TECHNISCHE
UNIVERSITÄT
WIEN

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Institut für
Computertechnik
Institute of
Computer Technology

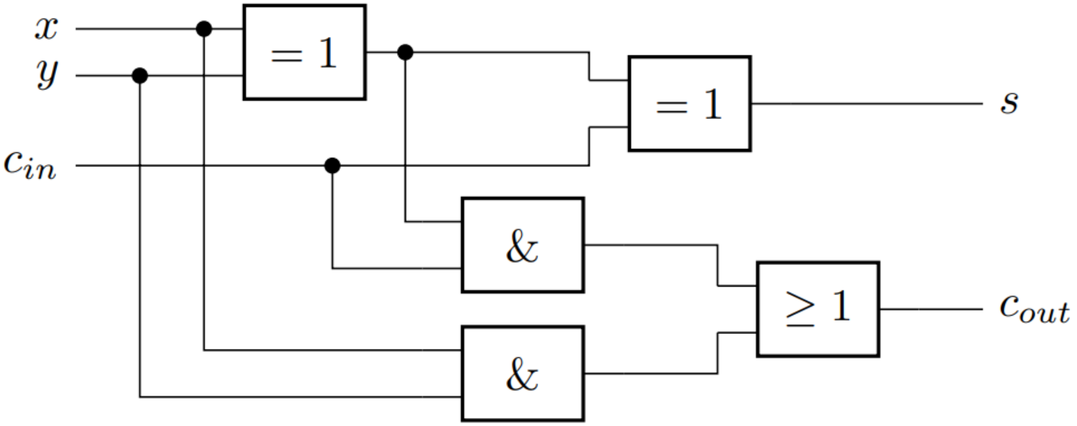
Midterm Presentation - P4 Accuracy of Approximate Circuits

Fabian Garber, Simon Howind, Kagan Özten,
Martin Resetarits, Peter Traunmüller

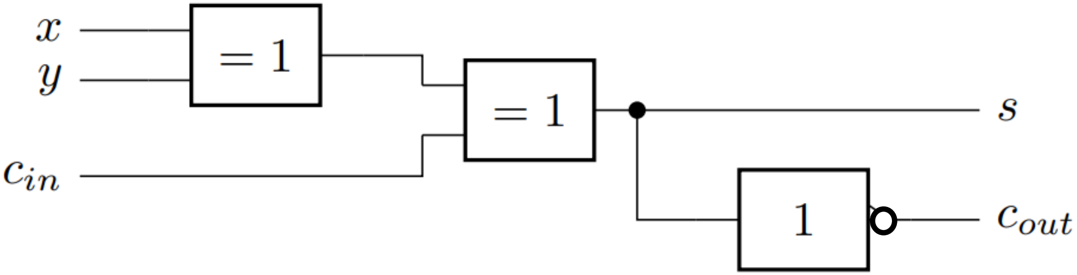
Introduction

- Short Recap
- Binary Decision Diagram Progress / Error Analysis
- VHDL Progress
- Power & Area Analysis
- Outlook

Recap: Circuit Designs

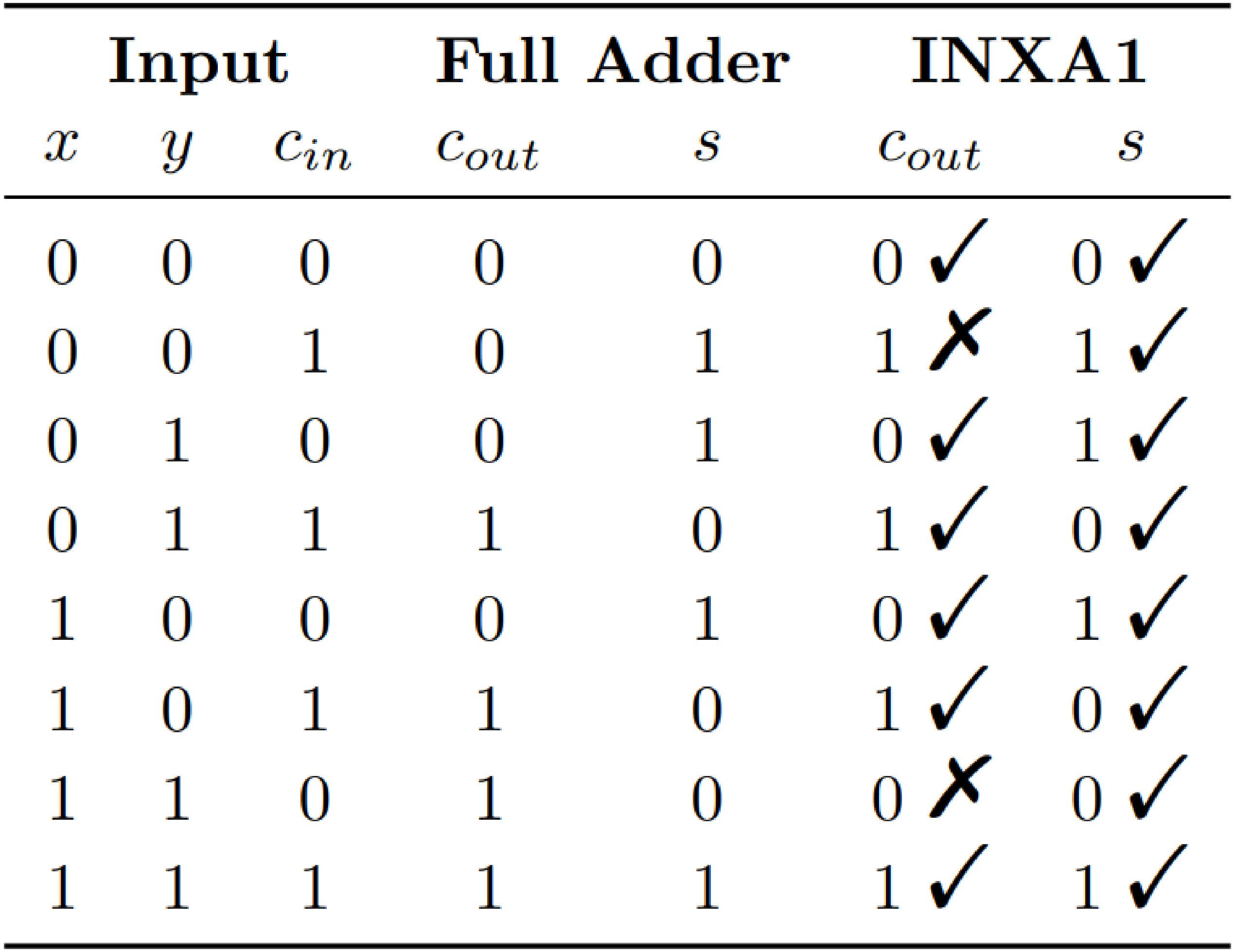


Conventional Full Adder Circuit Design



Approximate Full Adder Circuit Design as proposed by Priyadharshni et al

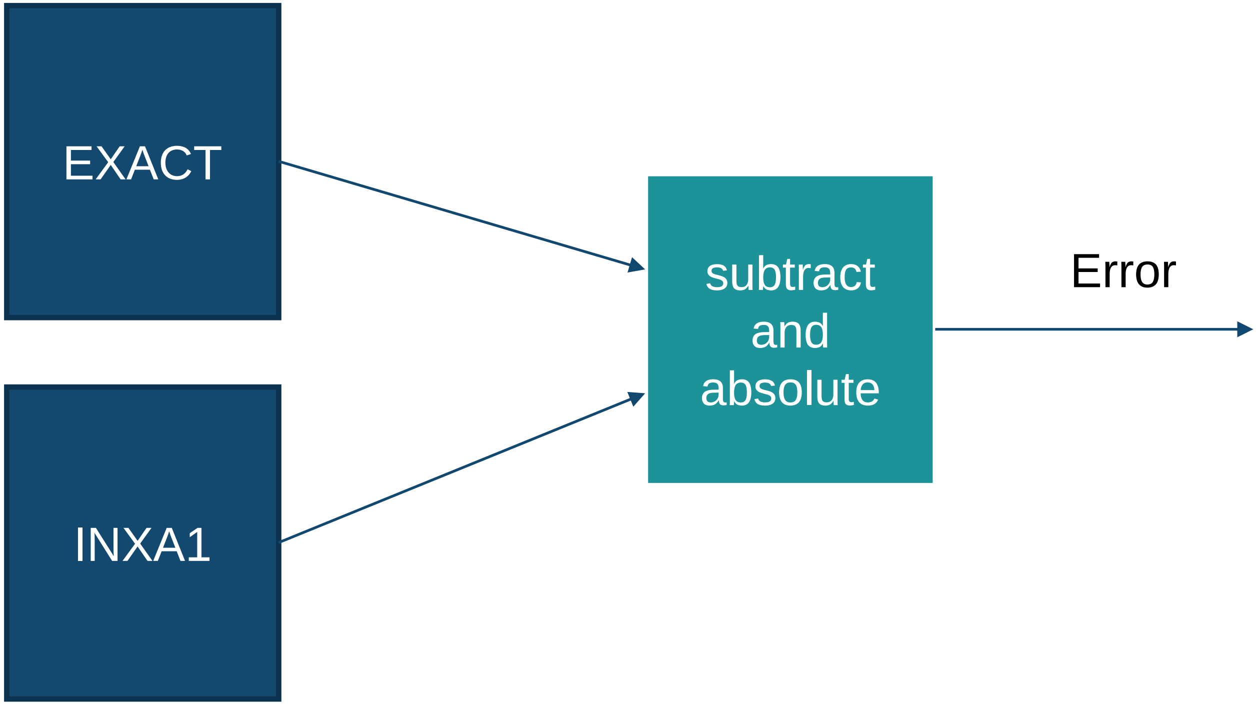
Recap: Truth Table



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 ✓

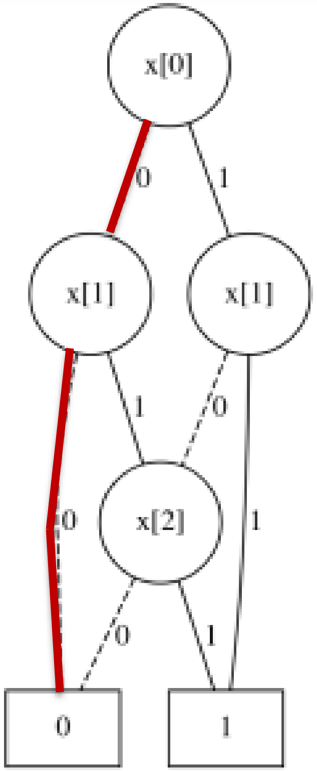
Recap: Error Analysis

- For multi bit inputs, the error is not Hamming distance
- Error must be interpreted as number



Recap: Binary Decision Diagram

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



What’s New

- BDD generation
- Worst case error analysis (partly) finished
- Area and power analysis of the VHDL design

Generating BDD for a 2 bit adder

- Generate outputs for exact and approx. adder

00 + 00 = approx: 010, exact: 000
00 + 01 = approx: 101, exact: 001
00 + 10 = approx: 100, exact: 010
00 + 11 = approx: 011, exact: 011
01 + 00 = approx: 101, exact: 001
01 + 01 = approx: 010, exact: 010
01 + 10 = approx: 011, exact: 011
…

Generating BDD for a 2 bit adder

- Generate outputs for exact and approx. adder
- Calculate error

```
00 + 00 = approx: 010, exact: 000, error: 1
00 + 01 = approx: 101, exact: 001, error: 1
00 + 10 = approx: 100, exact: 010, error: 1
00 + 11 = approx: 011, exact: 011, error: 0
01 + 00 = approx: 101, exact: 001, error: 1
01 + 01 = approx: 010, exact: 010, error: 0
01 + 10 = approx: 011, exact: 011, error: 0
...
```

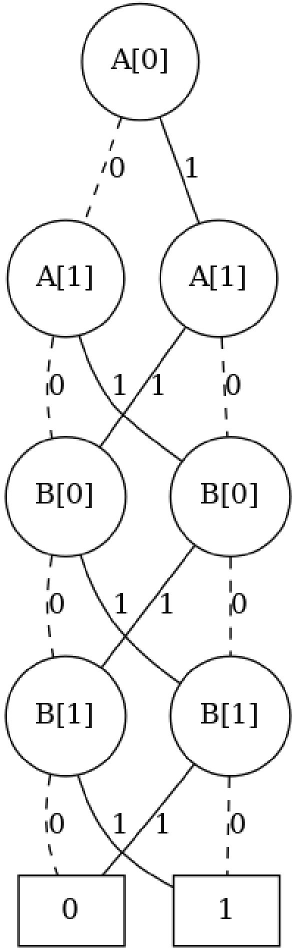
Generating BDD for a 2 bit adder

- Generate outputs for exact and approx. adder
- Calculate error
- Extract truth tables

```
                bit 0
00 + 00 = approx: 010, exact: 000, error: 1
00 + 01 = approx: 101, exact: 001, error: 1
00 + 10 = approx: 100, exact: 010, error: 1
00 + 11 = approx: 011, exact: 011, error: 0
01 + 00 = approx: 101, exact: 001, error: 1
01 + 01 = approx: 010, exact: 010, error: 0
01 + 10 = approx: 011, exact: 011, error: 0
...
```

Generating BDD for a 2 bit adder

- Generate outputs for exact and approx. adder
- Calculate error
- Extract truth tables
- Draw for bits



bit 0

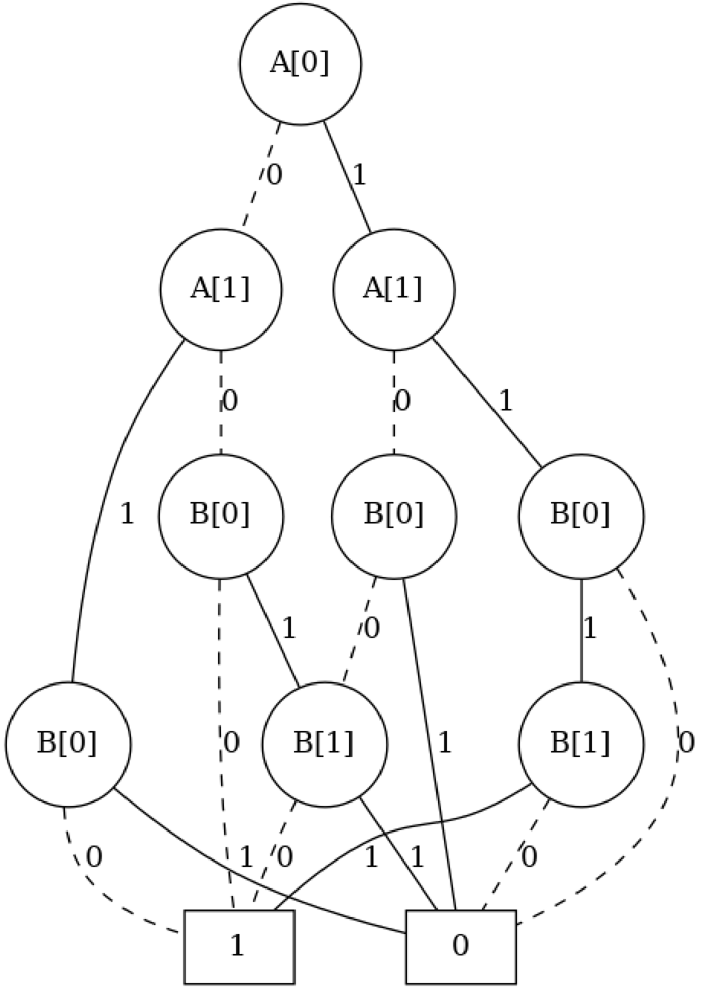
00 + 00 = approx: 010, exact: 000, error: 1
00 + 01 = approx: 101, exact: 001, error: 1
00 + 10 = approx: 100, exact: 010, error: 1
00 + 11 = approx: 011, exact: 011, error: 0
01 + 00 = approx: 101, exact: 001, error: 1
01 + 01 = approx: 010, exact: 010, error: 0
01 + 10 = approx: 011, exact: 011, error: 0
...

Generating BDD for a 2 bit adder

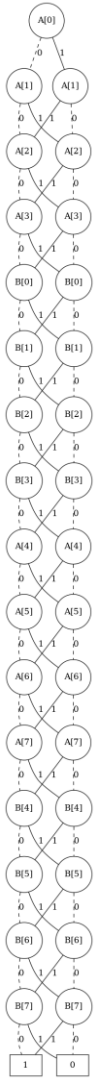
- Generate outputs for exact and approx. adder
- Calculate error
- Extract truth tables
- Draw BDD for bits
- Draw BDD for error

error

00 + 00 = approx: 010, exact: 000, error: 1
00 + 01 = approx: 101, exact: 001, error: 1
00 + 10 = approx: 100, exact: 010, error: 1
00 + 11 = approx: 011, exact: 011, error: 0
01 + 00 = approx: 101, exact: 001, error: 1
01 + 01 = approx: 010, exact: 010, error: 0
01 + 10 = approx: 011, exact: 011, error: 0
...



BDD: Complexity



Example:

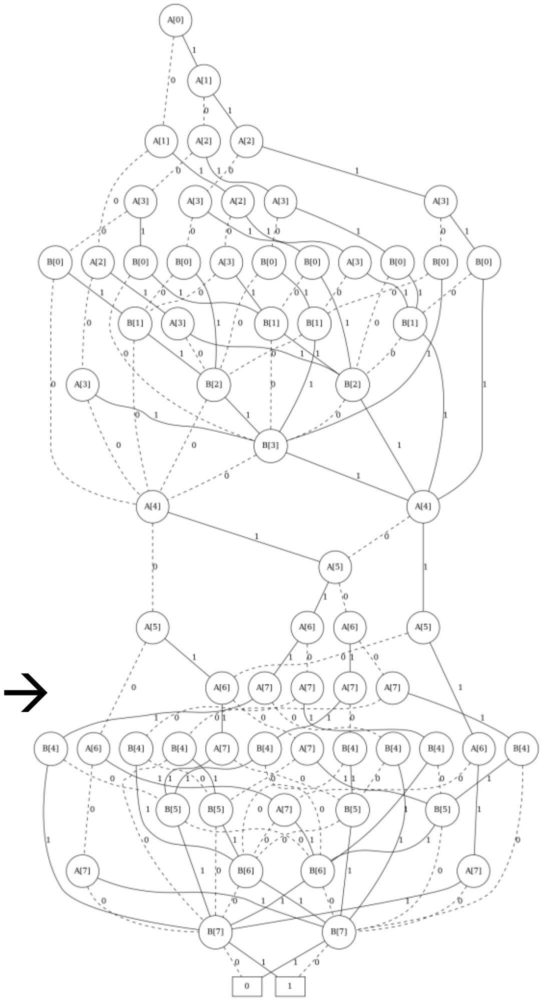
* 8 bit full adder

* 2^(8*2) = 65536 input combinations

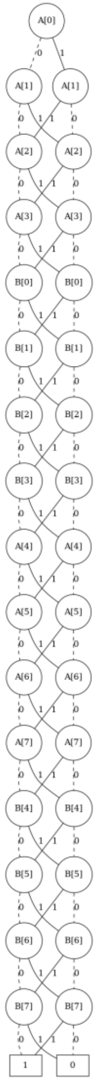
* BDDs for output bit „1“

← Approximate: 31 nodes

Exact: 49 nodes →



BDD: Complexity



Example:

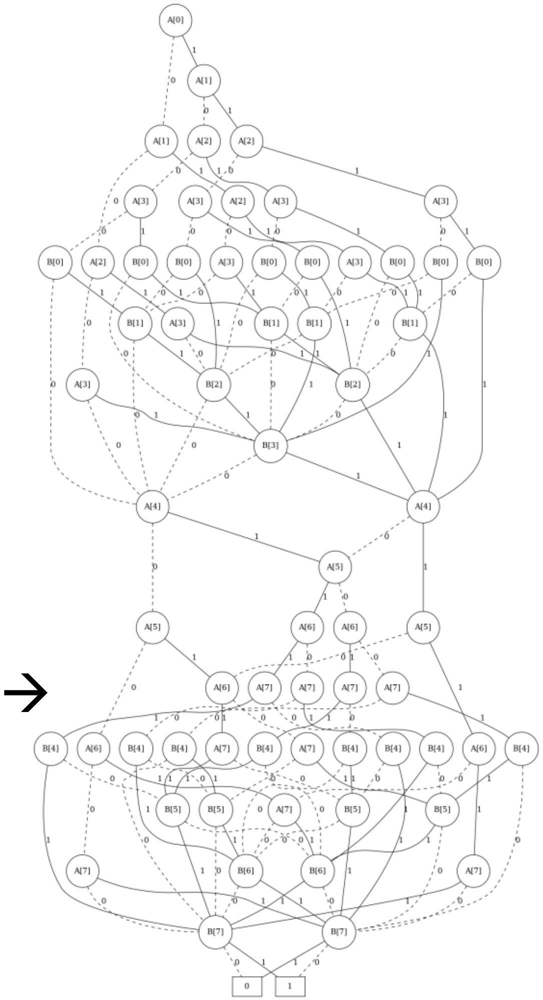
* 8 bit full adder

* 2^(8*2) = 65536 input combinations

* BDDs for output bit „1“

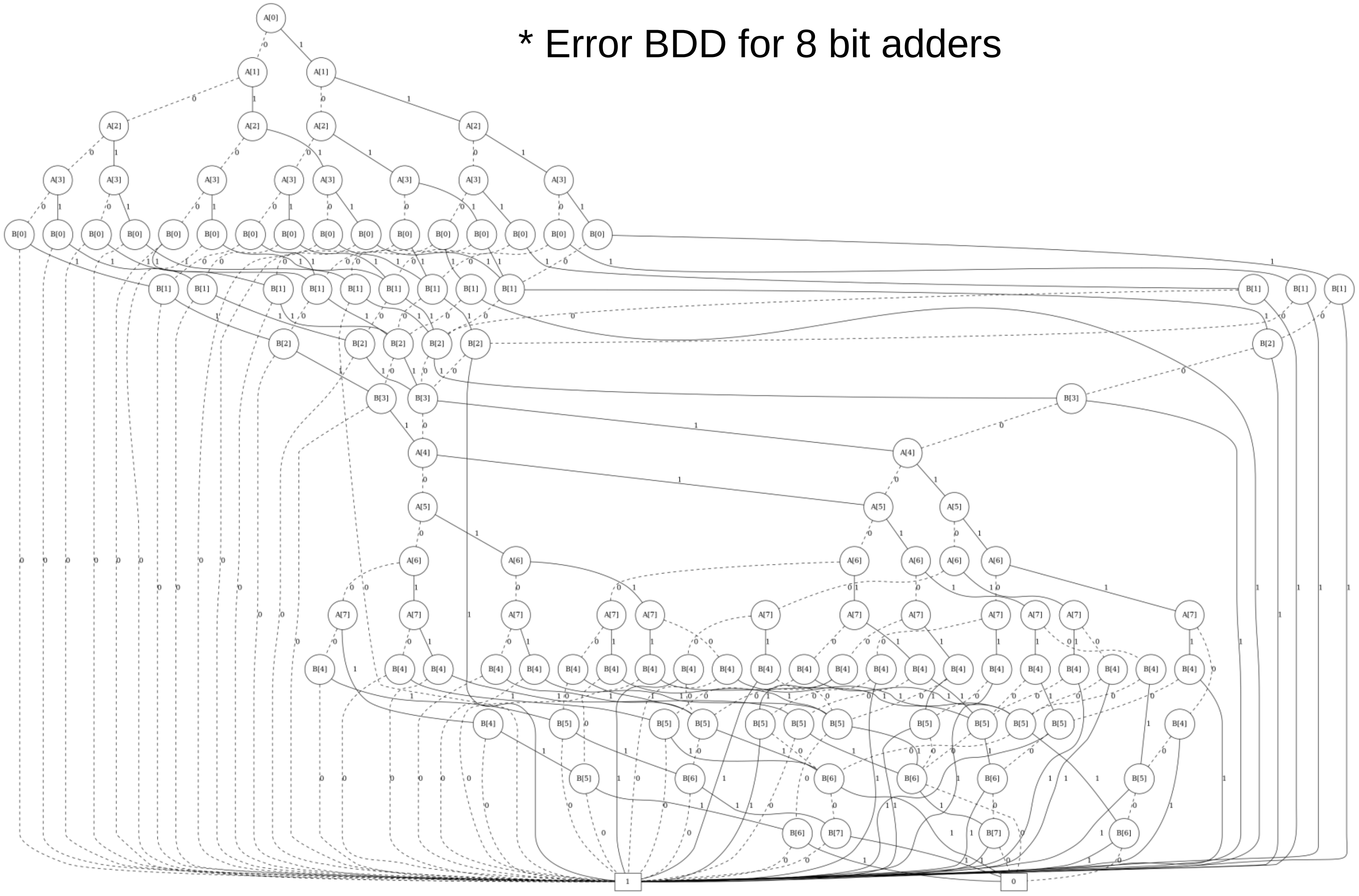
← Approximate: 31 nodes

Exact: 49 nodes →



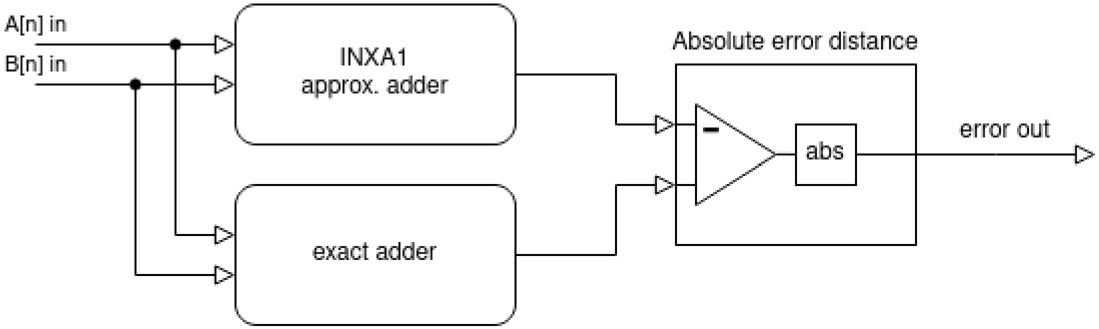
* 64 bit adder ?
* 2^(64*2) = 3.4E38 input combinations

BDD: Error



* Error BDD for 8 bit adders

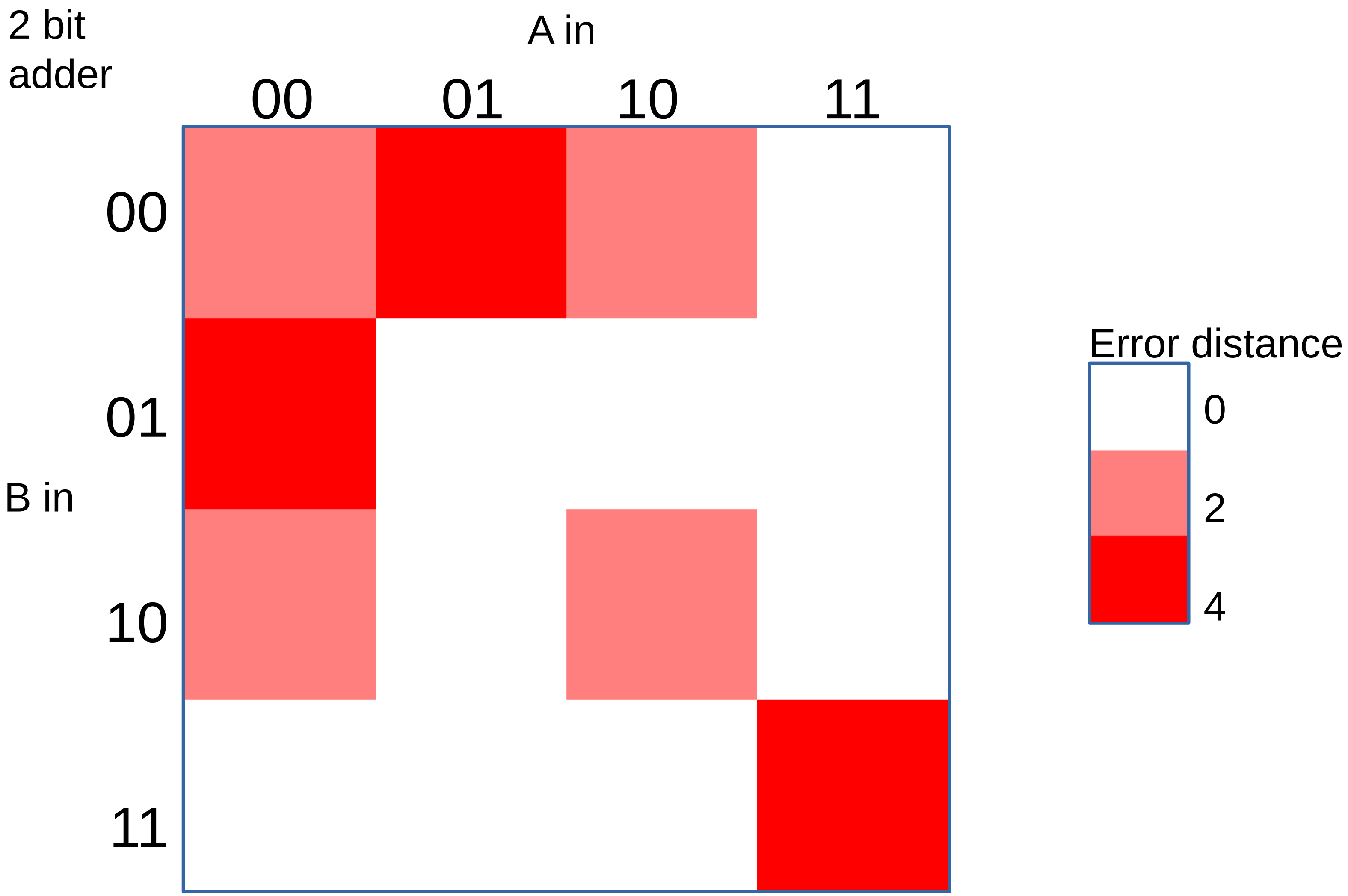
BDD: Error Analysis



00 + 00 = approx: 010, exact: 000, error out: 2
00 + 01 = approx: 101, exact: 001, error out: 4
00 + 10 = approx: 100, exact: 010, error out: 2
00 + 11 = approx: 011, exact: 011, error out: 0
01 + 00 = approx: 101, exact: 001, error out: 4
01 + 01 = approx: 010, exact: 010, error out: 0
01 + 10 = approx: 011, exact: 011, error out: 0
...

Adapted from: Z. Vasicek: Formal Methods for Exact Analysis of Approximate Circuits

BDD: Error Analysis



VHDL in Detail

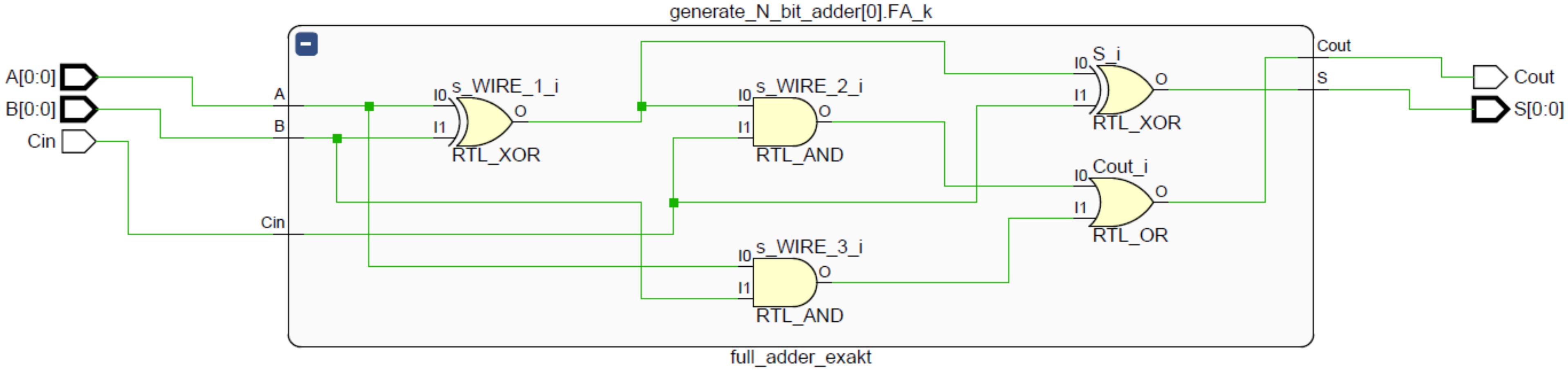
What has been done so far:

- 1-, 4-, 8- and 16-bit full adders in exact and approximate form are completed
- Schematics regarding these designs are realised
- Area and power analysis of the VHDL design are partially done

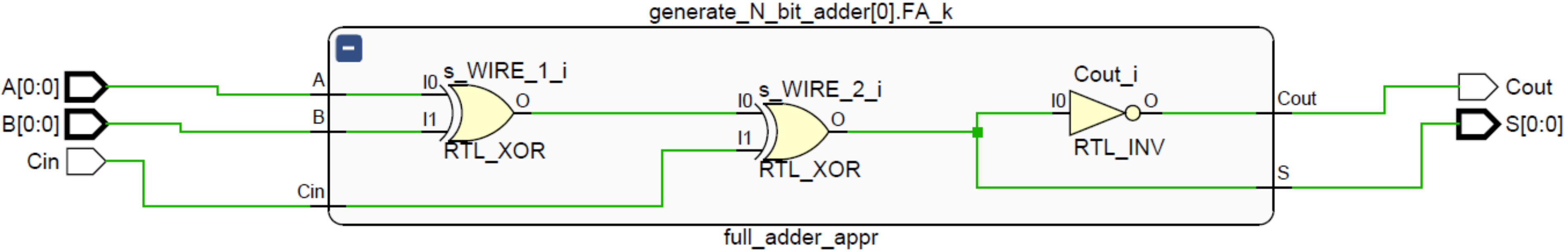
VHDL: Schematics

- Exact vs. Approximate: reduced basic logic gate number

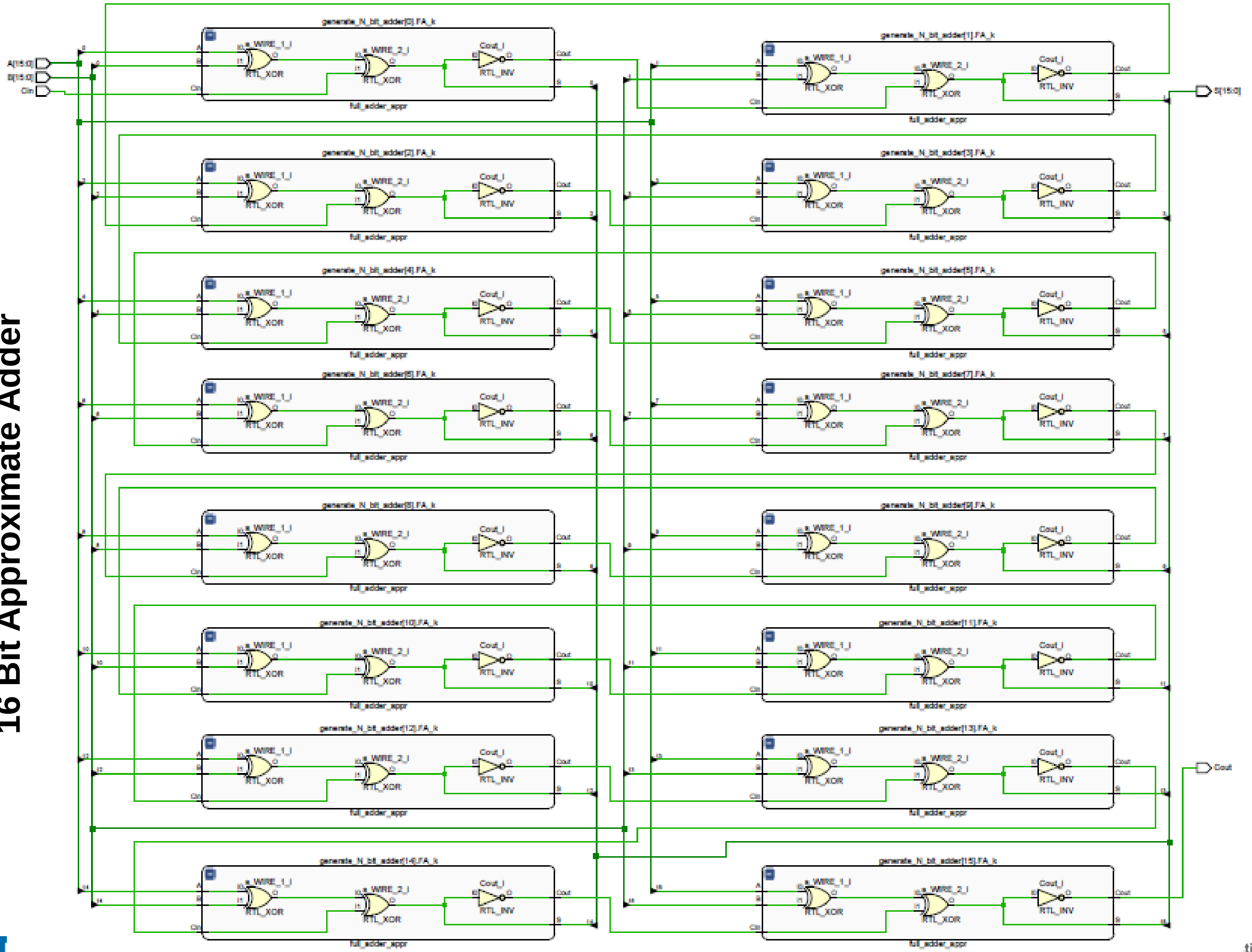
1-Bit Full Adder:



1-Bit Approximate Full Adder (INXA1):



16 Bit Approximate Adder


A[15:0]
B[15:0]
Cin
S[15:0]
Cout
generate_N_bit_adder[0].FA_k
generate_N_bit_adder[1].FA_k
generate_N_bit_adder[2].FA_k
generate_N_bit_adder[3].FA_k
generate_N_bit_adder[4].FA_k
generate_N_bit_adder[5].FA_k
generate_N_bit_adder[6].FA_k
generate_N_bit_adder[7].FA_k
generate_N_bit_adder[8].FA_k
generate_N_bit_adder[9].FA_k
generate_N_bit_adder[10].FA_k
generate_N_bit_adder[11].FA_k
generate_N_bit_adder[12].FA_k
generate_N_bit_adder[13].FA_k
generate_N_bit_adder[14].FA_k
generate_N_bit_adder[15].FA_k
a_WIRE_1_i
a_WIRE_2_i
Cout_i
I0
I1
O
RTL_XOR
RTL_INV
A
B
Cin
Cout
Si
full_adder_appr

UNIVERSITÄT WIEN

titut für Computertechnik Institute of Computer Technology

Area Analysis

Used metric for area: Gate Count in adders

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 Appro.	2	X	X	1	9 (-3)
4 Bit Appro.	8	X	X	4	36 (-8)
8 Bit Appro.	16	X	X	8	72 (-16)
16 Bit Appro.	32	X	X	16	144 (-32)

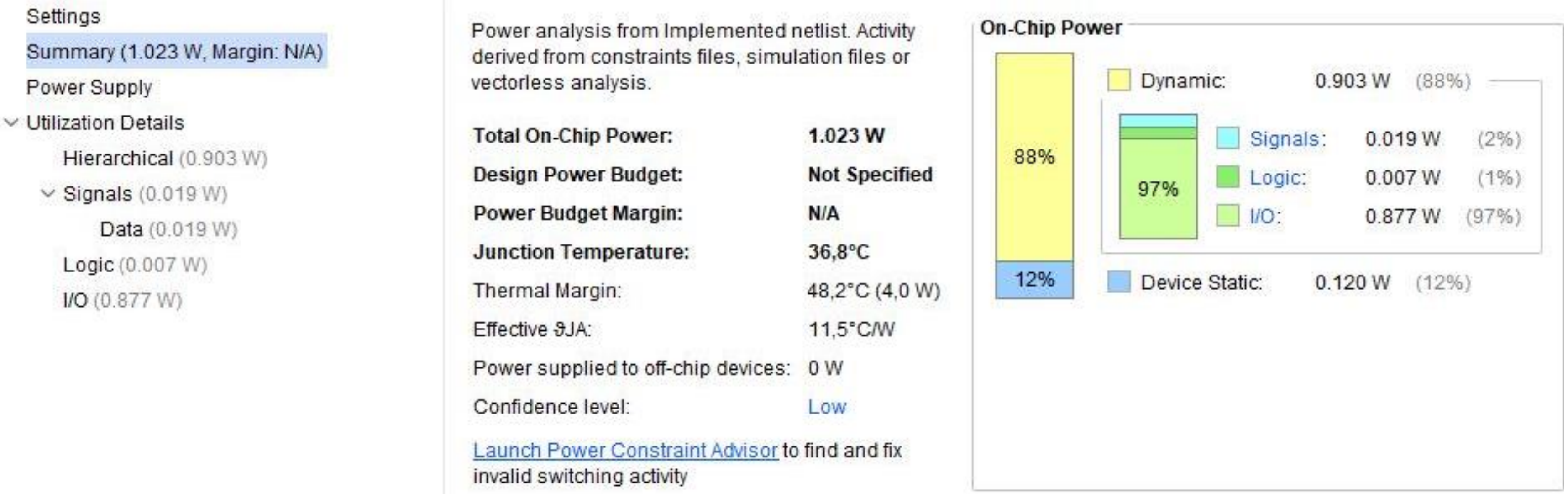
= ~18.2% less

XOR = 2 AND + 1 OR + 1 NOT = 4 Basic Gates

Power Analysis

1 BIT

EXACT



Settings
Summary (1.023 W, Margin: N/A)
Power Supply
Utilization Details
- Hierarchical (0.903 W)
- Signals (0.019 W)
  - Data (0.019 W)
- Logic (0.007 W)
- I/O (0.877 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

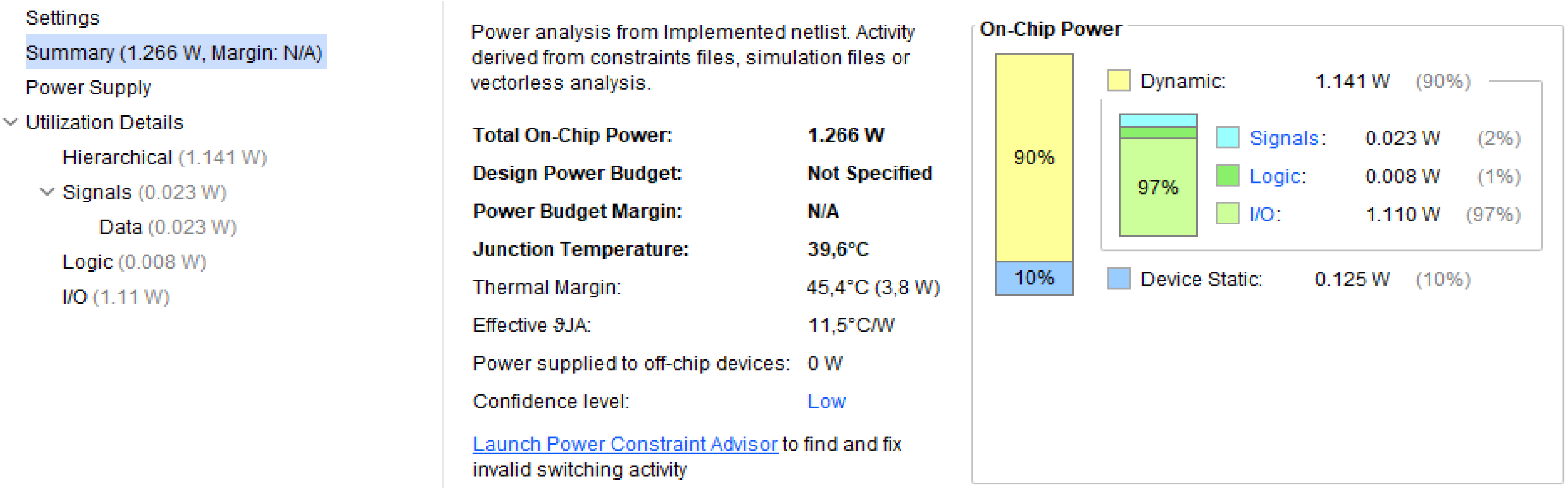
Total On-Chip Power:	**1.023 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**36,8°C**
Thermal Margin:	48,2°C (4,0 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:		0.903 W	(88%)
	Signals:	0.019 W	(2%)
	Logic:	0.007 W	(1%)
	I/O:	0.877 W	(97%)
Device Static:		0.120 W	(12%)

APPROXIMATE



Settings
Summary (1.266 W, Margin: N/A)
Power Supply
Utilization Details
- Hierarchical (1.141 W)
- Signals (0.023 W)
  - Data (0.023 W)
- Logic (0.008 W)
- I/O (1.11 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**1.266 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**39,6°C**
Thermal Margin:	45,4°C (3,8 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

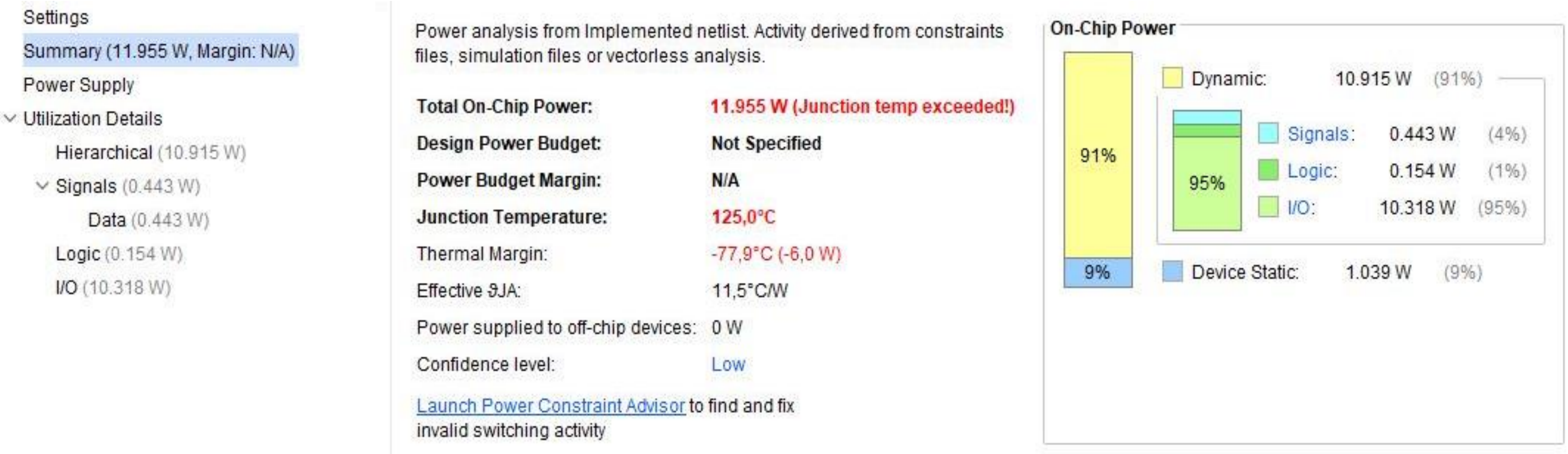
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:		1.141 W	(90%)
	Signals:	0.023 W	(2%)
	Logic:	0.008 W	(1%)
	I/O:	1.110 W	(97%)
Device Static:		0.125 W	(10%)

Power Analysis

16 BIT



EXACT

- Settings
- Summary (11.955 W, Margin: N/A)
- Power Supply
- Utilization Details
  - Hierarchical (10.915 W)
  - Signals (0.443 W)
    - Data (0.443 W)
  - Logic (0.154 W)
  - I/O (10.318 W)

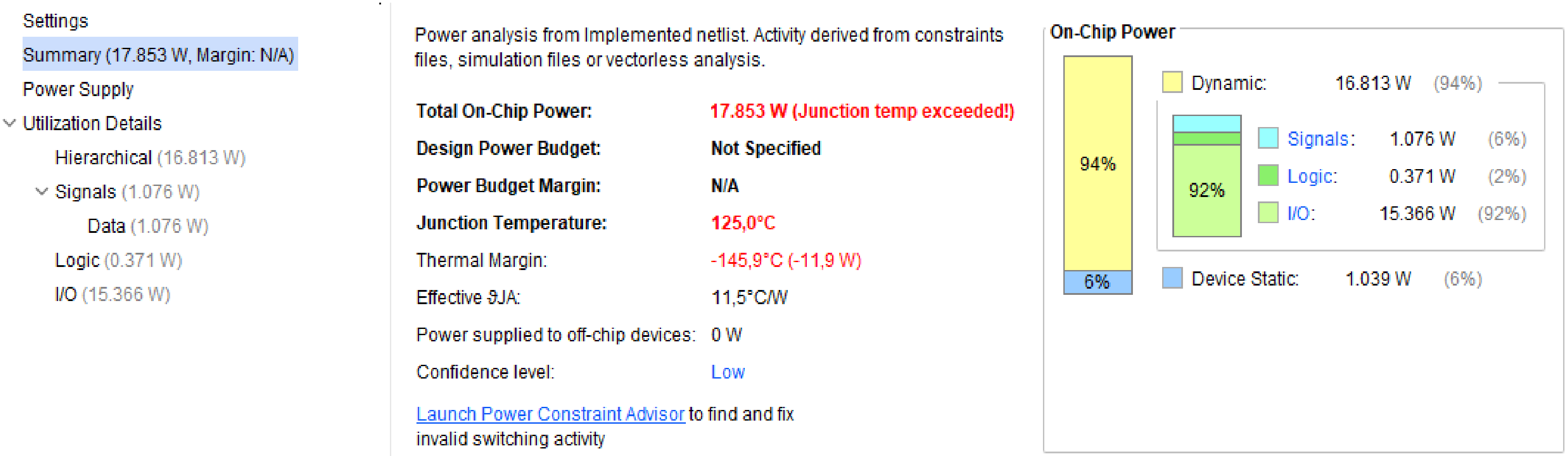
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**11.955 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**125,0°C**
Thermal Margin:	-77,9°C (-6,0 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:		10.915 W	(91%)
	Signals:	0.443 W	(4%)
	Logic:	0.154 W	(1%)
	I/O:	10.318 W	(95%)
Device Static:		1.039 W	(9%)



APPROXIMATE

- Settings
- Summary (17.853 W, Margin: N/A)
- Power Supply
- Utilization Details
  - Hierarchical (16.813 W)
  - Signals (1.076 W)
    - Data (1.076 W)
  - Logic (0.371 W)
  - I/O (15.366 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**17.853 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**125,0°C**
Thermal Margin:	-145,9°C (-11,9 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:		16.813 W	(94%)
	Signals:	1.076 W	(6%)
	Logic:	0.371 W	(2%)
	I/O:	15.366 W	(92%)
Device Static:		1.039 W	(6%)

Problem in Power Analysis

What could be the reason:

- Approximate design is actually faster in calculating inputs.
  - Therefore, in given certain amount of time it processes more calculations, which leads to more power consumption.

Signal Rate (Mtr/s)	A	B	Cin	S	Cout
16 Bit Exact	12500	12500	12500	32304,17	19238,91
16 Bit Appro.	12500	12500	12500	46986,90	49996,23

- Without a Testbench we don‘t have a controlled test environment.
- Vivado is sometimes inconsistent with its‘ analysis tools.

Future Plans

- Adding functionality for generic N bit Adders
- Final Worst Case Error Analysis with BDD
- Several Graphical Error Visualizations
- Designing a Testbench for VHDL design
- Implementing on Zedboard (Hardware)

Thank you!