HAL open science

Contributions to the scalability of automatic precision tuning

Van-Phu Ha

To cite this version:

Van-Phu Ha. Contributions to the scalability of automatic precision tuning. Other [cs.OH]. Université de Rennes, 2023. English. NNT : 2023URENS007 . tel-04189422

HAL Id: tel-04189422

https://theses.hal.science/tel-04189422v1

Submitted on 28 Aug 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
MAthématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Informatique*

Par
Van-Phu HA

Contributions to the Scalability of Automatic Precision Tuning

Thèse présentée et soutenue à IRISA/Inria, le 10/03/2023
Unité de recherche : IRISA/Inria, Rennes
Thèse N° :

Rapporteurs avant soutenance :

Florent de Dinechin	Professeur, INSA Lyon
Gabriel Caffarena	Professeur, Universidad CEU San Pablo, Madrid

Composition du Jury :

Président :	Daniel Ménard	Professeur, INSA Rennes, IETR
Examinateurs :	Florent de Dinechin	Professeur, INSA Lyon
	Gabriel Caffarena	Professeur, Universidad CEU San Pablo, Madrid
	Fabienne Jézéquel	Maître de Conférences HDR, Sorbonne Université, LIP6, Paris
	Daniel Ménard	Professeur, INSA Rennes, IETR
Dir. de thèse :	Olivier Sentieys	Professeur, Université de Rennes, Inria, IRISA

Acknowledgement

In the following lines, I would like to take the opportunity to express my gratitude to everyone who has given their support to me over this research period.

To begin, I would want to express my appreciation to both of my advisers, Prof. Olivier Sentieys and Dr. Tomofumi Yuki, for all of the assistance and support they have consistently provided. In addition to all of the technical advice they have given me, they have consistently kept me inspired and pushed me to push myself and aim for the greatest possible objectives.

I would like to give warm thanks for Prof. Daniel Ménard and Dr. François Charot in the CSI jury. They have been monitoring my research every year and giving me sincere advice that has kept me on the right track in my research and opened up new research directions.

My thanks also go to all the members of the jury, namely Prof. Florent de Dinechin, Prof. Gabriel Caffarena, Mrs. Fabienne Jézéquel, Prof. Daniel Ménard, for giving me the privilege of being a part of the thesis defense and evaluating my work.

My sincere thanks also go to Dr. Hoang Thanh Tung, who provided me with frequent conversation on issues relevant to both study and technology. He has been of great support to me by providing me with astute advice and the best possible direction with regard to new research topics.

I would like to express my gratitude to Silviu-Ioan Filip, who took the time to talk to me about regression technique and was a great assistance to me. Thanks to Nadia Derouault for her administrative support and to Ali Hassan El Moussawi for his support with Gecos tools.

The big thanks go to my colleagues and friends in TARAN team (formerly CAIRN team) at IRISA/Inria Laboratory. They all made the time spent at work enjoyable, with fun or serious conversations, technical or not.

In addition, I feel obligated to acknowledge my close friends Nguyen Khanh Linh, Cong Minh Thanh, and Nguyen Thi Thanh Tam for the valuable time they spent supporting me.

For my family, I would like to express my gratitude to my parents, Ha Hong Thanh and Bui Thi Khe, and the rest of my family for their unwavering love and support throughout my life and academic pursuits. Last but not least, I want to thank my wife, Vu Thi Van Anh, who always encourages me and continuously fills my life with love and happiness.

Table of Contents

Résumé en français **8**

List of acronyms **13**

List of figures **16**

List of tables **17**

1 Introduction **19**

1.1 Energy-Efficient Computing in Post Moore's Law 19

1.2 Objective of Thesis - Application-Level Tuning of Accuracy 25

1.3 Thesis Organization 25

Bibliography 27

2 Theoretical Background and Related Work **31**

2.1 Approximate Computing 31

2.1.1 Approximate Computing for Error-Resilient Applications 32

2.1.2 Cross-Layer Approximate Computing Techniques 34

Approximate Computing at the Software Level 34

Approximate Computing at the Architecture Level 35

Approximate Computing at the Hardware Level 36

2.2 Binary Arithmetic Number Representations 37

2.2.1 Floating-Point Arithmetic 39

Floating-Point Exceptions 40

Rounding modes 41

2.2.2 Fixed-Point Arithmetic 41

2.3 Fixed-Point Conversion Process 42

2.3.1 Integer Word Length (IWL) Selection 45

Interval-Based Approaches 46

Statistical Approaches 47

Stochastic Approaches 47

2.3.2 Fractional Word Length (FWL) Selection 48

Cost Evaluation 48

Accuracy Evaluation 48
Optimization Approaches 50
Brute-force with upper and lower bounds. 51
Heuristics. 51
Min+1 bit. 51
Max-1 bit. 52
Hybrid algorithms. 52
Divide-and-conquer approaches. 53
Analytical approaches. 54
2.4 Conclusions 55
Bibliography 56

3 Towards Generic and Scalable Word-Length Optimization 67
3.1 Introduction 67
3.2 Background and related work 68
3.2.1 Word-Length Optimization 68
3.2.2 Noise Budgeting 69
3.3 Approach Overview 70
3.4 Model Construction 73
3.4.1 Data Points for Cost Function ($\hat{\boldsymbol{C}}$) 73
3.4.2 Data Points for Quality Function ($\hat{\boldsymbol{\lambda}}$) 75
3.4.3 Polynomial Fitting 75
3.5 Evaluation 76
3.5.1 Experimental Setup 76
3.5.2 Image Signal Processor 77
3.5.3 Stereo Matching 78
3.5.4 Empirically Constructed Models 79
3.5.5 Exploration Time and Quality of Solution 79
3.5.6 FIR and IIR Filters 84
3.6 Conclusion 85
Bibliography 86

4 Leveraging Bayesian Optimization to Speed Up Automatic Precision Tuning 89
4.1 Introduction 89
4.2 Background and Related Work 91
4.2.1 Word-Length Optimization and Classical Approaches 91
4.2.2 Bayesian Optimization 92
4.3 Motivations 93

4.3.1 Performance Analysis: TPE vs. Tabu . . . 93
4.3.2 Initial Combinations of TPE and Tabu . . . 95
4.4 Proposed Hybrid Approach . . . 97
4.4.1 Loss Function . . . 97
4.4.2 Transition Point for Hybrid Approach . . . 97
4.5 Evaluation . . . 100
4.5.1 Experiment setup . . . 100
4.5.2 Performance Evaluation . . . 101
4.6 Conclusion . . . 103
Bibliography . . . 104

5 Resource-Constrained Word Length Optimization - A New Problem 107
5.1 Introduction . . . 107
5.2 Related Work . . . 109
5.3 Resource-Constrained WLO . . . 110
5.3.1 Room for Accuracy Improvement Given a Cost Budget . . . 111
5.3.2 Resource-Constrained WLO as a New Problem . . . 112
5.3.3 Limitations of Classical WLO Methods . . . 113
5.4 Solving Resource-Constrained WLO . . . 113
5.4.1 Loss Function . . . 115
5.4.2 TPE Algorithm . . . 115
5.5 Evaluation . . . 116
5.5.1 Experiment setup . . . 116
5.5.2 Performance Evaluation . . . 118
5.6 Conclusion . . . 122
Bibliography . . . 122

6 Conclusions and Perspectives 125

List of publications 128

Appendix A: TypeX - An Automatic Float to Fix Conversion toolbox 131

Appendix B: An Extension of TypEx in Python 148

Résumé en français

La consommation d'énergie est l'un des problèmes majeurs de l'informatique aujourd'hui, du calcul haute performance aux systèmes embarqués. Ces dernières années, l'approximation des calculs a reçu un regain d'intérêt pour améliorer l'efficacité énergétique. De nombreuses applications n'exigent pas une précision élevée, et les techniques de calcul approximatif augmentent l'espace de conception en fournissant de nombreux compromis entre la précision, les coûts et les performances. Cette thèse se concentre sur le développement de méthodes pour l'exploration systématique de cet espace de conception, y compris la modélisation de la performance et de la précision et l'automatisation de la conception. Nous utilisons la virgule fixe pour la représentation des données des données et nous optimisons la longueur du mot de chaque données et calcul pour chercher un bon équilibre entre le coût et la précision. Ce problème est appelé *Word length Optimization* (WLO) ou réglage automatique de la précision. Cette thèse contribue à trois directions de recherche. Premièrement, une méthode est proposée pour améliorer le passage à l'échelle du WLO pour les grandes applications. Pour réduire la complexité exponentielle de la nature de WLO, l'application d'entrée est décomposée en *noyaux* qui sont ensuite résolus indépendamment. Pour allouer les budgets de réduction de précision à chaque noyau, l'idée principale est de caractériser l'impact de l'approximation de chaque noyau sur la précision et le coût par simulation et régression pour construire les modèles empiriques. La deuxième direction de recherche est un algorithme hybride combinant l'optimisation Bayésienne (BO) et une recherche locale rapide pour accélérer la procédure WLO. Un mécanisme efficace est proposé pour obtenir de bons modèles en peu de temps. La dernière contribution ouvre une nouvelle voie de recherche sur le WLO avec contraintes de ressources. Les approches actuelles résolvent principalement les problèmes de WLO avec une contrainte de qualité (précision). Dans cette étude, un algorithme basé sur l'optimisation bayésienne a été proposé pour maximiser la qualité des calculs sous contrainte d'un budget de coût du matériel.

Motivations

Alors que nous entrons dans l'ère du *"dark silicon"*, où les avantages de la réduction de la taille des transistors sont limités par la puissance consommée, les scientifiques et les fabricants de puces se trouvent au défi de trouver des solutions alternatives efficaces. Pour l'instant, nous devons concentrer nos efforts sur le développement de technologies à haut rendement énergétique

qui augmentent considérablement les ratios énergie-performance des microprocesseurs tout en réduisant leur surface et leur consommation d'énergie. La demande d'architectures informatiques et d'approches alternatives qui consomment moins d'énergie et génèrent moins de chaleur pousse à réexaminer les microprocesseurs traditionnels.

Ces dernières années, parallèlement aux progrès de l'architecture des calculateurs informatiques, comme les processeurs multi-cœurs ou les systèmes hétérogènes, les informaticiens et les architectes de machines se sont particulièrement intéressés à la modification des méthodes de calcul pour améliorer encore l'efficacité énergétique. Le calcul approximatif (AxC) apparaît comme une solution potentielle pour améliorer les performances et la consommation d'énergie des systèmes embarqués. Le calcul approximatif vise à fournir de bonnes solutions à des problèmes difficiles tout en utilisant beaucoup moins de ressources de traitement que les solutions plus précises. Par exemple, au lieu de chercher la valeur qui fournit le résultat le plus précis, les algorithmes approximatifs prennent une solution acceptable et garantissent un certain seuil d'erreur. Fondamentalement, AxC est conçu pour relaxer les calculs parfaitement précis à un niveau d'inexactitude acceptable pour gagner en performance et/ou réduire la consommation d'énergie. En effet, de nombreux systèmes embarqués ne nécessitent pas une précision (ou exactitude des calculs) élevée, et les concepteurs de matériel recherchent souvent un bon équilibre entre précision, vitesse, énergie et coût en surface. Diverses techniques de calcul approximatif élargissent l'espace de conception en fournissant un autre ensemble de points dans cet espace, permettant de trouver des compromis performance/précision. Aujourd'hui, la communauté des chercheurs et l'industrie déploient des efforts intensifs pour mettre en œuvre les méthodes de calcul approximatif proposées dans des systèmes commerciaux, tels que les plateformes FPGA, ASIC et GPU, afin de réduire la consommation d'énergie et d'améliorer les performances.

L'une des techniques les plus efficaces de l'AxC consiste à réduire la précision de la représentation arithmétique. Les représentations typiques à virgule flottante, telles que *float* (32 bits) et *double* (64 bits), peuvent être remplacées par des formats de données moins précis, tels que l'arithmétique à virgule fixe. De nombreuses techniques de traitement numérique du signal (DSP) utilisant une arithmétique à précision réduite permettent d'augmenter considérablement l'efficacité énergétique en diminuant les coûts de calcul et le stockage des données, tout en maintenant la précision requise du modèle.

Les systèmes embarqués sont plus performants et moins coûteux lorsqu'ils utilisent une arithmétique à précision réduite, mais il n'est pas facile de trouver un bon équilibre entre la perte de précision et l'amélioration de l'efficacité énergétique. L'état actuel de ces techniques consiste à concevoir des moyens d'automatiser l'optimisation de le nombre de bits (aussi longueur des mots) d'une manière qui tire le meilleur parti des avantages de la réduction de la précision des

opérateurs. Ce processus est appelé optimisation de la longueur des mots (WLO pour *Word Length Optimization*) ou réglage automatique de la précision *(Automatic Precision Tuning)*. Ce processus explore les meilleures longueurs de mots pour les variables d'une application donnée afin de réduire le coût d'implémentation (consommation d'énergie et/ou surface), tout en maintenant la précision à un niveau acceptable malgré la quantification des données. Pour les applications réelles, où de nombreuses variables sont impliquées, cette procédure est confrontée à un problème de passage à l'échelle en raison de l'augmentation exponentielle de la complexité. En réalité, on estime que 25% à 50% du travail de conception est encore consacré à l'obtention d'un choix optimisé de configurations de longueur de mots en virgule fixe.

Objectifs et contributions

Dans cette thèse, nous nous concentrons sur la modélisation et l'utilisation des approches AxC au niveau logiciel de conception. Cette thèse étudie de nouvelles méthodologies pour l'exploration systématique de l'espace de conception, y compris la modélisation du coût d'implémentation, qualité de service (QoS) et l'automatisation de la conception. Cela nous permet d'accélérer la sélection de la configuration optimale ou quasi-optimale qui (1) réduit le coût (surface et/ou consommation d'énergie) tout en répondant à la précision nécessaire à la sortie de l'application, ou (2) optimise la précision à la sortie de l'application dans un budget de coût. Cette thèse examine la question du WLO sous plusieurs angles, tels que le passage à l'échelle pour des applications larges et l'optimisation sous contrainte de ressources. Dans cette dissertation, trois contributions principales ont été proposées.

Dans la première contribution, nous proposons une approche pour atteindre une procédure d'optimisation de la longueur des mots (WLO) qui est plus évolutive pour les grands systèmes, et qui utilise des critères de qualité sophistiqués comme la similarité structurelle (SSIM) en traitement d'images. Pour éviter une croissance exponentielle et incontrôlée du temps d'exploration, cette méthode divise l'application d'entrée en noyaux *(kernels)* plus petits. La question fondamentale qui est étudiée dans cette recherche est le problème de l'allocation des budgets de bruit dus aux calculs aux noyaux individuels qui constituent l'application. Pour ce faire, il est nécessaire de modéliser les interactions entre les différents noyaux. L'idée centrale est d'utiliser la simulation et la régression pour évaluer l'effet de l'approximation de chaque noyau sur la précision ou le coût au niveau global. Notre méthode permet aux techniques de WLO de passer à l'échelle et de trouver de meilleures siolutions pour des applications complexes telles qu'un pipeline de traitement de d'images et la vision stéréo.

La deuxième contribution vise à accélérer le processus WLO et nous proposons une approche

hybride qui combine l'optimisation Bayésienne (BO) avec une recherche locale rapide. Les résultats expérimentaux de ce chapitre fournissent la première preuve que cette combinaison réduit le temps d'exploration. Nous proposons ensuite une nouvelle technique qui permet de trouver automatiquement un point de transition approprié entre les deux algorithmes. Afin d'identifier le moment où il faut passer de la BO à la recherche locale, nous effectuons une analyse statistique de la convergence des modèles probabilistes créés pendant l'optimisation Bayésienne, puis nous formulons une condition d'arrêt. Les résultats expérimentaux montrent que, pour les grands benchmarks utilisés, notre technique peut faire gagner jusqu'à 80% de temps d'exploration.

La dernière contribution s'est concentrée sur un autre aspect de l'optimisation de la longueur des mots qui est affecté par les limitations matérielles. Il est important de noter que la majorité des méthodes de pointe résolvent le problème de WLO sous contrainte de précision. Aucune étude n'a été menée sur la manière d'améliorer la qualité de service (i.e., ici la précision des calculs) tout en maintenant les coûts à un niveau bas. Pour les équipements électroniques à faible puissance ou à faible coût, il s'agit d'un obstacle majeur à l'amélioration des performances des applications. Dans un premier temps, nous démontrons l'importance du problème WLO sous contraintes de ressources, en nous concentrant sur les systèmes à consommation énergétique contrainte. Nous soulignons ensuite la difficulté de choisir un état initial approprié pour les problèmes WLO, inhérente aux méthodes traditionnelles comme la recherche Tabu. Nous proposons d'utiliser une méthode BO/TPE qui incorpore une fonction de perte ajustable. Nos résultats expérimentaux démontrent que notre technique est plus performante que les méthodes de l'état de l'art, WLO uniforme (UWLO) et recherche Tabu. Nous constatons également que la qualité de la solution trouvée par la recherche Tabu dépend fortement du contexte. Notre méthode est la première à s'attaquer au problème du WLO sous contrainte de ressources, et elle sert également de tremplin pour des travaux futurs qui pourraient améliorer le passage à l'échelle et l'optimalité du problème.

Contenu du manuscrit

La thèse est organisée en six chapitres. L'histoire des ordinateurs est présentée dans le chapitre 1, depuis les premiers jours de la technologie SIlicium jusqu'aux alternatives de la période du *Dark Silicon*. Dans ce chapitre, nous abordons le calcul approximatif, une des technologies émergente de l'ère post loi de Moore. La question de l'optimisation de la longueur des mots est le principal objectif de recherche de cette thèse parmi les méthodologies de calcul approximatif. Le chapitre 2 présente les fondements de l'arithmétique à virgule fixe et à virgule flottante, ainsi que le contexte du problème de l'optimisation de la longueur des mots, suivi d'une étude des

approches récentes utilisées pour résoudre ce problème de WLO. Le chapitre 3 présente notre première contribution visant à améliorer le passage à l'échelle des méthodes d'optimisation WLO pour les larges applications qui utilisent de plus des métriques de qualité complexes, telles que la similarité structurelle (SSIM) en traitement d'images. Le chapitre 4 présente notre deuxième contribution en proposant un algorithme hybride combinant l'optimisation Bayésienne (BO) et une recherche locale rapide pour accélérer la procédure WLO. Le chapitre 5 présente notre troisième contribution qui se concentre sur un autre aspect de l'optimisation de la longueur des mots lorsque des contraintes de ressources matérielles sont considérées. Il s'agit d'un problème critique pour améliorer les performances des applications sur les systèmes électroniques et informatiques dont les budgets d'énergie ou de coût sont limités. Nous proposons un algorithme basé sur BO pour résoudre ce problème. Le chapitre 6 résume les principaux résultats de cette thèse et discute des pistes d'études potentielles pour des travaux ultérieurs.

List of acronyms

AA	Affine arithmetic
AI	Artificial intelligence
ASIC	Application-specific integrated circuit
AxC	Approximate Computing
BER	Bit Error Rate
BO	Bayesian optimization
CPU	Central processing unit
DNN	Deep neural network
DSP	Digital signal processing
FPGA	Field-programmable gate array
FWL	Fractional Word Length
GP	Gaussian Processes
GPU	Graphics processing unit
GWLO	Global Word Length Optimization
IC	Integrated Circuit
IoT	Internet of Things
ISP	Image Signal Processor
IT	Information Technology
IWL	Integer Word Length
LSB	Least significant bit
LTI	Linear time-invariant system
MC	Monte-Carlo
MOSFET	Metal–oxide–semiconductor field-effect transistor
MSB	Most significant bit
MSE	Mean Square Error
MWLC	Minimum word length configuration
MWLO	Multiple Word-Length Optimization
NPU	Neural processing unit
ODG	Objective Degradation Grade
PDF	Probability density functions
PSNR	Peak signal-to-noise ratio

QoS	Quality of service
RAM	Random-access memory
SM	Stereo Matching
SoC	System on Chip
SQNR	Signal-to-Quantization-Noise Ratio
SRAM	Static random-access memory
SSIM	Structural Similarity
TPE	Tree-structured Parzen Estimator
UWLO	Uniform Word-Length Optimization
WL	Word Length
WLO	Word Length Optimization

List of Figures

1.1 50 years of microprocessor trend data 21
1.2 An example of Approximate Computing 23
1.3 Cost comparison of various operations 24

2.1 Design space with three variables (accuracy, energy consumption and performance) 33
2.2 A number represented in the floating-point format 39
2.3 A number represented in the fixed-point format 42
2.4 Floating-point to fixed-point conversion process 44
2.5 Techniques for assessing dynamic range are categorized into groups 45

3.1 All explored solutions of three WLO runs for Demosaic 74
3.2 Image Processing Pipeline used in Smart Phone Camera 77
3.3 Stereo Matching example 78
3.4 Computing blocks (kernels) of Stereo Matching. 80
3.5 Two numerical solutions 81
3.6 Two numerical solutions 82
3.7 Two numerical solutions 83
3.8 Comparison of exploration time for ISP and Stereo Matching (SM). 84
3.9 The quality of solutions by our approach compared to the best combination of the configurations used for accuracy model construction. These results are for ISP with 4 kernels. 85
3.10 Comparison of exploration time for filter applications. 86

4.1 Search process of Random search, Tabu and TPE. 94
4.2 Manually selected stopping points on the best solution obtained so far by TPE. . 95
4.3 A comparison in energy cost (top), total exploration time (middle) and separated exploration time (bottom) of Tabu and different combinations between TPE and Tabu given the selected transition points. 96
4.4 Stopping point for IIR, quality target PSNR = 40dB. 99
4.5 Energy cost of solutions (top) and exploration time (bottom) normalized to Tabu[bias 102

5.1 10^5 random solutions as a representative subset of all possible solutions, for the FIR filter benchmark. The red line represents an energy budget of 0.0075 nJ . . . 111
5.2 The classical WLO problem in the context of the cost-constrained WLO problem. 112
5.3 Exploration trajectories of Tabu search method with different starting points . . 114
5.4 Performance comparison between our approach and UWLO in terms of quality of results for different benchmarks. 119
5.5 Comparison of the normalized quality of solutions obtained by Tabu Search and our approach for different benchmarks. 120
5.6 The search direction of our approach compared with the Random Search 121

A.1 TypEx: a tool for type exploration and automatic floating-point to fixed-point conversion 131
A.2 TypEx Tool Screenshot 133
A.3 Example of histograms resulting from profiling of data values during exploration 134
A.4 General principle of the exploration algorithm 141
A.5 Example of pruning on a non-local mean (NLM) denoising kernel with two accuracy metrics (PSNR and SSIM) and nine variables 142
A.6 Example of solutions explored by the Tabu search algorithm 143
A.7 Energy model of multiplier for an FPGA target (Xilinx Virtex6, multipliers implemented in DSP blocks) as a function of input wordlength 145
A.8 Area model of an adder for a 28nm ASIC technology as a function of input wordlength 145
B.1 TypEx on Python repository 149
B.2 Exploration visualization 153

List of Tables

2.1 IEEE 754 floating-point representation types 40

3.1 Comparison of solutions for ISP and Stereo Matching (SM). 84
3.2 Comparison of solutions for FIR and IIR. 85

5.1 The benchmarks for the evaluation 118
5.2 Quality improvement of our solutions in percentage compared to UWLO solutions. Given value for each benchmark is the average of those obtained by different values of α. 118
5.3 Average quality improvement of the solutions provided by our approach over those obtained by Tabu Search (TS). The results are normalized with the range in Figure 5.5. 120

A.1 A part of data for the cost model. The data contains the synthesized result of different adders and multipliers (different input/output wordlengths). 144

Chapter 1

Introduction

1.1 Energy-Efficient Computing in Post Moore's Law

Information technology is linked to the development of modern society. Electronic computers are now prevalent in many industries, including business, education, entertainment, health, agriculture, and manufacturing industries, contributing to the overall growth in productivity and income. Continued growth in computer performance is the main factor driving innovation in IT. However, this growth is facing difficulties due to a lack of energy. Energy consumption is one of the major issues in computing today, shared by all domains in computer science, from high-performance computing to embedded systems. In particular, we are entering into the dark silicon era, where the benefits of transistor scaling are diminished by the power limitation wall, while the needs of improving the computational performance to address many problems in big data, IoT, Artificial Intelligence, and High Power Computing are a must. This situation poses challenges for scientists and leading chip manufacturing companies to find effective alternative solutions to continue to increase the performance of electronic chips in the dark silicon era.

In 1965, an American engineer, Gordon Moore, made a prediction (the well known Moore's Law), which stated that the number of transistors on a single monolithic chip doubles every two years, though the cost of computers is halved. This means that, every two years, we have a new chip generation with higher speed and performance with lower cost compared to the previous one. So far, Moore's Law has been the fundamental driving force for chip development for more than four decades. Although increasing the number of transistors guarantees performance growth, one of the main challenges of this process comes from the limitation of the power supply to the transistors.

More than three decades since 1971, Dennard's scaling theory, also known as MOSFET scale, is the main driving force behind Moore's law to provide a roadmap for the development of the manufacturing chip industry [1]. Acording to Dennard's scaling, the power density of transistors will keep to be constant as their dimensions get smaller, which allows chip manufacturers to raise clock frequencies to push the performance of the chips as high as possible without significantly increasing the overall circuit power consumption. In detail, we consider a 2-dimensional transistor. Based on Dennard's scaling rules, with each new generation created, the dimensions

of the transistor, i.e., width and length, are diminished by the factor $\alpha = \sqrt{2}$, which allows halving the area of the original transistor, and also means doubling the number of transistors in the same area. Scaling the dimensions with the factor α results in a reduction in supply voltage and capacitance by about 30% and an increase in input frequency by about 40% [1], which in turn reduces the dynamic power consumption by half, as stated by the following equation representing the dynamic power consumption of a CMOS gate

$$P_{dyn} = a \cdot C \cdot V_{dd}^2 \cdot f. \tag{1.1}$$

where a is the activity of the gate, C its output capacitance representative of the gate area, V_{dd} its power supply volage, and f its frequency. As a result, the power consumption of the transistors will decline at the same factor as their area reduces. In other words, the power consumption remains the same even if the transistor density doubles each time a new chip generation is created.

The Raise of the Dark Silicon Era

However, around 2005, Dennard's scaling law stopped. Although the transistor area continues to shrink by the traditional ratio, i.e. 1/2, to double the number of transistors in the same area, the power per transistor is no longer scaled to the same ratio. The main cause of this change comes from the transistor operating voltages (V_{dd} and also the transistor threshold voltage V_t) not being able to scale to the traditional 1/2 factor, due to the significant increase of leakage current. Indeed, leakage current increases exponentially as the threshold voltage decreases.

Additionally, due to the consequence of downsizing the transistor, leakage is also more likely in smaller devices with thinner dielectrics and shorter channels. In fact, The leakage current is the most important contributor to leakage power. Thus, lowering down the threshold operating voltage contributes to an increase in leakage power, causing an increase in the total power supply. As a consequence of this change, if the proportion of active transistors does not decrease from one technological generation to the next, the power supply needs to be increased to make up the shortfall since it cannot scale with the original rate. The failure of Dennard scaling has ushered in the era of "dark silicon" as known among chip designers. If the scaling rate of power supply for the transistor is less than that of the area, it is possible that not at all of the transistors available as a result of the scaling will be turned on and used. These non-functioning transistors are dark silicon, which accounts for a portion of the chip area. The failure of Dennard Scaling and the opening of the dark silicon era posed many challenges for the computing industry to maintain the advancement of computing performance. The lack of power used to activate all transistors disrupted the effectiveness of the techniques used to improve the performance of single-core microprocessors such as cache management, branch prediction and pipeline tech-

nique. This challenge pushes the semiconductor industry to find alternatives to further increase the performance of chips.

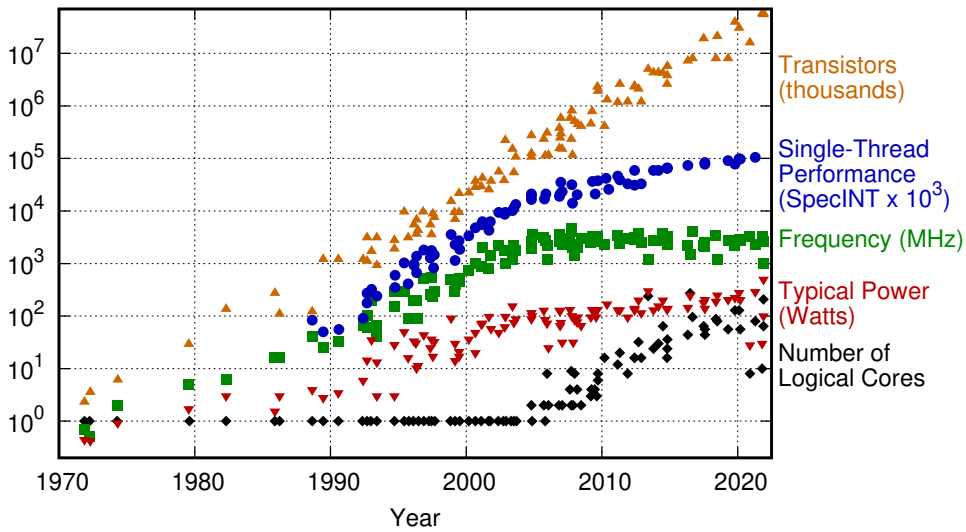


Figure 1.1: 50 years of microprocessor trend data. Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. New plot and data collected for 2010-2021 by K. Rupp

Indeed, in 2005, the era of multicore began, as shown in Figure 1.1. Some components, like specialized logic and cache memory, contribute to overall integrated circuit (IC) performance while drawing power only when necessary. As a result, the IC industry began focusing on multi-core architectures. The idea behind multicore technology is that it allows for parallel computing, which can significantly increase computing performance and efficiency by combining two or more central processing unit (CPU) cores into a single chip. The multicore chips do not necessarily run as fast as the highest performing single-core models, but they improve overall performance by handling more work in parallel. As a result, the heat and power consumption of the system are reduced thanks to managing the leakage power better. A multicore processor's design allows communication across all available cores, ensuring that processing jobs are split and assigned correctly. The processed data from each core is transmitted back to the motherboard via a single common gateway after the task is completed. When compared to a single-core CPU of comparable speed, this strategy considerably improves performance. However, one of the major challenges that are faced by multi-core processors is their memory systems. The execution of a program is often constrained by the memory bottleneck. A multicore system usually uses a 3-level cache structure, including L1, L2 and L3 caches. L1 and L2 are typically private, whereas L3 is the shared memory that synchronizes and facilitates data transmission across cores. When the number of cores increases, the amount of data transferred in L3 caches, buses, and interconnections

also increases, resulting in data traffic congestion and performance degradation.

Heterogeneous Computing

Heterogeneous computing is considered the third age of computing after the single-core and multi-core eras. In addition to breaking Moore's law, it can successfully deal with challenges like energy consumption and scalability to further improve the computing performance. Running some special tasks using GPUs or FPGAs as processing units – or accelerators–, along with standard multicores for general processing, can form heterogeneous systems aimed at speeding up program execution [2]. Indeed, the multi-core architecture is a group of identical single-core processors. This multi-core processor provides the ability to develop and exploit applications for single-cores in a homogeneous manner, which means that the ability to run applications on single cores is identical. However, over the last decade, application domains running on a single processor have become increasingly divergent in terms of functionality and resource requirements, mainly due to the much faster growth in application complexity, which limits the performance of these applications when run on a set of identical cores. In contrast, there is evidence that heterogeneous multi-core architectures are increasingly used to support the efficient processing of computationally demanding applications thanks to significant performance improvements in terms of power consumption and throughput [3, 4, 5]. Besides, heterogeneous multi-cores are perfectly suited for the dark silicon era because only the cores required for a given application must be turned on during program execution, while unused application cores are turned off to save power [6]

Approximate Computing is one of the Solutions

In recent years, along with improvements in computer architecture, computer scientists and leading chip manufacturers have been particularly interested in changing the computing methodology to further improve energy efficiency. Approximate Computing (AxC) methodology emerges as a potential solution to improve performance and energy consumption in embedded systems [7, 8, 9]. Figure 1.2 shows an example of how AxC is used to find a trade-off between the cost and the precision [1] of the representation of an image. The idea behind AxC is to relax the perfectly precise computations to an acceptable level of inaccuracy to gain in performance and/or to reduce energy consumption. This is entirely consistent with the fact that many applications in embedded systems do not require high precision/accuracy, and hardware designers often seek a good balance between accuracy, speed, energy, and area cost. The majority of these applications are not concerned with calculating an exact numerical response. Instead, "correctness" is defined as the ability to provide outcomes that are good enough, or of sufficient quality,

1. Precision is defined in this thesis as the number of bits used to represent data.

to provide an acceptable level of user satisfaction [10]. This intuitive insight opens a huge room for researchers and chip makers to rethink computing methodology for most of the computing components that span different layers: application, software, compiler, and hardware levels. Various techniques for approximate computing augment the design space by providing another set of design knobs for performance-accuracy trade-off. Recent methods concentrate on developing systematic procedures for automating the development and compilation of approximation software and hardware. The techniques are designed (1) to provide developers with a better understanding of how approximation hardware and software impact application correctness, and (2) to automate the control of application accuracy, energy consumption, and performance [11].

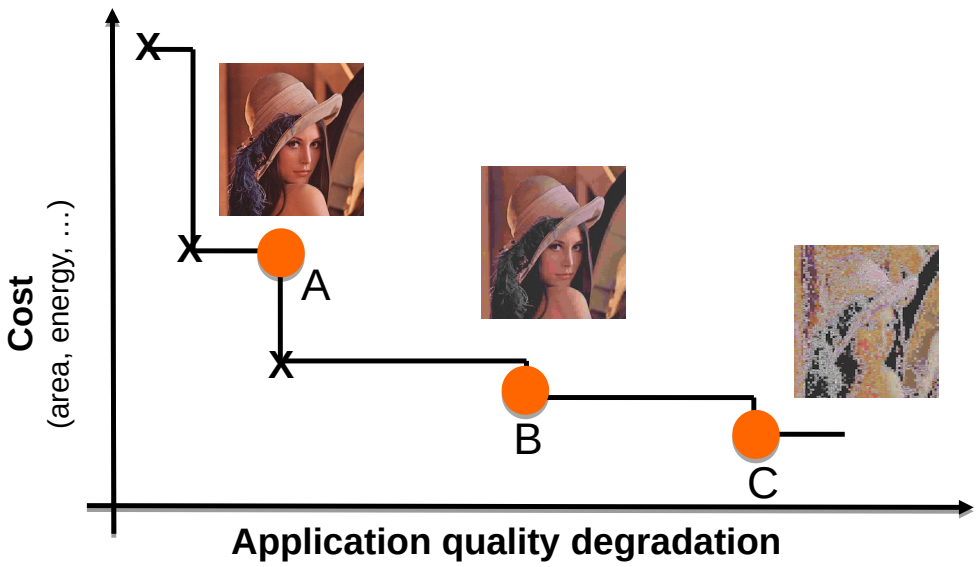


Figure 1.2: Approximate Computing example: trade off precision for gains in cost (area and/or energy) degradation. The representation A is of great quality but comes at a high cost; the representation B may be of sufficient quality at a reduced cost; and the representation C is of poor quality but has the lowest cost.

The Efficiency of Reduced Precision

Reducing precision in arithmetic representation is one of the most effective AxC methods. The standard floating-point representation provides high-precision computations with traditional single (32 bit) and double (64 bit) precision. These data types offer a high degree of accuracy in calculations, but at the expense of long execution time and high energy consumption. In recent applications, particularly in machine learning and deep learning applications, there has been a growing interest in the use of reduced-precision arithmetic. Hardware platforms for training and inference of deep neural networks (DNNs) are shifting towards 8-bit and 16-bit precision and

use fixed-point arithmetic (i.e., integer) with a significant increase in energy efficiency thanks to reducing computational costs and data storage, while ensuring the required accuracy of the model [12, 13, 14].

Besides, precision has a close relationship to area and energy consumption. Figure 1.3 [15] compares the energy and area cost of synthesized operators (adders and multipliers) with varying precision (8/16/32 bits) and number representations (float, integer) and memory read operations in SRAM and DRAM. For adders, the cost reduction rate is similar to that of bit-width shrinking. It is worth noting that when reducing the bit-width in multipliers, the cost drops dramatically. Hence, truncating the precision of arithmetic operations creates a new room to further improve energy efficiency.

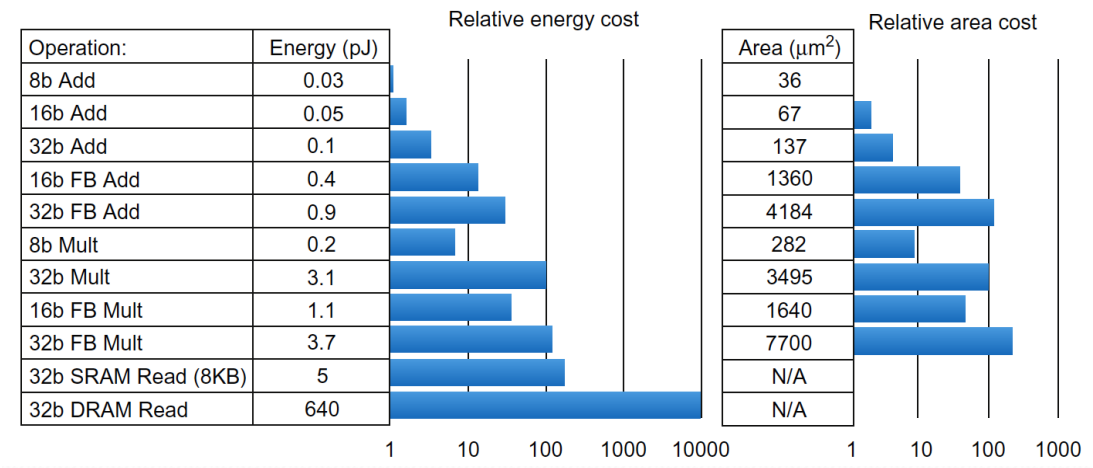


Figure 1.3: Energy numbers are from Mark Horowitz "Computing's Energy problem (and what we can do about it)", ISSCC 2014 [15]. Area numbers are from synthesized result using Design compiler under TSMC 45nm tech node. FP units used Design Ware Library

Although using reduced-precision arithmetic improves the performance and cost of embedded systems, finding an efficient way to optimize between accuracy loss and energy efficiency improvement is not an easy task. The state of the art focuses on design methodologies to efficiently automate bitwidth assignment to maximize the benefits from reducing the precision of operators. This problem is called **Word-Length Optimization** (WLO) –or **Automatic Precision Tuning**–. Word-length optimization is the process of determining suitable word-lengths for variables of a given algorithm in order to reduce the implementation cost (energy consumption and/or area) while maintaining acceptable accuracy due to data quantization.

1.2 Objective of Thesis - Application-Level Tuning of Accuracy

The thesis was initiated as a research direction of the ANR research project named AppRoximaTivE Flexible Circuits and Computing for IoT (ARTEFaCT) under the collaboration of CEA Leti, INRIA, INSA, EPFL and CSEM. The ARTEFaCT project aimed to build on the preliminary results on inexact and exact near-threshold and sub-threshold circuit design to achieve major energy consumption reductions by enabling adaptive accuracy control of applications. ARTEFaCT proposed to address, in a consistent fashion, the entire design stack, from physical hardware design, up to software application analysis, compiler optimizations, and dynamic energy management. Combining sub-near-threshold with inexact circuits on the hardware side and, in addition, extending this with intelligent and adaptive power management on the software side was expected to produce significant results in terms of energy reduction, i.e., at least one order of magnitude, in IoT applications. The project has contributed along three research directions:

1. Approximate, ultra low-power circuit design.
2. Modeling and analysis of variable levels of computation precision in applications.
3. Accuracy-energy trade-offs in software.

The thesis contributes to the third research direction, accuracy-energy trade-offs in software. In this research, we focus on applying AxC techniques at the application level of design. This thesis is about developing methods for systematic exploration of the design space, including implementation cost/quality of service (QoS) modeling and automation of designs. This allows us to speed up the optimal or near-optimal design selection that (1) minimizes cost (area and/or energy consumption) while still satisfying required accuracy at the application output or (2) maximizes the accuracy at the application output under a cost budget. We target emerging System on Chip platforms (Xilinx Zynq or Intel SoC) that feature FPGAs tightly coupled with embedded processors.

This thesis is addressing WLO problem with several aspects including the scalability and resource-constrained optimization. Fixed-point data types are considered for WLO throughout studies in the thesis thanks to the wide use in the implementation of Digital Signal Processing algorithms in ASIC and FPGA platforms.

1.3 Thesis Organization

The thesis is organized in six chapters.

This chapter has introduced the historical development of computers, from the beginnings of MOSFET technology when Moore made his prediction about the increase in transistor performance, to the alternatives in Dark Silicon era. In this chapter, we also discussed Approximate

Computing, one of the emerging solutions in the post-Moore's law era. Among approaches in Approximate Computing, the Word Length Optimization problem is the main research objective in this dissertation.

Chapter 2 gives a fundamental background of arithmetic number representations, including Fixed-Point and Floating-Point arithmetic, as well as the context for the Word Length Optimization problem, followed by a survey of recent approaches being used to address WLO.

Chapter 3 presents our proposed method to improve the scalability of Word-Length Optimization (WLO) for large applications that use complex quality metrics such as Structural Similarity (SSIM). In this approach, the input application is decomposed into smaller kernels to avoid an uncontrolled explosion of the exploration time, which is known as noise budgeting. The main challenge addressed in this research work is how to allocate noise budgets to each kernel. This requires capturing the interactions across kernels. The main idea is to characterize the impact of approximating each kernel on accuracy/cost through simulation and regression. Our approach improves the scalability while finding better solutions for an Image Signal Processor pipeline.

Chapter 4 presents our proposed hybrid algorithm combining Bayesian Optimization (BO) and a fast local search to speed up the WLO procedure. In this chapter, through experiments, we first show some evidence on how this combination can improve exploration time. Then, we propose an algorithm to automatically determine a reasonable transition point between the two algorithms. By statistically analyzing the convergence of the probabilistic models constructed during BO, we derive a stopping condition that determines when to switch to the local search phase. Experimental results indicate that our algorithm can reduce exploration time by up to $50\% - 80\%$ for large benchmarks.

Chapter 5 focuses on another aspect of Word Length Optimization with hardware resource constraints. It is worth noting that state-of-the-art approaches mainly solve WLO given a quality constraint. There is no research on enhancing quality of service under a cost budget. This is a critical problem in improving the performance of applications in electronic devices with a limited energy or cost budget. In this chapter, we first show challenging issues of classical methods in the WLO problem with resource constrains. Then, we propose a Bayesian-Optimization-based algorithm to maximize the accuracy of computations under some cost constraint. Experimental results indicate that our approach outperforms the classical approach in terms of the obtained solution quality.

Chapter 6 summarizes the main results of this thesis and discusses potential next study paths for further work.

Bibliography

[1] M. Bohr, "A 30 year retrospective on dennard's mosfet scaling paper," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, no. 1, pp. 11–13, 2007.

[2] T. Ju, Z. Zhu, Y. Wang, L. Li, and X. Dong, "Thread mapping and parallel optimization for mic heterogeneous parallel systems," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2014, pp. 300–311.

[3] T. Mitra, "Heterogeneous multi-core architectures," *Information and Media Technologies*, vol. 10, no. 3, pp. 383–394, 2015.

[4] D. Shin, J. Lee, J. Lee, J. Lee, and H.-J. Yoo, "Dnpu: An energy-efficient deep-learning processor with heterogeneous multi-core architecture," *IEEE Micro*, vol. 38, no. 5, pp. 85–93, 2018.

[5] H.-E. Zahaf, G. Lipari, M. Bertogna, and P. Boulet, "The parallel multi-mode digraph task model for energy-aware real-time heterogeneous multi-core systems," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1511–1524, 2019.

[6] J. Carabaño, F. Dios, M. Daneshtalab, and M. Ebrahimi, "An exploration of heterogeneous systems," in *2013 8th international workshop on reconfigurable and communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2013, pp. 1–7.

[7] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, and S. Tomov, "Accelerating scientific computations with mixed precision algorithms," *Computer Physics Communications*, vol. 180, no. 12, pp. 2526–2533, 2009.

[8] B. Barrois, O. Sentieys, and D. Menard, "The hidden cost of functional approximation against careful data sizing—a case study," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 181–186.

[9] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys, "Pushing the limits of voltage overscaling for error-resilient applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 476–481.

[10] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[11] A. Filieri, M. Kwiatkowska, S. Misailovic, and T. Mytkowicz, "Approximate and probabilistic computing: Design, coding, verification (dagstuhl seminar 15491)," in *Dagstuhl Reports*, vol. 5, no. 11. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[12] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[13] D. A. Gudovskiy and L. Rigazio, "Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks," *arXiv preprint arXiv:1706.02393*, 2017.

[14] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," *Advances in neural information processing systems*, vol. 31, 2018.

[15] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

Chapter 2

Theoretical Background and Related Work

In this chapter, we survey and classify recent approximate computing techniques by the level of design. Then, we focus on presenting arithmetic number representations, including Floating-Point (FlP) and Fixed-Point (FxP). After that, we provide the fundamental background and a survey of recent optimization techniques for the Word-Length Optimization (WLO) problem, the central theoretical background of this thesis. The primary optimization approaches are examined and discussed, as well as the advantages and disadvantages of existing approaches. Methods being used to model the cost/accuracy trade-offs during WLO are also discussed in this chapter.

2.1 Approximate Computing

Energy consumption is one of the important metrics in designing embedded systems. In many complex systems, such as scientific computing, social networking, and financial analysis. Improving the system performance by adding more features or replacing old functions with more advanced ones, sometimes comes at the cost of increasing the energy supply for the system. Besides, with an increasing amount of data being generated, queried, stored, accessed, and computed by the applications running in these systems, their energy consumption has increased significantly. Considering mobile devices with limited energy budget, such as coin cells or even low-power battery sources, the energy consumption of computer systems becomes a crucial metric.

Approximate Computing (AxC) is an emerging technique in designing embedded systems for energy-efficiency improvement, i.e., enhancing the performance and reducing the area and power consumption. AxC seeks to minimize computing accuracy, in order to achieve large computational performance increase and energy consumption reduction, while ensuring an acceptable quality of service (QoS). Approximate computing provides an alternative paradigm for analyzing large datasets by simulating "what-if" scenarios for entering parameters that capture the necessary trade-off between accuracy and performance. To find the appropriate design that seeks for the golden balance between accuracy and performance, various AxC techniques have been

proposed at various design levels, i.e., system-, algorithmic-, and circuit-level.

Error-tolerant applications relating to human perception and cognition drive the majority of approximate computing. As human beings, our perception and cognition are not perfect, which implies that many computation tasks can be done with certain imperfections or approximations. Multimedia processing, machine learning, signal processing, and scientific computing are examples of such application areas where approximate computing has been used [1]. In recent years, the research focus of the approximate computing community has expanded from fast approximate search algorithms (e.g., low-complexity signal and image processing) to wide applications in big data (e.g., complex analytics in web search, recommendation systems), and real-time/embedded applications (e.g., sensor networks). Numerous of these applications rely on statistical or probabilistic computations, such as the ability to make alternative approximations to better fit the intended goals [1]. These applications aim to provide means for developing such high-level abstractions by formulating approximation problems as parameterized data analysis tasks and proposing concrete methods to solve them. Examples include models for language understanding (such as latent semantic analysis), machine translation, object recognition, medical diagnosis, and DNA sequencing.

However, designing an efficient and accurate approximate computing techniques involves understanding multiple factors, such as the environment conditions, computing load and resource availability. The task of finding an optimal point on the accuracy-performance trade-off axis remains quite challenging in realistic applications. Due to the large number of parameters, it is non-trivial to select an appropriate subset of parameters, and this selection process will be quite time consuming. Hence, it is important to develop effective techniques that speed up the process of selecting the suitable design point.

2.1.1 Approximate Computing for Error-Resilient Applications

AxC techniques can be successfully applied to a variety of applications to reduce computational effort thanks to the intrinsic error tolerance properties of those applications. This allows us to simplify the computational elements of the program so that the errors they cause are tolerable. The error tolerance property is defined as the capacity of a program to generate acceptable results despite the fact that some of its underlying computations are faulty or inaccurate [2]. For example, AxC methods can be used in sound and image processing since little sound distortions or slightly discolored pixels go undetected [3]. There are three factors that contribute to the inherent resiliency of applications [2, 4]:

1. The inputs: These applications are designed to handle noisy and redundant input data. For example, the input to the digital image processing algorithms already includes noise or

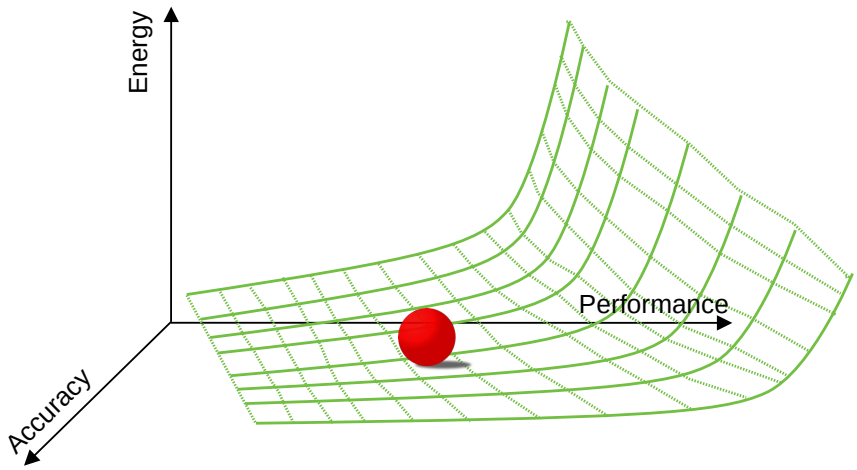


Figure 2.1: An example of design space that depends on three variables (accuracy, energy consumption and performance) for applications designed with AxC techniques. The third dimension, i.e., accuracy, opens a larger room for design optimization to further gains in energy efficiency (i.e., performance per Watt or energy per operations). The red point represents for the design flexibility in three dimensions.

quantization errors that come from digitizing analog data, data being also often sampled with noise.

2. The computations: The applications commonly utilize computing techniques like aggregation or iterative refinement to mitigate or repair approximation effects.

3. The outputs: The application of the concept that there is no such thing as a "golden" result, but rather a range of acceptable outputs or slight changes in the output that are undetectable by users. For example, a web search system can return similar results. Similarly, due to human limits in perception, occasional frame drops in video applications might go unnoticed.

Thanks to the error-tolerance property of those applications, we can adjust the accuracy of the application in a controlled manner to an acceptable level, which allows us to create a new level of design flexibility, which trade-offs the accuracy with traditional design metrics such as computational performance (number of computational units per second) and implementation cost (area utilization and/or energy consumption) as shown in Figure 2.1.

The use of AxC techniques in embedded system design leads to an optimization problem where the designers need to find a method to select an optimal or near-optimal solution from a design space exploration constructed by a variety of solutions with different parameters (performance, accuracy, and energy). With knowledge about components of the embedded system, the designers can create models that evaluate the interactions among these three parameters and

then formulate the optimization problem constrained by user-defined requirements for performance, energy and accuracy metrics. The optimization problem results in an estimated energy, performance and accuracy that can be compared to the specified constraints, enabling the selection of an optimal (or at least good enough) system design configuration. The selection of the solution using AxC techniques is based on a Pareto optimal design space, which includes a subset of solutions that achieve the highest performance level within a range of accuracy and energy cost. A properly designed AxC technique can improve the energy efficiency of the embedded application. It also helps designers to quickly obtain good design choices and hence reduce the development time, and eventually the time to market of electronic devices.

2.1.2 Cross-Layer Approximate Computing Techniques

Energy consumption is one of key design metrics in battery-operated embedded system. The energy consumption directly relates to dynamic power dissipation P_{dyn} which is calculated by Equation 1.1 through the product of switching activity factor (α), load capacitance (C_L), supply voltage (V_{dd}) and clock frequency (f_{clk}). The reduction in energy consumption is essentially the reduction of individual ingredients Equation 1.1. From that, many AxC techniques were proposed to reduce energy consumption at different layers ranging from algorithm level down to circuit level. In this thesis, we divide techniques into three categories: software, architecture, and hardware. The mentioned methods are the most frequently used approximation approaches in the state of the art. This survey does not include newer or less commonly used methods. It is also worth noting that some methods are shared across layers.

Approximate Computing at the Software Level

Some frameworks allow programmers to identify the impact of approximation in specific sections of code, giving them control over the system's accuracy at the software level. Green [5] is a framework that enables programmers to take advantage of AxC in a systematic way, while also ensuring statistical QoS. It allows for complex functions and loops to be approximated. Green develops a model that assesses the impact of approximation on QoS loss, allowing approximation decisions to be made based on the programmer's QoS constraints.

Sampson *et al.* proposed EnerJ [6] that uses type qualifiers to declare data signals to be expressed in an approximate manner in a system. The storage, computation, and algorithm components utilized for the variables designated with the approximation qualifier can be approximate. The system can ensure that the precise and approximate program components are isolated statically.

Some works [7, 6] use *Loop Perforation*, which systematically skips iterations of the loops without significant quality degradation to reduce the volume of processing in an application. This

can reduce directly the load capacitance that is equivalent to the logic circuit. The consideration of the number representation is also one of efficient approaches to reduce the energy consumption.

Task skipping is an approximation approach that enables code blocks to be skipped based on a specified run-time boolean condition [8]. If a statement block is executed when the runtime boolean condition fails, then the remaining code blocks in that control structure will not be executed. Typically, this method is used for the most computationally demanding portions of the code. In many scenarios, instead of running the loop to its completion, the paradigm may instruct a hardware system to process less data in order to save power and computing time. Some related existing works can be found in [9, 10, 11, 12].

Data sampling [13, 9] is also considered an AxC technique at the software level. This is a method in which the incoming data are not analyzed in their whole, but rather a selection is made before processing. This method decreases the computation complexity, the execution time and the computing power required, while providing acceptable accuracy of the result.

Approximate Computing at the Architecture Level

Some works exploit the room for approximation in instruction-set architecture (ISA) [14] and memories [15, 16]. Hadi Esmaeilzadeh *et al.* [14] proposed ISA extensions that define a set of special instructions that allow the compiler to convey what can be approximated. Truffle, a micro-architecture concept, was suggested to implement the ISA expansions effectively. Dual-voltage operation is the basis of the proposed design, with a high voltage for exact operations and a low voltage for approximate operations. The micro-architecture's reliance on the instruction stream to know whether to employ the low voltage is a significant feature. The paper [17] further proposed a low-power accelerator based on a neural processing unit (NPU) that can perform approximately compute-intensive code segments. The NPU is tightly coupled to the processor pipeline to speedup code regions. It is quicker and more energy efficient to offload approximable code areas to NPUs than to execute the original code.

Sampson *et al.* [16] proposed techniques that allow applications to approximate data storage, improving the performance, longevity, and density of solid-state memory. There are two mechanisms in operation. The first reduces the amount of programming pulses necessary to write multi-level cells, allowing for mistakes. By transferring approximation data into blocks that have exhausted hardware error correcting capabilities, the second approach reduces wear-out failures and increases memory endurance.

Flikker, an application-level approach for reducing refresh power in DRAM memory, is introduced in [15]. Flikker allows programmers to designate critical and non-critical data in their code, and the runtime system allocates this data to different memory locations. Critical data are updated at the standard refresh rate, whereas non-critical data are renewed at much lower rates. This partitioning saves energy at the expense of a little increase in non-critical data cor-

ruption. As a result, Flikker reveals and exploits a fascinating trade-off between energy usage and hardware accuracy.

Approximate Computing at the Hardware Level

Several AxC approaches minimize storage and computation costs by modifying the precision [1] of input or intermediate operands. Many works [18, 19, 20] use low-precision, efficient number representation, such as fixed-point arithmetic, to reduce the energy consumption. These works address the **Word Length Optimization (WLO)** problem to optimize fixed-point bit-widths of variables in the application to reduce the energy consumption with an acceptable quality at the application output. Indeed, using a shorter fixed-point word-length reduces the switching activity factor and the load capacitance since the operators compute with less bits. The problem of WLO to optimize fixed-point bit-widths of variables in the application is the main objective of this thesis and will be further detailed later in the chapter.

The bfloat16 (Brain Floating-Point) was developed by Google Brain, the company's artificial intelligence research group. Bfloat16 is a shortened version of the 32-bit IEEE 754 single-precision floating-point standard (binary32) designed to speed up machine learning and deep learning tasks. It uses the same 8 exponent bits as the 32-bit floating point standard, but it only supports an 8-bit precision instead of the 24-bit significand of the binary32 format [21]. Experiments conducted by Google have shown that it is feasible to shrink the mantissa as long as it can still represent tiny values closer to zero as part of the summation of minor differences during the training phase. The benefit of this change is the reduction of power consumption and implementation silicon area of the computation operator and memory storage [22]. Bfloat16 is used in Intel AI processors (Nervana NNP-L1000), Intel FPGAs, Google Cloud TPUs, and TensorFlow [21].

Inexact approximate arithmetic operators, such as adders and multipliers, have recently received a lot of attention as a way to enhance speed and energy efficiency with a minimal loss in accuracy for compute-intensive image and video processing and machine learning applications [23, 24, 25, 26]. Exploiting the error-tolerant characteristics of those applications, several approximation methodologies have been proposed for the implementation of inaccurate arithmetic units. Adder and multiplier being the fundamental operations of an arithmetic processor, they are crucial to obtain high performance.

Inexact adder has been explored extensively for approximation computation in order to reduce power consumption and latency [27, 28, 29, 30]. Liang *et al.* [31] have examined several approximate adders, using different metrics such as error distance (ED), mean error distance

1. or bit-width, word-length is also used as a synonym of precision in this thesis

(MED), and normalized error distance (NED), to evaluate approximate and probabilistic adders in approximation computing applications. Existing designs for an approximation multiplier can be divided into truncation and non-truncation methods. For the truncation-based design, the bottom portion of the partial products is eliminated, or the least significant partial products are approximated by a constant [32]. To improve the accuracy of truncated multipliers, many error compensation schemes have been developed [33, 34, 35, 36, 37]. A non-truncation technique assembles an approximate multiplier from approximate circuits [38, 39, 40, 41].

Dynamic-Voltage-Frequency-Scaling (DVFS) is a circuit-level approach that allows for a dynamic trade-off between energy consumption and computational precision. DVFS approaches enable the system to adjust the frequency and supply voltage to specific computer components. Many system components, including CPU cores, memory systems, last level caches, and interconnects, have exhibited considerable power and energy reductions when using DVFS [42]. In heterogeneous embedded systems and multi-processor systems, several investigations combine DVFS with machine learning methods such as regression and reinforcement learning to automatically develop a DVFS scheme that optimizes dynamic energy savings within an allowed level of performance deterioration [43, 44, 45, 46, 47]

2.2 Binary Arithmetic Number Representations

Digital signal processing (DSP) systems are critical for real-time processing of digitized data in the real world, providing high-performance numeric computations for a wide range of applications, from simple consumer electronics to complex industrial systems. DSP systems consist of software and hardware components. The software part provides the programming flexibility and modifiability. In the hardware part, the arithmetic units and memory units required for the computations are built for the DSP system. Many factors are taken into account while designing hardware to fulfill system output requirements, including latency, cost (area and energy consumption), and Quality of Service (QoS). Aside from choosing a hardware architecture, number representation is critical in hardware design to meet system requirements.

In DSP systems, the data are stored and processed through a specific arithmetic number representation system. Arithmetic is a discipline of mathematics that deals with number representation and numerical calculation. Arithmetic provides number representation systems such as floating point and fixed-point to represent integer or real values in digital signal processing algorithms. Suppose x and y are two numbers represented using a given arithmetic number representation system. It is usual in digital systems to perform the four basic arithmetic operations: addition ($s = x + y$), subtraction ($d = x - y$), multiplication ($p = y \times x$), and division ($q = x/y$).

Due to the limitations in computer memory and processing units, arithmetic operations on numbers using an arithmetic number representation system can cause round-off errors and overflows in the system operation. Round-off error is the difference between the output generated by an algorithm that uses precise arithmetic and the result produced by the same method that uses finite precision. Overflow occurs when the result produced by an arithmetic operation exceeds the range of results allowed by that number representation system. These errors or overflows may affect the stability of the system as well as the precision of the results generated by the system. Hence, the choice of number representation for a DSP system is closely related to the performance, precision and implementation cost of the whole system.

On modern computers, the native hardware mainly supports binary number systems other than octal, decimal, and hexadecimal number systems thanks to the optimization in storage and computation costs of the binary number systems. Floating-point (FlP) and fixed-point (FxP) are two basic types of binary arithmetic representation used in DSP systems or in many other computing systems. Floating-point arithmetic allows representing real numbers in a large range of values and is thus suitable for systems with very small and very large values. Popular programming languages like C/C++, Python, Matlab, etc., use the floating-point format as the major datatype to represent values of arithmetic operations. *Double* and *Float* are typical floating-point datatype being used in these programming languages. Meanwhile, for DSP systems where speed and implementation cost are more important than precision, the fixed-point datatype is often more widely used to represent real numbers. Depending on QoS and latency requirements, bit lengths commonly used in DSP applications are considered in 8/16/32 bits.

In floating-point and fixed-point representations, a basic digital unit, called a bit, can only have either logic "0" or logic "1". A number of bits can be joined together to represent a real number. The number of bits in a binary number is referred to as its width or precision. The greater the width of a number, the more accurate its computation will be. However, the greater the amount of bits in a number, the longer it will take to compute. Therefore, the width of a number must be chosen carefully to strike a balance between speed and accuracy. The unsigned and signed number representation are also supported in these two arithmetic number representations. Signed numbers can distinguish between negative and positive values by using a sign flag. Unsigned numbers, on the other hand, only store positive numbers and not negative ones. The *two's complement* system is usually used to represent the signed numbers. In the two's complement system, when the most significant bit (MSB) is a one, the number is signed as negative. Likewise, if the MSB is equal to zero, then the number is positive.

2.2.1 Floating-Point Arithmetic

Floating-Point (FlP) is a popular number representation to encode real numbers as a string of digits. FlP is widely used in DSP systems where data representation with high dynamic range is required. In general, an FlP number is represented approximately with a fixed number of *mantissa* digits (or *significand*) and scaled by a factor specified by a limited precision *exponent* in a given *base*. The base can be base-2, base-10, and base-16. In general, an FlP number is represented as

$$mantissa \times base^{exponent} \tag{2.1}$$

The term "floating-point" refers to the fact that the radix point can be placed anywhere between the significant digits. For example, with the radix-10, 123.45 can be written as 1.2345×10^2 or 0.12345×10^3. Likewise, with the radix-2, 11001.11 is equivalent with 1.100111×2^4. It is also worth mentioning that the floating-point system differs from the fixed-point system (discussed later) in being able to change the radix point to increase the dynamic range of the representation.

Computers have employed several floating-point representations throughout the years. Most CPUs now employ the IEEE 754 Standard for Floating-Point Arithmetic in base-2, which was standardized by the IEEE in 1985 and revised in 2008 [48]. This floating-point standard establishes protocols for representing floating-point numbers, managing subnormal numbers, and handling various circumstances, ensuring high portability of computing software.

Figure 2.2 illustrates the layout of a floating point number. Let x be a real signed number represented in floating-point with a base-2. The floating-point approximation $flp(x)$ of a given

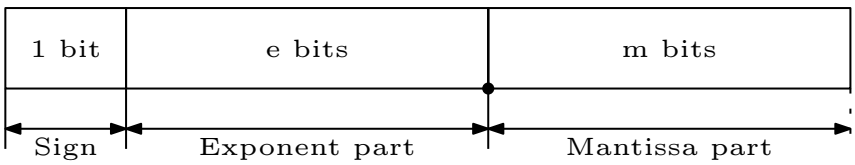


Figure 2.2: A number represented in the floating-point format

real number x is represented as

$$flp(x) = (-1)^s \times M \times 2^E. \tag{2.2}$$

An FlP number is encoded with one sign bit s, m bits of mantissa magnitude M (non-negative) and e bits of exponent E. With this format, every real number can be encoded using $1 + m + e$ bits. The types of the IEEE 754 binary formats currently in use in CPUs are detailed in table 2.1.

Table 2.1: IEEE 754 floating-point representation types

Type	Sign (s)	Mantissa part (m)	Exponent part (e)	Total bits
Half precision	1	5	10	16
Single precision	1	8	23	32
Double precision	1	11	52	64
Quadruple precision	1	15	112	128

The mantissa magnitude is normalized to the range $[1, 2[$. Using IEEE binary exchange protocols, the leading one-bit mantissa magnitude is not saved in the computer data. It is known as the "implicit" part. In fact, the m bits only store the fractional part of the mantissa magnitude M. The mantissa is then represented as

$$mantissa = (-1)^s \times M = (-1)^s \times 1.m \tag{2.3}$$

The exponent E is a signed integer number and represented by e bits. The value of E is in the range $[-2^{e-1} + 1, 2^{e-1}]$.

Floating-Point Exceptions

The IEEE floating-point standard provides a number of exceptions that might arise when the outcome of a floating point operation is ambiguous or undesirable. In IEEE 754 arithmetic, an exception might be reported alongside the result. This can be a status flag. When an exception's trapping is enabled, an error is alerted anytime the exception happens [49, 50]. The possible floating-point exceptions include: **invalid**, **divide-by-zero**, **overflow**, **underflow** and **inexact**

invalid: This exception is raised if the given operands are invalid for the operation to be performed. If trapping is enabled, the *extensions:floating-point-invalid* condition is signaled. Otherwise, the result of the operation is NaN. For example, $(+\infty) - (+\infty)$, $0/0$, $0 \times \infty$, ∞/∞, $\sqrt{-1}$.

divide-by-zero: This exception is thrown when a finite nonzero number is divided by zero. If trapping is enabled, the *divide-by-zero* condition is signaled. Otherwise, the appropriate infinity is returned. For example, $1/0$, $log(+0)$

overflow: This exception is raised when the rounded result with an unbounded exponent range has an exponent greater than 2^{e-1}. If trapping is enabled, the *floating-point-overflow* exception is signaled. Otherwise, the result relies on the sign of the intermediate result and the rounding mode. When the **overflow** exception is thrown, the **inexact** exception is thrown as well.

underflow: This exception is thrown when an intermediate result is too little to be computed precisely, or when the operation's result is too small to be normalized properly. If trapping is enabled, the *floating-point-underflow* condition is signaled. Otherwise, the operation returns the result in a *denormalized float* or *zero*.

inexact: This exception is signaled when the result of a floating-point operation is not exact, i.e., the result was rounded. If trapping is enabled, the *extensions:floating-point-inexact* condition is signaled. Otherwise, the rounded result is returned.

Rounding modes

When an operation on floating-point numbers produces a result that cannot be represented precisely in the chosen floating-point system, it needs to be rounded [49]. The rounding modes for a number x are:

round toward $-\infty$: denoted by $RD(x)$, returns the largest machine number (possibly $-\infty$) that is less than or equal to x;

round toward $+\infty$: denoted by $RU(x)$, returns the smallest floating-point number (possibly $+\infty$) greater than or equal to x;

round toward zero: denoted by $RZ(x)$, returns the closest floating-point number to x that is equivalent to $RD(x)$ if $x \geq 0$ and to $RU(x)$ if $x \leq 0$;

round to nearest: denoted by $RN(x)$, returns the closest floating-point number to x. A rule must be determined when x is precisely halfway between two successive floating-point values. *Round to nearest even* is a popular tie-breaking rule: x is rounded to the closest even integer significand of these two successive floating-point integers.

2.2.2 Fixed-Point Arithmetic

Fixed-point (FxP) arithmetic is a number representation to encode real numbers in binary number systems. A number represented in fixed-point arithmetic contains integer and fractional word-lengths (WLs), represented on m and n bits, respectively, as shown in Figure 2.3. The integer and fractional parts are separated by a binary point. The term "fixed-point" refers to a number of digits before and after the binary point that is fixed in the representation and during the computations, as opposed to the floating-point format.

A fixed-point encoding can express both unsigned and signed numbers. The unsigned numbers can only represent the values that are greater than or equal to zero; the signed numbers can represent both positive and negative values. For unsigned number, all the m bits are used to represent the integer part of this number. Whereas, for the signed number using the two's complement representation, the most significant bit (MSB) in the integer part, b_{m-1}, is reserved for the sign, and the remaining bits are used for the integer number. The unsigned and signed

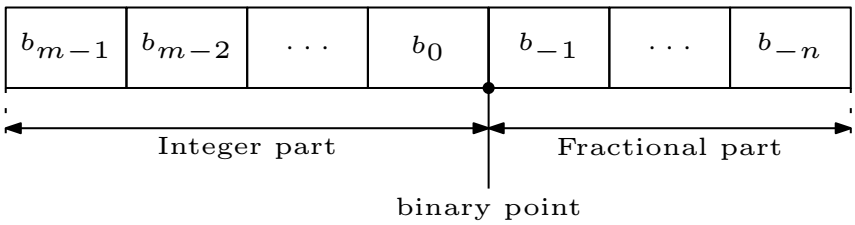


Figure 2.3: A number represented in the fixed-point format

FxP numbers are converted to decimal values $x_{unsigned}$ and x_{signed} using Equations 2.4 and 2.5, respectively.

$$x_{unsigned} = \sum_{i=-n}^{m-1} b_i 2^i \tag{2.4}$$

$$x_{signed} = -2^{m-1} + \sum_{i=-n}^{m-2} b_i 2^i \tag{2.5}$$

The choice of m relates to the dynamic range, while n provides the accuracy of each representation. Unsigned and signed numbers are represented in the ranges $[0, 2^m - 2^{-n}]$ and $[-2^{m-1}, 2^{m-1} - 2^{-n}]$, respectively. The ranges must be determined before converting a real number to the fixed-point number so that the fixed-point representation can be chosen with enough number of bits to avoid overflow and underflow errors. m is also known as the integer word-length (IWL).

The term $q = 2^{-n}$ serves as the smallest resolution (known as the quantization step q) of a number determined by the fractional word-length (FWL) n of the representation. Numbers with the increment smaller than the quantization step q cause quantization error (or round-off error), which refers to the difference between the value of the real and fixed-point numbers. A fixed-point number with a higher precision results in a lower quantization error.

2.3 Fixed-Point Conversion Process

When designing low-power embedded CPUs, floating-point is not always a good choice, and the fixed-point representation is preferred instead. It is thus necessary to perform a **floating-point to fixed-point conversion** when applications are developed in the form of a code relying on floating-point data types. In this conversion, floating-point data and arithmetic operations will be replaced by fixed-point formats with suitable word-lengths. Then, the goal of the fixed-point word-length determination process (the WLO process) is to pick the most cost-effective set of word-lengths while restricting the accuracy degradation to a level acceptable to the ap-

plication at hand. This process is known as the Fixed-Point Conversion Process [51].

As discussed in Section 2.2, a fixed-point number is composed of an integer and a fractional part. Each floating-point data must be converted to a fixed-point format to determine the number of bits needed to represent its integer and fractional parts. Overflows and quantizations may occur during the process of determining the integer and fractional s. Especially, a fatal error may occur while determining the integer part, if this specified integer word-length does not adequately cover the dynamic range of the data, resulting in an overflow error. Meanwhile, when selecting the fractional part of the data, fraction truncation and rounding errors can occur, causing considerable quantization errors. It is therefore important to control the choice of the s in order to ensure that the computation accuracy stays within an acceptable level as required by the application specifications.

Given an application with N variables in the floating-point format. The floating-point to fixed-point conversion is the process of determining the fixed-point W_i for each variable in the application, where i is integer numbers in the range $[0, N-1]$, under some accuracy constraints. Let W_i^{int} and W_i^{frac} be the integer word-length and fractional word-length of the ith variable, respectively. The total word-length assigned to the ith variable is

$$W_i = W_i^{int} + W_i^{frac}, \quad i = [0, N-1]. \tag{2.6}$$

The objective of this floating-point to fixed-point conversion (*float-to-fixed*) is to minimize the cost ($\boldsymbol{C}$) of the application. This cost can be, e.g., area or energy. The word-length of the variables has a direct impact on the application's implementation cost. Therefore, to minimize this cost, the word-length of each variable should be optimized to its appropriate minimal value. This process requires optimizing both integer and fractional s, i.e., W_i^{int} and W_i^{frac}, respectively. When the number of bits is reduced, there are unavoidable errors between the values with finite precision and the ones with infinite precision, which results in a drop in the quality of the application output as a result. Hence, the implementation cost minimization through word-length reduction should be done in such a manner that the output quality is guaranteed with a quality constraint, $\boldsymbol{\lambda_{obj}}$. In this context, an optimization criterion for the output quality is introduced to determine the required bits W_i of variable i, taking into account both the implementation cost and the quality constraint. The WLO problem is therefore defined as

$$\min(\boldsymbol{C}(\boldsymbol{W})) \quad \text{subject to} \quad \boldsymbol{\lambda} \leq \boldsymbol{\lambda_{obj}}, \tag{2.7}$$

where $\boldsymbol{W} = [W_0, W_1, \ldots, W_i, \ldots, W_{N-1}]$ is a word-length configuration vector that contains the s of the N variables in the application, $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are functions that express cost (area or energy

consumption) and output quality, respectively, and the target quality is given as $\boldsymbol{\lambda_{obj}}$.

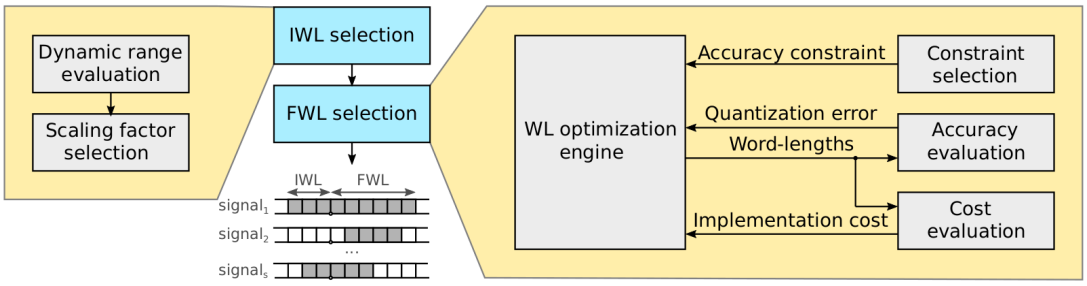


Figure 2.4: Floating-point to fixed-point conversion process [52]

Figure 2.4 depicts the process of floating-point to fixed-point conversion, which is divided into two main steps: the Integer Word Length (IWL) determination and the Fractional Word Length (FWL) determination. The IWL determination examines the dynamic value range of each variable and intermediate result and determines the minimal IWL necessary to represent each range's variable. Indeed, overflows arise when W_i^{int} is smaller than the word-length that can cover the whole dynamic range of the value, resulting in non-linearity in the computation and a large amplitude of the error as compared to infinite precision. The goal of the IWL determination is to maintain the low non-linearity, which implies that the potential error is small even though the computation takes place in finite precision arithmetic.

Generally, the IWL determination consists of two steps: *dynamic range evaluation* and *scaling factor selection*. In the dynamic range evaluation step, analytical or simulation-based methods can be used to estimate the upper and lower bounds of the dynamic range that each signal/variable represents. In certain cases, a dynamic range distribution of each signal can be obtained through this step and used to consider truncating values with a low occurrence probability of happening and not have a serious impact on the accuracy of the application by reducing integer word-lengths. Once the bounds are determined, An appropriate scaling factor is selected to achieve a certain number of accurate digits in a computable range without important non-linearity. Then, a scaling factor calculation is applied to align fixed-point formats using shift operation.

The FWL determination involves adjusting the fractional bit-width so that the accuracy loss of the application output is within an acceptable level. FWL optimization is an NP-Hard problem, which can take up to 25−50% of the whole design time of a DSP system [53]. Optimizing fractional bit-width causes quantization errors. Quantization errors are relatively small, but they have the ability to propagate throughout the computer system, reducing progressively the accuracy of the system output as a result. The FWL determination is an optimization process for fractional word-lengths to find an appropriate balance between cost and accuracy of the

system output. An FWL determination solver may include three ingredients:

— The *WL optimization engine* is basically an optimization solver to optimize fractional word-lengths given inputs from other components.

— The *cost evaluation* provides an estimated cost (e.g., implementation area or energy consumption) given the current fractional word-length configuration of the WLO optimization engine. Usually, cost models will be used for this component instead of the exact cost obtained through synthesizing all systems.

— The *accuracy evaluation engine* measures the accuracy loss due to the quantization error caused by the current fractional configuration. Both analytical and simulation-based approaches can be considered for accuracy evaluation.

2.3.1 Integer Word Length (IWL) Selection

The first step of the fixed-point conversion method is to determine the integer of each variable in the target application. The objective of this step is to keep integer word-length as short as possible to reduce the implementation cost, while safeguarding against overflows that severely decrease application quality. The number of integer bits of each variable depends on its dynamic range, hence the goal is to estimate the probability density function (PDF) of x that is associated with the application functions and the input signal distribution. Recent techniques to determine the dynamic range are classified in Figure 2.5.

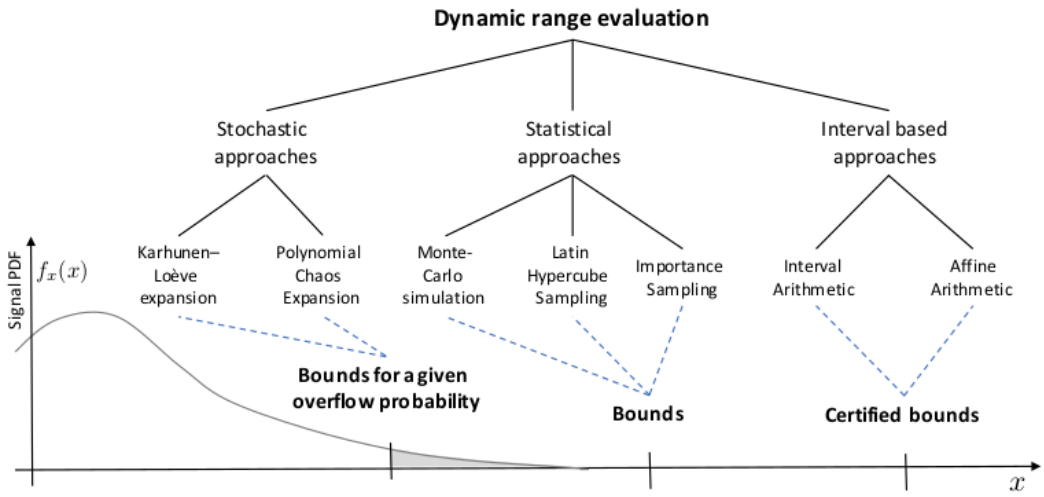


Figure 2.5: Techniques for assessing dynamic range are categorized into groups [52]

Critical systems use techniques that guarantee the absence of overflow. Indeed, when a system cannot accept any computational error, it is necessary to determine a minimum integer

word-length that will guarantee the absence of overflow on each variable, which means the integer of each variable must span the whole range of its feasible values. To ensure that these systems do not have overflow errors on any variables, approaches based on interval or affine arithmetic are applied. However, the drawback of those methods is that the dynamic range can be overestimated by a wide margin, leading to an increase in implementation cost. Statistical methods for determining boundaries from a collection of simulation results may help avoid overestimation, but they cannot guarantee that there will not be any overflows [54]. In many cases, if the likelihood of an overflow occurring is low enough, the program can tolerate this overflow. It is important to note that, in this scenario, the IWL selection is simply an optimization trade-off between implementation cost and loss of system quality. The objective of this task is to make integers as small as feasible, while maintaining an overflow probability of less than the approved probability. The PDF function of application variables may be determined using a variety of stochastic approaches [55]. The impact of overflow on application quality may also be assessed using simulation-based methodologies. As a result of the need for a large number of samples in order to achieve a high level of accuracy, simulation-based approaches take a long time to run, even if they are relevant to all systems.

Interval-Based Approaches

Interval-based methods [56, 57] aim to determine the interval of the output by propagating the input interval towards the output through every operation (addition, subtraction, multiplication, and division) present in the system. The primary benefit of **Interval Arithmetic** is that it can easily provide verified boundaries for all potential function outcomes. However, it has three sorts of issues: the dependence problem, the cancellation problem, and the wrapping effect. [52]. The dependence problem states that, when a variable appears several times in an interval calculation, it is considered as a separate variable each time, resulting in the widening of calculated intervals and making it more challenging to produce tight intervals [58]. The cancellation issue arises when the interval width is not cancelled in the inverse functions. This happens when an expression evaluates to zero. In this situation, the calculated interval enclosure must include zero, making it impossible to derive the expression's sign unless the interval is $\{0\}$ [58]. The wrapping effect is a result of the fact that the image of an interval vector under a map is not an interval vector, leading to an overestimation when enclosing the image with an interval vector [59].

Affine Arithmetic (AA) [60, 61, 62, 63] is another method for improving interval arithmetic. Unlike interval arithmetic, it keeps track of the relationships between the numbers that are computed and the numbers that are put in. This makes it resistant to the catastrophic loss of precision that is often seen in long interval computations [64]. The primary benefit of AA is that

it monitors noise symbols and eliminates all first-order uncertainty. AA achieves quadratic convergence in nonlinear systems, but the increase in the amount of noise components in nonlinear operations makes calculations less precise and more time-consuming [52].

Statistical Approaches

According to statistical techniques, the application is simulated multiple times using stimuli, and the values of the outputs are recorded. The values are then used to determine the dynamic range of the application using certain statistics. Several approaches have been studied. The **Monte-Carlo (MC) method** generates random test vectors based on input probability density functions (PDFs), propagates them through the system, and stores the statistics of all signals for analysis. Statistical properties include mean, variance, higher order statistics, and PDF. From the obtained statistics, suitable integer WLs are chosen. The MC approach is safe and theoretically straightforward, making it a common method for evaluating dynamic range. This approach takes a large number of simulations and is usually slow, particularly when high accuracy findings or input signals are required [52].

The **Latin Hypercube Sampling (LHS)** is a variation of MC that divides input signal PDFs into equiprobable portions and combines samples based on correlations [52]. In [65], LHS is evaluated in comparison with MC method. For a given number of samples, the accuracy is almost the same, but the processing speed is 12 times slower.

The **Importance Sampling approach** is based on the premise that certain simulation input random values have a greater influence on the parameter under study than others. The estimator variance may be minimized if these important values are sampled more often. The accuracy of this technique depends on the system being analyzed, providing certain baseline conditions are satisfied [52]

Stochastic Approaches

To analyze the data dynamic range, stochastic algorithms compute the PDF of the system inputs. The PDF represents the range of all variables, which is derived by propagating the variability characterisation across the system. The PDF is used to determine the range of all variables with regard to a coverage probability [54]. In [55], the authors mentioned several stochastic methods. The **Karhunen–Loève expansion** (KLE) is used in [66, 67] to stochastically discretize the input into random variables. It is feasible to derive the matching description of the output and a good removal of the temporal and spatial correlation using the superposition property of linear time-invariant systems (LTI), resulting in tighter constraints. Superposition is no longer possible for non-linear systems and hence the **polynomial chaos expansion** (PCE) is employed in [68]. Using PCE arithmetic, non-linear systems may statically propagate input variability, and the PCE representation for all variables can be derived.

2.3.2 Fractional Word Length (FWL) Selection

After the IWL selection procedure, the integer word-length of each variable, i.e., W_i^{int}, was obtained to ensure that no overflow problem occurs. The remaining part that needs to be determined in the configuration is the fractional word-lengths, which are optimized through the FWL selection procedure. The goal of the FWL selection procedure is to reduce implementation costs by optimizing fractional bit-widths while satisfying the output quality requirements defined in Equation 2.7. This procedure necessitates the solution of a constrained optimization problem, also known as the Word Length Optimization (WLO) problem, where the objective function is the cost function $\boldsymbol{C}()$ and the constraint function is the accuracy function $\boldsymbol{\lambda}()$. The solution of the problem is the fractional configuration that consumes the minimum implementation cost, while still meeting the quality requirement at the output.

Recent techniques mostly solve WLO using an iterative process that explores different WL configurations to obtain the appropriate solution. Each iteration, depending on the evaluation criteria of each algorithm, a WL configuration will be selected and served as the basis for choosing the next point. The evaluation criteria is usually constructed from a mathematical expression of cost and/or accuracy given a specific WL configuration. In the optimization problem 2.7, functions $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are responsible for returning the implementation cost and the output quality for each WL configuration. The process stops when the optimization algorithm can not find any other better point and will return the best-found solution.

Cost Evaluation

In fact, the cost and accuracy functions are built using estimation models or simulations that are able to return the values quickly, allowing the WLO process to be executed faster. Indeed, the implementation cost is precisely obtained just after the system is synthesized on FPGA or ASIC flow with a specific technology, which requires a long time for optimizing NP-complete problems such as scheduling and resource binding in the synthesis process. Therefore, a cost model is reasonable to replace the actual cost to speed up the optimization process. A good cost model is the one that can reflect relatively accurately the impact of changing the of each signal on the total cost. Energy, area, latency, and total bit-width are the most frequently utilized cost models.

Accuracy Evaluation

The reduction of word-lengths represented for signals and operations in an application causes quantization error. The quantization error is evaluated through an accuracy evaluation. The goal of accuracy evaluation is to quantify or measure the correctness of a fixed-point solution.

The evaluation of accuracy is a measure that determines how well the solution meets the requirements. Many metrics can be used to evaluate the quantization error, such as Bit Error Rate (BER), which is commonly used to characterize the performance of data channels when transmitting data between data point stations in telecommunication applications, Signal-to-Quantization-Noise Ratio (SQNR), which is commonly used to compare the level of a desired signal to the level of background noise in image/signal processing applications, Peak signal-to-noise ratio (PSNR), which is the ratio of the greatest feasible signal strength to the power of corrupting noise that influences its representation, or structural similarity index measure (SSIM), a method for predicting the perceived quality of digital images.

The accuracy can be estimated via bit-true simulations [69, 56, 70, 71, 72, 73]. The accuracy of the fixed-point configuration is statistically evaluated using the output obtained from simulating the application in both fixed-point and floating-point versions. The fixed-point version represents the application with fixed-point variables. The floating-point version represents the application for a precise execution. The single- or double-precision floating-point versions are usually used as the output references. Since the floating-point version can be considered as accurate, any differences between a fixed-point simulation result and a floating-point simulation result reflect the accuracy of the fixed-point solution. The simulation approaches have the benefit of being simple to use and producing accurate results for any type of system and quality metrics. However, these approaches do not scale well due to the long simulation time. In large applications, the number of variables needed to be optimized is high, increasing the number of iterations required for the solution exploration in the WLO process exponentially. This might lead to a very long exploration time.

Besides, some techniques use analytical models to estimate the quantization error [74, 51, 75, 76, 77]. The objective of analytical models is to obtain the mathematical expression that represents the function of quantization noise and fixed-point variables. Then, this function is used to quickly estimate the quantization error given any fixed-point solutions, which allows scaling the system better than simulation-based approaches. However, analytical approaches are limited to linear time-invariant (LTI) systems [78, 76] and differential nonlinear systems [79, 77].

In [80], a hybrid method of analytical and simulation approaches is used simultaneously to improve the accuracy evaluation. The technique divides the system into smooth and unsmooth components. For unsmooth blocks, an unsmooth error occurs, which cannot be captured by Taylor series expansion and has to be evaluated by simulations. For smooth blocks, an analytical approach can be used to evaluate the quantization error.

Optimization Approaches

Recall the WLO problem stated by Equation 2.7, we need to find an optimal set of fractional word-lengths that is the solution to this equation. The number of variables to be optimized is N, which is also the number of signals to be represented in the fixed-point datatype in the given application. Let M be the number of choices for the bit-width of each variable (e.g., from 8 to 32 bits). In real applications, M may be different for each variable depending on the user's settings. However, to describe the complexity of the problem simply and intuitively, we consider M to be the same for all variables. The number of WL configurations to evaluate is N^M. This is known as the multiple word-length assignment (MWLO) paradigm. If you use *brute-force* to solve this problem, it is equivalent to evaluating all combinations using simulations. This is impossible in industrial-sized problems that may contain hundreds to thousands of variables to be optimized. We take here an example to calculate the worst case scenario to find the optimal solution for WLO problem with an application including 100 variables. Assume the number of choices for bit length for a variable is 10 and the time to simulate one configuration is one second. This simulation time depends on the application complexity and the amount of input it takes; usually the simulation time for the fixed-point version of an application falls in the range of seconds. Then, the worst case scenario for finding the optimal configuration is $100^{10} \times 1$ seconds, which is equivalent to 1.16×10^{15} days. This optimization time is therefore unreasonable.

In the past, some techniques were proposed to reduce the complexity of the WLO problem by assigning the same word-length format to each variable. This technique is known as uniform word-length optimization (UWLO). Only a short time is needed to obtain the best solution since the worst-case scenario is M evaluations. However, the quality of the returned solution is far from the solution obtained by the MWLO approaches [81]. Therefore, recent techniques mainly focus on addressing scalability and optimality in MWLO problem.

Recent techniques solve WLO problems either by using analytical models or simulation-based methods, or a combination of both. Analytical approaches [18, 82, 83, 84] construct the objective and constraint functions of the WLO problem to be a convex optimization problem, and then apply convex optimization techniques to directly get the solution. Most of these approaches try to find a good approximation of the optimal solution (which is however generally unknown).

On the other hand, simulation-based approaches [19, 72, 85, 78, 20] use brute-force (with some extensions) and iterative search as well as cost-accuracy simulations to obtain the solution. Recently, some divide-and-conquer-based approaches were proposed to solve the scalability problem in industrial-size applications, which may contain hundreds to thousands of variables to be optimized [86, 87]. These approaches cut large problems into smaller problems to reduce optimization complexity. These smaller problems are then optimized using well-researched methods.

After obtaining solutions to the smaller problems, the authors use special techniques to combine these solutions to find a global solution of the original problem. Inspired by a review of the methods for WLO [52], we divide the methods into three main groups: heuristics, divide-and-conquer approach and analytical approach.

Brute-force with upper and lower bounds. As mentioned above, evaluating all configurations, i.e., the brute-force method, to find the optimal solution is impossible due to very long execution time. Some methods have improved the application of the brute-force method by searching for all possible word-length combinations in a narrower space, which reduces the number of configurations that need to be simulated and thus greatly reduces the search time. This method is referred to as *complete search* [88, 89, 72]. The idea of *complete search* is to apply lower and upper bounds for all word-lengths to restrict the design search space. The upper bound is determined by the minimum uniform configuration where all variables are assigned the same, and this is the minimum to meet the accuracy constraint. The lower bound is selected by the configuration where each variable is assigned by its *minimum WL*, i.e., the WL satisfying the quality target when other variables are set to the highest precision. Indeed, the process of choosing the minimum WL for each variable is done sequentially and independently. To select the minimum WL for a given variable, the variables are first assigned by the maximum number of bits, e.g., 32 bits or 64 bits, thus representing the highest possible precision of each variable. Then, a sequential reduction of the bit-width by 1 bit for the selected variable is performed while the highest bit-width is still kept for the other variables. The configuration obtained by each bit-width reduction is then evaluated to get the accuracy value. This process is carried out until a minimum WL is obtained for which the configuration is generated, that still satisfies the accuracy requirement. It is important to note that this method does not guarantee the global optimal since this point may be missing in the narrowed search space, as proved in [78].

Heuristics. Due to the NP-hard nature of the word-length optimization problem, numerous optimization strategies are based on greedy algorithms thanks to their simplicity of implementation and low iteration count. The greedy approach is an iterative search method, which at each iteration, considers possible non-overlapping solutions and selects the best solution following a loss function. The chosen solution at each iteration is the basis for selecting the solution at the next iteration. The algorithm stops when the next solution is not better than the current one and the current one is considered the best found solution. While the greedy algorithm is a very efficient method for solving many optimization problems, it is prone to getting stuck in a local minimum rather than reaching the global solution.

Min+1 bit. This algorithm [72, 85] consists of three steps. The first step is the process of finding the minimum WL for each variable (this process was mentioned above). In the second step,

the minimum WL is set for each variable. The configuration that is created by the combination of minimum WLs is called the starting point of the algorithm. We name this point as minimum WL configuration (MWLC). The third step is an iterative process to reach the final solution. At each iteration, there is a competition between different temporary configurations, which are created by only increasing one variable by one bit. The number of variables corresponds to the temporary configurations created. At each iteration, the best configuration whose accuracy is maximum is selected. The algorithm stops when a selected configuration at an iteration satisfies the accuracy constraint. This configuration is selected as the best solution. An extension of this procedure is **Min+b bit** [90]. In this method, the temporary configuration is created by increasing one variable by b bit(s), and b is also increased gradually to $b = 1, 2, 3, \ldots$ bit(s). If the accuracy constraint is met, b will be reset to 1 bit.

Max-1 bit. This algorithm [90] starts with the highest precision for all variables. Then, a competition to reduce variables by one bit is performed; configurations created by reducing one bit to one variable are evaluated to get the accuracy and cost. The configuration whose accuracy is maximum is kept. The procedure continues until the accuracy constraint is not met. Some techniques extend the use of Max-1 with a modified selection policy. The technique in [91] selects the configuration that produces a maximum cost reduction. In [92], Han *et al.* use a function considered as the trade-off between accuracy and cost to select the best configuration at each iteration.

In [88], an algorithm proposes to start at MWLC and then iteratively increase all variables by one bit until the accuracy constraint is met. The exploration time of this method is trivial, but the solution obtained is not good due to the lack of fine search.

Hybrid algorithms. The authors of [90] combine the Min+b bit algorithm and the Max-1 bit algorithm to produce a bi-directional search. Another heuristic algorithm is proposed in [69], which also uses bi-directional search to find the best solution. This algorithm starts at MWLC, then all variables are uniformly increased by one bit until the accuracy constraint is satisfied. Then, the Max-1 bit algorithm is applied to reduce the cost until the accuracy constraint is not met. A comparison in [90] shows that by combining both increment and decrement procedures like in [90, 69], the result obtained outperforms the one found by Min+b and Max-1 alone.

Tabu search [20, 93] also uses the bi-direction search strategy to improve the exploration result. First, the algorithm is initialized at MWLC. Then, a min+1 bit algorithm is used to improve the accuracy gradually with a trade-off to a minimal cost degradation. The procedure optimizes the bitwidths until a solution that satisfies the accuracy constraint is found. Finally, this solution is fine-tuned with Tabu search algorithm. Tabu search algorithm is a combination

of Min+1 and Max-1 algorithms that allows to search both directions, ascending and descending directions, to fine-tune the solution. This procedure uses a *Tabu* list to skip some explored variables during the exploration. This algorithm iterates until all variables are registered in the Tabu list. A variable is registered in the Tabu list when its word-length reaches its maximal value in the ascending direction or its minimal value in the descending direction. In the ascending direction, whenever the accuracy constraint is satisfied, the direction is inverted and the operator that leads to this accuracy satisfaction is also added into the Tabu list. In the descending direction, the search direction is inverted when the accuracy is no longer satisfied.

The Greedy Randomised Adaptive Search Procedure (**GRASP**) [20] combines a bi-directional search and stochastic optimisation. GRASP is an iterative two-phase procedure. In the construction phase, the search algorithm (similar to Min+1) randomly selects one of the best neighboring candidates during gradient descent. Then, a **Tabu search** is applied to refine the solution found by the first phase. Tabu allows movements in both directions (increase and decrease by one bit) and uses a Tabu list to skip some variables from the exploration. These two phases are iterated and the randomization of the construction phase avoids to stay in local minima. It is worth noting that, in this method, both cost and accuracy values are used to define the rule to select the best solution at each iteration during the Tabu search phase.

Divide-and-conquer approaches. Although simple to implement and directly applicable to all kinds of systems to solve WLO problems, simulation-based heuristics suffer from scalability issues. For industrial-size applications, which may include many variables, the complexity of WLO increases exponentially, meaning that the number of required fixed-point simulations also increases exponentially compared to simple applications. Besides, as mentioned above, the accuracy evaluation based on bit-true fixed-point simulations takes a long time to execute. Hence, simulation-based heuristics do not scale well in large applications. Recently, some approaches have been proposed to deal with the scalability problem in WLO. The idea behind those approaches is to divide the original WLO problem into sub-problems. These sub-problems can be solved independently using analytical or simulation-based approaches. Then, the solutions of the sub-problems are combined to find the best solution for the original WLO problem.

Parashar *et al.* [86] propose a hierarchical approach to solve WLO. They divide the system into different subsystems, each of which contains only smooth operators or unsmooth operators. Smooth operators can be treated with an analytical approach, while unsmooth operators are handled with a simulation-based approach. At the output of each hierarchical block, a optimization is performed to minimize the block cost under a noise budget constraint, which is provided by an optimization procedure at the system level. At the system level, an optimization procedure based on the Min+1 bit algorithm optimizes for the noise budgets of blocks to minimize

the system cost under a global accuracy constraint. In each iteration, a competition to reduce the noise power of one block by δ_P is performed. This strategy is regarded as acceptable in terms of simulation time if the number of unsmooth operators is appropriate. However, in the case of systems with many complex structures or operators (e.g., conditional structures) and/or sophisticated quality metrics that are not easily constructed analytically, this approach cannot be easily applied.

Novo *et al.* [87] propose a divide-and-conquer method to solve WLO for some complex wireless applications in practical time. The method separates the system into groups and uses a Data Flow Graph to model the separated systems. For sub-systems, the authors propose replacing the use of BER simulations to evaluate the quantization error with the expectation of noise power because the convergence speed is higher while still ensuring the quality of the result. After that, a combination step to obtain a global solution for the original system is taken using the Max-1 bit algorithm to choose from 10% of the best solutions from the groups. The approach is used for the fixed-point refinement of an advanced wireless algorithm, delivering a speedup of almost nine times over a reference statistical method without compromising the quality of the result.

In Chapter 3, we propose a divide and conquer approach that comes up with an empirical model to give the noise budgets to sub-systems (kernels). We also indicate some drawbacks of the recent techniques [86, 87] and compare our approach with them on some experimental results.

Analytical approaches. The optimization problem is an integer programming problem, i.e., all variables in the objective and constraint function are integers. Several approaches have been developed to relax the discrete constraint of each variable, that is, to allow variables to take non-integer values, and at the same time transform the original problem into an associated convex relaxation, which can result in a lower bound on the optimal value. The transformed program can be solved to obtain a fractional optimal solution, i.e., the lower bound, which is then rounded to obtain an integral feasible solution. It is important to note that the optimal relaxation solution may be arbitrarily far from the optimal integer program solution.

Chan *et al.* [18] propose analytical method to solve WLO for linear time-invariant systems (LTI). The method attempts to relax the integer constraints of word-lengths. The cost and accuracy functions are then analytically modeled as convex functions. The transformed problem is solved by Lagrange Multipliers to obtain an optimal fractional solution, which is then rounded to the next largest integers. An extension of the method using Geometric Programming to formalize the WLO is presented in [82]. The two methods are limited to LTI non-recursive systems and noise power as the accuracy metric. For other systems and more complex accuracy metrics, the method gets difficult to apply.

Parashar *et al.* propose a method [84] that relaxes the integrality constraint of word-lengths to solve the problem in the continuous solution space. The authors use noise power to build the accuracy constraint function and the energy consumption of operators to construct the cost function. The two functions are proved to be convex. The problem can be solved by solvers to get optimal noise power for each operator. Finally, a conversion step from the optimal noise powers to fixed-point word-lengths is performed using a fine-tuning procedure.

2.4 Conclusions

In this chapter, we examine and categorize current approximation computing methods according to their design complexity. We first present the Floating-Point (FlP) and Fixed-Point (FxP) data formats. We then detail the most recent solutions to solve the Word-Length Optimization (WLO) problem. The main optimization strategies are evaluated and analyzed, along with the benefits and drawbacks of the current strategies. This chapter also discusses the approaches used to simulate the cost/accuracy trade-offs during WLO.

For large applications involving many variables or different system properties (e.g., linear/non-linear, time-invariant/time-variant, deterministic/stochastic), there are some issues with current WLO approaches. WLO can be quickly handled analytically, but these methods are only effective mostly for linear and time-invariant (LTI) systems (with some extensions). On the other hand, approaches based on simulation are applicable to all systems and quality metrics. However, since the number of simulations grows significantly as the number of variables increases and the simulation time also increases with the size of the application, these approaches do not scale well in large applications with numerous variables.

In this thesis, we propose two contributions that improve the scalability of the WLO problem. The first contribution described in Chapter 3 proposes a noise budgeting methodology that separates the original problem into independent smaller problems by giving them noise budgets. The noise budgets represent the minimum accuracy constraints that sub-problems have to meet. Through empirically generated models, we must capture the interplay between approximations made to a kernel with its cost and approximations applied to other kernels. The generated models are then utilized to estimate the optimal noise budget allocation. To further enhance the scalability of simulation-based techniques, the second contribution presents a hybrid algorithm that combines Bayesian Optimization (BO) with local search, which is described in Chapter 4. The BO method may rapidly reduce the design space and identify a solution from which the local search algorithm can fine-tune the cost, resulting in a substantial decrease in exploration

time.

Another aspect that is lacking in the current state of the art is the investigation of the resource-constrained WLO problem. In Chapter 5, we identify issues with the present WLO approaches for handling WLO problems with cost constraints. Then, we give an effective Bayesian Optimization based technique for maximizing computation quality under an energy cost budget.

Bibliography

[1] "Approximate computing," https://en.wikipedia.org/wiki/Approximate_computing, accessed: 2022-05-30.

[2] 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*, 2013, pp. 1–9.

[3] "Oh, that's near enough," https://www.economist.com/technology-quarterly/2012/06/02/oh-thats-near-enough, accessed: 2022-02-11.

[4] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Cross-layer approximate computing: From logic to architectures," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6.

[5] W. Baek and T. M. Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation," in *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2010, pp. 198–209.

[6] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 164–174, 2011.

[7] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing performance vs. accuracy trade-offs with loop perforation," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 124–134.

[8] "Task skipping," https://en.wikipedia.org/wiki/Task_skipping, accessed: 2022-06-04.

[9] I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, "Approxhadoop: Bringing approximations to mapreduce frameworks," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2015, pp. 383–397.

[10] Y. Tian, Q. Zhang, T. Wang, F. Yuan, and Q. Xu, "Approxma: Approximate memory access for dynamic precision scaling," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, 2015, pp. 337–342.

[11] S. T. Chakradhar and A. Raghunathan, "Best-effort computing: Re-thinking parallel software and hardware," in *Design Automation Conference.* IEEE, 2010, pp. 865–870.

[12] Z. Wang, C. Huang, H. Kim, W. Li, and Q. Zhu, "Cross-layer adaptation with safety-assured proactive task job skipping," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–25, 2021.

[13] J. Ansel, Y. L. Wong, C. Chan, M. Olszewski, A. Edelman, and S. Amarasinghe, "Language and compiler support for auto-tuning variable-accuracy algorithms," in *International Symposium on Code Generation and Optimization (CGO 2011).* IEEE, 2011, pp. 85–96.

[14] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 301–312.

[15] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: Saving dram refresh-power through critical data partitioning," in *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, 2011, pp. 213–224.

[16] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate storage in solid-state memories," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 3, pp. 1–23, 2014.

[17] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture.* IEEE, 2012, pp. 449–460.

[18] S.-C. Chan and K. M. Tsui, "Wordlength determination algorithms for hardware implementation of linear time invariant systems with prescribed output accuracy," in *IEEE Int. Symp. on Circ. and Syst. (ISCAS)*, 2005.

[19] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[20] H.-N. Nguyen, D. Ménard, and O. Sentieys, "Novel algorithms for word-length optimization," in *19th European Signal Processing Conf.* IEEE, 2011, pp. 1944–1948.

[21] "bfloat16 floating-point format," https://en.wikipedia.org/wiki/Bfloat16_floating-point_format, accessed: 2022-05-30.

[22] "Brain floating-point format (bfloat16)," https://en.wikichip.org/wiki/brain_floating-point_format, accessed: 2022-05-30.

[23] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269–284, 2014.

[24] M. Brandalero, A. C. S. Beck, L. Carro, and M. Shafique, "Approximate on-the-fly coarse-grained reconfigurable acceleration for general-purpose applications," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[25] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, and M. Shafique, "Px-cgra: Polymorphic approximate coarse-grained reconfigurable architecture," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 413–418.

[26] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "Approxann: An approximate computing framework for artificial neural network," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 701–706.

[27] 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, 2009.

[28] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: Imprecise adders for low-power approximate computing," in *IEEE/ACM International Symposium on Low Power Electronics and Design*. IEEE, 2011, pp. 409–414.

[29] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[30] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel, "Probabilistic error modeling for approximate adders," *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 515–530, 2016.

[31] 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, 2012.

[32] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1435–1441, 2017.

[33] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 522–531, 2004.

[34] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified booth multipliers for lossy applications," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 19, no. 1, pp. 52–60, 2009.

[35] Y.-H. Chen, C.-Y. Li, and T.-Y. Chang, "Area-effective and power-efficient fixed-width booth multipliers using generalized probabilistic estimation bias," *IEEE Journal on Emerging and selected topics in Circuits and Systems*, vol. 1, no. 3, pp. 277–288, 2011.

[36] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic estimation bias circuit for fixed-width booth multiplier and its dct applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 4, pp. 215–219, 2011.

[37] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 3, pp. 594–603, 2011.

[38] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an under-designed multiplier architecture," in *2011 24th Internatioal Conference on VLSI Design*. IEEE, 2011, pp. 346–351.

[39] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, 2014.

[40] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[41] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 418–425.

[42] R. Nath and D. Tullsen, "The crisp performance model for dynamic voltage and frequency scaling in a gpgpu," in *Proceedings of the 48th international symposium on microarchitecture*, 2015, pp. 281–293.

[43] S. Yang, R. A. Shafik, G. V. Merrett, E. Stott, J. M. Levine, J. Davis, and B. M. Al-Hashimi, "Adaptive energy minimization of embedded heterogeneous systems using regression-based

learning," in *2015 25th international workshop on power and timing modeling, optimization and simulation (PATMOS)*. IEEE, 2015, pp. 103–110.

[44] R. Jain, P. R. Panda, and S. Subramoney, "Machine learned machines: Adaptive co-optimization of caches, cores, and on-chip network," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 253–256.

[45] G. Dhiman and T. S. Rosing, "Dynamic voltage frequency scaling for multi-tasking systems using online learning," in *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED'07)*. IEEE, 2007, pp. 207–212.

[46] H. Shen, J. Lu, and Q. Qiu, "Learning based dvfs for simultaneous temperature, performance and energy management," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2012, pp. 747–754.

[47] M. Clark, A. Kodi, R. Bunescu, and A. Louri, "Lead: Learning-enabled energy-aware dynamic voltage/frequency scaling in nocs," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[48] "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.

[49] J.-M. Muller, N. Brisebarre, F. De Dinechin, C.-P. Jeannerod, V. Lefevre, G. Melquiond, N. Revol, D. Stehlé, S. Torres *et al.*, *Handbook of floating-point arithmetic*. Springer, 2018, vol. 1.

[50] "Cmu common lisp user's manual," http://users.umiacs.umd.edu/~resnik/ling645_sp2002/cmu_manual/node19.html, accessed: 2022-02-22.

[51] D. Menard, R. Rocher, and O. Sentieys, "Analytical fixed-point accuracy evaluation in linear time-invariant systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 10, pp. 3197–3208, 2008.

[52] D. Menard, G. Caffarena, J. A. Lopez, D. Novo, and O. Sentieys, "Fixed-point refinement of digital signal processing systems," 2019.

[53] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger, "Accelerating fixed-point design for MB-OFDM UWB systems," https://www.design-reuse.com/articles/9559/, 2005.

[54] R. Nehmeh, "Quality evaluation in fixed-point systems with selective simulation," Ph.D. dissertation, Rennes, INSA, 2017.

[55] R. Nehmeh, D. Menard, E. Nogues, A. Banciu, T. Michel, and R. Rocher, "Fast integer word-length optimization for fixed-point systems," *Journal of Signal Processing Systems*, vol. 85, no. 1, pp. 113–128, 2016.

[56] H. Keding, M. Willems, M. Coors, and H. Meyr, "Fridge: a fixed-point design and simulation environment," in *Proceedings Design, Automation and Test in Europe*. IEEE, 1998, pp. 429–435.

[57] M. Willems, V. Bursgens, T. Grotker, and H. Meyr, "Fridge: An interactive code generation environment for hw/sw codesign," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 1997, pp. 287–290.

[58] H. Brönnimann, C. Burnikel, and S. Pion, "Interval arithmetic yields efficient dynamic filters for computational geometry," *Discrete Applied Mathematics*, vol. 109, no. 1-2, pp. 25–47, 2001.

[59] R. B. Kearfott, "Interval computations: Introduction, uses, and resources," *Euromath Bulletin*, vol. 2, no. 1, pp. 95–112, 1996.

[60] J. Cong, K. Gururaj, B. Liu, C. Liu, Z. Zhang, S. Zhou, and Y. Zou, "Evaluation of static analysis techniques for fixed-point precision optimization," in *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*. IEEE, 2009, pp. 231–234.

[61] C. F. Fang, R. A. Rutenbar, M. Püschel, and T. Chen, "Toward efficient static analysis of finite-precision effects in dsp applications via affine arithmetic modeling," in *Proceedings of the 40th annual Design Automation Conference*, 2003, pp. 496–501.

[62] L. H. De Figueiredo and J. Stolfi, "Affine arithmetic: concepts and applications," *Numerical Algorithms*, vol. 37, no. 1, pp. 147–158, 2004.

[63] S. M. Rump and M. Kashiwagi, "Implementation and improvements of affine arithmetic," *Nonlinear Theory and Its Applications, IEICE*, vol. 6, no. 3, pp. 341–359, 2015.

[64] J. Stolfi, M. Andrade, J. Comba, and R. Van Iwaarden, "Affine arithmetic: a correlation-sensitive variant of interval arithmetic," *Web document*, 1994.

[65] C. N. Zeeb, P. J. Burns, and F. Collins, "A comparison of failure probability estimates by monte carlo sampling and latin hypercube sampling," *Sandia National Laboratories*, 1998.

[66] B. Wu, J. Zhu, and F. N. Najm, "An analytical approach for dynamic range estimation," in *Proceedings of the 41st annual Design Automation Conference*, 2004, pp. 472–477.

[67] A. Banciu, E. Casseau, D. Menard, and T. Michel, "Stochastic modeling for floating-point to fixed-point conversion," in *2011 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2011, pp. 180–185.

[68] B. Wu, J. Zhu, and F. N. Najm, "Dynamic range estimation for nonlinear systems," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* IEEE, 2004, pp. 660–667.

[69] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[70] K.-I. Kum and W. Sung, "Word-length optimization for high-level synthesis of digital signal processing systems," in *1998 IEEE Workshop on Signal Processing Systems. SIPS 98. Design and Implementation (Cat. No. 98TH8374).* IEEE, 1998, pp. 569–578.

[71] S. Kim and W. Sung, "Fixed-point simulation utility for c and c++ based digital signal processing programs," in *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, vol. 1. IEEE, 1994, pp. 162–166.

[72] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196)*, vol. 5. IEEE, 2001, pp. 53–56.

[73] J. Hormigo and G. Caffarena, "Fpga acceleration of bit-true simulations for word-length optimization," in *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH).* IEEE, 2021, pp. 119–122.

[74] D. Menard and O. Sentieys, "Automatic evaluation of the accuracy of fixed-point algorithms," in *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition.* IEEE, 2002, pp. 529–535.

[75] R. Rocher, D. Menard, P. Scalart, and O. Sentieys, "Analytical approach for numerical accuracy estimation of fixed-point systems based on smooth operations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 10, pp. 2326–2339, 2012.

[76] J. A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Fast and accurate computation of the round-off noise of linear time-invariant systems," *IET Circuits, Devices & Systems*, vol. 2, no. 4, pp. 393–408, 2008.

[77] G. Caffarena, C. Carreras, J. A. López, and Á. Fernández, "Sqnr estimation of fixed-point dsp algorithms," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, pp. 1–12, 2010.

[78] G. A. Constantinides, P. Y. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1432–1442, 2003.

[79] D. Menard, R. Rocher, P. Scalart, and O. Sentieys, "Automatic sqnr determination in non-linear and non-recursive fixed-point systems," in *2004 12th European Signal Processing Conference*. IEEE, 2004, pp. 1349–1352.

[80] K. N. Parashar, D. Menard, and O. Sentieys, "Accelerated performance evaluation of fixed-point systems with un-smooth operations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 4, pp. 599–612, 2014.

[81] G. C. Fernández, "Combined word-length allocation and high-level synthesis of digital signal processing circuits," Ph.D. dissertation, Universidad Politécnica de Madrid, 2008.

[82] S.-C. Chan and K. M. Tsui, "Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming," *IEEE Trans. on Circ. and Syst.*, vol. 54, no. 4, pp. 845–854, 2007.

[83] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Trans. on Computers*, vol. 57, no. 11, pp. 1561–1570, 2008.

[84] K. N. Parashar, D. Menard, and O. Sentieys, "A polynomial time algorithm for solving the word-length optimization problem," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2013, pp. 638–645.

[85] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for cdma demodulator," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, 2001, pp. 290–293.

[86] K. Parashar, R. Rocher, D. Menard, and O. Sentieys, "A hierarchical methodology for word-length optimization of signal processing systems," in *23rd Int. Conf. on VLSI Design (VLSID)*, 2010, pp. 318–323.

[87] D. Novo, I. Tzimi, U. Ahmad, P. Ienne, and F. Catthoor, "Cracking the complexity of fixed-point refinement in complex wireless systems," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2013, pp. 18–23.

[88] W. Sung and K.-I. Kum, "Word-length determination and scaling software for a signal flow block diagram," in *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2. IEEE, 1994, pp. II–457.

[89] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Advances in Signal Processing*, vol. 2006, pp. 1–14, 2006.

[90] M.-A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353)*, vol. 2. IEEE, 2002, pp. II–II.

[91] H. Choi and W. Burleson, "Search-based wordlength optimization for vlsi/dsp synthesis," in *Proceedings of 1994 IEEE Workshop on VLSI Signal Processing.* IEEE, 1994, pp. 198–207.

[92] K. Han, B. L. Evans, and E. E. Swartzlander, "Data wordlength reduction for low-power signal processing software," in *IEEE Workshop onSignal Processing Systems, 2004. SIPS 2004.* IEEE, 2004, pp. 343–348.

[93] D. Menard, N. Herve, O. Sentieys, and H.-N. Nguyen, "High-level synthesis under fixed-point accuracy constraint," *Journal of Electrical and Computer Engineering*, vol. 2012, 2012.

Chapter 3

Towards Generic and Scalable Word-Length Optimization

In this chapter, we present a method to improve the scalability of Word-Length Optimization (WLO) for large applications, which also uses complex quality metrics such as Structural Similarity (SSIM). The input application is decomposed into smaller kernels to avoid uncontrolled explosion of the exploration time. The main challenge addressed in this chapter is how to allocate noise budgets to each kernel, which is known as noise budgeting. This requires capturing the interactions across kernels. The main idea is to characterize the impact of approximating each kernel on accuracy/cost through simulation and regression. We evaluate our approach on two application structures. One is made up of smaller units known as kernels that are organized in a sequential order, whereas the second is made up of forked and serialized kernels. Our approach improves the scalability while finding better solutions for applications such as Image Signal Processor pipeline and Stereo Matching.

3.1 Introduction

Fixed-point arithmetic is widely used for implementing Digital Signal Processing (DSP) systems on electronic devices. Since initial specifications are often written using floating-point arithmetic, conversion to fixed-point is a recurring step in hardware design. The primary objective of this conversion is to minimize the cost (energy and/or area) while maintaining an acceptable level of quality at the output. This process, called Word-Length Optimization (WLO), is a time consuming process taking up to $25 - 50\%$ of design time [1].

In WLO, each variable/operator may be assigned a different fixed-point encoding, which means that the design space grows exponentially as the number of variables increases. This is especially true when targeting hardware accelerators implemented in FPGA or ASIC. Thus, most approaches for WLO involve heuristic search algorithms [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. A key component in such search algorithms is the evaluation of output quality. There are two broad evaluation categories: simulations and analytical models. Simulation-based methods suffer from scalability issues as the number of required simulations, as well as the simulation time, increase

drastically with the size of the application [2, 3, 5, 6, 8, 9, 10]. Methods based on analytical models scale well, but are limited by the ability to construct adequate models. Existing techniques are limited to modeling noise power metrics of Linear and Time-Invariant (LTI) systems (with some extensions) [4, 7, 11].

One technique to address scalability issues, called *noise budgeting*, decomposes an application into smaller chunks, or kernels, and assigns separate quality constraints, called *noise budgets*. The smaller sub-problems can be explored much faster, at the cost of ignoring possible inter-kernel interactions. The allocation of noise budgets plays a critical role in this technique, as it is the only parameter that indirectly captures the inter-kernel interactions. However, there is still little work on how to find good allocations of the noise budgets. An existing approach [11] makes heavy use of analytical models, making it difficult to support quality metrics other than noise power and its variants. More complex quality metrics, such as Structual Similarily (SSIM) used for images or Objective Degradation Grade (ODG) used for audio, do not directly correlate with noise power, and are much harder to model.

In this chapter, we propose a WLO method to address both scalability (in simulations) and generality (in analytical models). The key in our work is capturing the interactions between approximations applied to a kernel with its cost and approximations applied to other kernels through *empirically* constructed models. Since the models are constructed through simulations, our approach can be used for any quality metric. The constructed models are then used to predict the best allocation of noise budgets. We show that the predicted noise budgets give better solutions than those found by WLO on the whole program (without decomposition) and that it can be used for both Peak Signal to Noise Ratio (PSNR) and SSIM.

The rest of the chapter is organized as follows. We introduce necessary background and discuss related work in Section 3.2. We formulate our problem, give an overview of our approach in Section 3.3, and describe model construction in Section 3.4. We evaluate our approach in Section 3.5, and conclude in Section 3.6.

3.2 Background and related work

3.2.1 Word-Length Optimization

Fixed-point representation contains two parts: integer word-lengths and fractional word-lengths (WL). The integer WL is closely related to dynamic range; the fractional WL controls the precision. In this work, we focus on the WLO for fractional WL, which is the time consuming part of the exploration. The main objective of WLO is to determine a WL configuration that

minimizes cost while satisfying quality constraints.

Let $\boldsymbol{W}$ denote a WL configuration. Then, the WLO problem is formulated as

$$\min \boldsymbol{C}(\boldsymbol{W}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W}) \geq \boldsymbol{\lambda_{obj}} \tag{3.1}$$

where $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are functions that express cost and quality, respectively, and the target quality is given as $\boldsymbol{\lambda_{obj}}$. How the functions $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are realized varies across work, ranging from simulations to analytical models.

A direct approach to obtain the optimal solution is exhaustive search [12]. However, it is only feasible for small kernels due to exponential growth in possible WL configurations as the number of variables increases. Many approaches [2, 9, 3, 8, 5, 6, 10] were proposed based on iterative search using heuristics to address WLO. Most approaches use variants of gradient decent algorithms that evaluate neighboring solutions, typically constructed by changing one (or a few) variables in the current solution, at each iteration. Since the number of neighboring solutions increases as the number of variables increases, the number of solutions that must be evaluated during the iterative search quickly increases. This is the main reason why simulation-based approaches suffer from scalability issues.

Some approaches construct analytical models of noise power [4, 7, 11] to avoid costly simulations during the exploration. These analytical approaches take advantage of a property of errors under linear systems that their propagation do not interfere with each other. Hence, the error introduced at a noise source may be propagated through the system independently and aggregated afterwards. Thus, these approaches cannot be directly extended to handle general programs. Moreover, complex quality metrics, such as SSIM, do not have a direct relationship with noise power.

3.2.2 Noise Budgeting

Noise budgeting [13, 14, 11] is a technique to address the scalability of WLO. It decomposes a problem into smaller sub-problems that takes less time to solve, and combines the solutions to sub-problems to form the final solution.

Consider an application that is decomposed into N kernels. The WLO is now formulated as

$$\min \sum_{i=1}^{N} \boldsymbol{C}(\boldsymbol{W_i}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W_i}) \geq \boldsymbol{\lambda_i}, \tag{3.2}$$

where the WL configuration and the constraint (*noise budget*) for the i-th kernel are denoted as $\boldsymbol{W_i}$ and $\boldsymbol{\lambda_i}$, respectively. The above decomposition gives N smaller sub-problems (subsets of the above for each i may be solved independently), limiting the explosion in number of configurations to explore.

How the quality of each kernel $\boldsymbol{\lambda}(\boldsymbol{W_i})$ is computed, may depend on the approach. In our work, we define the quality of a kernel considered in isolation as the quality of the application output when all other kernels are not approximated (i.e., computed with floating-point).

The main challenge in noise budgeting is in the allocation of the budgets. Decomposition into sub-problems makes it impossible to capture the possible interactions spanning multiple kernels. For example, errors introduced at a kernel may be masked (or amplified) at a later kernel, which affects how much loss in quality in the former kernel is tolerated. Moreover, such interactions make it difficult to tell if the solutions to the sub-problems using noise budgets would satisfy the constraint on application output when the individual solutions are combined.

Parashar *et al.* [11] use analytical models to capture the inter-kernel interactions, and solve problem (3.2) for the optimal allocations of noise budgets. However, this approach cannot be easily extended to general programs and/or sophisticated quality metrics due to the difficulty of constructing analytical models as discussed in Section 3.2.1.

Another body of work uses similar decomposition into sub-problems, but does not allocate noise budgets [13, 14]. In these work, local WLO is first performed for each kernel to identify designs that expose different quality-cost trade-off. Then, the combinations of the local solutions are explored to find the global solution. The main limitation of this approach is that the final solution space is restricted by the initial local WLO for the kernels. The number of design points available at each kernel is proportional to the amount of time spent on local WLO, and it is difficult to know *a priori* if the "right" design for the global combination has been found. Thus, there is a risk of missing important designs (when local search is too coarse) or having scalability issues beyond more than a few kernels (when local search is too fine). Our approach addresses these limitations by allocating the noise budgets using models constructed after a handful of local WLOs.

3.3 Approach Overview

Recall the decomposed WLO problem in (3.2). We are interested in modeling the following functions:

— $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i})$: Cost of kernel K_i as a function of the quality constraint at its output.

— $\hat{\boldsymbol{\lambda}}(\boldsymbol{\lambda_1}, \cdots, \boldsymbol{\lambda_N})$: Application output quality as a function of quality constraints at each kernel.

These functions enable the optimal choice of noise budgets to be formulated as:

$$\min \sum_{i=1}^{N} \hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i}) \quad \text{Subject to} \quad \hat{\boldsymbol{\lambda}}(\boldsymbol{\lambda_1}, \cdots, \boldsymbol{\lambda_N}) \geq \boldsymbol{\lambda_{obj}} \tag{3.3}$$

The functions $\hat{\boldsymbol{C}}_{\boldsymbol{i}}$ model the impact of approximating each kernel K_i to cost. This allows us to identify the kernel that gives best savings in cost for some loss in quality. However, how the individual approximations in the kernels interact must be taken into account to determine the impact on the application output. The function $\hat{\boldsymbol{\lambda}}$ models this behavior to optimize the budget allocation.

Our approach empirically constructs the functions $\hat{\boldsymbol{C}}_{\boldsymbol{i}}$ and $\hat{\boldsymbol{\lambda}}$ to obtain optimal allocation of noise budgets as described above. The overview of our approach is as follows:

1. For each kernel K_i, perform a few WLOs with different quality constraints. The Pareto-optimal designs found by these explorations are used as data points to construct the models ($\hat{\boldsymbol{C}}_{\boldsymbol{i}}$).
2. Run simulations for combinations of the designs used above to evaluate the accuracy of combined solutions. These simulations provide the data to construct the model of inter-kernel interactions ($\hat{\boldsymbol{\lambda}}$).
3. Solve the equation 3.3 for noise budgets.
4. For each kernel K_i, perform a single WLO with the obtained noise budgets. The result of these local WLOs are combined to obtain an initial solution.

One significant advantage of the proposed method is that it is parametric to the WLO algorithm, and how the cost is evaluated. The choice of WLO algorithms strongly influences the quality of designs found; the Pareto frontier is only among the designs explored and is potentially far from the true optimal. Similarly, the cost models used influence the design space and how efficiently an algorithm can explore this space. Thus, the constructed models are specialized for designs that can be found by the given combination of cost model and WLO algorithm. If another model/algorithm is to be used, a separate set of models must be constructed.

The approach is presented in more detail in Algorithm 1. The output of the algorithm is the appropriate quality constraints, $\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \cdots, \boldsymbol{\lambda_N}$. The algorithm contains two procedures, GetModel and SolveOpt. The procedure GetModel is called to obtain the cost functions

of the quality constraints, $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i})$, and the quality output function of the quality constraints, $\hat{\boldsymbol{\lambda}}(\boldsymbol{\lambda_1}, \cdots, \boldsymbol{\lambda_N})$. The procedure SolveOpt constructs and solves the constrained optimization problem to obtain the appropriate quality constraints. The parameter N is the number of kernels in the application. An array $\boldsymbol{\kappa}$ contains m quality targets, κ_1, κ_2, $\cdots$, κ_m, which are used for the heuristic search to explore the cost and quality data samples to each kernel.

Algorithm 1 Find $\boldsymbol{B_\lambda} = [\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \cdots, \boldsymbol{\lambda_N}]$

```
1:  Input
2:      N   Number of kernels
3:      m   Number of quality targets
4:      κ = [κ_1, κ_2, ···, κ_m]
5:  Output
6:      B_λ  Quality constraint array
7:  procedure [Ĉ_i(B_λ), λ̂(B_λ)]=GetModel(N, m, κ)
                                                        ▷ Data collection
8:      for i ← 1 to N do
9:          T_i ← ∅
10:         for j ← 1 to m do
11:             T_i = T_i ∪ getAllSols(κ_j, K_i)
12:         end for
13:     end for
                                                        ▷ cost function of the application
14:     Ĉ_i(B_λ) ← 0
15:     for i ← 1 to N do
16:         O_i = getSolsOnPareto(T_i)
17:         C(λ_i) = getParetoFront(O_i)
18:         C(B_λ) = C(B_λ) + C(λ_i)
19:     end for
                                                        ▷ quality function of the application
20:     λ̂(B_λ) = getCombination(O)
21: end procedure
22: procedure B_λ=SolveOpt(Ĉ_i(B_λ), λ̂(B_λ), λ_obj)
23:     opt = buildConstrainedOptProb(Ĉ_i(B_λ), λ̂(B_λ), λ_obj)
24:     B_λ = solve(opt)
25: end procedure
```

In the procedure GetModel, the data collection step is firstly performed to each kernel (lines 8-13). We define an array $\boldsymbol{T} = [T_1, T_2, \cdots, T_N]$ consisting of N elements corresponding to the N kernels. Each element, T_i, contains all solutions for an individual kernel that is found through the initial exploration targeting m quality targets. The function *getAllSols* is responsible for performing the heuristic search algorithm and storing all considered solutions in the exploration

process (line 11).

The function *getSolsOnPareto* is called to obtain and store $x\%$ of the Pareto frontier in O_i (line 16). The function *getParetoFront* is invoked with the Pareto-optimal points in O_i served as the input; Pareto-front curve is built based on a regression approach and considered as $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i})$ of each kernel (line 17). Afterwards, because of the independent nature in cost functions, the cost function of entire application, $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{B_\lambda})$, is obtained by summing up all individual cost functions $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i})$. The quality function of the application $\hat{\boldsymbol{\lambda}}(\boldsymbol{B_\lambda})$ is constructed through function *getCombination* which combines pareto-optimal points of one kernel to those of others (line 20). The combined quality points are then measured to obtain the relation among quality contribution from individual kernels to the application output. A multiple-dimensional regression is then applied to get the quality function of the application.

In the procedure SolveOpt, since two functions, $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{B_\lambda})$ and $\hat{\boldsymbol{\lambda}}(\boldsymbol{B_\lambda})$, were constructed, the function *buildConstrainedOptProb* is invoked to establish an optimization problem with a quality constraint $\boldsymbol{\lambda_{obj}}$ (line 23). The problem can be solved by existing solvers to obtain the appropriate quality constraints $\boldsymbol{B_\lambda} = [\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \cdots, \boldsymbol{\lambda_N}]$.

As an example, we can illustrate the previous paragraphs with the Image Signal Processor (ISP) application detailed in Section 3.5. ISP contains$N = 4$ kernels: NLM, Demosaic, GC and Unsharp. m defines the number of accuracy targets for simulations corresponding to each kernel. An example with the kernel Demosaic is given in Figure 3.1 on page 74, where the solutions are explored by TABU search for $m = 3$ SSIM quality targets: 0.91, 0.95 and 0.99. Then, $\hat{\boldsymbol{C}}_{\boldsymbol{Demosaic}}(\boldsymbol{\lambda_{Demosaic}})$ is constructed by determining a Pareto-front curve from the data points stored in $T_{Demosaic}$ (lines 14-19).

3.4 Model Construction

In this section, we describe how we construct the models, which is the core of our approach. We first present how the data points are collected, followed by a description of polynomial fitting we use to construct models.

3.4.1 Data Points for Cost Function ($\hat{C}$)

For each kernel K_i, a model of its cost as a function of target quality ($\hat{\boldsymbol{C}}_{\boldsymbol{i}}$) is constructed. The data points used to construct these models are collected by performing WLO of the kernel using different target quality constraints. Among the designs explored by the WLO runs, we use only those in the Pareto-frontier. This is because we are interested in the best designs (for each accuracy) found by the WLO algorithm in use.

It is necessary to perform multiple instances of WLO, especially when the algorithm is not a greedy search. This is because the designs near the target quality will be explored in detail, finding much better designs only for the region of interest. Figure 3.1 shows the designs explored during three WLO runs targeting different quality constraints with one of the kernels from our benchmark detailed in Section 5.5. In these explorations, we used Tabu search [10], which first performs greedy gradient descent to reach the target quality, followed by local search to minimize cost. It is clearly visible in the figure that the local search significantly reduces the cost without affecting quality. Thus, it is important that we run multiple instances of WLO to gather useful data points to accurately model the relationship between quality constraint and cost.

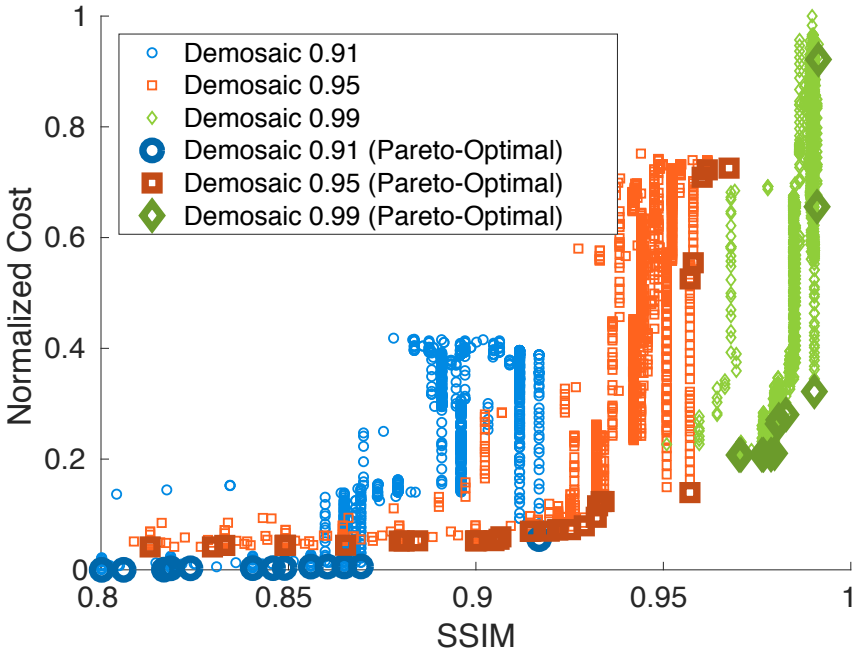


Figure 3.1: All explored solutions of three WLO runs for a kernel (Demosaic) in our benchmark. The number in the legend indicates the quality constraint (SSIM) targeted for each run. The Pareto-optimal points (within each WLO run) are emphasized in the plot. More details about the benchmark and the cost are in Section 5.5.

The design space is explored using three target quality constraints to collect the data points. Given a target quality constraint $\boldsymbol{\lambda_{obj}}$ (for the original problem), and the quality without approximation $\boldsymbol{\lambda_{max}}$, we target the following: $\boldsymbol{\lambda_{obj}} + \delta$, $\boldsymbol{\lambda_{max}} - \delta$, and $0.5(\boldsymbol{\lambda_{obj}} + \boldsymbol{\lambda_{max}})$ where $\delta = 0.1(\boldsymbol{\lambda_{max}} - \boldsymbol{\lambda_{obj}})$. In other words, we target the boundaries and the mid-point under consideration. These targets are motivated by the fact that polynomial fitting over an interval works better when the approximation nodes follow a Chebyshev-like distribution [15], with more nodes towards the boundaries.

The offset δ slightly moves the target constraints inwards. This is because $\boldsymbol{\lambda_{max}}$ is not a realistic target with fixed-point approximation, and because $\boldsymbol{\lambda_{obj}}$ is usually not an appropriate target for any kernel. If a kernel aggressively approximates to the extent that the quality is reduced to $\boldsymbol{\lambda_{obj}}$ by its effect alone, then the combination of such a design with even slight approximations in other kernels usually does not satisfy $\boldsymbol{\lambda_{obj}}$. As an example, if all kernels in an image processing pipeline satisfy an accuracy of SSIM=0.9, it is unlikely that the whole application would have an SSIM=0.9 at its output.

3.4.2 Data Points for Quality Function ($\hat{\boldsymbol{\lambda}}$)

The quality function $\hat{\boldsymbol{\lambda}}$ models the output quality of the combined solutions as a function of output qualities under isolated approximations. Thus, simulations of combined solutions are necessary to collect sufficient data points.

In our current implementation, we take a subset of the data points collected for modeling cost functions, and simulate all combinations to collect the data points. We consider X uniform segments of the quality in the range under consideration, and take the best design within each segment. Then the X^N combinations are simulated to collect data points. (We use $X = 5$ in our evaluation.)

This step may be optimized by having a more sophisticated method to select the subset. An initial model may be constructed by using a few samples from each kernel and simulating their combinations. Then, further simulations that would provide important data points may be predicted using the initial model. This process may be repeatedly applied until the improvement in accuracy of the model starts to diminish. The current implementation does not use this optimization.

3.4.3 Polynomial Fitting

The functions $\hat{\boldsymbol{C}}_{\boldsymbol{i}}$ and $\hat{\boldsymbol{\lambda}}$ are constructed with the data points collected using polynomial fitting. These models are expected to be non-linear, especially for complex quality metrics and/or non-linear systems, motivating the use of non-linear regression. We use polynomial fitting since it does not require a model (template) to be designed. We use Gaussian Process (GP) [16] regression to learn the cost models ($\hat{\boldsymbol{C}}_{\boldsymbol{i}}$) and least squares for the accuracy model ($\hat{\boldsymbol{\lambda}}$). We use a squared exponential covariance function as the kernel for GP. The least squares polynomial fit degree is manually selected by trying a range of values.

Most of the tuning effort for these models comes from preprocessing of the data. The size of the training data is relatively small, and there is no precise control over its distribution. These limitations make the regression analysis susceptible to common pitfalls (over-fitting, oscillation). We apply the following to fine-tune the models:

— obvious outliers are removed using our insights about the WLO algorithm, and

— designs within $x\%$ of the Pareto frontier are included in training data, where x is selected for each kernel.

Such tuning effort is necessary to improve the model quality. How to fully automate this process is a separate subject on its own, which we do not discuss here. In our evaluation, manual tuning time was within several minutes. The models constructed for our benchmarks are discussed in Section 3.5.4.

3.5 Evaluation

In this section, we evaluate our approach against global WLO, where all variables are concurrently considered for WLO, by comparing exploration time and quality of solutions.

3.5.1 Experimental Setup

All experiments are performed on Linux machine with 2x4 cores Intel Xeon E5640 at 2.67GHz and 4GB memory. Recall that there are two parameters to our approach: WLO algorithm, and cost model. In our evaluation, we have used Tabu search [10], which is a heuristic search algorithm based on gradient descent, to perform WLO. We use an energy model as our cost model. The energy model counts the number of operations performed by each operator, and calculates the total cost based on the energy consumption of an operation. The energy per operation is empirically gathered from several ASIC synthesis, simulation and power estimation for different WLs. An operator is characterized by the WLs of the operands, the WL of the result, and the arithmetic operation performed. Characterization is performed using Synopsys Design Compiler and Prime Time using a 28nm technology.

Our approach is implemented using GeCoS [1] [17], an open-source compiler framework. The WLO is performed at the source-level, where the variables define the granularity of the exploration. We use a set of variables per loop nest so that each loop can run with its own WL configurations. In addition, some of the variables in the code are forced to have the same format, since they are aliases of each other. We use the number of *effective variables*, i.e., the number of individual fixed-point formats being explored, as a measure of the complexity.

We use the constraint solver from Matlab optimization toolbox to solve for the noise budgets. In our experiments, the optimizer returned a solution almost instantaneously.

Each kernel is explored once with the same WLO algorithm using the derived noise budgets. The solutions are then combined to form the final solution. The combined solutions may sometimes overshoot or undershoot the target quality slightly due to inaccuracies in our models. If

1. `https://gitlab.inria.fr/gecos/gecos-float2fix`

the quality constraint was not met, we perform a greedy search to find the nearest design that satisfies the constraints. This calibration step takes less than 2% of the total time.

3.5.2 Image Signal Processor

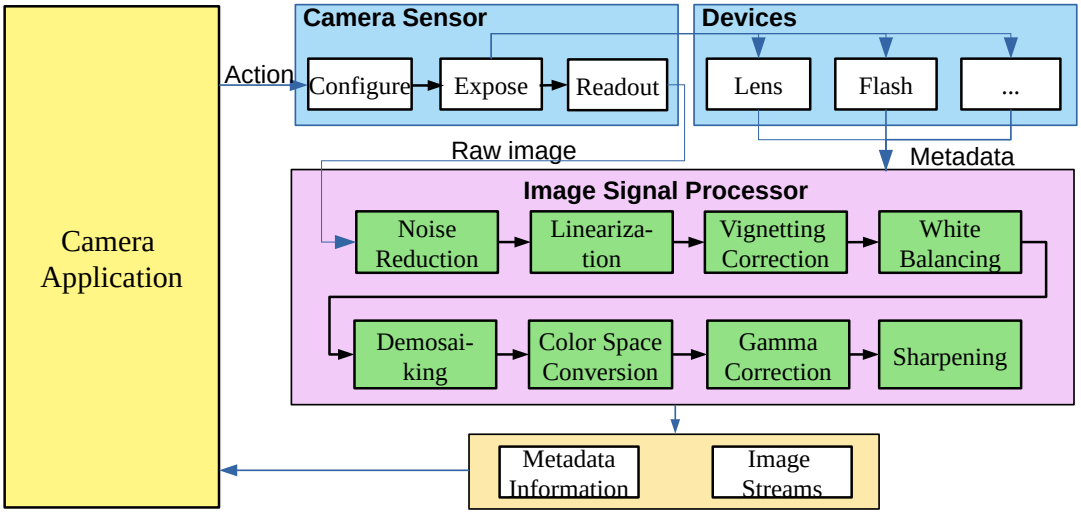


Figure 3.2: Image Processing Pipeline used in Smart Phone Camera

We use Image Signal Processor (ISP), a post-processing pipeline for digital cameras illustrated in Figure 3.2, to evaluate our approach. This application takes raw data from camera sensors, and applies a sequence of processing kernels to produce a color image. It is an interesting benchmark because it has various filters that naturally serve as kernels, and its primary quality metric is SSIM. We target up to four stages of the ISP pipeline; additional kernels make exploration time for global WLO too long. The four kernels are:

— *NLM*: Non-Local Means denoising filter [18]. This denoising stage takes means of 5 windows, weighted by the distance from the target pixel, to filter noise.

— *Demosaic*: Demosaicing reconstructs a full color image from sensor data that captures color information as a mosaic of primary colors. This stage is also expensive consisting of 7 filters of size 3×3 or 5×5.

— *GC*: Gamma Correction adjusts the brightness of the image to suit human eyes. The image is converted into gray scale to derive the amount of brightness correction, which is then applied to the image.

— *Unsharp*: Unsharp masking sharpens the image by masking it with its blur. It computes a blurred image with a two-pass (vertical + horizontal) filter to use as the mask. It also includes the conversion to/from `YCbCr` color space, since the filter is performed in `YCbCr`.

The number of effective variables to be optimized are 19 for *NLM* and *Demosaic* and 17 for *GC* and *Unsharp*. To evaluate our approach in solving the scalability problem, we create three experiments on ISP with increasing complexity for WLO. Depending on each case, some kernels are selected for WLO while others are kept at highest possible precision. In the first experiment, *NLM* and *Unsharp* are chosen for WLO. In the second experiment, the *Demosaic* kernel is added. The last experiment considers all four kernels.

We use two SSIM targets in our experiments: 0.9 and 0.99. Preserving SSIM = 0.99 is considered to have small impact on human perception, which is suitable for photos. To evaluate our approach, we also target SSIM = 0.9, which has a much larger solution space compared to the 0.99 case and which is representative of, e.g., video capturing.

3.5.3 Stereo Matching

The Stereo Matching (SM) algorithm is widely used in computer vision to extract the depth information from a scene. In most cases, SM uses two images taken from two cameras placed side by side and horizontally as the inputs, as shown in the left part of Figure 3.3. The output

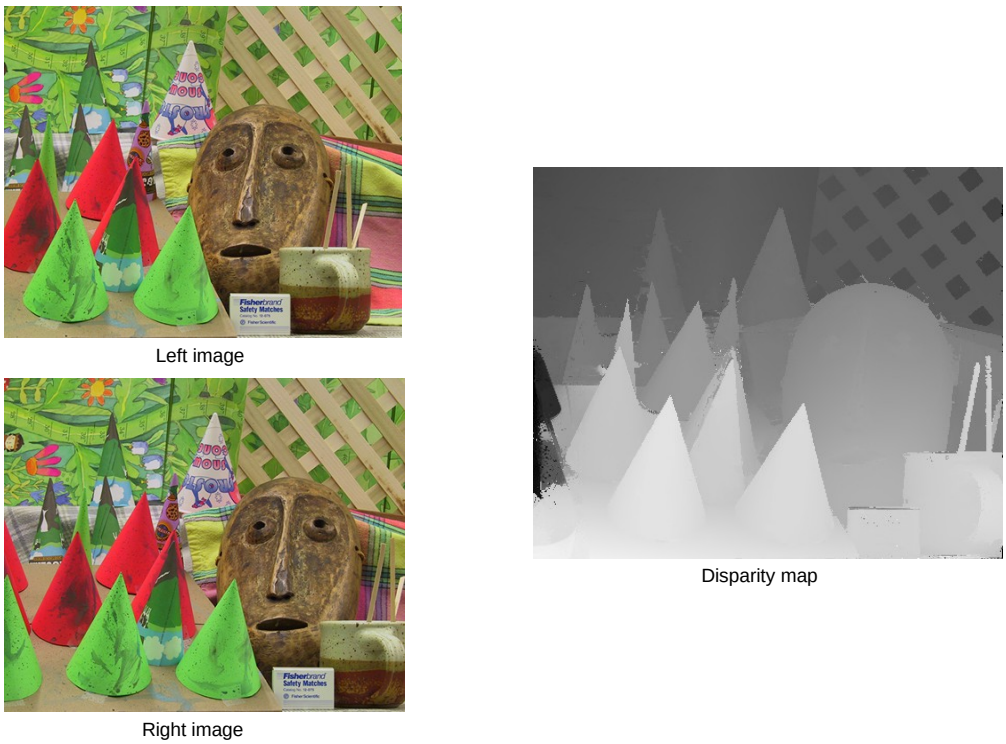


Figure 3.3: An example of Stereo Matching operation [19]

of a stereo matching algorithm is a disparity map whose size is the same as that of the original image but contains the depth information of each pixel. This disparity map (see right part of Fig-

ure 3.3) is obtained from the horizontal distance between corresponding pixels in the two images.

Figure 3.4 describes the computing flow graph of the Stereo Matching application in more details. The stereo matching method works by first choosing a pixel on the left image and then looking for an identical pixel on the right image. The left image pixel that was searched for represents the same actual location in the scenario. The disparity of a pixel is then determined by computing the horizontal distance between those two pixels. The pixel depths in the input images are determined by minimizing a cost function, which is related to its disparity. The cost function to minimize works out how much it would cost to match a pixel in the left image to a pixel in the right image. In Figure 3.4, after converting to gray images via *rgb2Gray*, the left and right gray images are passed through the *costConstruction* function. This function computes the cost of matching a pixel from the left image to a pixel from the right image with a specified disparity, which reflects the distance between the two pixels. This step is performed for every pixel and disparity level. To generate precise disparity maps, *aggregateCost* employs an adaptive weight approach to sum the matching costs of adjacent pixels and iterates in both the horizontal and vertical directions. The kernel *computeWeights* attempts to change the weight of each pixel based on the intensity of its neighbors and the geometric connection between the considered pixel and its neighbors. To put it simply, a pixel's proximity to the target pixel in terms of both intensity and distance should be given more consideration.

3.5.4 Empirically Constructed Models

It is difficult to evaluate these models since simple metrics, such as (Root) Mean Square Error, are not sufficient to assess their quality with respect to unseen data. In this work, these models are ultimately evaluated by the quality of the noise budgets derived. Figures 3.5 and 3.6 present the models constructed for ISP (excluding those that are hard to visualize) as a partial evaluation of their quality. Figure 3.7 illustrates the cost models constructed for Stereo Matching.

3.5.5 Exploration Time and Quality of Solution

Figure 3.8 and Table 3.1 summarize the exploration of GWLO and our approach for ISP and Stereo Matching (SM). The model construction time is significant due to the time consuming WLO algorithm, and hence our approach takes longer for smaller problems. However, the difference in scalability becomes clear with more kernels. For SM, our approach saves up to 23% of energy consumption while our execution time is saved up to 75% compared to GWLO approach.

The quality of the solutions improves for large problem instances. The explanation is that our approach performs local search around the derived noise budgets, finding better solutions (as explained with Figure 3.1). There are many local minima in the design space, and GWLO

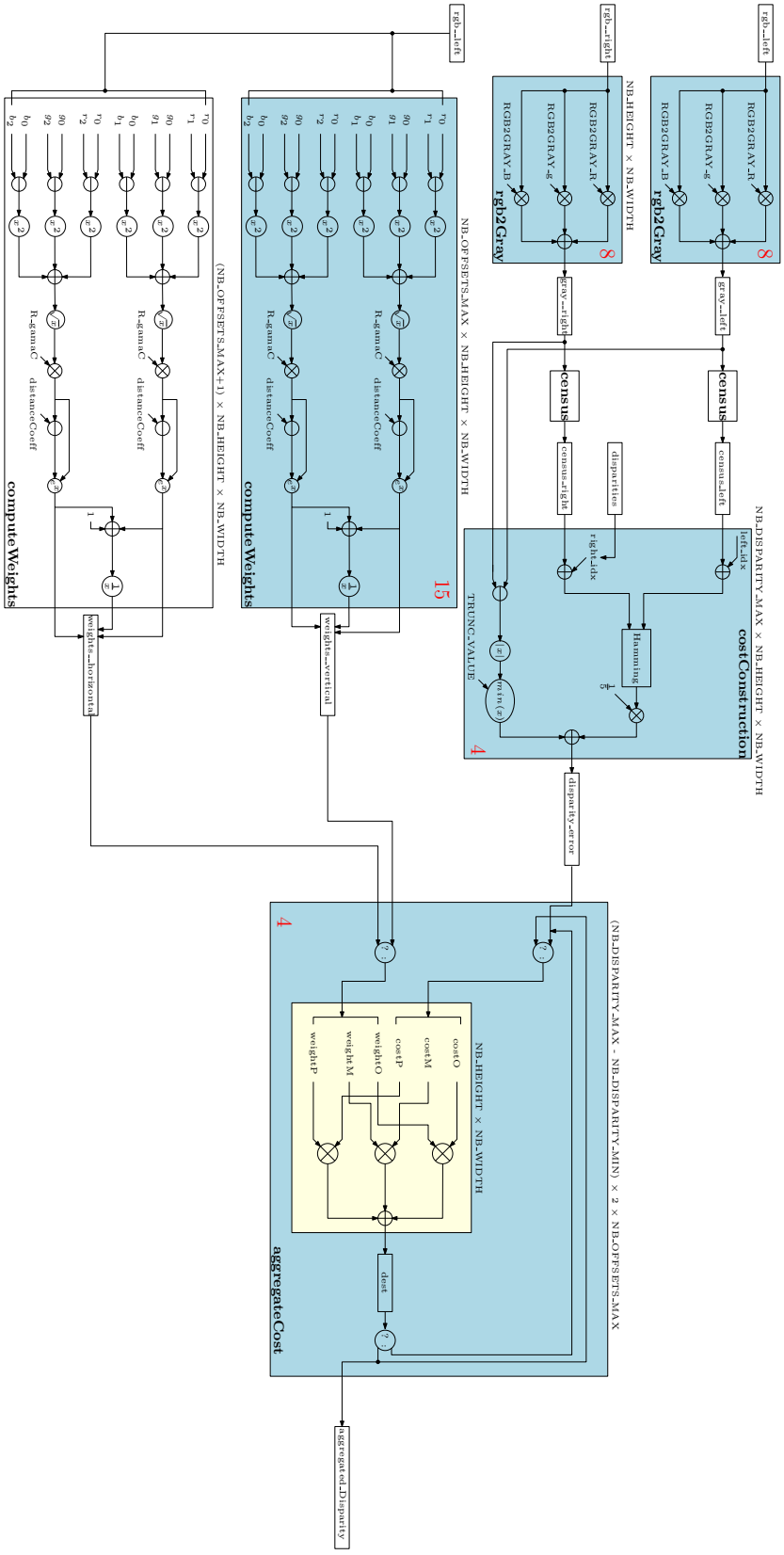
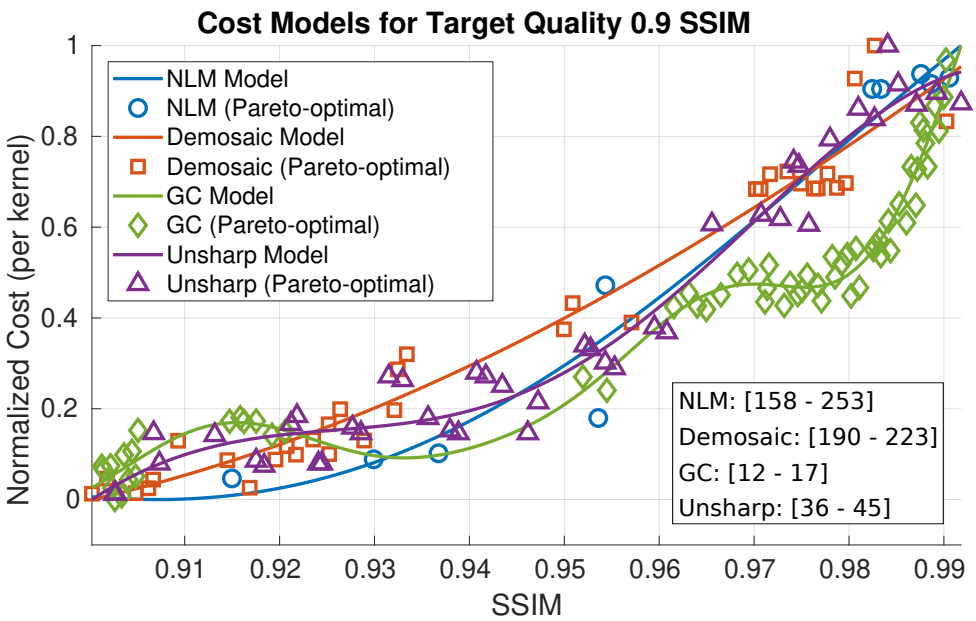
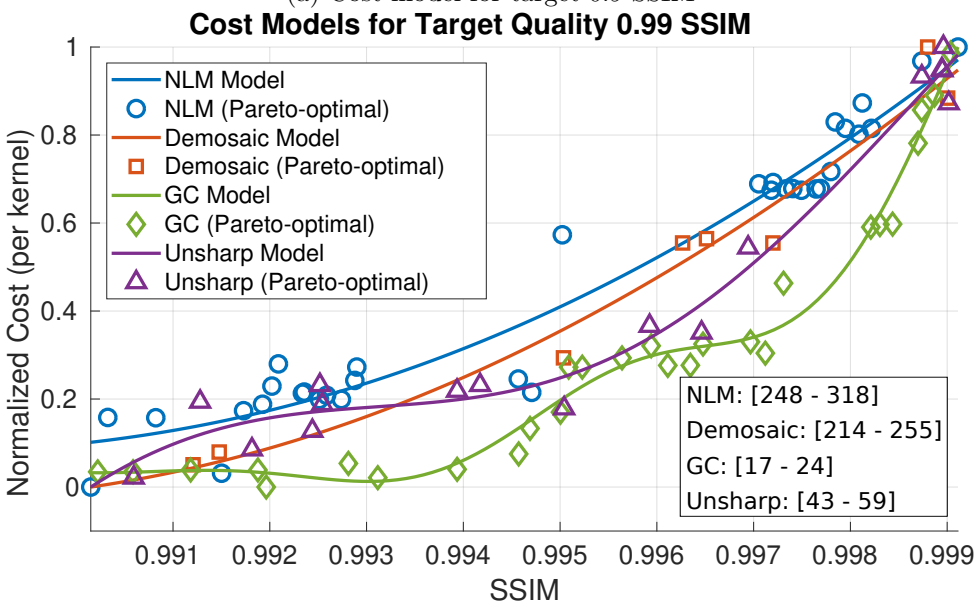


Figure 3.4: Computing blocks (kernels) of Stereo Matching. The kernels colored in cyan are the one optimized with fixed-point bit-widths. The kernel named *computeWeights* in white color uses the same word-length configurations as the one colored in cyan. The numbers colored in red are the number of variables of each kernel considered for the WLO problem. The computation operations described inside each box are the number of computations per pixel. These computations must be applied for the whole image with a certain size mentioned in the top of each kernel.

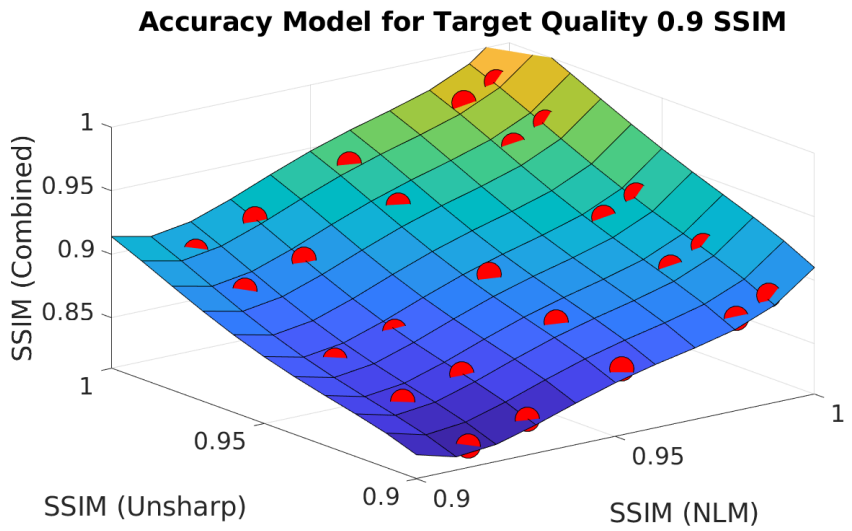


(a) Cost model for target 0.9 SSIM

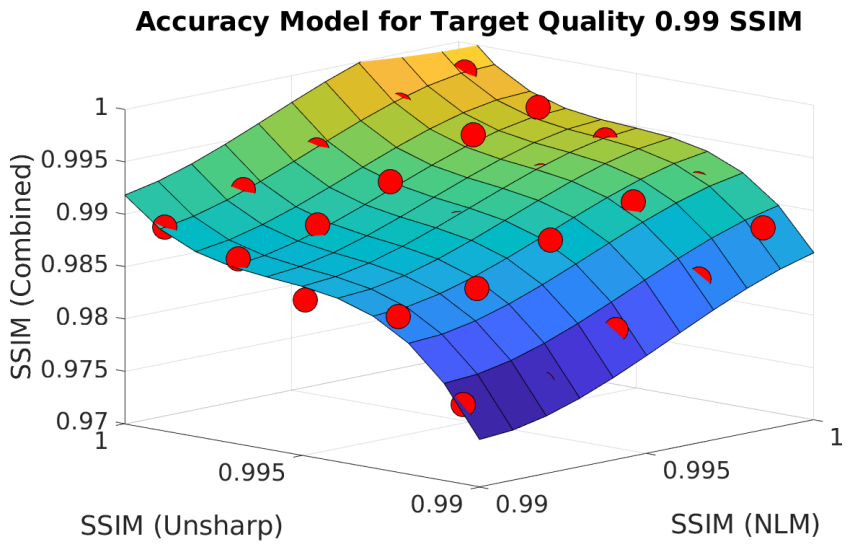


(b) Cost model for target 0.99 SSIM

Figure 3.5: Cost models for ISP constructed for each kernel using Gaussian Process. The costs are individually normalized for each kernel to present the models within a figure. The ranges of unnormalized energy cost (nJ) are shown at the bottom right.

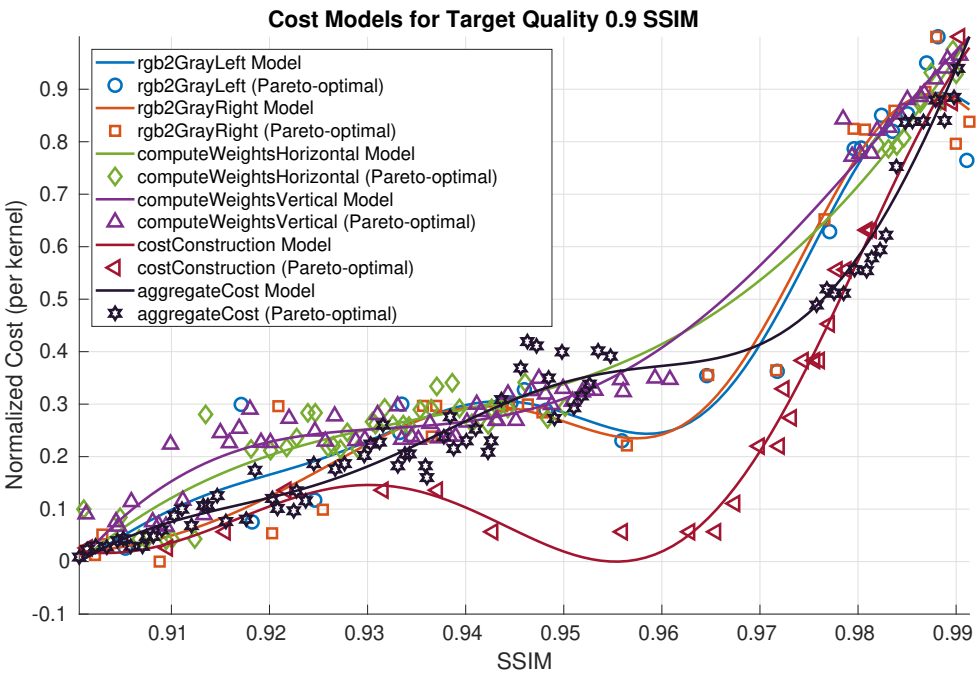


(a) Accuracy model for target 0.9 SSIM

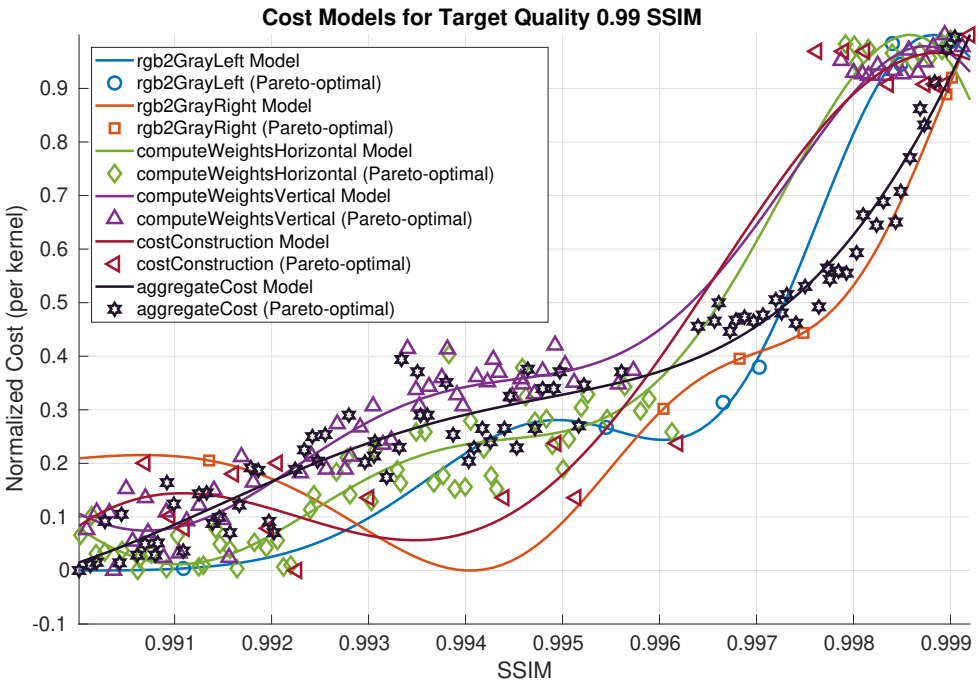


(b) Accuracy model for target 0.99 SSIM

Figure 3.6: Models for inter-kernel interaction of quality constraints in ISP. These models were constructed with least square fit with degree four and three polynomials for 0.9 SSIM and 0.99 SSIM cases, respectively.



(a) Cost model for target 0.9 SSIM (Stereo Matching)



(b) Cost model for target 0.99 SSIM (Stereo Matching)

Figure 3.7: Cost model construction for the Stereo Matching application with two different quality targets.

becomes more likely to be stuck with sub-optimal solutions for larger problems.

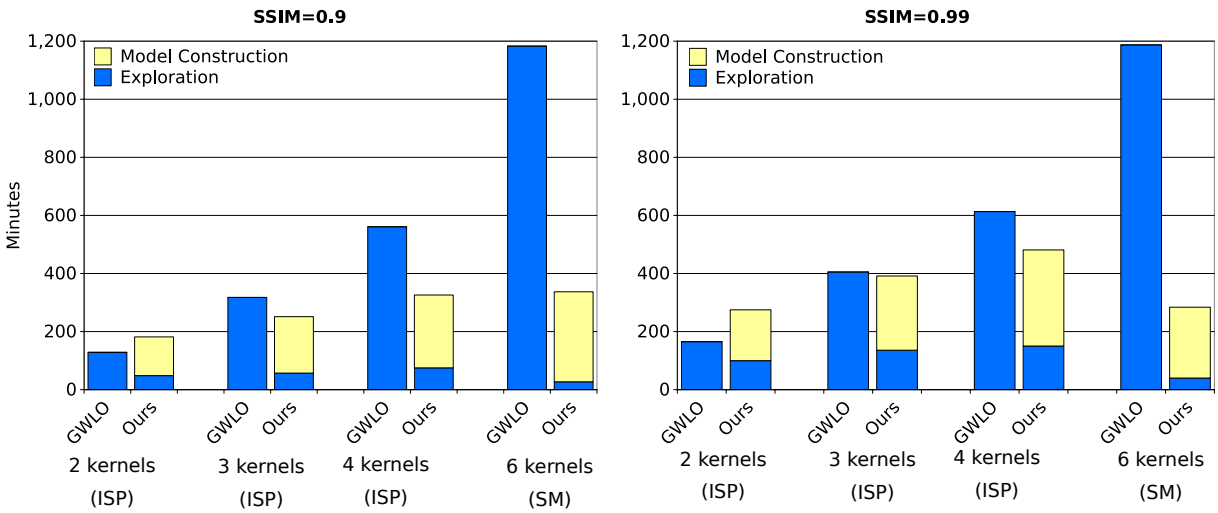


Figure 3.8: Comparison of exploration time for ISP and Stereo Matching (SM).

Table 3.1: Comparison of solutions for ISP and Stereo Matching (SM).

		Target 0.9 SSIM		Target 0.99 SSIM	
		SSIM	Cost (nJ)	SSIM	Cost (nJ)
2 Kernels (ISP)	GWLO	0.901	212	0.990	326
	Ours	0.915	207	0.991	309
3 Kernels (ISP)	GWLO	0.900	438	0.990	604
	Ours	0.906	427	0.990	587
4 Kernels (ISP)	GWLO	0.901	474	0.991	695
	Ours	0.907	444	0.990	612
6 Kernels (SM)	GWLO	0.904	6970	0.990	9183
	Ours	0.901	5311	0.994	8185

We have also compared our solutions to those that could be found without empirical models. A number of combined solutions are simulated during the accuracy model construction, which may already include a good design. In such cases, there is no need to perform further exploration, i.e., finding these solutions take the same time as model construction in our approach. We observed that for some accuracy targets, this is indeed the case. For some other accuracy targets, shown in Figure 3.9, more than 15% improvement in cost may be realized by using the derived noise budgets. These are instances that support our claim in Section 3.2.2.

3.5.6 FIR and IIR Filters

We have also applied our approach with cascaded FIR and IIR filters to test how it works for another quality metric and linear systems. FIR is decomposed into 2 kernels with 9 and 6 effective variables. IIR is partitioned into 3 kernels with 12, 12, and 7 variables. We used

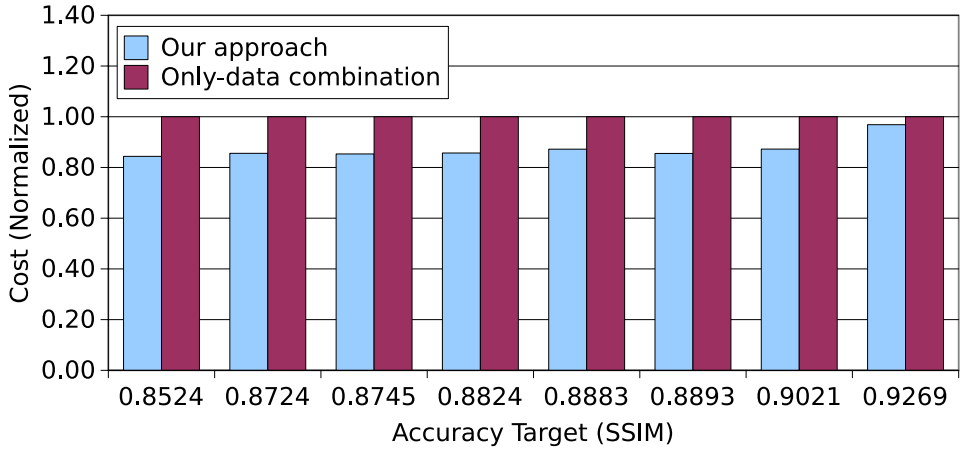


Figure 3.9: The quality of solutions by our approach compared to the best combination of the configurations used for accuracy model construction. These results are for ISP with 4 kernels.

PSNR equal to 50 dB and 60 dB as quality targets. The cost of solutions found by GWLO and our approach are shown in Table 3.2. The cost of our solutions were slightly worse than that of GWLO's solutions, with about 4% for FIR and 2% for IIR. The solution space of FIR and IIR is considered as relatively small and with few sub-optimal solutions compared to ISP. Thus, with considering all kernels during the optimization time, GWLO can find a good solution and avoid local minima. Figure 3.10 summarizes the exploration time between GWLO and our approach for FIR and IIR with constraints of 50 dB and 60 dB. The important result is that the exploration time of our approach scales much better than GWLO, and that the overall behavior is consistent with ISP.

Table 3.2: Comparison of solutions for FIR and IIR.

		Target 50 dB		Target 60 dB	
		PSNR (dB)	Cost (nJ)	PSNR (dB)	Cost (nJ)
FIR	GWLO	50.0	64.15E-04	60.1	72.95E-04
	Ours	50.2	66.94E-04	60.6	73.76E-04
IIR	GWLO	50.2	9.22E-04	60.1	10.66E-04
	Ours	50.6	9.60E-04	60.7	10.91E-04

3.6 Conclusion

In this chapter, we presented our proposed approach that uses empirically constructed models to solve the generality and scalability problems of WLO in large applications. The key idea in our approach is to characterize the impact of approximating each kernel to accuracy/cost through an empirical model. We show that for sufficiently large applications that justify the time spent on modeling, our approach can significantly reduce exploration time *and* improve the

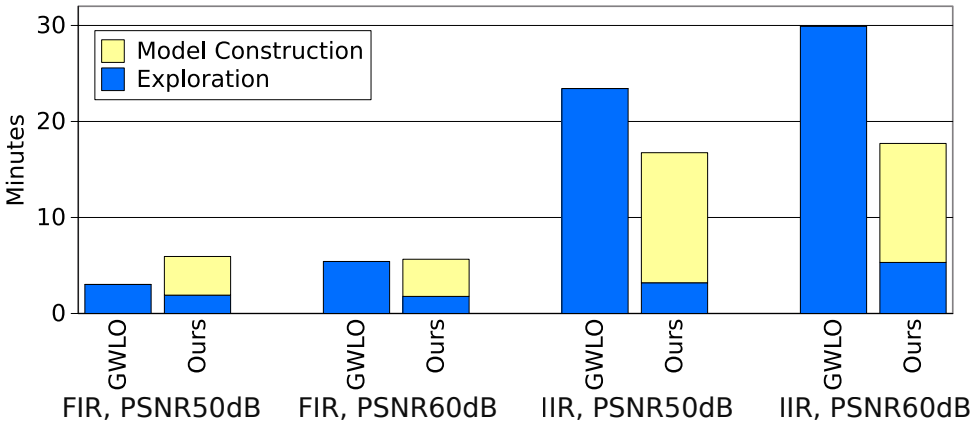


Figure 3.10: Comparison of exploration time for filter applications.

quality of the solutions.

In the future work, this approach can be extended to predict the noise budget for multiple accuracy targets with the model built once. The quality of the solution given the noise budget depends on the available data used for the model construction.

Acknowledgment

This work was supported in part by the French National Research Agency; ARTEFaCT project (ANR-RF-2015-01). We thank Silviu-Ioan Filip for his help with regression techniques.

Bibliography

[1] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger, "Accelerating fixed-point design for MB-OFDM UWB systems," https://www.design-reuse.com/articles/9559/, 2005.

[2] T. Arslan and D. H. Horrocks, "A genetic algorithm for the design of finite word length arbitrary response cascaded iir digital filters," in *Proceedings of the Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1995, pp. 276–281.

[3] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *Proceedings of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, 2001, pp. 53–56.

[4] S.-C. Chan and K. M. Tsui, "Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 4, pp. 845–854, 2007.

[5] H. Choi and W. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proceedings of 1994 IEEE Workshop on VLSI Signal Processing*, 1994, pp. 198–207.

[6] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1432–1442, 2003.

[7] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1561–1570, 2008.

[8] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for CDMA demodulator," in *Proceedings of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, 2001, pp. 290–293.

[9] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[10] H.-N. Nguyen, D. Menard, and O. Sentieys, "Novel algorithms for word-length optimization," in *Proceedings of the 19th European Signal Processing Conference*, 2011, pp. 1944–1948.

[11] K. N. Parashar, D. Menard, and O. Sentieys, "A polynomial time algorithm for solving the word-length optimization problem," in *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 638–645.

[12] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[13] J. Chung and L.-W. Kim, "Bit-width optimization by divide-and-conquer for fixed-point digital signal processing systems," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3091–3101, 2015.

[14] D. Novo, I. Tzimi, U. Ahmad, P. Ienne, and F. Catthoor, "Cracking the complexity of fixed-point refinement in complex wireless systems," in *Proceedings of the IEEE International Workshop on Signal Processing Systems (SiPS)*, 2013, pp. 18–23.

[15] L. N. Trefethen, *Approximation theory and approximation practice.* SIAM, 2013.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* MIT press Cambridge, MA, 2005.

[17] A. Floc'h *et al.*, "Gecos: A framework for prototyping custom hardware design flows," in *IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2013, pp. 100–105.

[18] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ser. CVPR'05, vol. 2, June 2005, pp. 60–65.

[19] "Patch match stereo," https://github.com/ethan-li-coding/PatchMatchStereo.

Chapter 4

Leveraging Bayesian Optimization to Speed Up Automatic Precision Tuning

Using just the right amount of numerical precision is an important aspect for guaranteeing performance and energy efficiency requirements. Word-Length Optimization (WLO) is the automatic process for tuning the precision, i.e., bit-width, of variables and operations represented using fixed-point arithmetic. However, state-of-the-art precision tuning approaches do not scale well in large applications where many variables are involved. In this chapter, we propose a hybrid algorithm combining Bayesian optimization (BO) and a fast local search to speed up the WLO procedure. Through experiments, we first show some evidence on how this combination can improve exploration time. Then, we propose an algorithm to automatically determine a reasonable transition point between the two algorithms. By statistically analyzing the convergence of the probabilistic models constructed during BO, we derive a stopping condition that determines when to switch to the local search phase. Experimental results indicate that our algorithm can reduce exploration time by up to 50%-80% for large benchmarks.

4.1 Introduction

The rapid development in scientific and technological innovations during the last decade opens a new era for intelligent and sophisticated systems. The demand for integrating many large applications in a limited silicon area poses new challenges for energy-efficient computing. Recently, Approximate Computing (AC) is considered as a good solution to address energy efficiency issues. The primary objective of approximation techniques is to trade quality of service for cost saving. One of the popular AC techniques is to use Fixed-Point arithmetic for low-precision computation in Digital Signal Processing or Machine Learning systems. This technique always requires a floating-point to fixed-point conversion that optimizes the fixed-point word-lengths for a good compromise between cost and quality requirement. This procedure, called Word-Length Optimization (WLO), accounts for 25-50% of design time [1] and is still considered as a problem

of interest to reduce time-to-market.

Approaches to address WLO can be classified in two groups: analytical and simulation-based approaches. Analytical approaches relax the WLO problem for convexity and then apply some convex optimization algorithms to directly obtain the optimal solution [2, 3]. Despite handling WLO quickly, these approaches require the accuracy to be modeled as convex functions, which cannot be analytically constructed in general. Simulation-based approaches solve WLO by iterative search using simulations [4, 5, 6, 7, 8]. They are thus generic with all systems and quality metrics. However, these approaches do not scale well in large applications where many variables are involved since the number of simulations increases dramatically with the number of variables. In addition, most of these iterative searches are based on local search that moves with a short distance in the discrete domain. Hence, convergence speed is slow if the initial point is far from a local minima.

Bayesian Optimization (BO) is a popular approach for tuning hyper-parameters in machine learning algorithms [9]. This approach aims to optimize problems where the mathematical expression of the target function is unknown and without derivatives. BO constructs a probabilistic model based on past samples to suggest new points [10]. Thanks to this model, at a certain state, BO can ignore neighboring points if they produce a low probability of being good solutions to search more quickly. This feature might overcome weakness of the simulation-based approaches. However, there is no study yet indicating the performance of Bayesian optimization for WLO.

In this chapter, we propose a hybrid algorithm combining Bayesian optimization and a local search to improve the scalability of simulation-based approaches. We first show how this combination can lead to large improvements in exploration time. Then, we design a reasonable transition point between the two approaches to leverage the efficiency of the combination, which is also the core of our approach. Using design points sampled by BO, we derive a statistical metric to evaluate the convergence of the models during BO. Experimental results show that our hybrid algorithm outperforms latest simulation-based approaches, reducing exploration time by up to 50%-80%, while leading to similar cost solutions.

The rest of the chapter is organized as follows. We introduce necessary background and discuss related work in Section 4.2. A motivation for our work is presented in Section 4.3. Then, we describe our proposed hybrid approach in Section 4.4. We show the performance of our approach in a comparison with different benchmarks and latest approaches in Section 4.5 followed by a conclusion in Section 4.6.

4.2 Background and Related Work

In this section, we introduce the WLO problem and discuss earlier work before presenting Bayesian optimization.

4.2.1 Word-Length Optimization and Classical Approaches

A number represented in fixed-point arithmetic contains integer and fractional word-lengths (WLs), represented on I and F bits, respectively. The integer WL covers the dynamic range whereas the fractional WL controls the precision. In this work, we focus on the WLO for fractional WL, which is the time consuming part of the exploration. Let the vector $\boldsymbol{W} = [W_0, W_1, \ldots, W_{N-1}]$ denote a word-length configuration with N effective variables to be explored for fixed-point conversion. The main objective of WLO is to determine a good-enough word-length configuration that minimizes a cost function under a quality constraint:

$$\min \boldsymbol{C}(\boldsymbol{W}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W}) \geq \boldsymbol{\lambda_{obj}} \tag{4.1}$$

where $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are functions that express cost and quality, respectively, and the quality target is given as $\boldsymbol{\lambda_{obj}}$. How the functions $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are realized varies across work, ranging from analytical models to simulation-based approaches.

Some approaches [2, 3, 11] construct analytical models for noise power through mathematical expression to avoid costly simulations during the exploration process. These analytical approaches take advantage of a property of errors, under the hypothesis of linear and time-invariant (LTI) systems, that their propagation do not interfere with each other. Hence, the error introduced at a noise source may be propagated through the system independently and aggregated afterwards. Thus, these approaches cannot be directly extended to handle general non-LTI programs. Moreover, complex quality metrics, such as Structural Similarity (SSIM), which are not directly related to noise power, are hard to model analytically.

Many simulation-based approaches [7, 4, 6, 5, 8] were proposed based on iterative search using heuristics to address WLO. These approaches use variants of gradient descent algorithms that evaluate neighboring solutions which differ by one or few bits from the current solution at each iteration. Since the number of neighboring solutions increases with the number of variables, the number of solutions that must be evaluated at each iteration quickly increases with the complexity of the application. Additionally, due to short movements at each iteration (one or few bits), these approaches require many iterations to converge if the starting point is far from a local minima. These are the main drawbacks causing a huge number of simulations especially in large applications and hence leading to an extremely long exploration time. Min+1, Min+b [4, 6]

and Max−1 [4] are typical approaches of this group. The Min+1 (resp. $+b$) algorithm begins with a design where each variable is assigned by its minimum WL (MWL), i.e., the WL satisfying the quality target when other variables are set to the highest precision. At each iteration, the algorithm moves towards neighboring candidates by increasing by 1 (resp. b) bit(s) the best variable among the candidates until obtaining a solution satisfying the quality constraint. By contrast, Max−1 initializes variables with the highest possible WL and decreases variables by 1 bit until to reach the final solution.

Afterwards, many variants of these two procedures were proposed. Heuristic approaches [12] or Hybrid combinations of Min+b and Max−1 outperform the original algorithms [13]. Likewise, the Greedy Randomised Adaptive Search Procedure (GRASP) [8] combines local search and stochastic optimisation. GRASP is an iterative two-phase procedure. In the construction phase, the search algorithm (similar to Min+1) randomly selects one of the best neighboring candidates during gradient descent. Then, a Tabu search is applied to refine the solution found by the first phase. Tabu allows movements in both directions (+1/−1) and uses a Tabu list to skip some explored variables. These two phases are iterated and the randomization of the construction phase avoids to stay in local minima.

Some noise budgeting techniques [14, 15] decompose large applications into smaller kernels to break down the exponential complexity of WLO. However, these methods still face the limitation of the classical above-mentioned approaches, which are used for each kernel to solve local WLO problems.

4.2.2 Bayesian Optimization

Bayesian Optimization (BO) is a machine-learning-based optimization method [9] aiming to optimize functions which usually have no mathematical expression and/or derivatives. Several BO frameworks are popularized as open sources such as Spearmint [1] [16], SMAC [2] [17] and Hyper-opt [3] [18]. and BOHB [4] [19] Despite being widely applied in many real world problems, there is no study of BO for WLO. Generally, BO has two key elements: i) a probabilistic surrogate model for modeling the unknown objective function based on already observed samples and ii) an acquisition function that optimizes over the surrogate model to suggest next samples. One main difference in BO methods is the process of selecting the surrogate models. While Gaussian Processes (GP) are often used for continuous-domain moderate-size problems, tree-based models like Random Forest and Tree-structured Parzen Estimator (TPE) facilitate discrete-domain large-size problems [20].

1. `https://github.com/JasperSnoek/spearmint`
2. `https://www.cs.ubc.ca/labs/beta/Projects/SMAC`
3. `https://github.com/hyperopt/hyperopt`
4. `https://github.com/automl/HpBandSter`

TPE works by identifying points that could have been drawn, and that appear promising on the basis of the evaluation of a loss function at other points. In detail, TPE models two density functions $p(\mathbf{X}|y < \alpha)$ and $p(\mathbf{X}|y \geq \alpha)$ where $\mathbf{X}$ is a set of hyper-parameters, y is the value of loss function $f(\mathbf{X})$ and α is a threshold that separates bad and good samples. It chooses value of $\mathbf{X}$ that maximizes the ratio $\frac{p(\mathbf{X}|y<\alpha)}{p(\mathbf{X}|y\geq\alpha)}$. Hyperopt [18] is a common framework for TPE. The loss function is very important to guide the optimizer, as detailed in Section 4.4.1. In many cases, it is more efficient to optimize on the loss function instead of the original one because BO can evaluate how far away from the current solutions to the optimal solution by following the gradient from the loss function.

Unlike gradient-based algorithms, BO inspects past iterations to construct the interplay among variables to evaluate a loss function via a probabilistic model. Based on the knowledge from this model, BO is able to evaluate and ignore neighboring designs which have low probability of being good solutions to search more globally. As a result, it can speedup the search and better avoid local minima if current state is on a plateau or a bad local minima. Thus, BO is a promising candidate to overcome limitation of classical simulation-based WLO approaches. However, the computational complexity of BO is quadratic $O(i^2)$, whereas most heuristic approaches have a linear complexity $O(i)$, i being the number of iterations. Thus, BO still faces scalability issues, especially for large applications in which BO requires more evaluations to obtain a good solution. Based on their features and properties, we propose a new method that combines Bayesian optimization and local search with a reasonable transition condition to improve the exploration efficiency. TPE is selected for BO because it is suited to discrete problems as WLO, and Tabu, a state-of-the-art WLO method, is used for local search.

4.3 Motivations

In this section, we compare the performance of TPE with Tabu combined with Min+1 gradient descent for WLO. Then, we indicate benefits of combining TPE and Tabu via empirical evidence, which motivates the proposed hybrid approach presented in Section 4.4. We used an Infinite Impulse Response (IIR) filter with Peak Signal to Noise Ratio (PSNR) equal to 40 dB as a quality constraint. Note that the trends in this section are consistent with the other benchmarks used in Section 4.5.2.

4.3.1 Performance Analysis: TPE vs. Tabu

Figure 4.1 shows the search process of Tabu and TPE during WLO of the IIR. Tabu searches in a narrow range constituted by near neighboring solutions which are some bits different to each

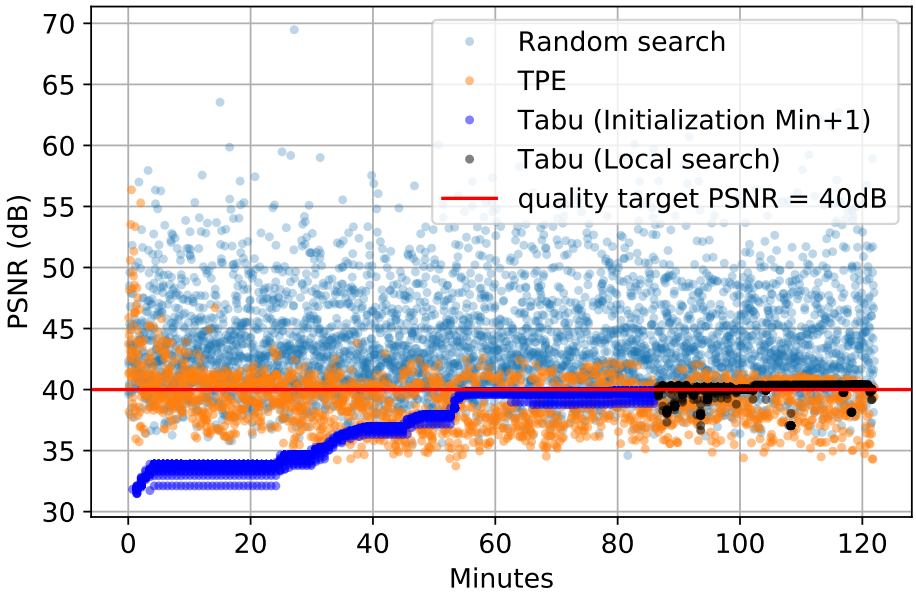


Figure 4.1: Search process of Tabu and TPE. Random search is used as a reference. Points correspond to quality of different designs evaluated in the search process (for Tabu, all neighboring solutions in a iteration are plotted at a corresponding time). Tabu has two procedures: i) **Initialization** uses Min+1 to search in infeasible region until a solution satisfying the quality target is found and ii) **Local search**, a combination of Min+1 and Max-1, optimizes locally around the quality target.

other. Thus, it moves slowly towards optimal region. At some points in the initialization procedure, the search takes a significant period to surpass the plateau region which has no quality improvement. As a result, local search must wait for a long time and is only started when a feasible solution is found in the initialization procedure. Meanwhile, TPE first searches randomly in the solution space (very first solutions scattered in around 35-55 dB). Afterwards, it explores solutions using probabilistic models in a wide range and quickly finds feasible solutions around the quality target. Note that by using models, TPE can stick around the quality target to enhance the search efficiency instead of whole solution space as random search.

For cost comparison (see Section 4.5.1 for more details on cost), Figure 4.2 shows the best solution obtained so far by TPE as a function of exploration time and the final solution obtained by Tabu. TPE converges quickly in the beginning. However, the cost is improving slowly for latest iterations, taking up a large portion of the total execution time. With this behavior, we emphasize the interest of stopping TPE pruning phase to switch to a more efficient local search, such as Tabu.

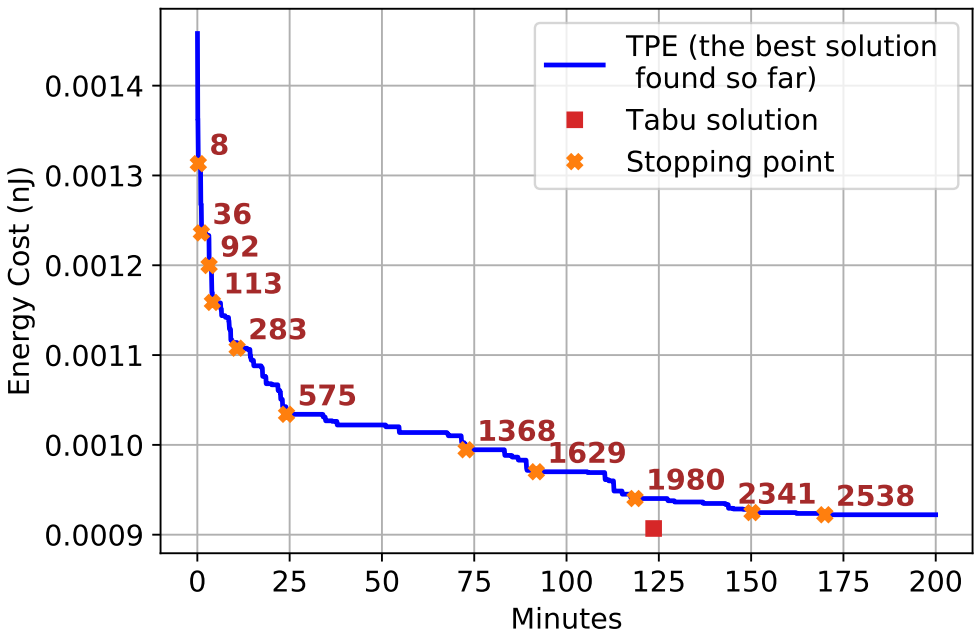


Figure 4.2: Manually selected stopping points on the best solution obtained so far by TPE. The curve was constructed by best designs, satisfying the quality target with minimum cost, at a certain time. The numbers indicate index of the selected points.

4.3.2 Initial Combinations of TPE and Tabu

From the performance analysis of TPE and Tabu, we empirically perform initial combinations of the two algorithms. TPE is stopped at a certain moment returning a temporary configuration, defined as a *transition point*, which is then served as starting point for Tabu. Among the designs explored by TPE, we use only the best solution obtained so far. These transition points (orange crosses in Figure 4.2) are selected if they have a significant improvement in terms of cost compared to the previous adjacent selected one. This ensures a high chance of having different WL configurations.

In Figure 4.3, we compare the performance of Tabu with TPE+Tabu for different transition points. The cost of the final solutions depends on the moment when TPE is stopped. Stopping TPE too early (H-8 to H-283) does not lead to good solutions compared to Tabu alone, even though their convergence speed is much faster. Stopping TPE after a certain amount of time (H-578, H-1368) can lead to an equivalent or even better solution cost, still in a shorter time. The benefit of the superior convergence speed is gradually decreased if TPE is stopped too late (H-1629 to H-2538). It is important to notice that, except for the early ones, the choice of the exact transition point does not impact much the cost of the final solution.

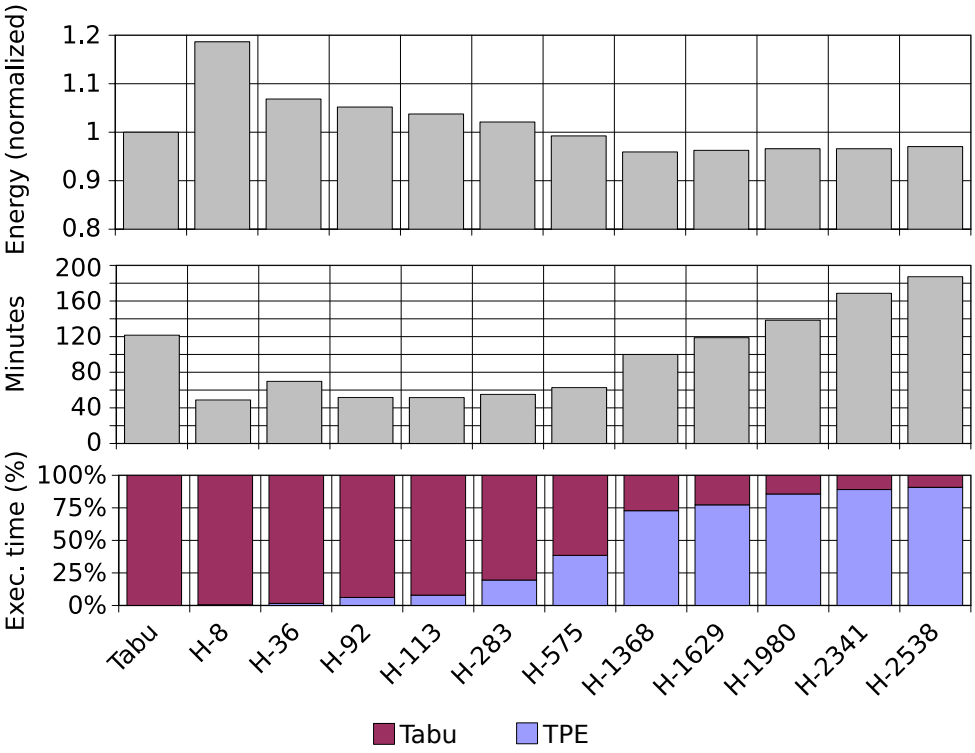


Figure 4.3: A comparison in energy cost (top), total exploration time (middle) and separated exploration time (bottom) of Tabu and different combinations between TPE and Tabu given the selected transition points. The x-axis separates the data for different cases; H-X represents for the different combinations where X is the index of the stopping point of TPE.

From this analysis, we highlight that TPE can quickly prune the design space and find a solution from which Tabu will be able to fine-tune the cost, resulting in a significant reduction in exploration time. This reduction depends on choosing the right transition point, which is not an obvious task especially in an automated way, and is the aim of the next section.

4.4 Proposed Hybrid Approach

In this section, we construct the loss function used in our Bayesian optimization and define a method to automatically find the transition point between TPE and Tabu.

4.4.1 Loss Function

Derived from the original WLO problem of Eq. 4.1, we construct the loss function using the Lagrangian form as

$$f(\boldsymbol{W}) = \boldsymbol{C}(\boldsymbol{W}) - \alpha(\boldsymbol{\lambda}(\boldsymbol{W}) - \boldsymbol{\lambda_{obj}}), \tag{4.2}$$

with a positive and big enough α. TPE will then follow $f(\boldsymbol{W})$ to find solutions that minimize the original problem.

Based on many WLO experiments, we experienced that the cost and quality functions tend to be proportional to $\boldsymbol{W}$. A high quality solution mostly corresponds to a high cost. Thus, the best solutions to satisfy the constraint with a small cost are likely to be around the quality target $\boldsymbol{\lambda_{obj}}$. Therefore, for faster convergence, we force the loss function to cover only a narrow range $[q_l, q_h]$ around $\boldsymbol{\lambda_{obj}}$. The loss function is then defined as

$$f(\boldsymbol{W}) = \begin{cases} \boldsymbol{C}(\boldsymbol{W}) - \alpha(\boldsymbol{\lambda}(\boldsymbol{W}) - \boldsymbol{\lambda_{obj}}) & \text{if } \boldsymbol{\lambda}(\boldsymbol{W}) \in [q_l, q_h] \\ +\infty & \text{otherwise} \end{cases} \tag{4.3}$$

Moreover, to penalize solutions below $\boldsymbol{\lambda_{obj}}$ with regards to solutions above the target, we choose $\alpha = 0.5$ for solutions below $\boldsymbol{\lambda_{obj}}$, and $\alpha = 0$ for other solutions in the range $[q_l, q_h]$.

4.4.2 Transition Point for Hybrid Approach

In Bayesian optimization relying on TPE [10], hyper-parameters –WL configurations $\boldsymbol{W}_i$ in our case– are chosen uniformly over the search space and evaluated by the loss function $f(\boldsymbol{W})$. Then, obtained samples $\{\boldsymbol{W}_i, f(\boldsymbol{W}_i)\}$ stored in an observation history $\boldsymbol{\mathcal{H}}$ are divided into two groups. The first group contains *good* samples where the loss f_i is less than a threshold γ^*, whereas the second group consists of the remaining, considered as *bad* samples. TPE uses these two sample groups to model two likelihood probability density functions $l(\boldsymbol{W}) = p(\boldsymbol{W}|f(\boldsymbol{W}) <$

γ^*) and $g(\boldsymbol{W}) = p(\boldsymbol{W}|f(\boldsymbol{W}) \geq \gamma^*)$, respectively. Then, it decides which hyper-parameter to try in the next iteration by maximizing the ratio

$$\frac{l(\boldsymbol{W})}{g(\boldsymbol{W})} = \frac{p(\boldsymbol{W}|f(\boldsymbol{W}) < \gamma^*)}{p(\boldsymbol{W}|f(\boldsymbol{W}) \geq \gamma^*)}.$$

Algorithm 2 describes TPE algorithm with a stopping condition checked at each iteration. The algorithm stops if the stopping condition is satisfied and then returns a WL configuration $\boldsymbol{W}_{stop}$ which serves as the starting point of Tabu. How the condition is constructed is described as follows.

At the tth iteration, we consider the vector $\boldsymbol{S_L^t}$ comprising the L top-performing solutions from the observation history $\boldsymbol{\mathcal{H}}$. Our main objective is to evaluate if the next iteration of TPE can lead to a new top-performing solution, that is significantly different in terms of WL configuration compared with the solutions in $\boldsymbol{S_L^t}$. Then, we decide to stop TPE if the solutions in $\boldsymbol{S_L^t}$ share similar WL configurations. Otherwise, we still wait for further exploration iterations. The similarity of WL configurations in $\boldsymbol{S_L^t}$ reflects that TPE is likely to converge to a local region formed by a number of similar WL solutions, which only differ by few bits. Thus, continuing to spend time with TPE is likely to be inefficient. Instead, a local search like Tabu is more reasonable to converge quickly in that local region. This is the key idea in our method for the combination. We use TPE for narrowing the solution space down to a good region, from which Tabu will continue the search for fine-tuning to provide the final solution following its greedy gradient.

Each solution in $\boldsymbol{S_L^t}$ is represented by a WL configuration with N effective variables. Let the $L \times N$ matrix $\boldsymbol{W_t}$ contain WL configurations of the L top performing solutions at iteration t. Elements $w_{i,j}^t$ of $\boldsymbol{W_t}$ represent the number of bits of variable $j \in [1, N]$ from solution $i \in [1, L]$. We use the standard deviation to evaluate the distribution of WL values for each variable (i.e.,

Algorithm 2 TPE with stopping condition

```
1: H ← {}
2: for i ∈ [1, ..., T] do
3:     W* = argmax l(W)/g(W)
4:     Evaluate f(W*)
5:     H ← H ∪ (W*, f(W*))
6:     if stopping condition is satisfied then
7:         Return W_stop
8:     end if
9:     Update l(W) and g(W) given H
10: end for
```

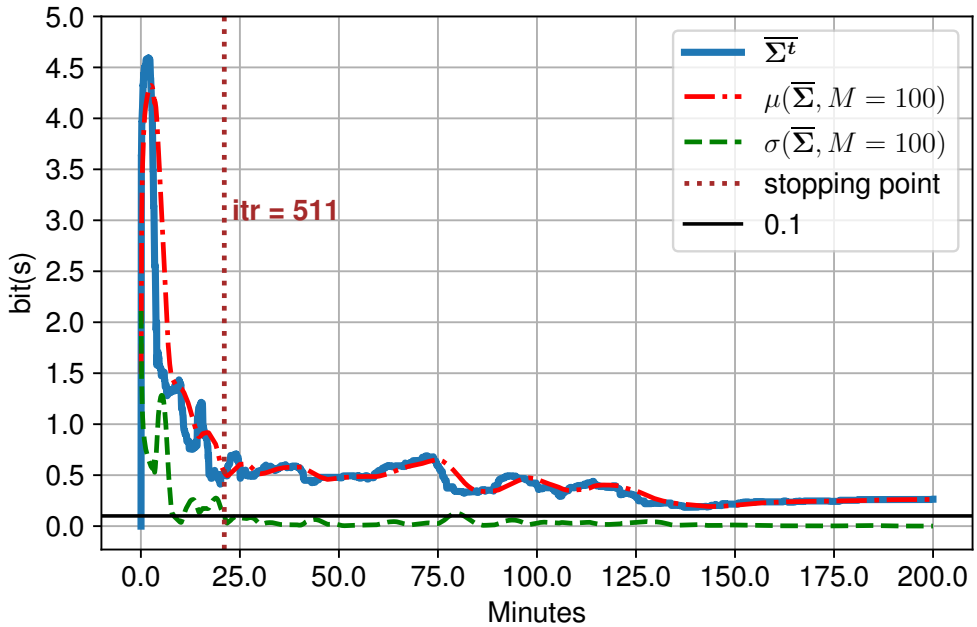


Figure 4.4: Stopping point for IIR, quality target PSNR = 40dB.

each column of $\boldsymbol{W_t}$) as

$$\boldsymbol{\Sigma_t} = [\sigma_0^t, \sigma_1^t, \ldots, \sigma_{N-1}^t], \tag{4.4}$$

where σ_j^t is the individual standard deviation of WL values of the jth variable and is calculated as

$$\begin{aligned} \sigma_j^t &= \sqrt{\frac{1}{L-1}\sum_{i=0}^{L-1}(w_{i,j}^t - \overline{W_j^t})}, \\ \overline{W_j^t} &= \frac{1}{L}\sum_{i=0}^{L-1} w_{i,j}^t. \end{aligned} \tag{4.5}$$

The smaller the standard deviation, the more similar the solutions. To evaluate the similarity on all WL configurations at a certain iteration, we aggregate standard deviations by the average of all σ_j^t with $j \in [0, N-1]$ as

$$\overline{\boldsymbol{\Sigma^t}} = \frac{1}{N}\sum_{i=0}^{N-1} \sigma_i^t. \tag{4.6}$$

$\overline{\boldsymbol{\Sigma^t}}$ represents the average difference on the number of bits for the WL configurations of the L top-performing solutions. By evaluating the convergence of TPE through $\overline{\boldsymbol{\Sigma^t}}$ at each iteration, we obtain the same behavior on all our benchmarks. In Figure 4.4, we show an illustration of this behavior for the IIR example. In the beginning, the curve of $\overline{\boldsymbol{\Sigma^t}}$ varies dynamically with a high variance, around 4.5 bits. It constantly finds new good solutions which are relatively different to each other, resulting in dissimilar WL configurations in $\boldsymbol{S_L^t}$ and then producing a high value of $\overline{\boldsymbol{\Sigma^t}}$. Afterwards, it enters a more stable area with a value of $\overline{\boldsymbol{\Sigma^t}}$ less than 1 bit. This indicates

that TPE is likely to converge to a local region. So, we should stop TPE and switch to a greedy algorithm like Tabu to search locally around this region. Based on that behavior, we derive two indicators to detect the stable region to decide to stop TPE. At the tth iteration, we evaluate the stability of $\overline{\mathbf{\Sigma}^t}$ in the M latest samples via mean and standard deviation as

$$\mu(\overline{\mathbf{\Sigma}}, M) = \frac{1}{M} \sum_{i=t-M-1}^{t} \overline{\mathbf{\Sigma}^i} \tag{4.7}$$

$$\sigma(\overline{\mathbf{\Sigma}}, M) = \sqrt{\frac{1}{M-1} \sum_{i=t-M-1}^{t} (\overline{\mathbf{\Sigma}^i} - \mu(\overline{\mathbf{\Sigma}}, M))} \tag{4.8}$$

$\mu(\overline{\mathbf{\Sigma}}, M)$ indicates the average magnitude of $\overline{\mathbf{\Sigma}}$. $\sigma(\overline{\mathbf{\Sigma}}, M)$ is chosen with a small value to ensure the stability of $\overline{\mathbf{\Sigma}}$. Then, two corresponding thresholds are chosen to decide the adequate stopping moment for TPE. In our evaluation, we monitor $L = 25$ top-performing solutions at each iteration and choose two conditions, $\mu(\overline{\mathbf{\Sigma}}, M) \leq 1$ and $\sigma(\overline{\mathbf{\Sigma}}, M) \leq 0.1$, with $M = 100$ to stop TPE. Figure 4.4 indicates with a dashed line the iteration where TPE is stopped.

4.5 Evaluation

4.5.1 Experiment setup

Experiments are performed on an Intel Xeon E5640 2.67GHz with 4GB memory running Linux. We used Adaptive TPE (ATPE), an extension on top of TPE implemented on Hyperopt [18], which provides a machine learning model to automatically tune the hyper-parameters of TPE. Tabu search is described in [8]. We use energy as our cost model. The energy model counts the number of operations performed by each operator, and calculates the total cost based on the energy consumption of an operation empirically gathered from several ASIC synthesis/simulation for different WLs. An operator is characterized by the WLs of the operands, the WL of the result, and the arithmetic operation performed. Characterization is performed using Synopsys Design Compiler and Prime Time using a 28nm FDSOI technology.

We compare our hybrid approach with reference algorithms (Max−1, Min+1, Tabu search and GRASP, as in Section 4.2.1) in terms of solution cost and exploration time. GRASP is configured with $T_{RCL} = 3$ running in 10 iterations as in [8]. During the execution of Min+1 and Tabu, we experienced that the search process usually gets stuck in situations that keep increasing the number of bits and thus cost, without a significant quality improvement. Actually, in such situations there are more distant neighbors that gain a significant amount of quality. As a result, more quality can be achieved without increasing the number of bits too much, which reduces redundant cost. The bias versions of Min+1 and Tabu search add a bias to the selection criterion

to reduce the priority of the candidates for which a significant number of bits has already been added, which leads to optimized versions of the original algorithms.

We evaluate cost and exploration time on five applications. The description of benchmarks is detailed as follows:

— Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) are implemented with cascaded 5-stage structure using 33th and 2nd order filters, respectively. The number of effective variables of FIR and IIR is 17 and 33.

— Block-Matching and 3D Filtering (BM3D) [21] is a widely used for noise reduction algorithm in digital image processing. This algorithm use a Collaborative Filtering to filter 3D groups created by similar 2D same-size image fragments (i.e. blocks). The filtered 3D groups are then transformed back into 2D fragments to reproduce the output image. The implementation of BM3D uses 45 effective variables.

— Image Signal Processor (ISP) is an image signal processing chain which takes raw data from camera sensors as the input, and applies a sequence of processing kernels to produce a color image. 4 kernels are considered for WLO. Non-Local Means (NLM) denoising filter [22] computes means of 5 windows, weighted by the distance from the target pixel, to filter noise. Demosaic reconstructs a full color image from sensor data that captures color information as a mosaic of primary colors. Gamma Correction adjusts the brightness of the image to suit human eyes. Unsharp sharpens the image by a mask computed by a two-pass (vertical + horizontal) filter. ISP has 74 effective variables.

— Stereo Matching (SM) algorithm is widely used in computer vision to extract a depth information of a scene. In most cases, SM uses two images taken from two cameras placed side by side and horizontally as the inputs. 3D information is represented by a disparity map obtained by the horizontal distance between corresponding pixels in the two images. For evaluation, we explore 85 effective variables in SM.

We use PSNR as the quality metric for filters and Structural Similarity (SSIM) for image processing applications. Two quality constraints $\boldsymbol{\lambda_{obj}}$ are evaluated for PSNR (40 dB and 50 dB) and SSIM (0.9 and 0.99). We use the narrow range $[q_l, q_h]$ as ± 2 dB around $\boldsymbol{\lambda_{obj}}$ for PSNR, ± 0.09 for SSIM=0.9, and ± 0.009 for SSIM=0.99.

4.5.2 Performance Evaluation

Figure 4.5 presents the comparison between our hybrid approach and the references in terms of solution cost (top) and exploration time (bottom) for our five benchmarks and two accuracy constraints. Max-1 converges fast but always leads to the worst solution cost. Conversely,

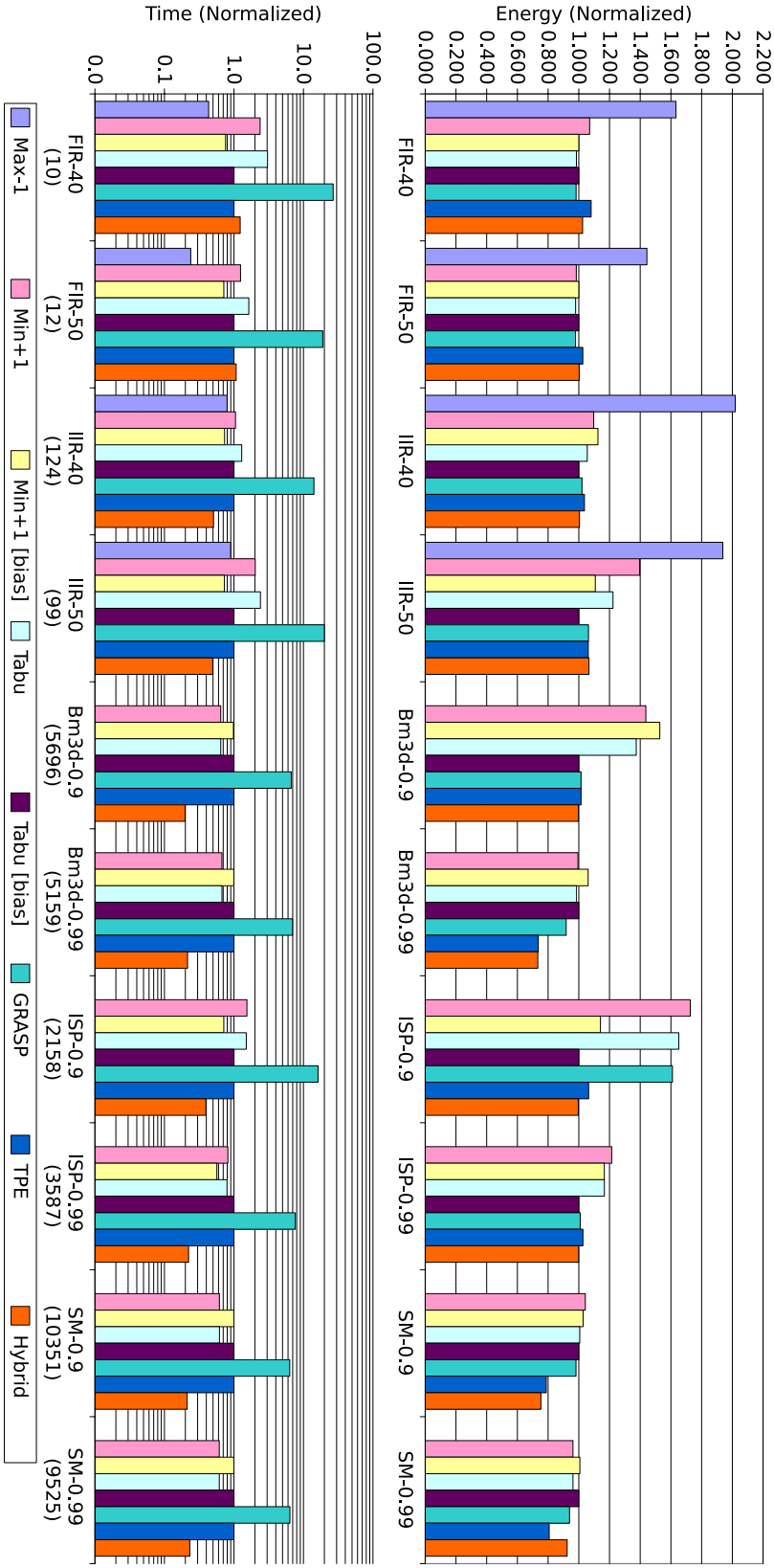


Figure 4.5: .

] Energy cost of solutions (top) and exploration time (bottom) normalized to Tabu[bias]. Numbers in parentheses are the execution time in minutes of Tabu[bias] used for the normalization. Results for Max−1 are removed for better readability when leading to bad solutions.

GRASP obtains relatively competitive solutions but with a long exploration time. Time of GRASP is proportional to the number of iterations used to obtained different randomized starting points to avoid local minima. Most solutions found by GRASP are slightly better than those obtained by Min+1, Min+1[bias] and Tabu. However, GRASP does not outperform Tabu[bias] and is therefore not that efficient in spending longer time through randomization (see e.g., ISP-0.9 example where GRASP still gets stuck in bad local minima).

Thanks to the local bi-directional search, Tabu always find better solutions compared to Min+1, and sometimes the bias version is better, sometimes not. However, for a fair comparison, we always use the best reference as a competitor to our Hybrid approach. We also provide the solutions obtained by TPE only iterating for the same time as Tabu[bias]. Solutions obtained by TPE are relatively competitive to Tabu[bias]. For BM3D-0.99, SM-0.9, and SM-0.99, TPE found better solutions than Tabu, which indicates that TPE is able to avoid bad local minima better than Tabu.

The results show that our Hybrid approach always find competitive solutions in shorter time than other methods. For small problems solved in minutes such as FIR-40 and FIR-50, it is sufficient to use classical approaches because they are able to converge in short time. For larger problems solved in a few hours to a few days, our approach can reduce by 50% to 80% (66% on average) exploration time compared to the best reference with similar cost. Our approach even defeats TPE in most benchmarks with slightly better solutions found in a much shorter time (2x-5x faster than TPE, 3.7x on average). Apart from IIR-50 and SM-0.99 where solutions obtained by our approach is 0.4% and 14.5% worse than TPE, respectively, the other cases record a 0.4%-6.3% solution improvement (3.3% on average) compared to TPE. It is clear that the speedup of our approach comes from the benefits of i) TPE to speedup the first phase and ii) our transition condition to stop TPE early. The solution cost improvement takes advantage of the good convergence of Tabu in local regions. In absolute execution time, our approach reduces exploration from 7.2 days to 1.5 days compared to TPE for SM-0.9, from 4 days to 18.8 hours compared to Tabu[bias] for BM3D-0.9, and from 2.5 days to 13.3 hours compared to Tabu[bias] for ISP-0.99.

4.6 Conclusion

In this chapter, we propose a hybrid method leveraging Bayesian Optimization and local search, to improve the scalability issue that WLO faces for large applications. First, we show that TPE is good at narrowing down the search on good regions due to its randomness and Tabu is well suited for fine-tuning because of its bi-directional local search. Then, the core of our

approach is to derive a statistical metric to evaluate model convergence of TPE, to automatically find the right transition moment between TPE and Tabu and to maintain the search efficiency. We show that for large applications, our approach significantly outperforms state-of-the-art approaches with a 50%-80% reduction in exploration time, while still providing similar or better cost solutions. The exploration of the parameters used in our methods is important but left as future work.

Bibliography

[1] M. Clark *et al.*, "Accelerating fixed-point design for mb-ofdm uwb systems," *CommsDesign, January*, vol. 4, 2005.

[2] S.-C. Chan and K. M. Tsui, "Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming," *IEEE Trans. on Circ. and Syst.*, vol. 54, no. 4, pp. 845–854, 2007.

[3] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Trans. on Computers*, vol. 57, no. 11, pp. 1561–1570, 2008.

[4] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 5, 2001, pp. 53–56.

[5] G. A. Constantinides, P. Y. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. on CAD of Int. Circ. and Syst.*, vol. 22, no. 10, pp. 1432–1442, 2003.

[6] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for cdma demodulator," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, 2001, pp. 290–293.

[7] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[8] H.-N. Nguyen, D. Ménard, and O. Sentieys, "Novel algorithms for word-length optimization," in *19th European Signal Processing Conf.* IEEE, 2011, pp. 1944–1948.

[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[10] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[11] K. N. Parashar, D. Menard, and O. Sentieys, "A polynomial time algorithm for solving the word-length optimization problem," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2013, pp. 638–645.

[12] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[13] M.-A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 2, 2002, pp. II–II.

[14] V.-P. Ha, T. Yuki, and O. Sentieys, "Towards generic and scalable word-length optimization," in *DATE 2020-23rd IEEE/ACM Design, Automation and Test in Europe.* IEEE, 2020, pp. 1–6.

[15] D. Novo, I. Tzimi, U. Ahmad, P. Ienne, and F. Catthoor, "Cracking the complexity of fixed-point refinement in complex wireless systems," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2013, pp. 18–23.

[16] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[17] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Int. Conf. on learning and intelligent optimization.* Springer, 2011, pp. 507–523.

[18] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, 2015.

[19] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *arXiv preprint arXiv:1807.01774*, 2018.

[20] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," in *NIPS workshop on Bayesian Optimization in Theory and Practice*, vol. 10, 2013, p. 3.

[21] M. Lebrun, "An analysis and implementation of the bm3d image denoising method," *Image Processing On Line*, vol. 2, pp. 175–213, 2012.

[22] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Confe. on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, pp. 60–65.

Chapter 5

Maximizing Computing Accuracy on Resource-Constrained Architectures

With the growing complexity of applications, designers need to fit more and more computing kernels into a limited energy or area budget. Therefore, improving the quality of results of applications in electronic devices with a constraint on its cost is becoming a critical problem. Word Length Optimization (WLO) is the process of determining bit-width for variables or operations represented using fixed-point arithmetic to trade-off between quality and cost. State-of-the-art approaches mainly solve WLO given a quality (accuracy) constraint. In this chapter, we first show that existing WLO procedures are not adapted to solve the problem of optimizing accuracy given a cost constraint. It is then interesting and challenging to propose new methods to solve this problem. Then, we propose a Bayesian optimization based algorithm to maximize the quality of computations under a cost constraint (i.e., energy consumption is taken into account). Experimental results indicate that our approach outperforms conventional WLO approaches by improving the quality of the solutions by more than 170%.

5.1 Introduction

The rapid development in scientific and technological innovations during the last decade has driven the demand for more powerful chips with higher performance to handle complex tasks in various fields, such as artificial intelligence (AI), big data, and the Internet of Things (IoT). The requirement for integrating more applications and services, while ensuring at the same time high performance and quality of service (QoS) in a limited cost budget (energy and/or area), poses new challenges for energy-efficient computing. As an example, Deep Neural Networks (DNN) are used in multiple applications such as object detection, speech recognition, navigation, etc. Many of these applications operate on battery-powered intelligent embedded systems such as autonomous drones, self-driving cars, smart phones, and wearable devices. Thus, increasing the performance and/or QoS under a limited energy budget is a key competitive advantage among

electronic device manufacturers.

Likewise, implementing applications on ASIC or FPGA devices also faces challenges of optimizing the performance and/or QoS within a constrained cost budget. In theses cases, besides energy, relevant cost constraints include either limited area inside a System on Chip (SoC) or limited resources like the number of LUTs, DSP blocks, memory size in the context of FPGA architectures. As an example, in the development life cycle of some SoC products, upgrading only some modules while keeping the remaining ones untouched helps shortening the time to market. Replaced modules usually have new features that enhance the performance of the system. This requires optimization in terms of maintaining high implementation performance and/or QoS within the available silicon area. This process takes more effort to optimize because the new modules are usually more sophisticated and complex than older ones, requiring more time and effort to create, test, and debug them to fit into the limited, unchanged area (or a given energy budget). Therefore, a design technique that optimizes for the performance and/or QoS within a constrained cost budget of the system would be of great concern, but is not really covered in the state of the art.

Approximate Computing (AC) emerged as an effective solution to adjust the trade-off between QoS and cost constraints, while satisfying design requirements. The use of fixed-point arithmetic in embedded systems, combined with a reduction of the precision, is one of the AC techniques for reducing cost and energy. This technique requires to determine the optimal fixed-point word-lengths (i.e., bit-widths) for representing all variables of the application that still fulfil some quality (i.e., accuracy of computations) requirement. This procedure is called Word-Length Optimization (WLO). Many methods are proposed to solve the WLO problem with QoS constraints quickly and efficiently [1, 2, 3]. However, these methods traditionally explore the design space to minimize a cost function (e.g., area or energy) under a given constraint on the accuracy at the output of the system.

However, as mentioned previously, it might be of great interest to limit the area or energy cost budget of a computing kernel below a specified bound. In this case, the stringent constraint becomes the cost, and it is important to maximize accuracy (or the minimize accuracy degradation) as a quality metric under this cost constraint. In this chapter, we propose new methods to solve this problem of maximizing QoS within a given cost budget. We first highlight the limitations of traditional methods to solve WLO under cost constraints. Then, we propose a method relying on Bayesian optimization to address the resource-constrained WLO problem. Experimental results show that our proposed algorithm outperforms the latest conventional approaches, improving solution quality by up to 300% compared to Uniform Word-Length Optimization (UWLO) and

by up to 170% compared to a state-of-the-art classical WLO approach.

The remaining sections are organized as follows. In Section 5.2, we provide background information and discuss related work, followed by an explanation of our motivations and problem statement in Section 5.3. In Section 5.4, we describe our proposed Bayesian optimization-based method. In Section 5.5, we compare the performance of our approach with traditional approaches on various benchmarks, before to conclude in the last section.

5.2 Related Work

In this section, the well-studied accuracy-constrained Word-Length Optimization (WLO) problem is recalled, followed by the related state of the art. Then, we motivate the need for a new method to solve a cost-constrained WLO problem.

A number expressed using fixed-point arithmetic has both integer and fractional word-lengths (WLs), which are correspondingly represented by I and F bits, respectively. The dynamic range of the number represented with this format is covered by the integer WL, while its accuracy is controlled by the fractional WL. In this chapter, we concentrate on the WLO for the fractional WL since it is the most time-consuming exploration in the design process. The dynamic range is usually evaluated through static analysis or simulation to determine the integer WL to guarantee no or low-probability overflow during computations.

Let the vector $\boldsymbol{W} = [W_0, W_1, \ldots, W_{N-1}]$ denote a word-length configuration with N effective variables from the considered application to be explored for fixed-point conversion. The main objective of WLO is to determine a good-enough word-length configuration that minimizes a cost function under a quality (accuracy) constraint:

$$\min \boldsymbol{C}(\boldsymbol{W}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W}) \geq \boldsymbol{\lambda_{obj}}, \tag{5.1}$$

where $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are the cost and accuracy functions depending on $\boldsymbol{W}$, respectively. $\boldsymbol{\lambda_{obj}}$ is the maximal degradation of the accuracy at a given output of the application that is acceptable for the required quality of service. Analytical models and simulation-based methodologies are both used to construct $\boldsymbol{C}$ and $\boldsymbol{\lambda}$.

There are two approaches to solve this well-studied accuracy-constrained WLO problem: simulation-based and analytical approaches. The aim of analytical approaches is to approximate quality and cost functions of the WLO problem to be convex functions which can be solved quickly by some convex optimization algorithms, e.g., CVX [4]. Existing techniques are limited to modeling noise power metrics of Linear and Time-Invariant (LTI) systems (with some exten-

sions) [5, 6, 1]. Simulation-based approaches [7, 8, 9, 10, 11] use simulations and iterative search to address WLO. Uniform WordLength Optimization (UWLO) can quickly evaluates several solutions which are constructed by the same wordlength for each variable to obtain the best one. However, the quality of the solution obtained is much worse than the ones obtained by the non-uniform wordlength optimization methods [12]. Some more advanced methods leverage noise budgeting techniques [2, 13] to reduce the exploration time in solving WLO for large applications.

Bayesian Optimization (BO) is a machine-learning-based optimization technique [14] designed to optimize functions that often lack mathematical expressions and/or derivatives. BO typically consists of two essential components: i) a probabilistic surrogate model for modeling the unknown objective function based on previously observed samples and ii) an acquisition function that optimizes over the surrogate model to recommend further samples. The procedure of choosing surrogate models is a key distinction between BO approaches and those of others. While Gaussian Processes (GP) are often employed for moderate-size continuous-domain issues, tree-based models such as Random Forest and Tree-structured Parzen Estimator (TPE) are advantageous for large-size discrete-domain problems [15]. TPE identifies points that might have been drawn based on the assessment of a loss function at other points. BO was first used to speed up WLO in a hybrid method [3].

5.3 Resource-Constrained WLO

All the previously mentioned techniques solve the Word-Length Optimization problem without any constraint on the cost of the solution. The objective of the optimizer is to find a solution with minimum cost that guarantees the accuracy being higher than a given constraint. However, as motivated in the introduction, with the growing complexity of applications, it might be of great interest to limit the area or energy cost budget of a computing kernel under a restricted bound. In this case, the stringent constraint becomes the cost and it is important to find the maximal accuracy (or the minimum accuracy degradation) under this cost constraint.

The resource-constrained WLO problem can be stated as

$$\max \boldsymbol{\lambda}(\boldsymbol{W}) \quad \text{Subject to} \quad \boldsymbol{C}(\boldsymbol{W}) \leq \boldsymbol{C}_{budget}, \tag{5.2}$$

where $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ represent for the cost and accuracy functions of a WL configuration vector $\boldsymbol{W}$. The objective of this problem is to find a WL configuration $\boldsymbol{W}_o$ that maximizes the accuracy at the application output given $\boldsymbol{C}_{budget}$ as a cost constraint. The WL components W_i of $\boldsymbol{W}$ can be bounded as

$$m_i \leq W_i \leq n_i \qquad m_i, n_i \text{ and } W_i \in \mathbb{Z}^+, \tag{5.3}$$

to limit the exploration space.

5.3.1 Room for Accuracy Improvement Given a Cost Budget

We use a subset of the solution space of a 5-stage 33-tap Finite Impulse Response (FIR) filter as an example to motivate our work. The filter has 17 different variables represented in fixed-point arithmetic.

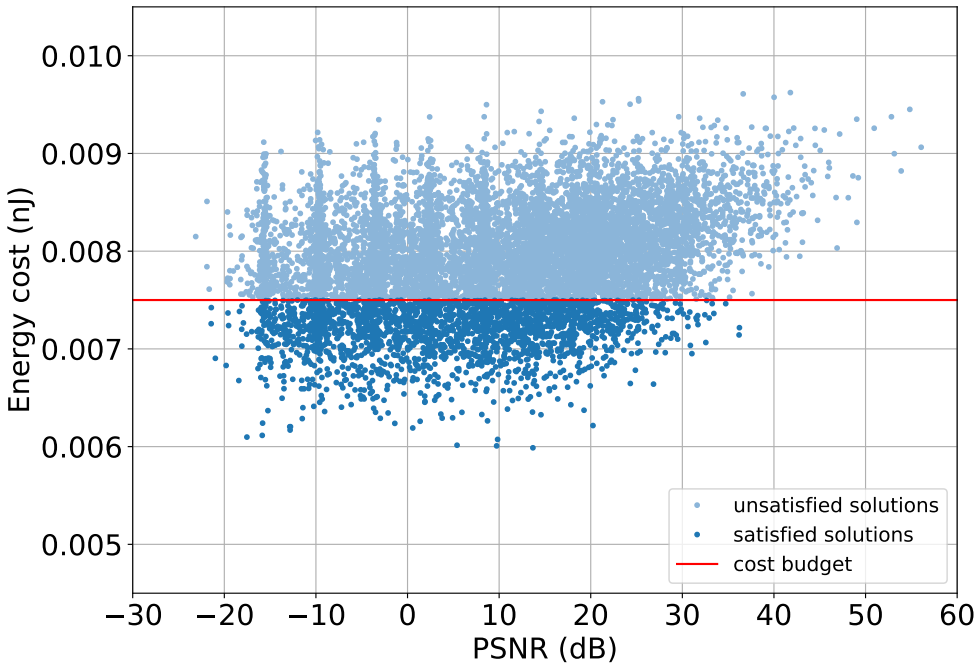


Figure 5.1: 10^5 random solutions as a representative subset of all possible solutions, for the FIR filter benchmark. The red line represents an energy budget of 0.0075 nJ

Fig 5.1 shows 10^5 solutions obtained from a random search. Each solution evaluates the FIR filter output accuracy (PSNR) and the associated cost (energy in nJ) for a random WL configuration. The integer WL is set to 12 bits for all variables to avoid overflow and the fractional WL of each variable is varied in the range of 3 to 20 bits, which can be considered large enough to reduce the chance of missing good solutions. The energy is evaluated based on a library of arithmetic operators characterized for various WL in a 28nm FDSOI technology (see Section 5.5.1 for more details). Given an energy level, there is a large set of solutions consuming the same amount of energy but resulting in a significantly different quality at the application output. For instance, with 0.0075 nJ considered as a maximum energy budget, the quality of satisfied solutions can vary from -20dB to 60dB. Hence, to maximize the performance or quality of the system given a cost budget, it is worth to design an optimization method that can obtain the highest quality solution satisfying the cost constraint from possible solutions.

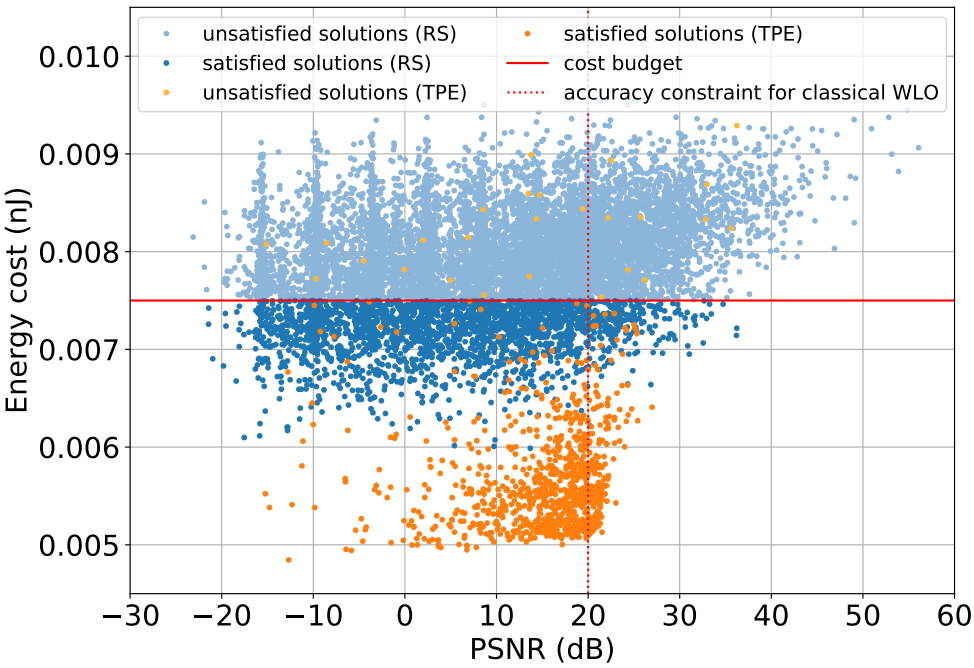


Figure 5.2: The classical WLO problem in the context of the cost-constrained WLO problem. The data points includes the 10000 randomized solutions and 1000 solutions found by TPE for the classical WLO problem with an accuracy constraint as 25 dB of PSNR.

5.3.2 Resource-Constrained WLO as a New Problem

The classical accuracy-constrained WLO focuses on minimizing cost as much as possible while satisfying QoS at the system output. Meanwhile, the objective of the resource-constrained WLO is to exploit the available resource (energy and/or area) of the system to enhance its overall performance. The objective of these two problems are totally different, leading to the different exploration targets. For the design purpose that enhances quality on a system with specific resources as much as possible, the search objective of the original problem is no longer relevant. Hence, it is important to design new methods to solve the resource-constrained WLO problem. We illustrate the objective target of the classical WLO for FIR filter in the context of a system with an energy budget. We use TPE as a search method for the classical problem with a PSNR quality target of 25 dB. The use of TPE follows the setup in [3].

In Fig. 5.2, the explored solutions discovered by TPE for the classical problem are compared to those discovered by Random Search (RS) in the context of a system with an energy budget as 0.0075 nJ. Clearly, TPE focuses on exploring low-cost solutions around the accuracy constraint, i.e. around the vertical line 20 dB, missing possible solutions with higher accuracy and still satisfying the energy budget, i.e., solutions found by RS range in 30 to 40 dB of PSNR. Similar to TPE, recent simulation-based techniques mentioned in [11, 3] such as Min+1 and Tabu Search

also rely on a search model to restrict the exploration space, followed by a fine tuning around the quality constraint to find a better solution.

5.3.3 Limitations of Classical WLO Methods

Numerous methods to address the WLO problem have been proposed in the literature. In [16, 3], the performance and execution time of several state-of-the-art techniques, including Min+1, Max-1, Heuristic approach, Tabu Search, and GRASP, have been compared. The comparison shows that variants of bi-directional searches that combine steepest-descent and mildest-ascent procedures, such as heuristic approach [17] and Tabu search [11], outperform the mono-directional searches. However, due to the greedy nature of those algorithms, the final solution is not guaranteed to be a global minima and is dependent on the starting point. Additionally, most of these searches are based on local iterative searches that travel across small distances in the discrete domain. As a result, if the starting point is distant from a local minimum, convergence will be slow [3].

Fig. 5.3 illustrates the limitations of the classical methods. We choose Tabu search as a typical method to address the resource-constrained WLO. The FIR filter with a maximum energy supply as 0.0075 nJ is still used for the illustration. The search procedure is initialized at different points. Each initial point is constituted from fixed-point variables with UWL. The integer word-length of variables is fixed as 12 bits. The fractional word-length is incremented every two bits from 2 bits to 20 bits, thus the UWL varies from 14 bits to 32 bits for each starting point. The quality of obtained solutions with Tabu search and different starting points significantly differs and strongly depends on the initialization. At starting points with UWL equal to 30 or 32, the obtained solutions have a bad PSNR of around -30 dB. At the remaining initial points, Tabu search produced better solutions than RS from 10 to 50 dB of PSNR. However, the solution quality is varying dramatically. Besides, procedures that start farther away from the cost constraint tend to converge more slowly than those that start closer to it. The trends are consistent with other benchmarks evaluated in Section 5.5.

5.4 Solving Resource-Constrained WLO

In this section, we introduce a BO-based method to address the resource-constraint WLO problem. The loss function is constructed first, followed by the description of our BO-based approach.

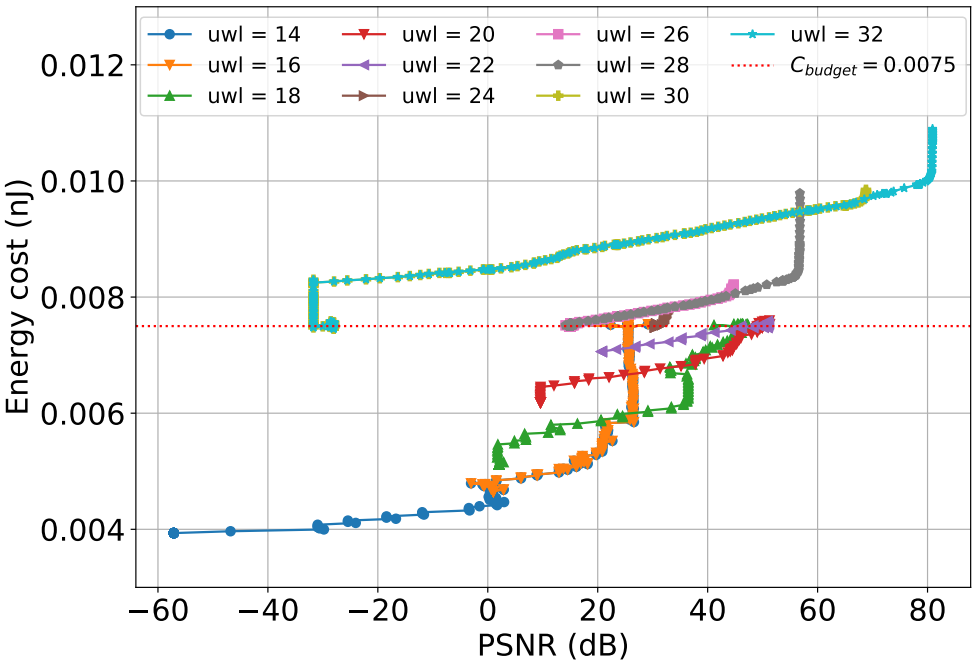


Figure 5.3: Different exploration trajectories of Tabu search method corresponding to different starting points for the resource-constrained WLO problem on the FIR application. The energy budget is 0.0075 nJ. Each color corresponds to a search procedure with a given starting point. Fixed-point variables of each procedure are chosen with uniform word-lengths (UWLs) at the starting point. The number in the legend indicates the word-length given for the UWL. For each procedure, the starting point is further from the cost budget line than the ending point. Some procedures overlapped during the exploration process.

5.4.1 Loss Function

The resource-constrained WLO problem statement of Eq. 5.2 is changed as follows using the Lagrange multipliers approach to turn a constrained problem into an unconstrained one:

$$f(\boldsymbol{W}) = -\boldsymbol{\lambda}(\boldsymbol{W}) + \alpha(\boldsymbol{C}(\boldsymbol{W}) - \boldsymbol{C}_{budget}), \tag{5.4}$$

with a positive and big enough α. By minimizing the loss function, TPE tends to sample more frequently solutions that are of high quality and satisfy the cost constraint. The factor α is important for exploration. If α is very small, TPE will ignore the restricted cost condition and concentrate on exploring solutions with quality as high as possible. If α is very large, the solution of TPE would be highly dependent on the restricted cost condition. That is, TPE focuses on finding solutions that cost less than $\boldsymbol{C}_{budget}$ while neglecting solutions of high quality.

Based on many WLO experiments, we found that the cost and quality functions tend to be proportional to $\boldsymbol{W}$. A high quality solution usually comes at a high cost. Thus, the best solutions to satisfy the cost constraint with high quality are likely to be those around the cost budget $\boldsymbol{C}_{budget}$. Therefore, for faster convergence, we force the loss function to cover only a narrow range $[c_l, c_h]$ around $\boldsymbol{C}_{budget}$. Solutions with costs outside this range of interest are penalized with the positive infinite loss value. The loss function is then defined as

$$f(\boldsymbol{W}) = \begin{cases} -\boldsymbol{\lambda}(\boldsymbol{W}) + \alpha(\boldsymbol{C}(\boldsymbol{W}) - \boldsymbol{C}_{budget}) & \text{if } \boldsymbol{C}(\boldsymbol{W}) \in [c_l, c_h], \\ +\infty & \text{otherwise.} \end{cases} \tag{5.5}$$

5.4.2 TPE Algorithm

For the resource-constrained WLO problem of Eq. 5.2, a WL configuration formed by the choice of each variable, i.e., $W_i \in [m_i, n_i]$ represents a solution. All possible combination of different values of W_i creates a solution space of this WLO problem. With the leverage of Bayesian Optimization relying on TPE [18] to solve the WLO problem, each W_i is considered as a hyper-parameter to be tuned. Each W_i is initially mapped to a prior of quantized uniform distribution in which the value is sampled uniformly in $[m_i, n_i]$ and then rounded to the nearest integer value.

Algorithm 3 describes the TPE algorithm for the resource-constrained WLO problem. The algorithm is first initialized by uniform word-length (UWL) configurations and corresponding loss values evaluated by the loss function. These configurations are then served as first samples in an observation history $\boldsymbol{\mathcal{H}}$. TPE works as an iterative approach. At each iteration, a WL configuration $\boldsymbol{W}_i$ is evaluated by the loss function $f(\boldsymbol{W}_i)$. Then, obtained samples $\{\boldsymbol{W}_i, f(\boldsymbol{W}_i)\}$ stored in the observation history $\boldsymbol{\mathcal{H}}$ are divided into two groups. The first group contains *good* samples where the loss f_i is less than a threshold γ^*, whereas the second group consists of the

remaining, considered as *bad* samples. TPE uses these two sample groups to model two likelihood probability density functions $l(\boldsymbol{W}) = p(\boldsymbol{W}|f(\boldsymbol{W}) < \gamma^*)$ and $g(\boldsymbol{W}) = p(\boldsymbol{W}|f(\boldsymbol{W}) \geq \gamma^*)$, respectively. Then, the TPE algorithm decides which hyper-parameter to try in the next iteration by maximizing the ratio

$$\frac{l(\boldsymbol{W})}{g(\boldsymbol{W})} = \frac{p(\boldsymbol{W}|f(\boldsymbol{W}) < \gamma^*)}{p(\boldsymbol{W}|f(\boldsymbol{W}) \geq \gamma^*)}.$$

Algorithm 3 TPE algorithm

1: $\boldsymbol{\mathcal{H}} \leftarrow$ Uniform Word-Length Initialization
2: **for** $i \in [1, \ldots, T]$ **do**
3: $\quad \boldsymbol{W}^* = argmax \frac{l(\boldsymbol{W})}{g(\boldsymbol{W})}$
4: $\quad$ Evaluate $f(\boldsymbol{W}^*)$
5: $\quad \boldsymbol{\mathcal{H}} \leftarrow \boldsymbol{\mathcal{H}} \cup (\boldsymbol{W}^*, f(\boldsymbol{W}^*))$
6: $\quad$ Update $l(\boldsymbol{W})$ and $g(\boldsymbol{W})$ given $\boldsymbol{\mathcal{H}}$
7: **end for**

Our initialization procedure constructed from UWL configurations samples the solution space at some design points ranging from low-accuracy-and-low-cost solutions to high-accuracy-and-high-cost solutions. These design points are then used to construct density functions at the very first iteration of the algorithm. Under the evaluation of the loss function, TPE can start exploring from the points that yield low loss values. This initialization procedure provides better results and faster convergence than an initialization constructed from random configurations.

5.5 Evaluation

5.5.1 Experiment setup

We choose Uniform Word-Length Optimization (UWLO) as our baseline. Our goal is to evaluate how much our approach can perform better than UWLO, a fast and simple method considering all variables having the same WL. In our approach, we employed Adaptive TPE (ATPE), an improved extension of TPE that was implemented on Hyperopt [18] to update the hyper-parameters of TPE in real time. The cost and quality functions, $\boldsymbol{\lambda}(\boldsymbol{W})$ and $\boldsymbol{C}(\boldsymbol{W})$, in the loss function are normalized within a range. First, UWLO is performed to obtain the quality and the cost of uniform-wordlength solutions. $\boldsymbol{\lambda}(\boldsymbol{W})$ is normalized between upper and lower bounds. The lower bound is the solution which satisfies the cost constraint and has the highest quality. The upper bound is the highest quality value obtained from those solutions without the cost constraint. The normalization range of $\boldsymbol{C}(\boldsymbol{W})$ is $[c_l, c_h]$, where c_l and c_h are separated from the central value, $\boldsymbol{C}_{budget}$, by a distance of 10% of the difference between the minimum and the maximum cost values obtained from the UWL solutions. We conduct experiments on

our benchmarks with five values of α, i.e., $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5\}$, to evaluate its impact to the approach. The number of iterations for each experiment is chosen as 5000 which can cover mostly the possibility of improving the solution quality as much as possible. Note that more complex applications would need more iterations to increase the possibility of obtaining better solutions. To further demonstrate the limitations of classical methods, we conduct several experiments with Tabu search [11] on our benchmarks. In each benchmark, Tabu Search is initialized at different uniform wordlengths.

All experiments were conducted on a Linux-powered Intel Xeon E5640 processor at 2.67GHz with 4GB of RAM. Energy is the basis for our cost model. The energy model counts the operations carried out by each operator and determines the overall energy cost using the empirically determined energy consumption of an operation from several synthesis for various WLs. An operator is characterized by the WLs of the operands, the WL of the result, and the arithmetic operation performed. A 28nm FDSOI technology is used for characterization along with Synopsys Design Compiler and Prime Time.

We use three applications as the benchmarks to evaluate our approach: FIR, IIR and NLM. The FIR filter, also used in previous sections, is implemented with a 5-stage cascaded structure of 33 taps each. The IIR filter has a 5-stage cascaded structure of 2nd order filters. The number of effective variables to optimize is 17 for FIR and 33 for IIR. Non-local means (NLM) is a denoising approach in image processing. To denoise the target pixel, the algorithm calculates the mean of all pixels in the picture, weighted by their similarity to the target pixel. The implementation of NLM uses 19 effective variables. We explore only the fractional word-lengths in the resource-constrained WLO problem. The integer word-length of each signal in the applications is determined beforehand so that the dynamic range of each signal is covered. The integer word-length of FIR, IIR and NLM are set to 12, 12 and 6, respectively. The word-lengths of the variables used in FIR and IIR range from 13 to 32 bits, whereas NLM is explored with bit-width ranging from 8 to 32 bits.

We use Peak Signal to Noise Ration (PSNR) for filters and Structural Similarity Index Measure (SSIM) for NLM as quality metrics. We choose three cost budget objectives for every application as shown in Table 5.1, thus resulting in nine different benchmarks. The targets are selected proportionally to the operation counts of each application, i.e., the number of additions and multiplications. The cost budgets of NLM are higher than the remaining applications due to its higher computation complexity.

Table 5.1: The benchmarks for the evaluation

Application	Target 1 (nJ)	Target 2 (nJ)	Target 3 (nJ)
FIR	0.0065	0.0070	0.0075
IIR	0.0010	0.0012	0.0014
NLM	250	300	400

5.5.2 Performance Evaluation

Figure 5.4 presents a comparison between our approach and UWLO serving as a baseline. In overall, our approach outperforms UWLO by obtaining solutions with better quality for a given cost budget. The results show that solutions obtained by our approach with different values of α are always better than those obtained by UWLO. Table 5.2 summarizes the quality improvement of our solutions compared to those obtained by UWLO. In several cases, including FIR (Target 1, 2 and 3), IIR (Target 1 and 2), and NLM (Target 1), our approach significantly improves the quality of the solutions. This indicates that a solution can be obtained quickly by UWLO, but which is far from the optimal; this also confirms the conclusion in [12].

Table 5.2: Quality improvement of our solutions in percentage compared to UWLO solutions. Given value for each benchmark is the average of those obtained by different values of α.

Application	Target 1	Target 2	Target 3
FIR	321.19%	200.48%	90.68%
IIR	52.39%	50.67%	16.91%
NLM	160.50%	25.63%	3.32%

For some benchmarks, the quality of solutions obtained by explorations with small α, such as $\alpha = \{0.1, 0.2\}$, is worse than for higher α values. As explained in Section 5.4.1, the cost constraint will be not respected with small α. This means that the exploration strategy will focus more on improving the solution quality without being strictly constrained by the cost budget, which causes an excessive search in the infeasible region. As a result, this will reduce the possibility of improving the quality of solutions in the feasible region.

Figure 5.5 illustrates a comparison of the quality of solutions found by our proposed method and Tabu Search using various benchmarks. The quality of results found using Tabu Search varies a lot among benchmarks, the solution strongly depending on its starting point. In the meanwhile, the quality of solutions obtained via our method is much more stable. Figure 5.5 also shows that the average quality obtained by our method is always superior than that of Tabu search. Table 5.3 reports the average improvements of the quality of solutions provided by our method over Tabu search. The table shows that the solutions can be improved from 20% to 173% using our optimized BO-based algorithm to solve the resource-constrained WLO problem.

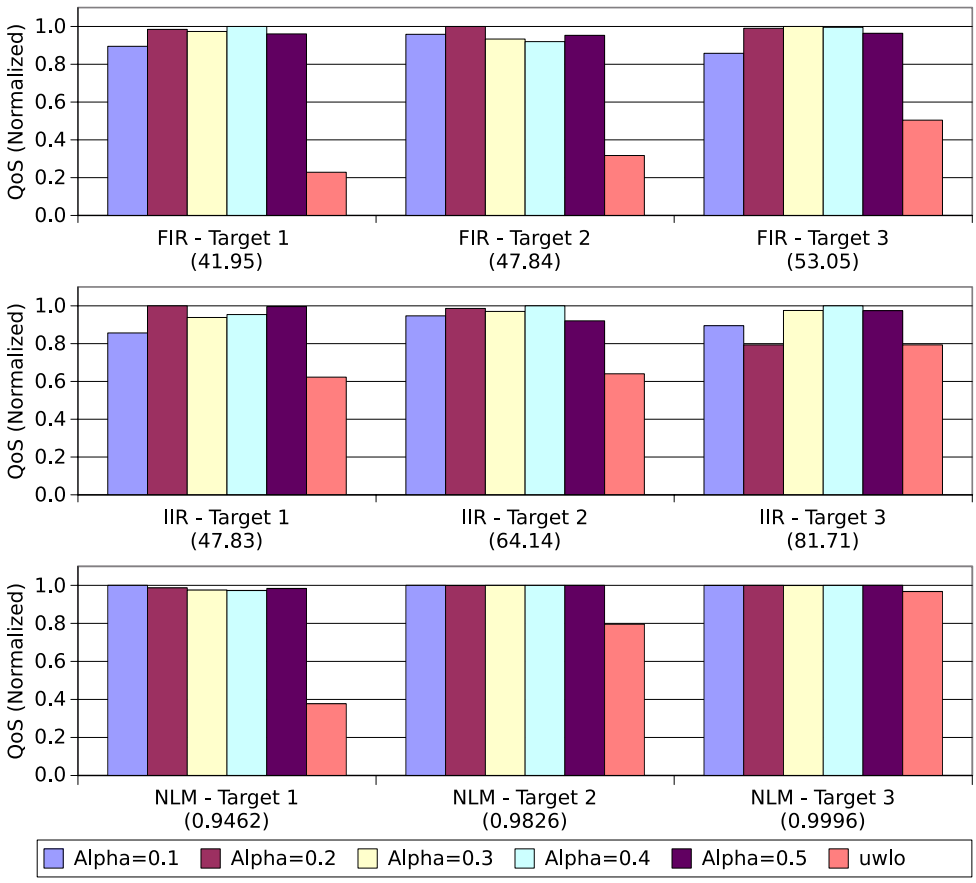


Figure 5.4: Performance comparison between our approach and UWLO in terms of quality of results for different benchmarks. The results are normalized with the highest quality value indicated in parentheses for each benchmark.

Especially, for IIR with target 1 and target 2, our method outperforms Tabu by up to 158% and 173%, respectively.

Figure 5.6 shows how our approach explores the design space of the problem, and how the search is evolving along with the iterations of the BO algorithm. We still use the FIR application with a cost budget of 0.0075 nJ, as in Section 5.3, to demonstrate how well our strategy performs in improving the quality of the solution. Indeed, few very early solutions (in the yellow color) randomly sample the solution space. Then, the later solutions of our technique concentrate quickly on the area of interest, which is located around the cost budget line. Solutions obtained during later iterations of the algorithm are represented by deeper colors. Our approach tries to sample at points of higher quality and still satisfy the cost constraint, which is clearly seen by the left-to-right search direction to improve the solution quality.

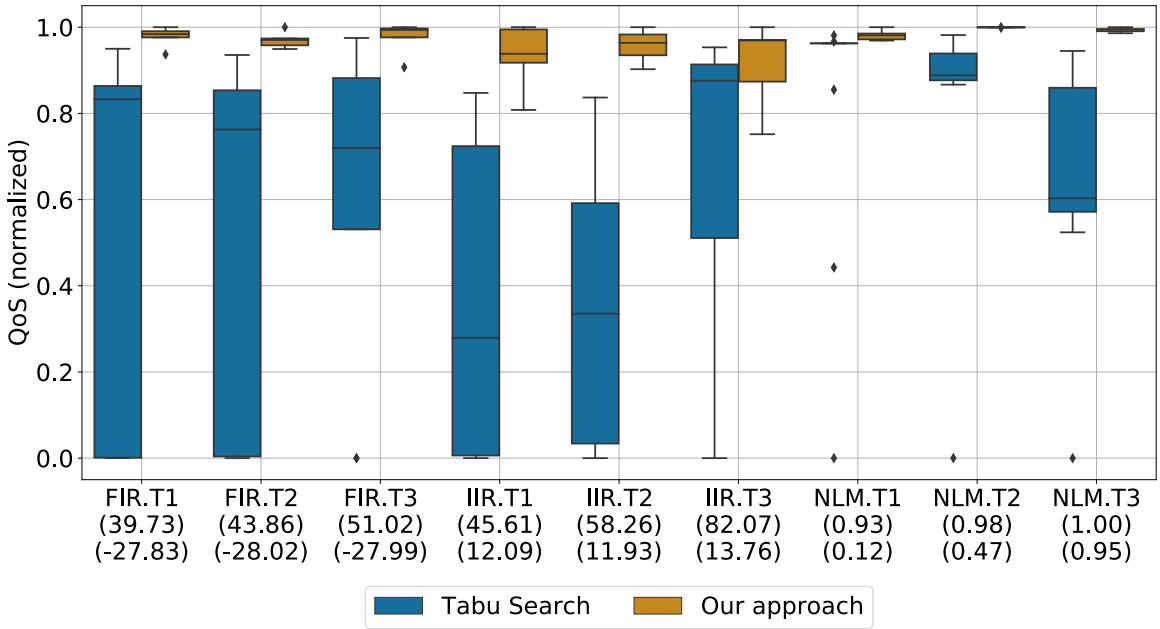


Figure 5.5: Comparison of the normalized quality of solutions obtained by Tabu Search and our approach for different benchmarks. The solutions obtained by Tabu search are the best found solutions under the target cost constraint, but for different starting points. Our solutions are obtained with different α. The normalization range are mentioned in parentheses, where the numbers above and below represent for upper and lower bounds, respectively. Letter "T" is the abbreviation of "Target".

Table 5.3: Average quality improvement of the solutions provided by our approach over those obtained by Tabu Search (TS). The results are normalized with the range in Figure 5.5.

Benchmark - Target	TS	Our approach	Avg. Impr.
FIR - Target 1	0.5258	0.9776	85.93%
FIR - Target 2	0.5017	0.9704	93.40%
FIR - Target 3	0.6116	0.9749	59.40%
IIR - Target 1	0.3599	0.9318	158.93%
IIR - Target 2	0.3499	0.9569	173.46%
IIR - Target 3	0.6717	0.9130	35.94%
NLM - Target 1	0.8058	0.9814	21.79%
NLM - Target 2	0.8288	0.9996	20.61%
NLM - Target 3	0.6388	0.9935	55.52%

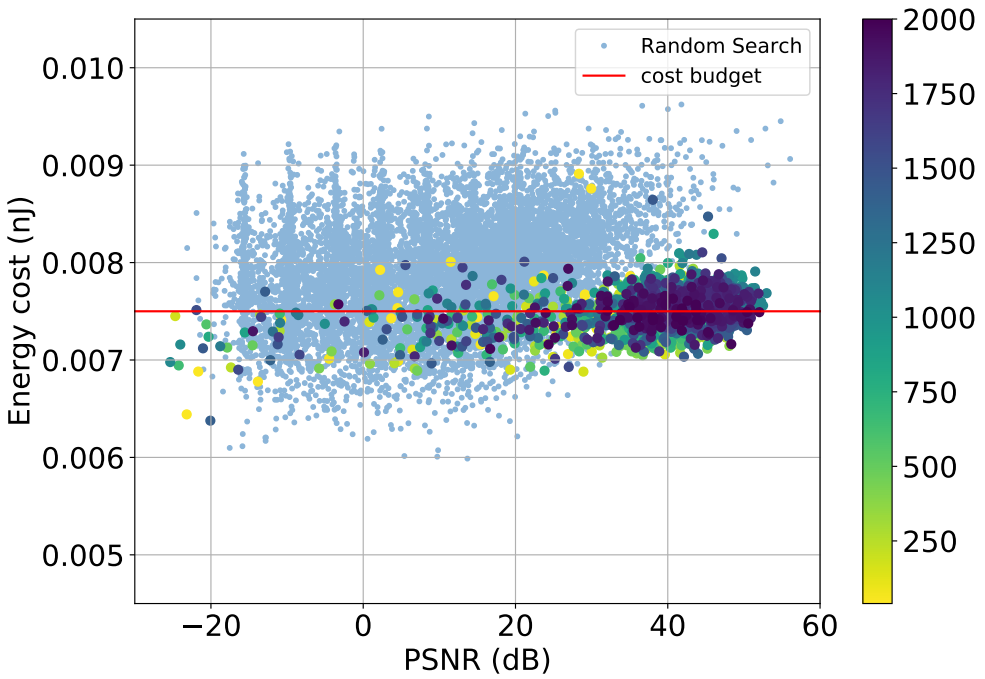


Figure 5.6: The search direction of our approach compared with the Random Search. The solutions found by RS (10^5 solutions) are colored in translucent blue. The solutions found by our approach (2000 solutions) are colored from a yellow to dark purple range of colors, corresponding to the solutions found at each iteration, from the first to the last iterations, respectively.

5.6 Conclusion

In this chapter, we present our Bayesian-Optimization-based approach to maximize the quality of digital applications implemented under a resource-constrained budget. We first show the importance of the resource-constrained word-length optimization (WLO) problem in systems which have limited silicon area and energy supply. Then, we highlight the limitations of classical approaches for WLO, such as Tabu search, which are not well adapted to solve the resource-constrained WLO problem. We thus propose a method based on BO and the TPE algorithm with an adaptive loss function. The results from our experiments show that our approach significantly outperforms UWLO and Tabu search, a state-of-the-art method. Our method is the first efficient approach addressing the resource-constrained WLO problem and also the starting point for further research that will address other aspects on the scalability and optimality of this problem, as well as other cost constraints, such as limited area or maximum number of resources.

Bibliography

[1] K. N. Parashar, D. Menard, and O. Sentieys, "A polynomial time algorithm for solving the word-length optimization problem," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2013, pp. 638–645.

[2] D. Novo, I. Tzimi, U. Ahmad, P. Ienne, and F. Catthoor, "Cracking the complexity of fixed-point refinement in complex wireless systems," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2013, pp. 18–23.

[3] V.-P. Ha and O. Sentieys, "Leveraging bayesian optimization to speed up automatic precision tuning," in *24th IEEE/ACM Design, Automation and Test in Europe (DATE)*, 2021.

[4] M. Grant, S. Boyd, and Y. Ye, "cvx users' guide," *online: http://www. stanford. edu/˜ boyd/software. html*, 2009.

[5] S.-C. Chan and K. M. Tsui, "Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming," *IEEE Trans. on Circ. and Syst.*, vol. 54, no. 4, pp. 845–854, 2007.

[6] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Trans. on Computers*, vol. 57, no. 11, pp. 1561–1570, 2008.

[7] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[8] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 5, 2001, pp. 53–56.

[9] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for cdma demodulator," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, 2001, pp. 290–293.

[10] G. A. Constantinides, P. Y. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. on CAD of Int. Circ. and Syst.*, vol. 22, no. 10, pp. 1432–1442, 2003.

[11] H.-N. Nguyen, D. Ménard, and O. Sentieys, "Novel algorithms for word-length optimization," in *19th European Signal Processing Conf.* IEEE, 2011, pp. 1944–1948.

[12] H. Choi and W. Burleson, "Search-based wordlength optimization for vlsi/dsp synthesis," in *IEEE Work. on VLSI Signal Processing VII*, 1994, pp. 198–207.

[13] V.-P. Ha, T. Yuki, and O. Sentieys, "Towards generic and scalable word-length optimization," in *DATE 2020-23rd IEEE/ACM Design, Automation and Test in Europe.* IEEE, 2020, pp. 1–6.

[14] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[15] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," in *NIPS workshop on Bayesian Optimization in Theory and Practice*, vol. 10, 2013, p. 3.

[16] M.-A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 2, 2002, pp. II–II.

[17] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[18] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

Chapter 6

Conclusions and Perspectives

Energy consumption is one of the most important challenges in computing today, a concern shared by all computer science subfields. We are entering into the dark silicon era, when the power constraint wall diminishes the benefits of transistor growth. This circumstance makes it difficult for scientists and chip-manufacturing corporations to develop viable alternative alternatives. In recent years, along with advancements in computer design, computer scientists and major chip makers have taken a keen interest in modifying the computing approach to increase energy efficiency. Approximate Computing (AxC) appears as a possible alternative for improving embedded system performance and energy consumption. The concept behind AxC is to decrease the accuracy of calculation to a tolerable level, in order to improve performance and/or save energy. Reducing the precision of the arithmetic representation is one of the most efficient AxC techniques for minimizing energy consumption and silicon area use. The current state of the art focuses on design techniques that effectively automate the precision tuning process in order to optimize the advantages of decreasing the accuracy of operators. This optimization process is referred to as Word-Length Optimization (WLO). Word-length optimization is the technique of establishing appropriate word-lengths for variables of a given algorithm in order to lower the implementation cost (energy consumption and/or area) while maintaining an acceptable level of accuracy owing to data quantization.

This dissertation focuses on the development of methodologies for systematic exploration of the design space, including implementation cost/quality of service (QoS) modeling and design automation. This enables us to accelerate the optimum or near-optimal design selection that (1) reduces cost (area and/or energy consumption) while still meeting the minimum required accuracy at the application output, or (2) optimizes accuracy at the application output within a constrained cost budget. In this dissertation, three main contribution have been proposed as follows:

1. Towards generic and scalable WLO based on the noise budgeting technique, as presented in Chapter 3.
2. Leveraging Bayesian Optimization to speed up automatic precision tuning, as presented in Chapter 4.

3. Maximizing Computing Accuracy on resource-constrained architectures, as presented in Chapter 5.

In Chapter 3, we detailed our proposed approach to reach a Word-Length Optimization (WLO) procedure that is more scalable for big systems, and that makes use of sophisticated quality criteria like Structural Similarity (SSIM). To prevent an uncontrolled growth in exploration time, this method splits the input application into smaller kernels. The fundamental issue that is being studied in this research is the problem of allocating noise budgets to individual kernels. To do so, it is necessary to record the connections between different kernels. The core idea is to use simulation and regression to assess the effect of approximating each kernel on accuracy/cost. Scalability is increased by our method, and better answers are discovered for complex applications such as an Image Signal Processor pipeline and the Stereo Matching.

To accelerate the WLO process, we proposed a hybrid approach in Chapter 4 that combines Bayesian optimization (BO) with a quick local search. Experimental results from this chapter provide the first proof that this combination reduces the amount of time spent exploring. We then propose a new technique that can automatically find an appropriate transition point between the two. In order to identify when to switch from BO into local search, we perform a statistical analysis of the convergence of the probabilistic models created during BO and then formulate a stopping condition. The experimental findings show that, for large benchmarks, our technique can save exploration time by as much as 80%.

In Chapter 5, we have examined another aspect of Word Length Optimization that is affected by hardware limitations. It is important to note that the majority of state-of-the-art methods solve WLO with a quality restriction in place. There has been no study on how to improve the quality of service, while keeping costs down. For low-power or low-cost electronic equipment, this is a major obstacle to enhancing the performance of applications. First, we demonstrate the significance of the WLO problem under resource constraints, focusing on systems with constrained energy consumption. We then point out the difficulty in choosing an appropriate initial state for WLO problems inherent in traditional methods like Tabu search. We proposed using a TPE method that incorporates an adjustable loss function. Our experimental findings demonstrate that our technique outperforms the state-of-the-art methods, UWLO and Tabu Search. We also note that the quality of the solution found via Tabu search is very context-dependent. Our method is the first to tackle the resource-constrained WLO issue, and it also serves as a springboard for future work that may enhance the scalability and optimality of the problem.

This thesis opens up several directions of research.

In Chapter 3, the quality function $\hat{\boldsymbol{\lambda}}$ models the output quality of the combined solutions from different kernels. We use the Least Squares technique to approximate this function on the dataset obtained by simulating combined solutions of the kernels. In our current implementation, we take a subset of the data points collected for modeling the cost functions and simulate all combinations to collect the data points. We consider X uniform segments of the quality in the range under consideration, and take the best design within each segment. Then, the X^N combinations are simulated to collect data points. This simulation encounters a scalability issue when the number of kernels N and/or segments X increases. This phase may be optimized using a more advanced subset selection mechanism. An initial model may be developed by simulating the combination of a few data points from each kernel. Using the initial model, further simulations that would provide significant data points may be anticipated. This procedure may be repeated until the improvement in the model's precision begins to decline.

In this thesis, the two independent contributions to accelerate the accuracy-constrained WLO problem solving are presented in Chapter 3 and Chapter 4. On the one hand, we proposed a methodology to reduce the complexity of the original problem by breaking it down into local problems (the kernels) and solving them independently. On the other hand, we proposed a hybrid algorithm with the combination of the Bayesian-Optimization and Tabu search algorithms to speed up the precision tuning in the WLO problem. To further improve the scalability of solving the accuracy-constrained WLO problem, a combination of the two approaches can be applied for larger applications. Indeed, the input application can be divided into smaller kernels. After that, the empirical models proposed in Chapter 3 can be applied to give the noise budget for each kernel. The optimization for kernels can be done using the Hybrid approach proposed in Chapter 4, which allows to quickly obtain the solutions for these kernels.

The proposed approaches in the thesis are generic for different applications. As a result, many assessments for applications in various areas may be used to further demonstrate the efficiency of the methodologies. For example, neural networks are the principal building blocks being used in a variety of applications such as computer vision, natural language processing, and robotics. The implementation of neural networks on embedded devices requires energy efficiency within a resource budget, which is important to maintain the performance of the devices for a long time of use. Neural network architectures are composed of several layers and many variables that express the values of the weight factors and the results of intermediate computations. They are

good resources for precision tuning to seek energy efficiency. However, it is difficult to analyze these designs and provide the precise accuracy for each kernel and/or variable. The noise budget technique proposed in Chapter 3 can be applied to separate the cost and accuracy impacts of kernels for independent optimizations, and then the heuristic search presented in Chapter 4 can explore the near optimal word-length configurations of each kernel. Finally, by combining the local settings, the word-length configuration of the full neural network architecture can be effectively obtained.

Apart from the fixed-point datatype, custom floating-point is also a promising number representation for approximate computing, especially in the context of low precision. As mentioned in Chapter 2, the floating-point format is comprised of mantissa and exponent parts. The exponent part controls both the accuracy and the dynamic range of the representation, whereas the mantissa only adjusts its precision. In the context of WLO, both exponent size and mantissa size therefore need to be adjusted. Various custom floating point datatypes need to be optimized to seek efficiency in specific applications and their implementation and there is still plenty of room to investigate in this direction. Current techniques need to be improved to properly capture the cost/accuracy trade-offs when both mantissa and exponent are customized. This opens a promising research direction to explore the use of highly customized floating-point types for various applications.

List of publications

[1] V.-P. Ha, T. Yuki, and O. Sentieys, "Towards generic and scalable word-length optimization," in *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1668–1673.

[2] V.-P. Ha and O. Sentieys, "Leveraging bayesian optimization to speed up automatic precision tuning," in *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1542–1547.

[3] ——, "Maximizing computing accuracy on resource-constrained architectures," in *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Antwerp, Belgium, Apr. 2023.

[4] V.-P. Ha, T. Yuki, and O. Sentieys, "Noise Budgeting in Multiple-Kernel Word-Length Optimization," in *4th Workshop on Approximate Computing (AxC)*, Florence, Italy, Mar. 2019, pp. 1–3.

Appendix A: TypEx - An Automatic Floating-point to Fixed-point Conversion Toolbox

The experiments in Chapter 3 are performed using the TypEx [1] tool, which is a part of the Gecos [2] project. In this appendix, we provide the user manual of TypEx. Tomofumi Yuki, Ali El-Moussawi, Olivier Sentieys and Van-Phu Ha also contributed to the development of TypEx as well as to the writing of this user manual. TypEx is designed to automatically determine

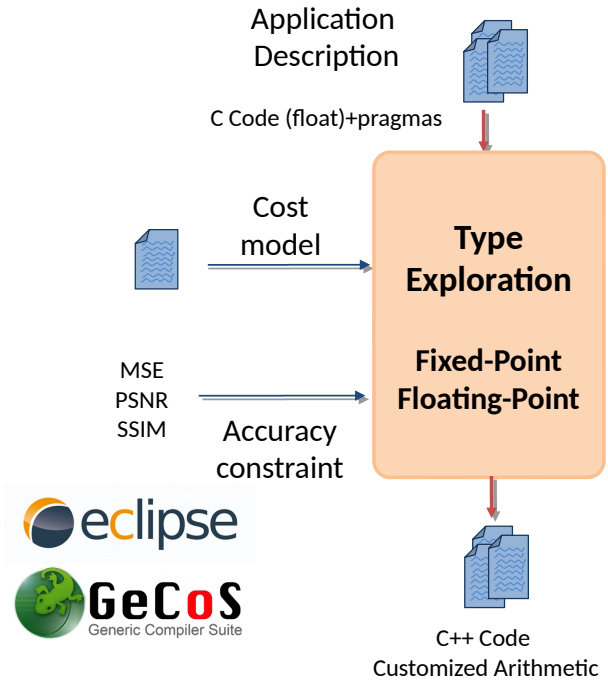


Figure A.1: TypEx: a tool for type exploration and automatic floating-point to fixed-point conversion

custom number representations and word-lengths (i.e., bit-width) for FPGA and ASIC designs at the C source level. The main goal of TypEx is to explore the design space spanned by possible number formats in the context of High-Level Synthesis. TypEx takes a C code written using

1. https://gitlab.inria.fr/gecos/gecos-float2fix
2. https://gitlab.inria.fr/gecos

floating-point datatypes specifying the application to be explored. The tool also takes as inputs a cost model as well as some user constraints and generates a C code where the floating-point datatypes are replaced by the word-lengths found after exploration. The best set of word-lengths is the one found by the tool that respects the given accuracy constraint and that minimizes a parameterized cost function. Figure A.1 presents an overview of the TypEx design flow.

The exploration flow follows the classical simulation-based exploration. The design space is explored by running simulations to evaluate the accuracy, guided by an algorithm of choice. The main flow consists of the following steps.

1. Write a C program using floating-point datatypes that implement the desired computation. The segments of the code that should run on hardware should be separated as one or more functions. The C code must follow some conventions to be properly processed by the flow (see Section Writing C Specification for more details).
2. Specify the design space by providing the range of wordlengths to explore. This may be provided for individual variables through pragmas, or as global default for all variables (see Section Selecting the Exploration Space for more details).
3. Select the parameters of exploration. The main parameters include cost metric to optimize, the accuracy metric, accuracy constraints, and exploration algorithm (see Sections Selecting the Exploration Space and Selecting Parameters of the Exploration for more details).
4. Run the exploration algorithm. It will report all the designs tested during exploration and return the design with lowest cost that satisfies the accuracy constraint (see Section Selecting Parameters of the Exploration for more details on exploration algorithms).

Figure A.2 illustrates a screenshot of TypEx toolbox. The tool is written in Java and runs under Eclipse. On the left of the figure is shown the manager of the different files imported in the project. This includes

— C files to be explored (.c and .h),

— Gecos scripts to run the tool (.cs), and

— configuration files defining user constraints (.properties).

The datatypes of variables in a C/C++ source code can be explored by using a `#pragma` primitive **#pragma EXPLORE_FIX W={a..b}, I={c}**, where a and b define the fractional bitwidth range for exploration, c is the integer bitwidth. Based on the selection criteria of each optimization technique, the word-length of the variables under consideration will be changed throughout the exploration process. The process repeats itself until a stopping condition is met.

Limitations:

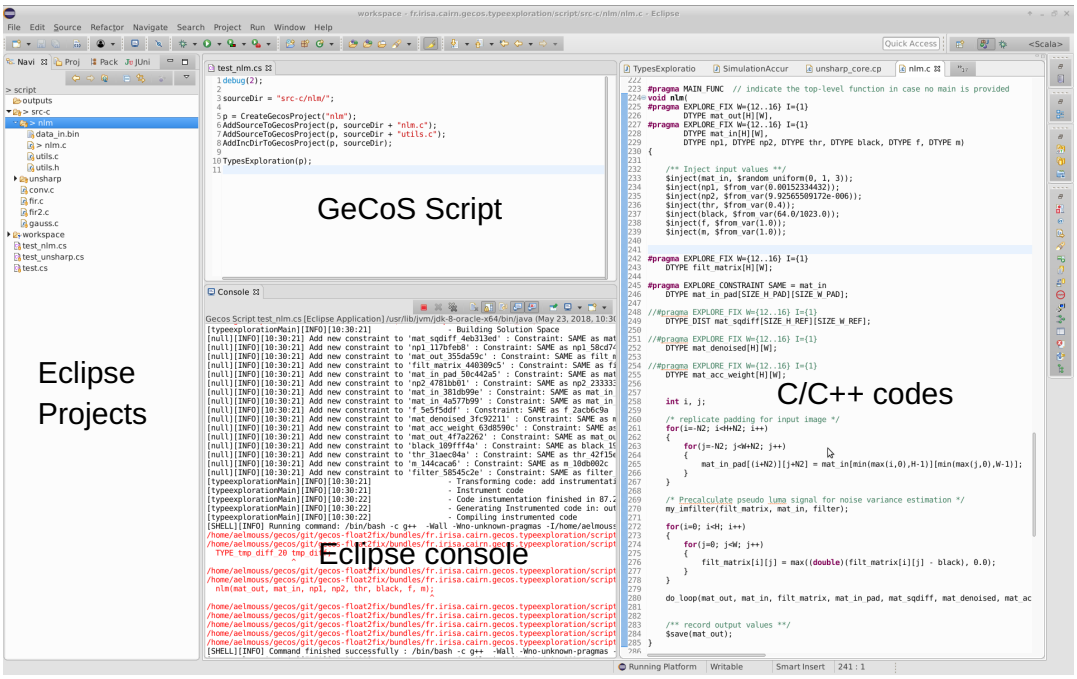


Figure A.2: TypEx Tool Screenshot

1. The flow does not perform dynamic range analysis. It is assumed to be performed independently and the approximate wordlength for the integer part (or exponent) should be specified. However, the tool provides some means to evaluate the dynamic range during simulation. Figure A.3 presents an example of histograms containing profiling results of data values during exploration that can be generated by the tool.
2. The flow mainly supports fixed-point representation. Support for custom floating points is planned, but not fully available yet.

Writing C Specification

The input specification is written as C programs. The flow **does not** support C++ code. The flow targets a function annotated by `#pragma MAIN_FUNC` as the top-level function. This is analogous to the top-level function specified in HLS tools, and defines the region under exploration. Within this top-level function, typical HLS restrictions (such as no malloc, no recursive calls) apply.

Granularity of Exploration

How the C program is written affects the design space being explored. The variables in the source code defines the granularity of the exploration. For example, consider the following:

Figure A.3: Example of histograms resulting from profiling of data values during exploration

```
float a, b;
a = ...
b = a + 1;

a = ...;
b = a + 100;
```

The above describes two independent computations that reuses the variables `a` and `b`. In other words, you may rename the latter occurrences of `a` and `b` with `x` and `y` without changing the semantics. Thus, it may seem appropriate to assign different number representations for the different uses of `a` and `b`. However, the type exploration flow will only assign a number format for each variable that is common to all uses of the variable.

If you want the two uses of `a` and `b` to have different number representations, then they must be explicitly given different names:

```
float a, b;
a = ...
b = a + 1;

float x, y;
```

```
x = ...;
y = x + 100;
```

What is explored by the flow are possible design points that can be expressed by modifying the variable declaration. It is possible to automatically apply additional transformations to refine the granularity, but this is not supported. How the variables are declared is viewed as a way to specify the granularity of the exploration.

Function Interface

The top-level functions may call other functions. For every function call, the exploration flow automatically constrains the function parameters to have the same number representation as the input arguments at the call cite.

For instance, if you have:

```
foo(float x) { ... };

float a, b;
a = ...
b = ...
foo(a);
foo(b);
```

Then all variables `a`, `b`, and `x` must have the same number representation.

If a function is called multiple times with different types, then there must be multiple instances of the function. In the above, example, if the `a` and `b` have different data types, then the function `foo` requires two different implementations, one for each data type.

If two separate instances of a function is desired, the function should be duplicated or inlined.

Directives for Profiling

The type exploration flow generates many versions of the top-level function for simulation. The top-level function is wrapped by code that populates the input variables, and code that writes output variables for analysis purposes.

The input generation and profiled variables are controlled by directives - a special function processed by GeCoS when generating code.

`$save` Directive

All variables to be profiled must be specified at the observation point (usually at the end of the top-level function) using the `$save` directive.

The syntax of the `$save` directive is:

```
$save(<symbol> [, <size_outermost>, ..., <size_innermost>]);
```

If the symbol is not a scalar, its dimension sizes must be specified in the `$save directive`.

`$inject` Directive

If `main` function is available in the input code, then the exploration flow assumes that the `main` does the necessary initializations and calls the top-level function.

When there is no `main`, then one will be generated by the flow. In this case, the variable initializations must also be generated. The `$inject` directive is used to control this initialization.

The `$inject` directive has the following syntax:

```
$inject(<symbol>, SOURCE, [, <size_outermost>, ..., <size_innermost>]);
```

If the symbol is not a scalar, its dimension sizes must be specified as additional arguments. `SOURCE` can be either:

- Random with uniform distribution: `$random_uniform(min, max [, seed])`
- Random with normal distribution: `$random_normal(mean, stddev [, seed])`
- C expression (as String): `$from_var(<any expression that evaluates as double>)`
- Read from file: `$from_file(<file_path>)`

  Currently, supports `.txt` files with the following format:

  ```
  ND_DIMS
  SIZE DIM 0
  ...
  SIZE DIM ND_DIMS-1
  first value
  second value
  ...
  ```

  `.png` image files in grayscale, and `.raw` image files. png files are normalized by 256. raw files are normalized by 1024 assuming 10-bit images of size $W = 3968$ and $H = 2976$ (this can be changed in the code if necessary).

Note that `$inject` directive can be used anywhere even if `main` is defined, it will simply override the symbol values at its location.

`$size` Directive

The `$size` directive may be used prior to `$save` or `$inject`:

```
$size(<symbol>, <size_outermost>, ..., <size_innermost>);
```

This is an alternate way to specify the size information that is needed for `$save` and `$inject` directive. Using `$size` avoids the need to specify the size in both `$save` and `$inject`.

Other Coding Guidelines

— Your code should not define any function named as any of the directives (`$save`, `$inject`, or `$size`)!

— Do **not** use operations between a floating-point symbol and a immediate constant. This may cause compilation errors when we later change the type of the symbol. More generally, keep in mind the fact that the floating-point symbols will be compiled with different types during the exploration. Operations that might cause conflicts or ambiguity in the eventual backend library (e.g., `ac_fixed`) should be avoided. Currently the tool can only use AC_DataTypes, but this could be easily extended to other libraries.

  For example, consider the following:

```
double x;
.. = x * 0.5;
```

  During the exploration the type of x might be changed to `ac_fixed<..>`, this will result in a compilation error (ambiguous overload for 'operator*') since the `ac_fixed` library does not define such behavior.

  This particular problem can be avoided by extracting the constant into a separate variable.

— Do **not** use type operations using explicit types, for example:

```
#define TYPE double
...
TYPE *p = (TYPE *)malloc(sizeof(TYPE)*4);
```

  In this case, the cast and the `sizeof` operations are not allowed, since they are not modified when the flow generates different designs by changing the data type of variable `p`. This problem can be avoided by using `typedef`. However, one `typedef` can currently be used by one symbol only. The flow does not handle the same type by a `typedef` for 2 different symbols. For example:

```
typedef double TYPE;
...
TYPE *p = (TYPE *)malloc(sizeof(TYPE)*4); // OK
TYPE x; // NOT SUPPORTED !
```

  uses the same type for two variables, which is not supported.

Selecting the Exploration Space

The space of exploration may be defined for each variable using pragmas or as a global configuration. In this section, fine tuning of the exploration space through pragmas is explained.

These pragmas are called `EXPLORE` annotations and are attached to variable declarations in the input code. There can be multiple annotations to a variable, as long as they do not conflict each other.

A default configuration is used for symbols with no `EXPLORE` annotations. This default configuration can be set in the configuration file explained in Section Selecting Parameters of the Exploration.

`pragma EXPLORE_FIX`

Annotate a variable with `pragma EXPLORE_FIX` to define the set of fixed-point configurations to be explored:

```
#pragma EXPLORE_FIX W={SET_VALUES} I={SET_VALUES}}
float symbol;
```

where

- `SET_VALUES := (<min>..<max> | <value>)[, (<min>..<max> | <value>)]+`
- `W`: values to be explored for the *total wordlength*
- `I`: values to be explored for the integer part

The specification follows the conventions by AC_DataTypes; note that the fractional part is implicitly defines as `W - I`.

`pragma EXPLORE_FLOAT`

Annotate a variable with `pragma EXPLORE_FLOAT` to define the set of custom floating-point configurations to be explored:

```
#pragma EXPLORE_FLOAT W={SET_VALUES} E={SET_VALUES}
float symbol;
```

where

- `SET_VALUES := (<min>..<max> | <value>)[, (<min>..<max> | <value>)]+`
- `W`: values to be explored for the *total wordlength*
- `E`: values to be explored for the exponent

`pragma EXPLORE_CONSTRAINT`

Annotate a variable with `pragma EXPLORE_CONSTRAINT` to define constraints on the choice of wordlengths.

There is only one constraint in the current version:

```
#pragma EXPLORE_CONSTRAINT SAME = <variable name>
```

where the number representation of the annotated variable with be forced to be the same as the variable specified by the pragma.

Note that you cannot have cycles with `SAME` constraints. The tool will detect cycles and complain when found. It is possible that a cycle is unintentionally created due to automatically addd `SAME` constraints for function calls (see Section Function Interface).

Selecting Parameters of the Exploration

The exploration flow exposes many parameters though a configuration file (`.properties`). A file with default properties may be automatically generated, which can be modified to customize the flow.

The exposed parameters are:

- `nbThreads`: Number of threads to be used.
- `nbSimulations`: Number of simulations to be performed when evaluating a design. The input can be made different for each of these simulations.
- `enableCharts`: When set to true, the progress is visualized during the exploration.
- `mainLogLevel`: Selects the logging level of the entire flow.
- `explorationLogLevel`: Selects the logging level of the exploration.
- `explorationMode`: Selects the target number representation (FIXED or FLOAT).
- `timeTagOutput`: When set to true, outputs are stored in different directories for each run (tagged by time).
- `nbOutputsToKeep`: Number of time-tagged outputs to keep.
- `pruneFirst`: If pruning is applied before starting the main exploration. See Section Pruning for details.
- `explorationAlgo`: Exploration algorithm to use. See Section Exploration Algorithms for details.
- `accuracyMetric`: The accuracy metric(s) to use. See Section Accuracy Metrics for details.
- `accuracyThreshold`: The threshold(s) on the selected accuracy metric(s).
- `costMetric`: The cost metric to use. See Section Cost Metrics for details.

— `SSIMenabled`: When set to true, SSIM is calculated. All other metrics such as PSNR is always computed. SSIM requires a specific flag since it is expensive to compute.

— `SSIMtarget`: Variable name used to compute the SSIM.

— `defaultFixedW`: Default range of total wordlength explored for fixed-point exploration. Accepts Comma-separated, positive integer value or value range (min..max)

— `defaultFixedI`: Default range of integer part length explored for fixed-point exploration. Accepts Comma-separated, positive integer value or value range (min..max)

— `defaultFloatW`: Default range of total wordlength explored for custom floating-point exploration. Accepts Comma-separated, positive integer value or value range (min..max)

— `defaultFloatE`: Default range of exponent length explored for custom floating-point exploration. Accepts Comma-separated, positive integer value or value range (min..max)

All the parameters are also explained in comments of the automatically generated property file.

Exploration Algorithms

Figure A.4 presents the general principle of the exploration algorithm. As already mentioned, the exploration takes as inputs some C code and user configurations, and generates a C code enriched with fixed-point (or custom floating-point) datatypes found by the tool. The exploration algorithm follows the general principle:

— A candidate solution is generated where the wordlength (and datatype) of each variable in the program has been decided by the exploration algorithm.

— This candidate solution is simulated using the set of inputs provided by the user. The wordlength chosen implies reduced precision and therefore a corresponding set of outputs is generated after this simulation.

— The simulated set of outputs is compared to the golden reference (obtained from a single simulation in floating-point (single- or double-precision) performed before exploration) and an accuracy metric is calculated (see Section Accuracy Metrics for details on the available accuracy metrics).

— The cost of the candidate solution is estimated (see Section Cost Metrics for details on the available cost metrics).

— Another candidate solution is generated by the exploration algorithm according to accuracy and cost values previously obtained. Then these different steps are repeated until the exploration algorithm finishes its iteration space and finds a solution. The best solution is one of the set of wordlengths explored by the algorithm that respects the accuracy constraint given and that provides the minimum cost.

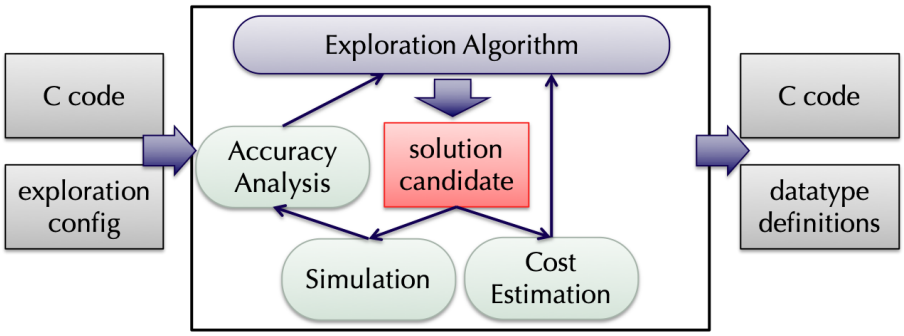


Figure A.4: General principle of the exploration algorithm

There are currently three exploration algorithms: brute force, min+1, and Tabu search. In addition, pruning can be optionally performed to reduce the design space.

Brute force exhaustively evaluates all possible solutions. This is infeasible except for cases with extremely small number of variables.

All algorithms (except brute force) are a form of gradient descent. At each iteration, neighboring designs are explored and the best one is selected. The main difference in the available algorithms are starting point and terminating condition.

The "best" design is selected from the neighbor as a function of how much accuracy and cost changed compared to the previously selected design. Currently, all methods use the following:

$$\frac{\delta\text{Accuracy}}{\delta\text{Cost}}$$

where

— δAccuracy is the increase in the accuracy compared to the previously selected design. The value is normalized to take values in $[-1, 1]$; 0 when the accuracy is unchanged.

— δCost is the increase in the cost compared to the previously selected design. The value is normalized to take values in $[-1, 1]$; 0 when the cost is unchanged.

The above favors designs that has highest ratio of accuracy improvement to cost degradation.

Pruning

Pruning keeps all but one variable in floating point, and determines the minimum wordlength that a variable can take without violating the accuracy constraint. This gives the lower bound on the valid wordlength for a variable, assuming that the error introduced by this variable will not be cancelled by others. The exploration flow performs a binary search for each variable, and prunes wordlength smaller than the minimum valid wordlength found. Figure A.5 illustrates the

pruning on a non-local mean (NLM) denoising kernel with two accuracy metrics (PSNR and SSIM) and nine variables. As an example (low part of Figure A.5), wordlengths under 11 bits for any variable can be ignored if the accuracy constraint is to keep SSIM near to 1.0 (e.g., 0.99). This enables to prune the design space by removing candidate solutions where it is likely that the accuracy constraint will not be met.

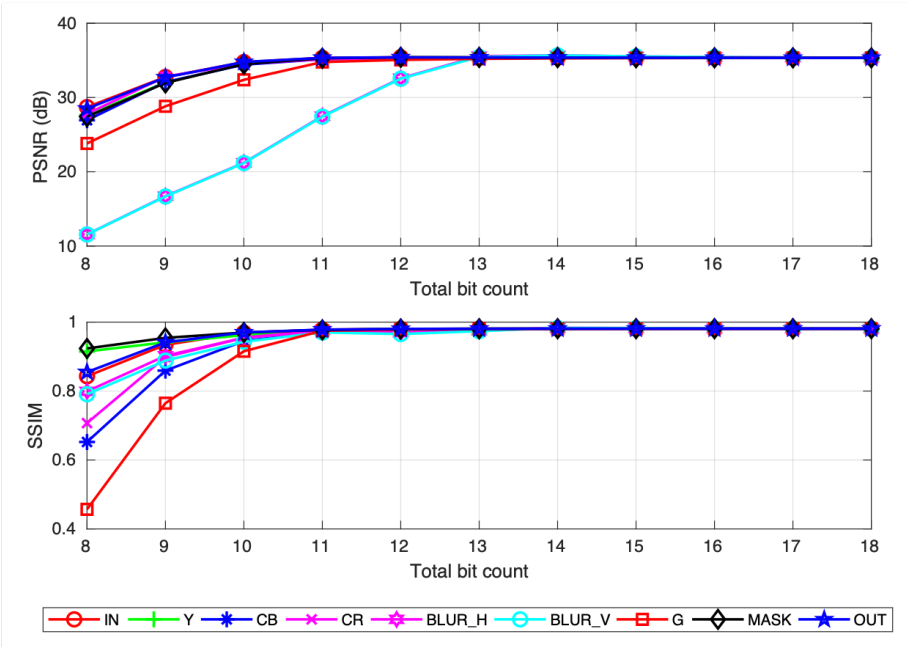


Figure A.5: Example of pruning on a non-local mean (NLM) denoising kernel with two accuracy metrics (PSNR and SSIM) and nine variables

min+1

The min+1 algorithm is a greedy gradient decent [3, 2]. It starts at the lowest wordlength choices in the valid range, and greedily follows the gradient. The gradient is followed until a design that satisfies the accuracy constraint is found.

Tabu search

Tabu search performs a more detailed search after reaching a design that satisfies the constraints in contrast to min+1 [4, 2]. The algorithm repeatedly switches the direction: increases wordlength when the accuracy constraint is not met, and decreases when it is. As the direction is switched, the most influential variable that is left (i.e., the one that changed in the selected design after evaluating the neighboring designs) is put in to the "tabu list", freezing its current choice of wordlength. Figure A.6 illustrates how the Tabu search algorithm tries to escape from

local minima by repeatedly switching the direction of the search space, as opposed to min+1 which would greedily follow the gradient descent and return the first solution found that meets the accuracy constraint.

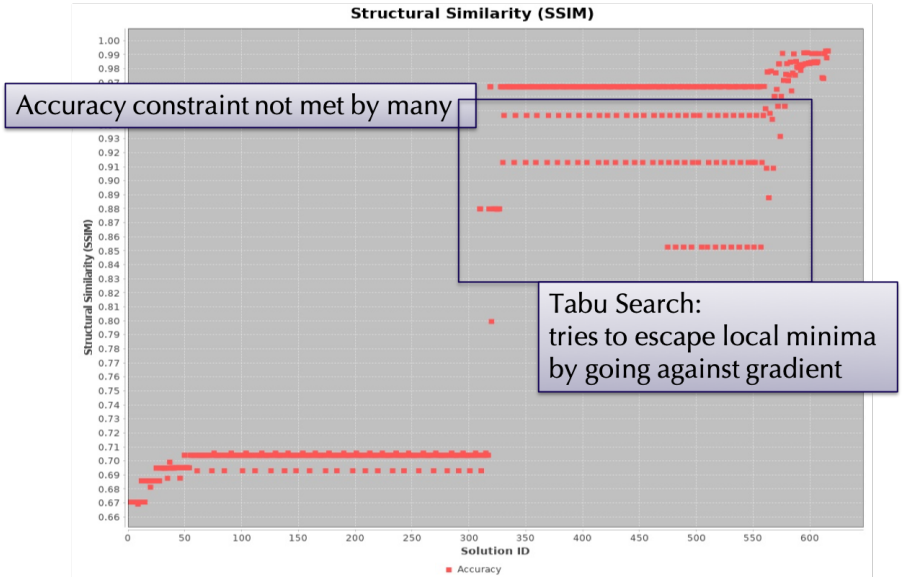


Figure A.6: Example of solutions explored by the Tabu search algorithm

Cost Metrics

There are currently three cost metrics: sum of wordlengths, area model, and energy model.

Sum of wordlengths

The sum of wordlengths is a naïve cost model mostly for debugging purposes. It defines the cost as the sum of wordlengths of all variables.

Area and energy models

The area and energy models are simple models that are aimed to capture the sharing of hardware resources (concrete operators) as a function of desired throughput. The model assumes that all operations can be executed in parallel and computes the number of concrete operators that needs to be instantiated to meet the throughput requirement. The area cost or the energy consumption is then calculated based on empirically measured cost/consumption of concrete operators for the target process technology. The cost models are estimated based on properties (area, power, delay and error) of synthesized operators (adder and multiplier) on

28nm technology (Table A.1). We synthesized the operators with different bitwidths of input operands and the result.

Table A.1: A part of data for the cost model. The data contains the synthesized result of different adders and multipliers (different input/output wordlengths).

OPE	IN1	IN2	OUT	AREA (μm^2)	POWER (W)	DELAY (ns)	MSE
ADD	4	4	4	28.0704	3.35E-06	0.15	0.007809
ADD	6	4	4	28.0704	3.35E-06	0.15	0.017118
ADD	6	4	6	33.9456	4.42E-06	0.17	0.000488
ADD	6	6	4	30.8448	3.40E-06	0.21	0.017101
ADD	6	6	6	39.4944	4.49E-06	0.22	0.000488
...	...	...	...	...	...	...	...
MUL	4	4	4	47.4912	3.66E-06	0.23	0.015915
MUL	4	4	6	59.2416	4.78E-06	0.27	0.000549
MUL	4	4	8	67.8912	5.78E-06	0.28	0
MUL	6	4	6	73.4400	4.77E-06	0.33	0.000995
MUL	6	4	8	83.2320	6.00E-06	0.35	3.43E-05
...	...	...	...	...	...	...	...

Figure A.7 shows the example of an energy model for a multiplier as a function of input wordlength. The energy is measured on an FPGA Xilinx Virtex6 with multipliers implemented in DSP blocks and for various wordlengths (from 4 to 48). Figure A.8 shows the example of an area model for an adder as a function of input wordlength. The adder is synthesized on a 28nm ASIC technology for various wordlengths (from 8 to 32) and various frequency constraints, and the area in square microns is reported.

These cost models of elementary operations (addition, subtraction, multiplication, division) are used during the exploration to estimate the cost (i.e., energy or area) of individual operation as a function of inputs and output wordlength and to construct the cost function of a candidate solution.

Specifying operation count for each variable

The model requires the number of (dynamic) operations for each pair of variables to be known. This could be automatically computed from the source code if the runtime parameters are fixed (which is always the case for simulations). However, it is not automated in the current version. The operation count and the target throughput is specified by a simple DSL, which looks like the following:

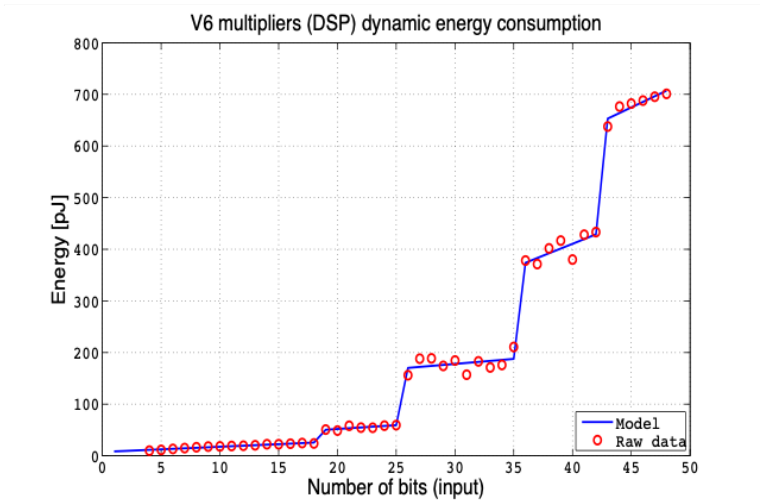


Figure A.7: Energy model of multiplier for an FPGA target (Xilinx Virtex6, multipliers implemented in DSP blocks) as a function of input wordlength

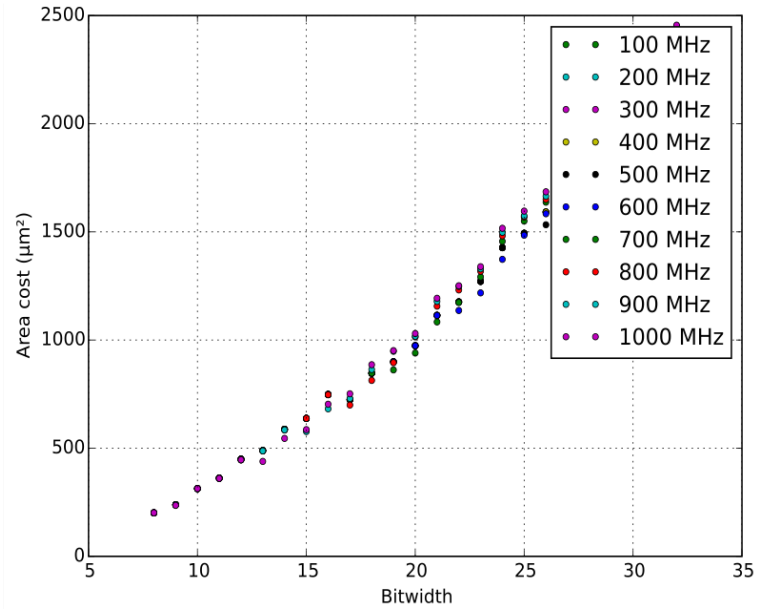


Figure A.8: Area model of an adder for a 28nm ASIC technology as a function of input wordlength

```
param H=384;
param W=512;

target {
    frequency = 500
    outputsPerCycle = 2
    problemSize = H W
}

block nlmADD {
        ADD : mat_in_pad = mat_in_pad op mat_in_pad x 2 H W;
        ADD : sqdiff_current = sqdiff_current op sqdiff_prev x H W;
        ADD : sqdiff_save = sqdiff_current op mat_sqdiff x H W;
        ADD : tmp_dist =  mat_sqdiff op  mat_sqdiff x 3 H W;
        ADD : pre_exp = tmp_dist op sigma2 x H W;
        ADD : mat_out = mat_out op denoised x H W;
        ADD : mat_acc_weight = mat_acc_weight op weight x H W;
        ADD : mat_out = mat_out op mat_in_pad x H W;
}

block nlmMUL {
        MUL : 2 mat_in_pad = mat_in_pad op mat_in_pad x H W;
        MUL : denoised = weight op mat_in_pad x H W;
        MUL : mat_out = mat_acc_weight op max(mat_out, mat_in_pad) x H W;
}
```

The number of operations of each operator are specified for each pair of variables in the program. The blocks are grouping that is user defined.
For the complete grammar of the DSL, see the Xtext grammar:
`https://gitlab.inria.fr/gecos/gecos-float2fix/blob/master/bundles/fr.irisa.cairn.gecos.typeexploration.computation.xtext/src/fr/irisa/cairn/gecos/typeexploration/Computation.xtext`

Accuracy Metrics

There are a number of supported accuracy metrics:

- Maximum Absolute Error
- PSNR (Peak Signal to Noise Ratio)
- Noise Power
- SSIM (Structural Similarity)

Any number of these accuracy metrics can be used to define the accuracy constraint.

Source Code Repository

The type exploration flow is available open source at:
`https://gitlab.inria.fr/gecos/gecos-float2fix`
See README.md for detailed instructions on how to install the flow.

Bibliography

[1] Menard D, Caffarena G, Lopez JA, Novo D, Sentieys O. Analysis of Finite Word-Length Effects in Fixed-Point Systems. In: Handbook of Signal Processing Systems. pp. 1063-1101, 2019.

[2] Menard D, Caffarena G, Lopez JA, Novo D, Sentieys O. Fixed-point refinement of digital signal processing systems. In: Digitally Enhanced Mixed Signal Systems. 2018.

[3] Kim S, Kum K, Sung W. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. In: Workshop on VLSI and Signal Processing. 1995.

[4] Nguyen HN, Menard D, Sentieys O. Novel Algorithms for Word-length Optimization. In: 19th European Signal Processing Conference (EUSIPCO). Barcelona, Spain. 2011. Available from: `https://hal.inria.fr/inria-00617718`.

Appendix B: An Extension of TypEx in Python

A Python framework that provides the solution space modeling and different solvers for word-length optimization has also been developed during this thesis. Van-Phu Ha is the main developer for this framework that extends the original TypEx tool with the addition of Bayesian Optimization algorithm. The development and expansion of TypEx on Python makes it easy to deploy available optimization algorithms to address WLO problem. Besides, several analysis and visualization toolboxes can be applied to enhance the efficiency of exploration strategies. We plan to add more optimization strategies to further improve the scalability and optimality of WLO problem. Note that the results of experiments in Chapter 4 and Chapter 5 are based on the features of the python TypEx toolbox.

Requirements

— `Hyper-Opt` package (`https://github.com/hyperopt/hyperopt`)

— `Scikit-Learn` library (`https://github.com/scikit-learn/scikit-learn`)

Structure of the framework

This section provides information about the organization of the different folders in the Python TypEx toolbox, following the structure reported in Figure B.1.

`application`

The folder `application` contains the applications written in C/C++ source code. A cost model written in Python `costModel.py` accompanies each application to describe the arithmetic operations (addition, subtraction, multiplication, and division) in the application. The cost model is then used to estimate the total cost of the application. The operators differ in word-length of input, output and the operation types (addition, subtraction and multiplication).

In the C code, the datatype used for variables is defined as `TYPE_XYZ` which is then assigned to a certain datatype during the compilation process. For example, a datatype customizing variable can be declared from a double format as follows:

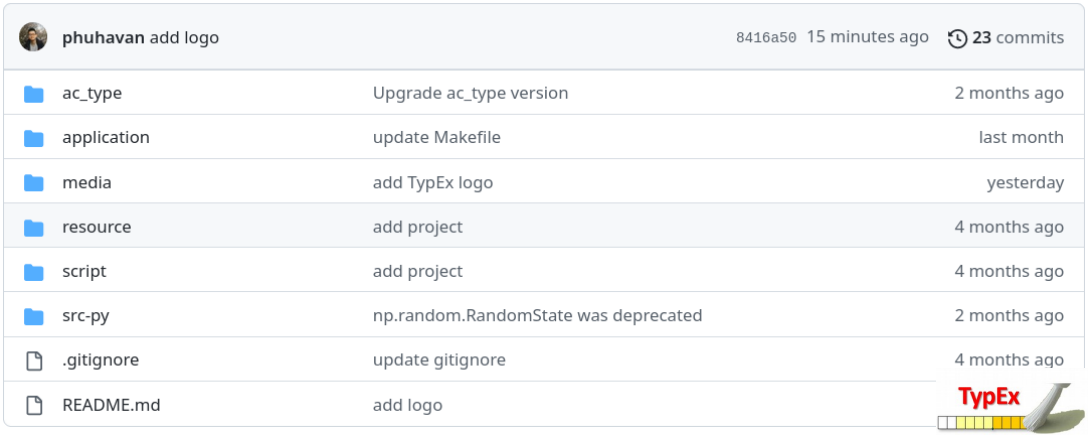


Figure B.1: TypEx on Python repository

— A number `x` is defined in double:

```
double x;
```

— A number `x` is defined in a format meant to be customized:

```
TYPE_X x;
```

A cost model is a combination of operation declarations in an application. The syntax of an operation declaration is as follows:

```
block.addOperation(OperatorConfiguration("OPT_TYPE",
solInfo['TYPE_IN1'] ,solInfo['TYPE_IN2'] ,solInfo['TYPE_OUT']), NUM)
```

— `OPT_TYPE` is `ADD` or `MUL`
— `TYPE_IN1` is the first input name. `solInfo['TYPE_IN1']` will return the bit-width of the first input. Similarly, `solInfo['TYPE_IN2']` will return the bit-width of the second input
— `TYPE_OUT` is the output name and `solInfo['TYPE_OUT']` represents the bit-width of the output
— `NUM` is the number of operations

ac_type

The folder `ac_type` includes libraries for customized datatype. The datatype is provided by Mentor Graphic.

resource

The folder `resource` includes data in area, power, delay and quantization error for synthesized operators (adder and multiplier) with different input/output wordlength. The technology

28nm is currently used to synthesize the operators.

scr-py

The folder `scr-py` includes implementation of optimization algorithms and infrastructure of the platform.

script

The folder `script` contains scripts to explore WLO with different search algorithms and user constraints.

Project creation

The first configuration is to setup the C/C++ compiler.

```
gcc="g++"
```

Each exploration will create a folder that contains all exploration results and outputs. We can configure the number of latest results to be kept.

```
nbOutputsToKeep = 10
```

For each exploration, we need to define the cost/accuracy metrics and the accuracy constraint. We provide three cost models: energy, area and total bit count. The accuracy metric can be chosen between PSNR [3] and SSIM [4]. The example below sets PSNR = 40dB as the accuracy constraint.

```
costType = "ENERGY_MODEL" # [ENERGY_MODEL, AREA_MODEL, SUM_W]
accType = "PSNR" # [PSNR, SSIM]
accTarget = userAccuracyConstraint(isHigherBetter=True,
accType=accType, value=40)
```

Define the problem

The example below shows the exploration setup for an `fir` (Finite Impulse Response) filter that contains three variables `data`, `coeffs` and `acc`. We can define the integer word-length for variables, the minimum and maximum fractional word-length to be explored. The function

3. <https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio>
4. <https://en.wikipedia.org/wiki/Structural_similarity>

HWSpec defines the frequency freq to be synthesized, the throughput outputsPerCycle and the problem size problemSize corresponding to the total elements in an input matrix/array/-dataframe to be processed.

```
app = "fir" # application name
hwspec = HWSpec(freq = 500, outputsPerCycle = 1, problemSize = 5)
integerWL = 4
firstW = 6
lastW = 20
symbolNames = ["TYPE_data","TYPE_coeffs","TYPE_acc"]
projPath = os.path.abspath("application/fir-example")
srcPath = os.path.join(projPath, "src-c")
srcFile = "fir_explore.c"
```

We apply the following template to create the design space exploration procedure:

```
ss = SolutionSpace(symbols)
proj = ProjectCreatation(ss, projPath, srcPath, srcFile, nbOutputsToKeep)
Metric(proj, gcc, accType, costType, hwspec)
sys.path.append(projPath)
```

Optimization procedures

We offers several exploration strategies: Tabu search, Bayesian Optimization, Min+1, Max-1 and GRASP. It is worth noting that we can use a pruning option to eliminate some infeasible solutions from the design space. This option can be used before the five strategies to speed up the exploration.

```
pruneSolutionSpace(ss, accTarget)
```

We provide API wrappers to call the optimization procedures:

— Min+1

```
MinPlusOneExploration(ss_algo, accTarget, costTarget)
```

— Max-1

```
MaxMinusOneExploration(ss_algo, accTarget, costTarget)
```

— Grasp

```
GRASPExploration(proj, ss_algo, accTarget, costTarget)
```

— Tabu Search

```
TabuExploration(ss_algo, accTarget, costTarget)
```

— Bayesian Optimization

```
# costRange = [min_cost, max_cost]
# accuracyRange = [min_accuracy, max_accuracy]
# costRange and accuracyRange are used for the normalization
hyper_opt_wrapper(ss_algo, kernel, accTarget, costRange,
                  accuracyRange, costRange, accuracyRange)
```

Figure B.2 shows a visualization of the exploration progress with 300 iterations. The figure gives us a good view of the search progress and provides different trade-offs between cost, accuracy and loss value by the iteration.

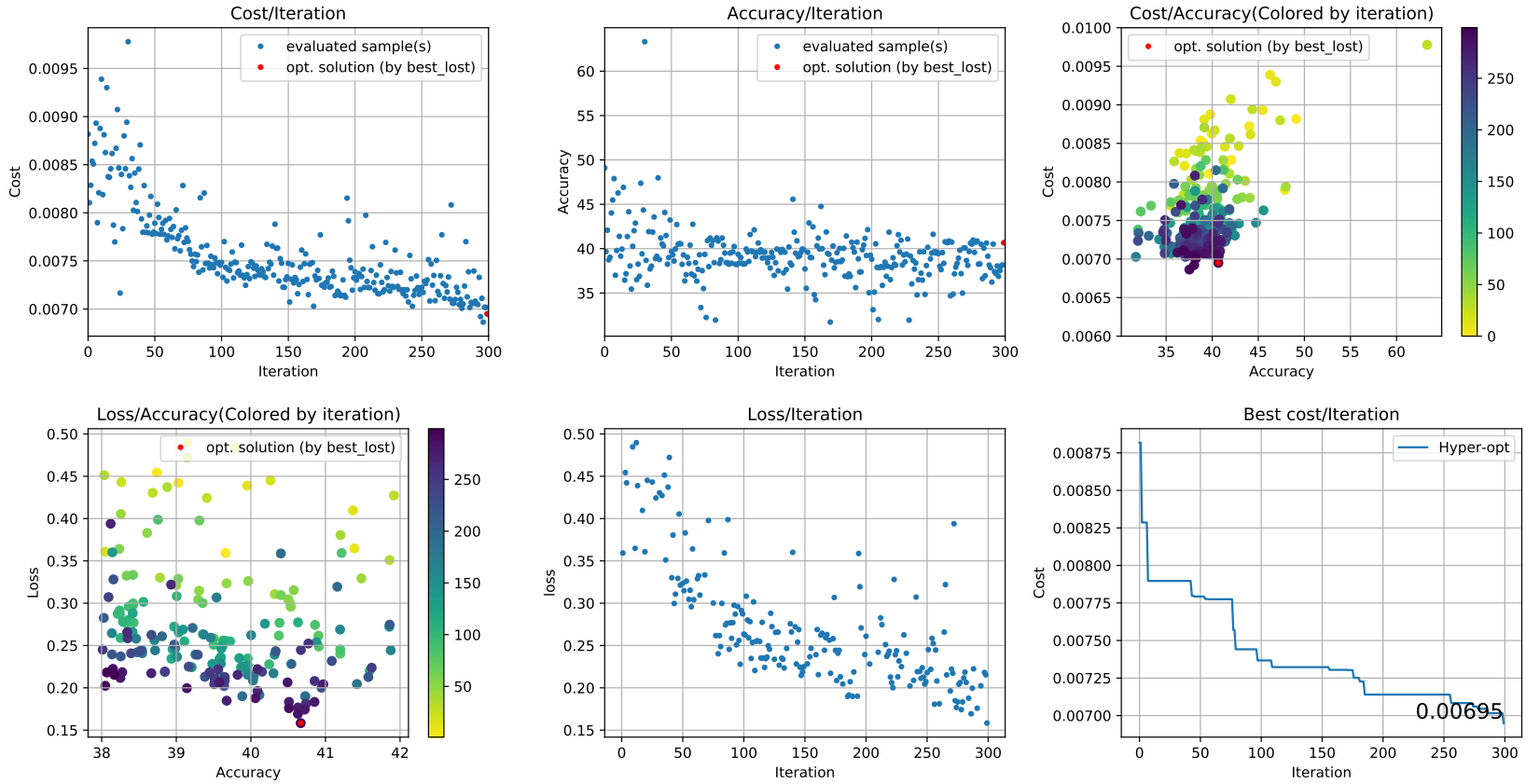


Figure B.2: Exploration visualization

Université de Rennes

Titre : Contributions au passage à l'échelle de l'optimisation de la précision des calculs

Mot clés : optimisation des largeurs, arithmétique virgule fixe, calcul approximatif

Résumé : La consommation d'énergie est l'un des problèmes majeurs de l'informatique aujourd'hui, du calcul haute performance aux systèmes embarqués. Ces dernières années, l'approximation des calculs a reçu un regain d'intérêt pour améliorer l'efficacité énergétique. De nombreuses applications n'exigent pas une précision élevée, et les techniques de calcul approximatif augmentent l'espace de conception en fournissant de nombreux compromis entre la précision, les coûts et les performances. Cette thèse se concentre sur le développement de méthodes pour l'exploration systématique de cet espace de conception, y compris la modélisation de la performance et de la précision et l'automatisation de la conception. Nous utilisons la virgule fixe pour la représentation des données des données et nous optimisons la longueur du mot de chaque données et calcul pour chercher un bon équilibre entre le coût et la précision. Ce problème est appelé *Word length Optimization* (WLO) ou réglage automatique de la précision. Cette thèse contribue à trois directions de recherche. Premièrement, une méthode est proposée pour améliorer le passage à l'échelle du WLO pour les grandes applications. Pour réduire la complexité exponentielle de la nature de WLO, l'application d'entrée est décomposée en *noyaux* qui sont ensuite résolus indépendamment. Pour allouer les budgets de réduction de précision à chaque noyau, l'idée principale est de caractériser l'impact de l'approximation de chaque noyau sur la précision et le coût par simulation et régression pour construire les modèles empiriques. La deuxième direction de recherche est un algorithme hybride combinant l'optimisation bayésienne (BO) et une recherche locale rapide pour accélérer la procédure WLO. Un mécanisme efficace est proposé pour obtenir de bons modèles en peu de temps. La dernière contribution ouvre une nouvelle voie de recherche sur le WLO avec contraintes de ressources. Les approches actuelles résolvent principalement les problèmes de WLO avec une contrainte de qualité (précision). Dans cette étude, un algorithme basé sur l'optimisation bayésienne a été proposé pour maximiser la qualité des calculs sous contrainte d'un budget de coût du matériel.

Title: Contributions to the Scalability of Automatic Precision Tuning

Keywords: Word-Length Optimization, Fixed-Point Refinement, Approximate Computing

Abstract: Energy consumption is one of the major issues in computing today, shared by all domains of computer science, from high-performance computing to embedded systems. In recent years, approximation during computation has received renewed interest to improve energy efficiency. Many applications do not require high precision, thus hardware designers often trade-off the accuracy for cost reduction and speed-up. Various techniques for approximate computing augment the design space by providing another set of design knobs for performance-accuracy trade-off. This thesis focuses on developing methods for systematic exploration of this design space, including performance and accuracy modeling and design automation. We use fixed-point for data representation of signals and the results of their computations. We optimize the word length of each signal to get the good balance between the cost and the accuracy of the final design. This problem is called Word length Optimization (WLO) or automatic precision tuning. The thesis contributes to three research directions. First, a method is proposed to improve the scalability of WLO for large applications. To reduce exponential complexity in the nature of WLO, the input application is decomposed into smaller kernels, which are then solved independently using noise budgets to reduce the exploration time. To allocate noise budgets to each kernel, the main idea is to characterize the impact of approximating each kernel on accuracy and cost through simulation and regression to construct the empirical models. These models are then used to obtain the noise budgets. The second research direction is a hybrid algorithm combining Bayesian optimization (BO) and a fast local search to speed up the WLO procedure. An efficient mechanism is proposed to switch between the BO and the local search to obtain good designs in a short time. The last contribution opens a new research direction on resource-constrained WLO. State-of-the-art approaches mainly solve WLO given a quality (accuracy) constraint. In this study, a Bayesian optimization based algorithm was proposed to maximize the quality of computations constrained by a cost budget.