figure 10, the power consumption of both methods are almost identical, although the approximate design is much faster in signal rate compare to exact design. If we were to keep the signal rate for both methods at the same level, it would result in lower power consumption for approximate design, thus makes it more efficient in that regard.

5.3 Error Discussion

In Figure 11 a visualization of the error of an 8-bit approximate adder is shown. The x-axis represents the input A and the y-axis the input B. The color shows the severity of the error. This means that more red equals a bigger absolute error and therefore the white area shows no error.

At first sight it seems that Figure 11 is point symmetric. This would mean that the sequence of summands does not matter, i.e. $a + b = b + a$. However, if you look closer, you can see that there are slight differences between the upper left corner and the bottom right corner, i.e. $a + b \neq b + a$. For the approximate adder, the commutative law is not valid anymore.

If $a = b$, the error of the approximate adder is 0, which can be seen in Figure 11, since there is a diagonal white line (no error) from the bottom left to the upper right corner.

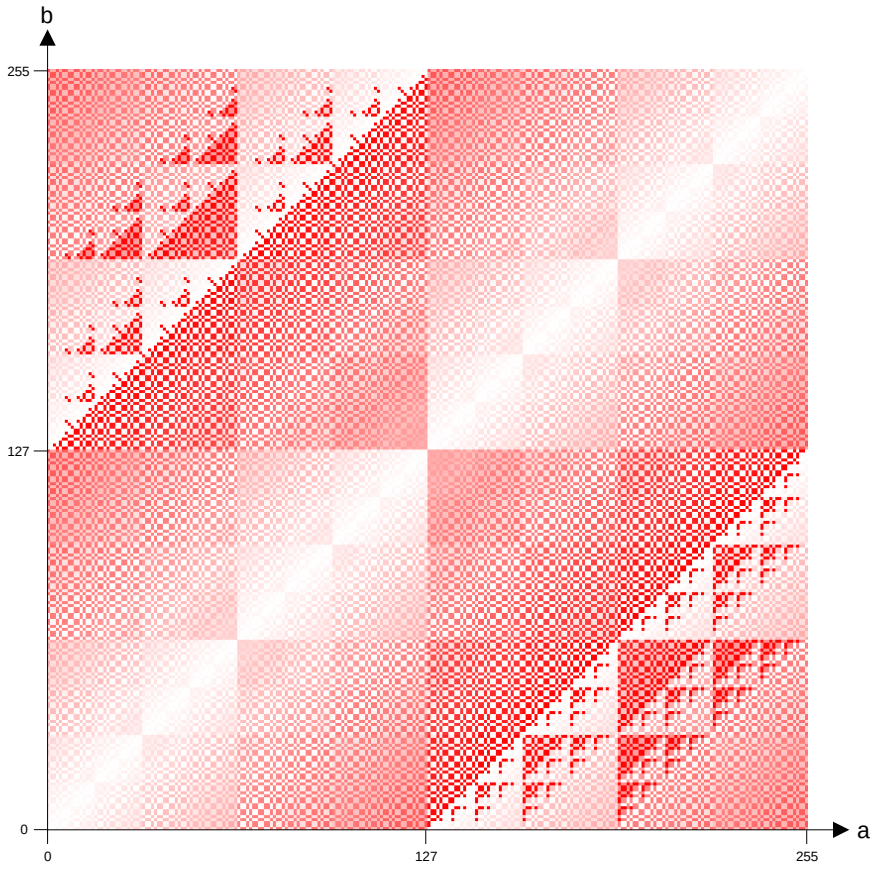

Figure 11: Visualization of the absolute error for an 8-bit approximate adder. The more red, the greater the error.

6 List of Acronyms

FA	Full-Adder
INXA1	Inexact Full Adder 1
BDD	Binary Decision Diagram
FPGA	Field Programmable Gate Array
PCB	Printed Circuit Board
Mtr	Mega transactions

References

[HO13] Jie Han and Michael Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6, 2013.

[PK19] M. Priyadharshni and S. Kumaravel. A comparative exploration about approximate full adders for error tolerant applications. In S. Rajaram, N.B. Balamurugan, D. Gracia Nirmala Rani, and Virendra Singh, editors, *VLSI Design and Test*, pages 61–74, Singapore, 2019. Springer Singapore.

[VCRR15] Swagath Venkataramani, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. Approximate computing and the quest for computing efficiency. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015.

[XMK16] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. Approximate computing: A survey. *IEEE Design Test*, 33(1):8–22, Feb 2016.