

Arun Chandrasekharan · Daniel Große  
Rolf Drechsler

# Design Automation Techniques for Approximation Circuits

Verification, Synthesis and Test

 Springer

# Design Automation Techniques for Approximation Circuits

Arun Chandrasekharan • Daniel Große  
Rolf Drechsler

# Design Automation Techniques for Approximation Circuits

Verification, Synthesis and Test



Springer

Arun Chandrasekharan  
OneSpin Solutions GmbH  
Munich, Germany

Daniel Große  
University of Bremen and DFKI GmbH  
Bremen, Germany

Rolf Drechsler  
University of Bremen and DFKI GmbH  
Bremen, Germany

ISBN 978-3-319-98964-8      ISBN 978-3-319-98965-5 (eBook)  
<https://doi.org/10.1007/978-3-319-98965-5>

Library of Congress Control Number: 2018952911

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To Keerthana,  
Nanno  
and  
Zehra*

# Preface

APPROXIMATE COMPUTING is a novel design paradigm to address the performance and energy efficiency needed for future computing systems. It is based on the observation that many applications compute their results more accurately than needed, wasting precious computational resources. Compounded to this problem, dark silicon and device scaling limits in the hardware design severely undermine the growing demand for computational power. Approximate computing tackles this by deliberately introducing controlled inaccuracies in the hardware and software to improve performance. There is a huge set of applications from multi-media, data analytics, deep learning, etc. that can make a significant difference in performance and energy efficiency using approximate computing. However, despite its potential, this novel computational paradigm is in its infancy. This is due to the lack of efficient design automation techniques that are synergetic to approximate computing. Our book bridges this gap. We explain algorithms and methodologies from automated synthesis to verification and test of an approximate computing system. All the algorithms explained in this book are implemented and thoroughly evaluated on a wide range of benchmarks and use cases. Our methodologies are efficient, scalable, and significantly advance the state of the art of the approximate system design.

# Acknowledgments

First and foremost, we would like to thank the members of the research group for computer architecture at the University of Bremen. We deeply appreciate the continuous support, the inspiring discussions, and the stimulating environment provided. Next, we would like to thank all coauthors of the papers which formed the starting point for this book: Mathias Soeken, Ulrich Kühne, and Stephan Eggersgluß. This book would not have been possible without their academic knowledge and valuable insight. Our sincere thanks also go to Kenneth Schmitz, Saman Fröhlich, and Arighna Deb for numerous discussions and successful collaborations.

Munich, Germany  
Bremen, Germany  
Bremen, Germany  
July 2018

Arun Chandrasekharan  
Daniel Große  
Rolf Drechsler

# Contents

<b>1</b>	<b>Introduction</b>	1
1.1	Approximate Computing IC Design Flow	4
1.2	Outline	7
1.3	AxC Software Framework and Related Tools	8
<b>2</b>	<b>Preliminaries</b>	11
2.1	General Notation and Conventions	11
2.2	Data Structures: Boolean Networks	12
2.2.1	Binary Decision Diagrams	14
2.2.2	And-Inverter Graphs	16
2.3	Boolean Satisfiability	17
2.3.1	CNF and Tseitin Encoding	19
2.3.2	Lexicographic SAT	19
2.3.3	Model Counting	20
2.3.4	Bounded Model Checking	21
2.4	Post-production Test, Faults, and ATPG	21
2.5	Quantifying Approximations: Error Metrics	22
2.5.1	Error-Rate	23
2.5.2	Worst-Case Error	24
2.5.3	Average-Case Error	24
2.5.4	Bit-Flip Error	24
<b>3</b>	<b>Error Metric Computation for Approximate Combinational Circuits</b>	27
3.1	Overview	28
3.2	BDD-Based Methods	29
3.2.1	Error-Rate Using BDDs	29
3.2.2	Worst-Case Error and Bit-Flip Error Using BDDs	30
3.2.3	Algorithms for max Function	31
3.2.4	Algorithm 3.2 to Find max	33
3.2.5	Average-Case Error Using BDDs	34



3.3	SAT-Based Methods .....	38
3.3.1	Error-Rate Using SAT .....	38
3.3.2	Worst-Case Error Using SAT .....	39
3.3.3	Bit-Flip Error Using SAT .....	41
3.3.4	Average-Case Error Using SAT .....	41
3.4	Algorithmic Complexity of Error Metric Computations .....	43
3.5	Implementation .....	43
3.5.1	Experimental Results .....	44
3.6	Concluding Remarks .....	50
<b>4</b>	<b>Formal Verification of Approximate Sequential Circuits</b> .....	<b>51</b>
4.1	Overview .....	52
4.2	General Idea .....	53
4.2.1	Sequential Approximation Miter .....	54
4.3	Approximation Questions .....	55
4.3.1	Question 1: What Is the Earliest Time That One Can Exceed an Accumulated Worst-Case Error of $X$ ? .....	55
4.3.2	Question 2: What Is the Maximum Worst-Case Error? .....	56
4.3.3	Question 3: What Is the Earliest Time That One Can Reach an Accumulated Bit-Flip Error of $X$ ? .....	57
4.3.4	Question 4: What Is the Maximum Bit-Flip Error? .....	57
4.3.5	Question 5: Can One Guarantee That the Average-Case Error Does Not Exceed $X$ ? .....	57
4.4	Experimental Results .....	57
4.4.1	Approximated Sequential Multiplier .....	58
4.4.2	Generality and Scalability .....	61
4.5	Concluding Remarks .....	64
<b>5</b>	<b>Synthesis Techniques for Approximation Circuits</b> .....	<b>65</b>
5.1	Overview .....	65
5.2	Approximate BDD Minimization .....	67
5.2.1	BDD Approximation Operators .....	68
5.2.2	Experimental Evaluation .....	70
5.3	AIG-Based Approximation Synthesis .....	73
5.3.1	And-Inverter Graph Rewriting .....	73
5.3.2	Approximation-Aware Rewriting .....	75
5.3.3	Implementation .....	76
5.3.4	Experimental Results .....	78
5.4	Concluding Remarks .....	86
<b>6</b>	<b>Post-Production Test Strategies for Approximation Circuits</b> .....	<b>87</b>
6.1	Overview .....	87
6.2	Approximation-Aware Test Methodology .....	90
6.2.1	General Idea and Motivating Example .....	90
6.2.2	Approximation-Aware Fault Classification .....	92

- 6.3 Experimental Results ..... 96
  - 6.3.1 Results for the Worst-Case Error Metric ..... 97
  - 6.3.2 Results for the Bit-Flip Error Metric ..... 100
- 6.4 Concluding Remarks ..... 102
- 7 ProACt: Hardware Architecture for Cross-Layer Approximate Computing** ..... 103
  - 7.1 Overview ..... 103
    - 7.1.1 Literature Review on Approximation Architectures ..... 106
  - 7.2 ProACt System Architecture ..... 107
    - 7.2.1 Approximate Floating Point Unit (AFPU) ..... 109
    - 7.2.2 Instruction Set Architecture (ISA) Extension ..... 110
    - 7.2.3 ProACt Processor Architecture ..... 111
    - 7.2.4 Compiler Framework and System Libraries ..... 112
  - 7.3 ProACt Evaluation ..... 112
    - 7.3.1 FPGA Implementation Details ..... 113
    - 7.3.2 Experimental Results ..... 114
  - 7.4 Concluding Remarks ..... 118
- 8 Conclusions and Outlook** ..... 119
  - 8.1 Outlook ..... 120
- References** ..... 123
- Index** ..... 129

# List of Algorithms

3.1	BDD maximum value using mask .....	32
3.2	BDD maximum value using characteristic function.....	34
3.3	BDD weighted sum .....	36
3.4	SAT maximum value.....	40
3.5	SAT weighted sum .....	42
4.1	Sequential worst-case error .....	56
5.1	Approximate BDD minimization .....	68
5.2	Approximation rewriting .....	75
5.3	Sequential bit-flip error .....	78
6.1	Approximation-aware fault classification.....	94

# List of Figures

Fig. 1.1	Design flow for approximate computing IC .....	5
Fig. 2.1	Homogeneous Boolean networks: AIG and BDD .....	13
Fig. 2.2	Non-homogeneous Boolean network: netlist .....	14
Fig. 3.1	Formal verification of error metrics .....	29
Fig. 3.2	Xor approximation miter for error-rate .....	30
Fig. 3.3	Difference approximation miter for worst-case error.....	31
Fig. 3.4	Bit-flip approximation miter for bit-flip error.....	31
Fig. 3.5	Characteristic function to compute the maximum value .....	35
Fig. 3.6	Characteristic function to compute the weighted sum.....	36
Fig. 4.1	General idea of a sequential approximation miter .....	53
Fig. 5.1	Approximation synthesis flow .....	66
Fig. 5.2	BDD approximation synthesis operators.....	71
Fig. 5.3	Evaluation of BDD approximation synthesis operators .....	72
Fig. 5.4	Cut set enumeration in AIG .....	74
Fig. 5.5	Approximation miter for synthesis .....	77
Fig. 6.1	Approximation-aware test and design flow .....	88
Fig. 6.2	Faults in an approximation adder.....	91
Fig. 6.3	Fault classification using approximation miter .....	95
Fig. 7.1	ProACt application development framework .....	105
Fig. 7.2	ProACt system overview .....	108
Fig. 7.3	ProACt Xilinx Zynq hardware.....	113

# List of Tables

Table 3.1	Error metrics: 8-bit approximation adders .....	45
Table 3.2	Error metrics: 16-bit approximation adders .....	46
Table 3.3	Evaluation: ISCAS-85 benchmark.....	47
Table 3.4	Evaluation: EPFL benchmark.....	48
Table 4.1	Evaluation of approximation questions.....	59
Table 4.2	Run times for the evaluation of approximation questions .....	62
Table 5.1	BDD approximation synthesis operators .....	69
Table 5.2	Synthesis comparison for approximation adders.....	80
Table 5.3	Error metrics comparison for approximation adders.....	81
Table 5.4	Image processing with approximation adders.....	83
Table 5.5	Approximation synthesis results for LGSynth91 benchmarks .....	84
Table 5.6	Approximation synthesis results for other designs .....	85
Table 6.1	Truth table for approximation adder .....	93
Table 6.2	Fault classification for worst-case error: benchmarks set-1 .....	98
Table 6.3	Fault classification for worst-case error: benchmarks set-2 .....	99
Table 6.4	Fault classification for bit-flip error .....	101
Table 7.1	ProACt FPGA hardware prototype details .....	114
Table 7.2	Edge detection with approximations .....	116
Table 7.3	Math functions with approximations .....	117

# Chapter 1

## Introduction



APPROXIMATE COMPUTING is an emerging design paradigm to address the performance and energy efficiency needed for the future computing systems. Conventional strategies to improve the hardware performance such as device scaling have already reached its limits. Current device technologies such as 10 nm are already reported to have significant secondary effects such as quantum tunneling. On the energy front, dark silicon and the power density is a serious challenge and limiting factor for several contemporary IC design flows. It is imperative that the current state-of-the-art techniques are inadequate to meet the growing demands of the computational power. Approximate computing can potentially address these challenges. It refers to hardware and software techniques where the implementation is allowed to differ from the specification, but within an acceptable range. The approximate computing paradigm delivers *performance* at the cost of *accuracy*. The key idea is to trade off correct computations against energy or performance. At a first glance, one might think that this approach is not a good idea. But it has become evident that there is a huge set of applications which can tolerate errors. Applications such as multi-media processing and compressing, voice recognition, web search, or deep learning are just a few examples. However, despite its huge potential, approximate computing is not a mainstream technology yet. This is due to the lack of reliable and efficient design automation techniques for the design and implementation of an approximate computing system. This book bridges this gap. Our work addresses the important facets of approximate computing hardware design—from formal verification and error guarantees to synthesis and test of approximation systems. We provide algorithms and methodologies based on classical formal verification, synthesis, and test techniques for an approximate computing IC design flow. Further, towards the end, a novel hardware architecture is presented for cross-layer approximate computing. Based on the contributions, we advance the current state-of-the-art of the approximate hardware design.

Several applications spend a huge amount of energy to guarantee correctness. However, correctness is not always required due to some inherent characteristics