Online AMS Frontend Reconfiguration for Sensor Network Applications and Other Continuously Changing Environments

A Dissertation Presented

by

Pengbo Sun

to

The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2009

Abstract of the Dissertation

Online AMS Frontend Reconfiguration for Sensor Network Applications and Other Continuously Changing Environments

by

Pengbo Sun

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2009

This thesis proposes a novel online analog and mixed-signal (AMS) frontend reconfiguration approach for sensor network applications and other continuously changing environments. The approach is based on a design point (DP) selection algorithm. The algorithm has two steps: DP sampling and DP pruning. This thesis also proposes a systematic methodology for reconfigurable $\Delta\Sigma$ modulator topology designs.

The system software of traditional embedded system optimizes the allocation of a fixed set of resources under static conditions. However, the functionality and performance constraints of data processing systems are far less predictable since the characteristics of the monitored environment are continuously changing. In this thesis, the concept of developing AMS frontend design strategies for anticipative management of metadata acquisition, processing, and communication in dynamic environments is proposed. The idea is to develop mathematical models and small-overhead algorithms for online monitoring of performance

requirements and comprehensive adaptation of embedded architecture for metadata processing.

While reconfigurable digital systems are very popular and well understood in terms of their capabilities and limitations, reconfigurable analog and mixed-signal (AMS) systems are, in contrast, much less studied or employed in practical applications. This prevents the more comprehensive harvesting of the possible benefits of reconfigurable systems, as a majority of embedded applications (e.g., embedded control and telecommunications) include significant amounts of analog signal processing. To address this major limitation, research must not only address new reconfigurable AMS architectural concepts, but also study the related design methodologies and EDA tools. More specifically, it is essential to develop efficient techniques for designing reconfigurable analog to digital converters (ADC) due to the importance of ADCs in embedded systems.

Reconfigurable systems simultaneously offer the advantages of *(i)* high performance processing, provided by hardware, and *(ii)* flexibility in tackling different applications, provided by software. Reconfigurable systems are attractive implementation platforms for many embedded applications due to their capability of offering low development costs and short design times, while being accessible to less experienced designers.

With the development of wireless communication technology, many wireless communication standards emerged, such as AMPS, GSM, CDMA, WCDMA, UMTS etc. A wireless communication system which can efficiently support multiple communication standards led to the interest in developing a reconfigurable multi-mode receiver that can meet the bandwidth and resolution requirements of multiple standards. The most difficult part in implementing a reconfigurable multi-mode receiver is the design of a high performance, compact reconfigurable analog to digital convertor that will reconfigure the convertor topology when

communication standard changes.

In this thesis, a methodology for designing reconfigurable discrete-time $\Delta\Sigma$ modulator topologies is proposed. Optimized topologies are selected from the set of all possible topologies expressed by a generic topology, such that they *(i)* minimize the complexity of the topologies, *(ii)* maximize the topology robustness with respect to circuit nonidealities, and *(iii)* minimize the total power consumption. A case study of the design of topologies for a three-mode reconfigurable $\Delta\Sigma$ modulator is presented. A reconfigurable topology implementation on a Programmable System-on-Chip (PSoC) device is also included.

This thesis presents a systematic methodology for producing reconfigurable $\Delta\Sigma$ modulator topologies with optimized flexibility in meeting variable performance specifications. To increase their flexibility, topologies are optimized for performance attributes pertaining to ranges of values rather than being single values. Topologies are implemented on switched-capacitor reconfigurable mixed-signal architectures. Since the number of configurable blocks is very small, it is extremely important that the topologies use as few blocks as possible.

A case study illustrates the methodology for specifications from telecommunications area and a dynamic reconfiguration methodology of mixed-domain embedded systems for applications with variable performance requirements is introduced. A methodology for designing cost effective, dynamically reconfigurable, mixed-domain systems for metadata processing is proposed. During operation, the system switches between different design points for analog and digital blocks, depending on the actual performance needs. A case study for sound based tracking is discussed.

Sensor networks are increasingly important for many applications in environmental monitoring, manufacturing, defense and infrastructure monitoring. Many applications introduce variable performance requirements which demand online architecture reconfiguration, including the analog-digital frontends to sensors. In the last part, this thesis presents design automation methods for deciding the design points used for dynamic reconfiguration of analog-to-digital converters and DSP circuits.

Contents

List of Tables **x**

List of Figures **xi**

1 Introduction **1**

1.1 Motivation for this thesis and method overview 1

1.2 Goals and contributions 10

1.3 Thesis outline 11

2 Systematic Methodology for Reconfigurable Switched-Capacitor $\Delta\Sigma$ Modulator Topology Design **13**

2.1 Introduction 13

2.2 Related work 14

2.3 Overview of the methodology for reconfigurable $\Delta\Sigma$ modulator topology design 16

2.4 Case studies 21

2.4.1 Triple-mode reconfigurable $\Delta\Sigma$ modulator topology design 21

2.4.2 Implementation of a reconfigurable $\Delta\Sigma$ modulator using $PSoC^{TM}$ mixed-signal SoC 26

2.5 Conclusions 31

3 Flexibility-oriented Design Methodology for Reconfigurable ΔΣ Modulators 32

3.1 Introduction 32

3.2 Related work 34

3.3 Reconfigurable Mixed-Signal Architecture 35

3.4 Reconfigurable Δ ADC Design Method 43

3.4.1 Description of flexible performance requirements 43

3.4.2 Optimal ΔΣ ADC topology design 46

3.4.3 Overall design flow 49

3.5 Case Study 51

3.6 Conclusions 55

4 Dynamic Reconfiguration of Mixed-Domain Embedded Systems for Applications with Variable Performance Requirements 56

4.1 Introduction 56

4.2 Problem Formulation 59

4.3 Modeling for Dynamic Reconfiguration 62

4.4 Case Study 71

4.5 Conclusions 73

5 Online AMS Frontend Reconfiguration for Sensor Network Applications 74

5.1 Introduction 74

5.2 Reconfigurable Mixed Analog-Digital Architecture 76

5.3 Synthesis of Reconfigurable AMS 82

5.4 Experiments 89

5.5 Conclusions 92

6 Conclusions **94**

6.1 Conclusions 94

Bibliography **97**

List of Tables

2.1 Design complexity 25

2.2 Values for the capacitor arrays. 28

5.1 DP selection for ΔΣ modulators: Case 1 90

5.2 DP selection for ΔΣ modulators: Case 2 91

List of Figures

1.1 Sound Localization and Tracking System 3

1.2 Camera-based target recognition and tracking [28, 30, 32] 4

2.1 Third order generic reconfigurable $\Delta\Sigma$ modulator topology 17

2.2 Nonideal blocks in *Simulink* 21

2.3 Reconfigurable modulator topology *opt2* 23

2.4 SNR degradation by circuit nonidealities for topology *opt2* (mode 2) 24

2.5 Analog subsystem of $PSoC^{TM}$ [40] 26

2.6 Dual-mode second-order modulator (a) topology and (b) $PSoC^{TM}$ implementation 27

2.7 SNR comparison of the dual-mode second-order modulator 28

2.8 Measurement result for $PSoC^{TM}$ based implementation. 29

3.1 Type C reconfigurable SC block in the $PSoC^{TM}$ architecture [40] 36

3.2 Type D reconfigurable SC block in the $PSoC^{TM}$ architecture [40] 37

3.3 Reconfigurable SC block based on the $PSoC^{TM}$ architecture [40] 39

3.4 Third order $\Delta\Sigma$ modulator topology 40

3.5 Implementation of Figure 3.4 topology on current $PSoC^{TM}$ SC architecture 41

3.6 Implementation of Figure 3.4 on proposed SC architecture 42

3.7 Performance requirement region (PRR) 43

3.8 Generic 3^{rd} order $\Delta\Sigma$ topology 46

3.9 Overall design methodology 51

3.10 Influence of discretization on DR 52

3.11 Pareto curves for different orders 54

4.1 Mixed-domain architecture 60

4.2 Dynamic reconfiguration of mixed-domain systems 62

4.3 (a) *RBlock* and (b) *RBlock* execution semantics 64

4.4 Parameterized SFG for DFT 65

4.5 DP selection for ranges 66

4.6 Design point compatibility graph 69

4.7 DPs for DFT 72

5.1 Coarse-grained/fine-grained reconfigurable AMS architecture 77

5.2 Coarse grained reconfigurable digital block 80

5.3 Synthesis model for overall reconfiguration of AMS system 81

5.4 Synthesis model for overall reconfiguration of AMS system 83

5.5 Dynamic reconfiguration 84

5.6 Parameterized Signal Flow Graphs 85

5.7 Design point selection 87

5.8 Design points for classifier circuit 93

Acknowledgements

It would not have been possible for me to finish this doctoral thesis without the help and support of the kind people around me, to only some of whom it is my great honor to give particular mention here.

First, and above all, I would like to acknowledge my Ph.D. supervisor, Dr. Alex Doboli, for his help, advice and patience all the time throughout my PhD years. It is difficult to overstate my gratitude to him. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make research fun for me. He provided encouragement, sound advice, good teaching, good company, and lots of good ideas. I would have been lost without him.

I also want to thank Dr. Edward H. Currie, Dave Van Ess, and Cypress Semiconductor Corporation for providing invaluable help, support and resources, especially for initiating the topic on reconfigurable mixed-signal embedded system architectures and dynamic reconfiguration.

Thanks to my committee members, Dr. Sangjin Hong, Dr. Alex Doboli, Dr. Monica Fernandez-Bugallo, and Dr. Edward H. Currie, who offered guidance, support and gave suggestions on my preliminary proposal.

I am indebted to my colleagues in VLSI Systems Design Laboratory for providing a stimulating and fun environment to learn and grow. I am especially grateful to Meng Wang, Cristian Ferent, Varun Subramanian, Michael Gilberti, Anurag Umbarkar, Yang Zhao, Sankalp Kallakuri, Ying Wei, Junling Zhou, Jing Gao, Hui Zhang and Hua Tang.

I would also like to express my gratitude to all the faculty members, staffs and the many individuals in the department who have made my stay at Stony Brook University a pleasant, unique and memorable one. Specially thanks to all my friends at Stony Brook University for giving me encouragement throughout.

And big thanks to all the people in my life who have kept me happy and positive with their companions. You all are important to me. I particularly thank those certain few, Bo qiu, Dan Yin, Bo Chen, Hao Huang, Bo Ning, Feng Gao, Hongying Yu, Jia Zuo, Lei Sun, Haibin Gong, Lulu Sun, Dongni Wang, Jiaohua Wang, Haojun Wang, Yang Song, Jing Gao, Yijun Hang, Shiyuan Zhang, Meng Wang, Xiaoyao Liang, Junling Zhou and Keyi Chen, who were supportive of me in difficult times. Your love and support, voice of reason, understanding ear and general good laughs, has certainly made for some great memories!

I wish to thank my extended family for their love, support and understanding all these years. My uncle Weiguo Liang, Weigang Liang and Weige Liang were particularly supportive. My dear grandparents, Yongchang Sun and Guiwen Yang gave me courage to began this journey, but sadly are not here to see me finish. I know you must be proud of me and I miss you two terribly.

Last, and most importantly, I wish to thank my parents, Yuancheng Sun and Yan Liang. They bore me, raised me, supported me, taught me, and loved me. Also thanks to my parents-in-law Songbai Liu and Ruiyuan Liu for their love and support. Specially thanks to my husband Feng Liu. He will always be my greatest love, my truest friend, and my closest companion. Thanks to my son, Vincent Hanjie Liu. His coming lights up my life. I could not have completed this thesis without them. To all of them I dedicate this thesis.

For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely my own.

Chapter 1

Introduction

This chapter explains the motivation, goals, and contributions of this thesis. It also gives an overview of the methodology, which introduces the essence of the methodology and illustrates how the methodology can be used for online AMS frontend reconfiguration. More precisely, this chapter (i) explains the advantages of reconfigurability and the design challenges; (ii) discusses the advantages of flexibility-oriented design methodology for reconfigurable $\Delta\Sigma$ modulators; (iii) gives an overview of the proposed algorithm for dynamic reconfiguration of mixed-domain embedded systems for applications with variable performance requirements; (iv) gives a brief introduction of the online AMS reconfiguration method for sensor network applications and other continuously changing environments.

1.1 Motivation for this thesis and method overview

Many applications have variable performance requirement that demand online architecture reconfiguration, from AMS frontends to sensors. For example, metadata processing and monitoring systems usually have continuously changing environments. Metadata (such as video, image, sound, olfactory and hyperspectral images) is the basis of supervisory and process control (SPC) systems. However, metadata handling is computationally cumbersome with current computing concepts because it has *(i)* expensive hardware, *(ii)* long computation time, *(iii)* large power consumption, *(iv)* low reliability, and *(v)* ineffective worst case design. The system software of traditional embedded systems optimizes the allocation of a fixed set of resources (including processing hardware, communication bandwidth and supply energy) under static conditions [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. However, the functionality and performance constraints of SPC systems are far less predictable since the characteristics of the monitored environment are continuously changing (including hard-to-predict situations). Hence, the urgent need for the development of middleware design strategies for anticipative management of metadata acquisition, processing, and communication in dynamic environments.

The system manages hardware usage and configuration through on-line learning and adaptation, when the environments change. There is currently some research in this area [12, 13, 14, 15].

SPC Systems in real environments should have the following features based on their characteristics:

1. *System functionality.* Applications including image sampling in intrusion detection, target recognition and tracking, temperature sampling and monitoring, hyperspectral imaging, etc., are all classes of applications addressed by the conceptual motivation of this thesis. In this section, sound localization and tracking are discussed as an illustrative example. In the sound localization and tracking system shown in Figure 1.1, sounds are periodically sampled using a microphone array. Subsequently, they are locally processed to localize the position and identify the moving speed (if the

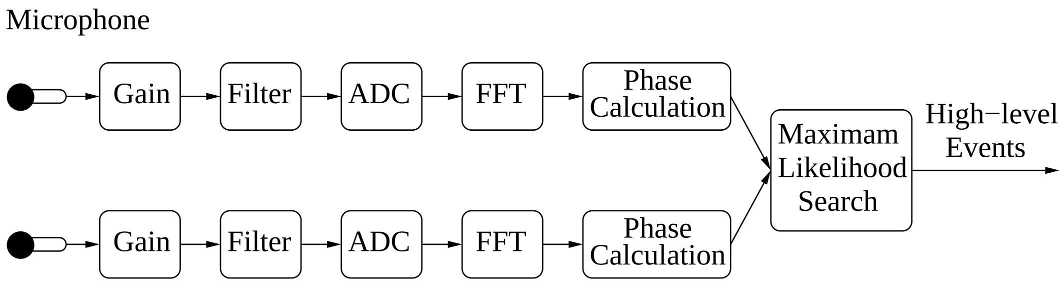


Figure 1.1: Sound Localization and Tracking System

target is moving). Then, compressed information about the objects is sent via a wireless network to other nodes or to the server.

2. *Dynamics of performance requirements.* SPC systems operate in dynamic environments. For example, the sound localization and tracking system might operate under a noisy or noiseless environment, the target sound could be either moving or still, and the quality of sound may require different processing precision, such as music versus a human voice. An efficient metadata processing system should be able to react to different environmental configurations and adapt accordingly to save hardware resources and energy.

3. *Customizable embedded architectures.* Embedded architectures offer multiple possibilities for customizing the hardware architecture to adapt to specific applications and environments, while offering the opportunity for effective metadata processing. Architecture customization [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27] is achievable through dynamic allocation of registers, dynamic voltage scaling or architecture modifications and so on.

The novelty of the proposed methodology includes the following.

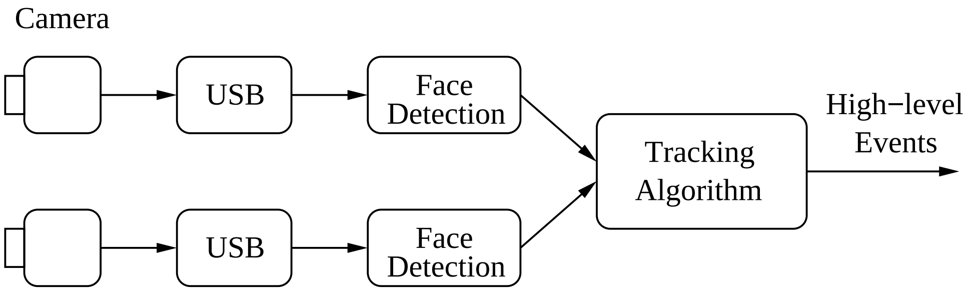


Figure 1.2: Camera-based target recognition and tracking [28, 30, 32]

- The changing semantics of data becomes important in optimized embedded processing. Current approaches only optimize the algorithm, or assume static requirements.

- The methodology requires an address system design of complete multi-domain processing (analog-digital-software). Existing methods cannot handle the complete chain.

- For the anticipative processing part, new design methods must be proposed because existing embedded processing is either reactive or data driven.

The proposed methodology can be used in various applications. One category is customization based on semantics, such as image sampling in intrusion detection systems, as shown in Figure 1.2, where the sampling scheme should be different, before and after event detection. Another category is customization in redundant systems. For example, in the sound localization application discussed earlier, the number of microphones used in tracking might be determined by the characteristics of the moving object. Among these, flexibility-oriented reconfigurable $\Delta\Sigma$ modulators design has proven to be a good application for this research topic.

With the rapid development in wireless communication technologies, a demand for reconfigurable receivers-on-a-chip that support multiple communication standards has emerged.

A major difficulty for multi-standard receiver implementation is the design of high performance, compact multi-mode analog to digital converters (ADC) [37].

The extant literature on reconfigurable multimode ADCs [37, 36, 35] is some what limited. So far, the design of reconfigurable, multimode, $\Delta\Sigma$ modulator relies mostly on manual techniques and relatively unsystematic. There is little understanding of how to develop optimal reconfigurable topologies for a new set of specifications. Also, to improve design closure and reduce cost, topology design must contemplate a fair amount of circuit nonidealities. Efficient methodologies are needed for systematically designing reconfigurable ADCs, while considering nonidealities at early stage in the design flow.

Current systematic design methodologies are for single-mode $\Delta\Sigma$ modulators only. Most of the work has been focused on on topology parameter optimization with limited modification of the modulator topology. Medico *et al.* [43] propose a design flow based on extensive analysis and modeling of the nonidealities that degrade the $\Delta\Sigma$ modulator performance. However, their methodology is limited to the four single-mode, discrete-time topologies supported by CAD tools – ADOPT and FRIDGE. Frances *et al.* [42] describe an approach for high-level simulation and synthesis of discrete-time $\Delta\Sigma$ modulators. Behavioral models for integrators, quantizer, and feedback digital-to-analog converter (DAC), are integrated in a synthesis tool (DAISY) that selects the modulator topology with the lowest power consumption for a given specification and constraints for the building blocks. An analytical integration method for the simulation of continuous-time $\Delta\Sigma$ modulators is suggested in [51]. Tang and Doboli [50] present a synthesis algorithm that finds the optimal topology for a given specification by solving a mixed-integer, nonlinear programming (MINLP) problem [53] with a cost function expressing the signal path complexity, sensitivity, and power consumption of the topology. This method considers only single-mode

$\Delta\Sigma$ modulators built out of ideal blocks.

This thesis proposes a systematic methodology for designing reconfigurable discrete-time $\Delta\Sigma$ modulator topologies optimized for minimum topology complexity, maximum robustness to circuit nonidealities, and minimum power consumption.

Reconfigurable systems attempt to simultaneously offer the two main advantages of hardware and software: *(i)* high performance processing, provided by hardware and *(ii)* flexibility in addressing different applications, provided by software. Reconfigurable systems are attractive implementation platforms for many embedded applications due to ability to provide ow development costs and short design times, while being accessible to less experienced designers.

While reconfigurable digital systems are very popular and well understood in terms of their capabilities and limitations, reconfigurable analog and mixed-signal (AMS) systems are, in contrast, much less studied and/or employed in practical applications. This prevents the more comprehensive harvesting of the possible benefits of reconfigurable systems, since the majority of embedded applications (e.g., embedded control and telecommunications) include significant amounts of analog signal processing. To address this major limitation, research must not only develop new, reconfigurable AMS architectural concepts, but also study the related design methodologies and EDA tools. More specifically, it is essential to develop efficient techniques for designing reconfigurable analog to digital converters (ADC) due to their importance in embedded systems.

Several general-purpose reconfigurable AMS architectures are mentioned in the literature [33, 34]. Continuous-time and switched-capacitor, reconfigurable ADCs have been presented in [35, 36, 37], but no design methodology, or EDA tools were considered. More recently the PSoC reconfigurable mixed-signal array has been offered by Cypress

Inc. [38, 40] as a cost-effective solution to embedded system implementation. While several ADCs have been incorporated into PSoC [41], there are no design methodologies or EDA tools that would allow effortless and rapid design of new ADCs. Filter synthesis methods for reconfigurable analog systems are presented in [33, 34]. Various techniques for single-mode (non-reconfigurable) $\Delta\Sigma$ ADC design have been described in [42, 43, 44]. A systematic design flow for continuous-time reconfigurable $\Delta\Sigma$ ADCs has been recently proposed in [45]. However, the produced modulator topologies are restricted to a set of predefined performance specifications (e.g., GSM, CDMA, and UMTS communication standards). The topologies have no flexibility in addressing new performance specifications. If another specification needs to be also addressed, then the entire process has to be repeated. This is an important limitation because flexibility ought to be one of the main strengths of reconfigurable AMS architectures.

This thesis proposes a systematic design methodology for creating flexible reconfigurable $\Delta\Sigma$ modulator topologies implemented on switched-capacitor (SC) reconfigurable AMS architectures. This work is based on an AMS architecture similar to that of PSoC's reconfigurable architecture. Since the number of programmable analog blocks is very limited in PSoC, it is extremely important that the modulator topologies use as few blocks as possible. In contrast to the existing work, the proposed methodology generates a set of topologies that are optimized to meet performance attributes pertaining to ranges of values, rather than being singular values.

Modern applications in environmental monitoring, security, robotics, military, and industrial control are based on (i) acquisition of high-level data (metadata) (video images, sound, olfactory and hyperspectral images), (ii) processing, storing, and communication of metadata, and (iii) intelligent control using higher-order semantics extracted from metadata [28, 29, 31, 39]. Systems continuously collect metadata to discover, and plan, new activities of interest, or to observe patterns of monitored situations. Then, this information is used to provide self-optimizing responses, including faster and more intelligent anticipation and reaction to critical events, superior utilization of hardware, and improved survival in harsh situations.

The performance constraints of these embedded systems are being dynamically modified, since operating conditions change continuously. Also, the amount of available hardware resources for a task depends on the computational demands of the other tasks, since all tasks share the same resources. The dynamics of processing and performance needs does not follow any particular mathematical rule for a mixture of different types of behavior, e.g., quasi-static, probabilistic, and performance constrained operation [28]. Hence, the system must continuously "comprehend" new processing demands, and accordingly customize its architecture and the dispatching of hardware resources to tasks. A naive approach would aggressively utilize high performance resources that are optimized for the worst-possible situation. However, this results in very expensive systems, that are likely to be unattractive in real life. Moreover, high performance processing is very energy hungry, hence unsuitable for mobile, autonomous systems.

Reconfigurable systems are attractive implementation platforms for many embedded applications due to their ability to provide low development costs and short design times, while being accessible to less experienced designers. To address this limitation, research must not only focus on new reconfigurable mixed-domain architectures, but also study the related design methodologies and EDA tools. More specifically, it is essential to develop efficient techniques for designing dynamically reconfigurable systems, which include both analog and digital functional blocks.

Techniques for dynamic reconfiguration of software or digital hardware are discussed in [54, 57, 58, 56]. Several reconfigurable mixed-domain architectures are mentioned in the literature [33, 38, 34]. More recently the PSoC reconfigurable mixed-signal array has become available from Cypress Inc. [40] as a cost-effective solution for embedded system implementation. Filter synthesis for reconfigurable analog systems is presented in [33, 34]. A systematic design flow for continuous-time reconfigurable $\Delta\Sigma$ ADCs has been recently proposed in [45]. However, the topologies produced have no flexibility in addressing new performance specifications. Also, there is no methodology available that takes into account the synergy between analog and digital blocks for producing more cost effective, dynamically reconfigurable designs.

This thesis proposes a methodology for designing cost-effective, dynamically reconfigurable mixed-domain implementations for metadata processing applications. To provide flexibility, performance requirements are specified as continuous ranges, or as discrete sets of values. During operation, the system switches between different design points (DPs) for its (reconfigurable) analog and digital blocks, depending on the actual performance constraints, so that a minimum amount of hardware is required. The proposed methodology includes two parts: (i) DP selection and (ii) system adaptation. This thesis presents two DP selection algorithms, one for performance requirements, expressed as continuous ranges (e.g., bandwidth, latency, range), and one for performance, defined as discrete sets (such as precisions, dynamic range). System adaptation is achieved by employing a centralized procedure that selects the most effective set of DPs for new requirements, so that all DPs for coupled blocks can operate together (they are compatible). A case study for sound based tracking is discussed.

1.2 Goals and contributions

The goals and contributions of this thesis are to develop:

- A methodology with continuous adaptation of the system capabilities depending on the changing semantics of the interaction with the environment especially in understanding the nature of the change (sometime in advance) while guaranteeing efficient operation in the new conditions.

- A systematic methodology for generating optimal topologies for discrete-time $\Delta\Sigma$ modulator. The generated topology is optimized for minimum topology complexity, maximum robustness to circuit nonidealities, and minimum power consumption.

- A flexibility-oriented design methodology for reconfigurable $\Delta\Sigma$ modulators. This work considered an AMS architecture based closely on the PSoC reconfigurable architecture. Since the number of PSoC programmable analog blocks is very limited , it is extremely important that the modulator topologies use as few blocks as possible. In contrast to the existing work, the proposed methodology generates a set of topologies that are optimized to meet performance attributes pertaining to ranges of values rather than being singular values.

- A methodology for designing cost-effective, dynamically reconfigurable mixed-domain implementations for metadata processing applications. System adaptation is realized as a centralized procedure that selects the most effective set of DPs for new requirements, so that all DPs for coupled blocks can operate together (they are compatible). A case study for sound-based tracking is discussed.

1.3 Thesis outline

This thesis is organized as follows:

- Chapter 2 presents a systematic methodology for designing reconfigurable discrete-time $\Delta\Sigma$ modulator topologies [61]. Optimized topologies are selected from the set of all possible topologies expressed by a generic topology, such that they minimize the complexity of the topologies, maximize the topology robustness with respect to circuit nonidealities, and minimize the total power consumption. A case study for designing topologies for a three-mode reconfigurable $\Delta\Sigma$ modulator is presented in Chapter 2. This chapter also offers a reconfigurable topology implementation for a Programmable System-on-Chip (PSoC) device.

- Chapter 3 presents a systematic methodology for producing reconfigurable $\Delta\Sigma$ modulator topologies with optimized flexibility in meeting variable performance specifications [62]. To increase their flexibility, topologies are optimized for performance attributes pertaining to ranges of values rather than being single values. Topologies are implemented on switched-capacitor reconfigurable mixed-signal architectures. Since configurable blocks are very valuable resources, it is extremely important that the topologies use as few blocks as possible, according to design specifications, rather than just worst case design. A case study illustrates the methodology for specifications from a telecommunications area.

- Chapter 4 proposes a methodology for designing cost-effective, dynamically reconfigurable, mixed-domain implementations for metadata processing applications [63].

The proposed methodology includes two parts: (i) DP selection and (ii) system adaptation. The chapter presents two DP selection algorithms, one for performance requirements expressed as continuous ranges (e.g., bandwidth, latency, range), and one for performance defined as discrete sets (such as precisions, dynamic range). A case study for sound-based tracking is discussed.

- Chapter 5 finalizes the DP selection and system adaptation idea proposed in chapter 4 into a complete algorithm [64]. This algorithm has two steps for finding the design points used in dynamic reconfiguration: (i) DP sampling and (ii) DP pruning. Experimental results are offered and discussed.

- The final chapter presents the conclusions.

Chapter 2

Systematic Methodology for Reconfigurable Switched-Capacitor $\Delta\Sigma$ Modulator Topology Design

2.1 Introduction

In this chapter, a methodology for designing reconfigurable, discrete-time, $\Delta\Sigma$ modulator topologies is proposed. Optimized topologies are selected from the set of all possible topologies expressed by a generic topology, such that they minimize the complexity of the topologies, maximize the topology robustness with respect to circuit nonidealities, and minimize the total power consumption.

This chapter is organized as follows. Section 2.2 summarizes related work. Section 2.3 presents an overview of the proposed ADC topology design methodology. Section 2.4

offers the case studies. Section 2.4.1 gives an design example on a triple-mode reconfigurable $\Delta\Sigma$ modulator topology showing the overall design flow when using it in a specific application. Section 2.4.2 is the implementation of a reconfigurable $\Delta\Sigma$ modulator using a $PSoC^{TM}$ mixed-signal SoC. Finally, conclusions are given in section 2.5.

2.2 Related work

There has not been much work on reconfigurable multimode ADCs [37, 36, 35]. So far, the design of reconfigurable multimode $\Delta\Sigma$ modulators is manual and relatively unsystematic. There is little understanding of how to develop optimal reconfigurable topologies for a new set of specifications. Also, to improve design closure and reduce costs, topology designs must take into account a fair number of circuit nonidealities. Efficient methodologies are needed for systematically designing reconfigurable ADCs while considering nonidealities early in the design flow.

Current systematic design methodologies are for single-mode, $\Delta\Sigma$ modulators only. Most of the work is on topology parameter optimization with limited modification of the modulator topology. Medico *et al.* [43] propose a design flow based on extensive analysis and modeling of the nonidealities that degrade the $\Delta\Sigma$ modulator performance. However, their methodology is limited to the four single-mode, discrete-time topologies supported by CAD tools – ADOPT and FRIDGE. Frances *et al.* [42] describe an approach for high-level simulation and synthesis of discrete-time $\Delta\Sigma$ modulators. Behavioral models for integrators, quantizer, and a feedback digital-to-analog converter (DAC), are integrated in a synthesis tool (DAISY) that selects the modulator topology with the lowest power consumption for a given specification and set of constraints for the building blocks. An analytical integration method for the simulation of continuous-time $\Delta\Sigma$ modulators is suggested in [51].

Tang and Doboli [50] present a synthesis algorithm that finds the optimal topology for a given specification by solving a mixed-integer, nonlinear programming (MINLP) problem [53] with a cost function expressing the signal path complexity, sensitivity, and power consumption of the topology. This method considers only single-mode $\Delta\Sigma$ modulators utilizing ideal blocks.

This chapter proposes a systematic methodology for designing reconfigurable discrete-time $\Delta\Sigma$ modulator topologies optimized for minimum topology complexity, maximum robustness to circuit nonidealities, and minimum power consumption. The methodology is based on the concept of generic topology that expresses all possible signal paths in a reconfigurable topology. The modeling of generic topologies is presented, including following nonidealities: integrator leakage and gain error, circuit noise, and circuit nonlinearity. Models are used in the methodology to provide a set of mixed integer-nonlinear (MINLP) equation set. Equations are solved for finding optimized reconfigurable $\Delta\Sigma$ topologies. The resulting topologies are then refined using a *Simulink* simulation of the models with more detailed nonidealities. The design methodology is presented in Section 2.3. This chapter discusses, in Section 2.4, a case study of a three-mode. reconfigurable, $\Delta\Sigma$ modulator design, and the implementation of a second-order reconfigurable topology using a $PSoC^{TM}$ mixed-signal SoC [40].

2.3 Overview of the methodology for reconfigurable ΔΣ modulator topology design

The proposed design methodology consists of two steps: (1) topology synthesis by mixed-integer nonlinear equation (MINLP) solving, and (2) topology refinement (post-optimization) by topology parameter optimization, while considering a more detailed modeling of circuit nonidealities.

In this section, a dual-mode, third-order, reconfigurable ΔΣ modulator is used as an example to present the design methodology. However, the methodology is applicable for single-loop reconfigurable ΔΣ modulators of any order.

The input to the topology design methodology is the specifications for the dual-mode reconfigurable ΔΣ modulator. For each target specifications, such as DR, a set of solutions that can achieve the target DR are found by using analytical expressions, such as the expressions embedded in the ΔΣ toolbox [52]. Each solution has four parameters: *(i)* the order of the loop filter, *(ii)* the oversampling ratio (OSR), *(iii)* the internal quantizer bits, and *(iv)* the noise transfer function (NTF) type, either Butterworth or Inverse Chebyshev. Any solution for a single mode can be combined with the solution for another mode to build a dual-mode ADC. Hence, the set of possible candidates for the dual-mode reconfigurable ADC is generated. Each candidate can be implemented using different modulator topologies. Optimal modulator topologies are generated by the methodology proposed next.

Optimal, reconfigurable, ΔΣ modulator topologies are designed for a set of specifications starting from a *generic modulator topology*. The generic topology contains all possible signal paths in a reconfigurable modulator. The generic topology for a third order modulator is shown in Figure 2.1. Similar topologies can be devised for modulators of

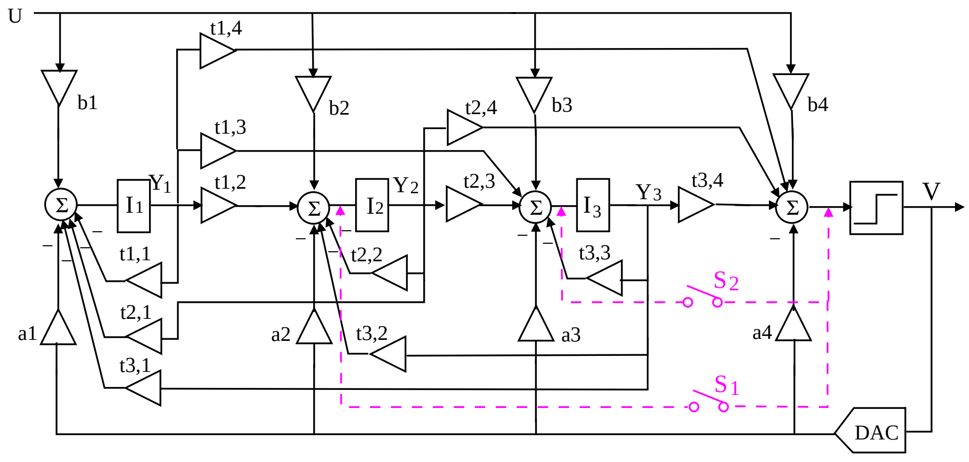


Figure 2.1: Third order generic reconfigurable ΔΣ modulator topology

higher order too.

The state-space description of the topology is expressed as follows:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \left(\begin{bmatrix} -t_{1,1} & -t_{2,1} & -t_{3,1} \\ t_{1,2} & -t_{2,2} & -t_{3,2} \\ t_{1,3} & t_{2,3} & -t_{3,3} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} + \begin{bmatrix} b_1 & -a_1 \\ b_2 & -a_2 \\ b_3 & -a_3 \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} \right) \cdot \begin{bmatrix} I_1 & & \\ & I_2 & \\ & & I_3 \end{bmatrix} + \begin{bmatrix} N_{th1} \\ N_{th2} \\ N_{th3} \end{bmatrix}$$

$$V = \begin{bmatrix} t_{1,4} & t_{2,4} & t_{3,4} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} + \begin{bmatrix} b_4 & -a_4 \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} + E_q \tag{2.1}$$

I_1, I_2, and I_3 are nonideal integrator transfer functions, which are of two possible types: delayed or delayless. E_q denotes the quantization noise. N_{th1}, N_{th2}, and N_{th3} denote the circuit noise in each integrating stage.

By solving the state-space equations, the transfer function of the modulator is written as the equation $V = L_0U + L_1V + E_q$ [47]. Variables L_0 and L_1 are loop filters, which can be analytically derived from the state-space equations.

MINLP formulation. For a cost function f (expressing the desired design goals), the problem of designing an optimized reconfigurable topology is formulated as:

$$\textit{minimize cost } f = f(x_{m,i}, wx_{m,i});$$

$$\textit{subject to: } g(x_{m,i}) = 0;$$

$$\textit{subject to: } x_{m,i} \geq 0,\ wx_{m,i} \in \{0,1\};$$

Variable $x_{m,i}$ denotes any of the unknown coefficients a_i, b_i, and $t_{i,j}$ for the m^{th} mode. Unknown $wx_{m,i}$ represents whether the signal path with coefficient $x_{m,i}$ is present in the optimal topology, or not. g are the 24 equality constrains obtained from equating symbolic expressions for L_0 and L_1 to the desired loop filter functions for all the modes [50].

The cost function for optimal reconfigurable modulator topology design is:

$$f = \alpha_1 \sum_m \sum_i wx_{m,i} + \alpha_2 \sum_{m_1} \sum_{m_2 \neq m_1} \sum_{i,j} |wx_{m_1,i} - wx_{m_2,j}|$$

$$+ \sum_m \beta_m (\sum_k \gamma_{m,k} \cdot h_{m,k}(x_{m,i})) + \gamma \sum_m \sum_i p_{m,i} \qquad (2.2)$$

The first two terms in the cost function represent the complexity of the modulator topology. The first term is the absolute complexity, considering the total number of signal paths in the topology. The second term is the relative complexity that accounts for the changes of the signal paths when the modulator is reconfigured. The third cost function term minimizes the sensitivity of the modulator with respect to circuit nonidealities. The term is the weighted sum of the impact of nonidealities on the modulator NTF. $h_{m,k}$, a function of $x_{m,i}$, minimizes the impact of the k^{th} nonideality on NTF for the m^{th} mode, which captures the following nonidealities.

Integrator leakage and gain error: for a switched-capacitor integrator, the integrator leakage and gain error are modeled by including the finite gain (A_i) of the OpAmp and capacitor mismatch factor (ε_i) in the integrator transfer function I_i, respectively, as shown below.

$$I_i = g_i(1-\varepsilon_i)\frac{(1,z)}{z-(1-g_i/A_i)}, \quad i = 1,2,3 \tag{2.3}$$

Where, g_i is the integrator gain, $(1,z)$ denotes the two types of integrators: delayed or delayless. Notice that the NTF depends only on the loop filter L_1. Therefore, minimizing the variation of the NTF is equivalent to minimizing L_1. If the integrator transfer function is replaced by equation (2.3), then L_1 becomes like equation (2.4).

$$\widetilde{L_1} = \frac{L_{1,num} + \sum_{i=1}^{3}\mu_i\left(\sum_{j=0}^{2} c_{\mu_i,j}z^j\right) + \sum_{i=1}^{3}\varepsilon_i\left(\sum_{j=0}^{2} c_{\varepsilon_i,j}z^j\right)}{L_{1,den} + \sum_{i=1}^{3}\mu_i\left(\sum_{j=0}^{2} d_{\mu_i,j}z^j\right) + \sum_{i=1}^{3}\varepsilon_i\left(\sum_{j=0}^{2} d_{\varepsilon_i,j}z^j\right)} \tag{2.4}$$

Variable $L_{1,den}$ and $L_{1,num}$ denotes the denominator and numerator of the ideal loop filter function L_1, respectively. $c_{\mu_i,j}$, $d_{\mu_i,j}$, $c_{\varepsilon_i,j}$, $d_{\varepsilon_i,j}$ are the symbolic coefficients as functions of signal path coefficients. To minimize the difference between $\widetilde{L_1}$ and L_1, the following function $h_{m,k}$ was added to the overall cost function (2.2) used in topology design:

$$h_{m,k} = \sum_{i=1}^{3}\sum_{j=0}^{2}\left(\left|c_{\mu_i,j}\right| + \left|d_{\mu_i,j}\right| + \left|c_{\varepsilon_i,j}\right| + \left|c_{\varepsilon_i,j}\right|\right) \tag{2.5}$$

Circuit noise and nonlinearity: Considering the noise of all integrating stages, the transfer function of the modulator was expressed as equation (2.6). And, equation (2.7) captures the circuit noise in the cost function.

$$\tilde{V} = L_0 U + L_1 \tilde{V} + \sum_{i=1}^{3} \left(\frac{\sum_{j=0}^{3} c_{th_i,j} z^j}{L_{1,den}} \times N_{th,i} \right) + E_q \tag{2.6}$$

$$h_{m,k} = \sum_{i=1}^{N} \sum_{j=0}^{N} |c_{th_i,j}| \tag{2.7}$$

In the frequency domain, nonlinearity can be modeled by calculating the frequency responses at different harmonics. For example, assuming that only the second order nonlinear component (k_2) of the first stage is considered, the frequency response of the modulator can be expressed as:

$$\widetilde{L_{1,o2}} = \frac{L_{1,o2,num} + k_2 \left(\sum_{j=0}^{3} c_{hd,j} z_0^j \right)}{L_{1,o2,den} + k_2 \left(\sum_{j=0}^{3} d_{hd,j} z_0^j \right)} \tag{2.8}$$

The corresponding term in the cost function (2.2) is:

$$h_{m,k} = \sum_{j=0}^{3} \left(|c_{hd,j}| + |d_{hd,j}| \right) \tag{2.9}$$

The forth cost function term minimizes the power consumption and takes into account the static and dynamic power consumption of the switched-capacitor loop filter, which has the same form as that in [50].

By formulating the reconfigurable modulator topology synthesis problem as a MINLP problem, and including the above four nonidealities into the cost function, a set of optimized reconfigurable $\Delta\Sigma$ modulator topologies is generated.

Reconfigurable topology refinement. In this step, the reconfigurable modulator topology is tuned (post-optimized), based on a more detailed circuit model simulation. Only local optimizations are conducted in this step. The topology parameters are adjusted, but signal paths are left unchanged. The *Simulink* models are used in this step for a more accurate capturing of nonidealities as shown in Figure 2.2. The coefficients of the signal paths are tuned, such that the SNR degradation because of circuit nonidealities is minimized. A simulated annealing algorithm was used for fine tuning the modulator coefficients. A 5%

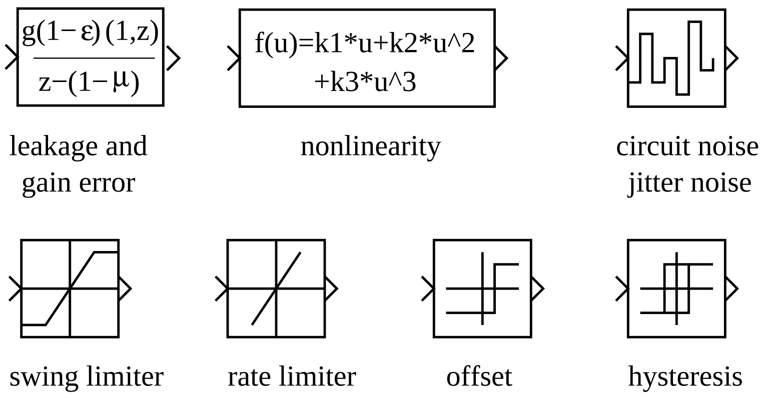


Figure 2.2: Nonideal blocks in *Simulink*

variation of the coefficient was allowed during tuning. By fine tuning, the solutions generated through MINLP solving are locally optimized according to the more detailed models. Hence, the two steps – topology synthesis by MINLP solving and topology refinement, optimize the reconfigurable topologies, both globally and locally. The resulting topologies are optimized in terms of the topology complexity and robustness with respect to circuit nonidealities, as discussed before. Design requirements for each building block are also derived, such that the performance degradation of the modulator is less than $3dB$.

2.4 Case studies

2.4.1 Triple-mode reconfigurable ΔΣ modulator topology design

The proposed design methodology were used to generate optimized reconfigurable ΔΣ modulator topologies that satisfy multiple specifications, and have reduced system complexity. The modulator operates in three modes that correspond to UMTS, CDMA2000, and GSM communication standards. The DR requirement for UMTS, CDMA2000, and

GSM are 11.5-bit, 13-bit, and 15-bit with the bandwidth of $1.92mHz$, $615kHz$, and $190kHz$, respectively [37].

One of the optimized topologies is shown in Figure 2.3. Table 2.1 presents the comparison of the two optimized topologies obtained using the proposed methodology, and a non-reconfigurable topology that includes three single-mode modulators. The comparison is in terms of the topology complexity, reduction in design effort (η_d) and power consumption (η_p). Topology *opt1* has orders $< 4,4,4 >$ for the three modes. Topologies *opt2* has orders $< 4,3,3 >$ for the three modes. The complexity of the system is analyzed with respect to the number of signal paths (N_p), the number of non-reconfigurable cells ($N_{p,r}$), and the number of reconfigurable cells ($N_{c,r}$).

Table 2.1 shows that reconfigurable ADCs are much more compact than the modulator with three single-mode architectures. For example, there are 33 signal paths in the non-reconfigurable topology, while there are only 25 and 20 signal paths in the generated topologies *opt1* and *opt2*, respectively. Fewer signal paths not only decreases the complexity of the circuit implementation, but also decreases the overall power consumption and chip area. Design effort is estimated from the number of signal paths, and reconfigurable cells.

The power consumption reduction (η_p) is estimated from the sum of the power consumption of the active building blocks for each mode. The total number of active paths ($N_{p,a}$) for all three modes is shown in Table 2.1. The generated optimized topologies present up to a 24.2% improvement in power consumption compared to the non-reconfigurable topology.

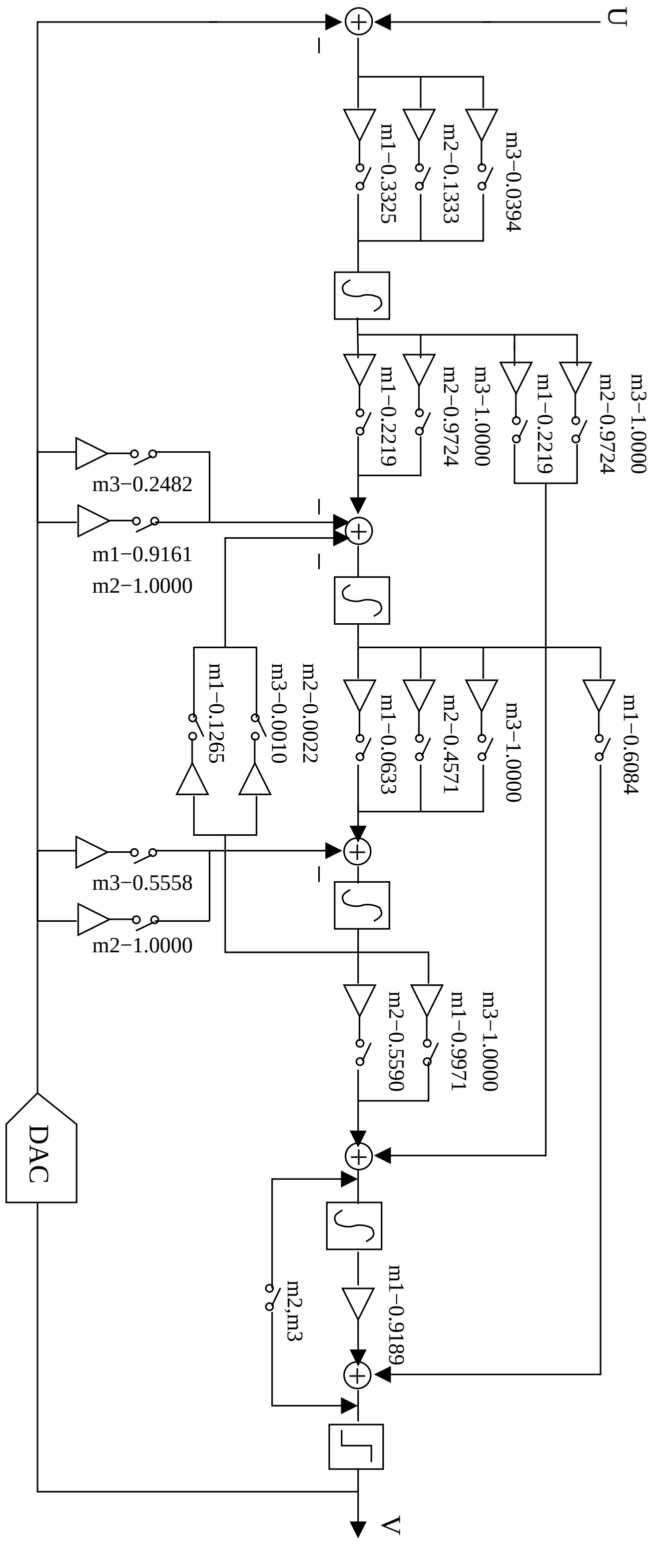
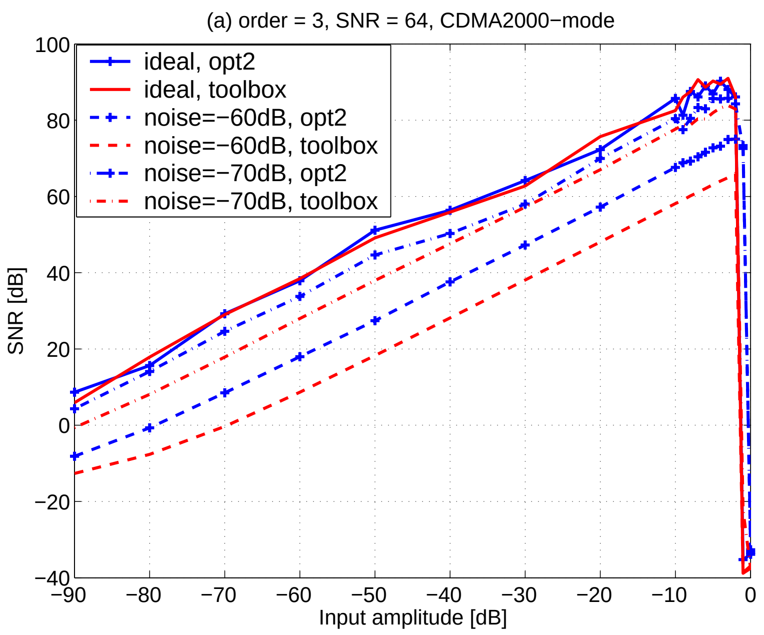


Figure 2.3: Reconfigurable modulator topology *opt2*



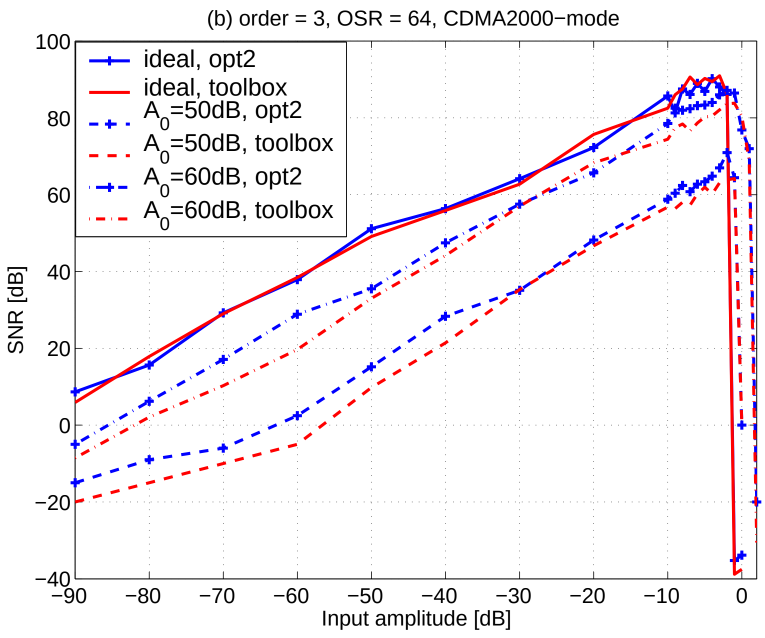


Figure 2.4: SNR degradation by circuit nonidealities for topology *opt2* (mode 2)

Topology	N_p	$N_{p,r}$	$N_{c,r}$	$N_{p,a}$	η_d	η_p
non-reconfigurable	33	–	–	33	–	–
opt1 $< 4,4,4 >$	25	18	7	30	24.2%	9.1%
opt2 $< 4,3,3 >$	20	15	5	25	39.4%	24.2%

Table 2.1: Design complexity

In order to analyze the robustness of the produced modulator topologies, the second mode of topology textitopt2 was simulated. Figure 2.4 presents the performance comparison for mode 2 of topology *opt2* and the corresponding $\Delta\Sigma$ toolbox topology in the presence of circuit noise and integrator leakage. The generated topology performs better. If circuit noise is considered (Figure 2.4(a)), the improvements of peak SNR for mode 2 of topology *opt2* is $3dB$ and $10dB$ as compared to the toolbox topology for noise levels of $-70dB$ and $-60dB$, respectively. The higher the noise level, the more the improvement. Figure 2.4(b) shows the SNR comparison for integrator leakage. For example, if the OpAmp of the first integrator has finite gain of $50dB$ or $60dB$, the degradation of SNR because of integrator leakage for mode 2 of topology *opt2* is $3dB$ less than that of the toolbox topology, as shown by the dashed line and the dashed-dotted line with plus markers, respectively. Therefore, although in the ideal case the generated topologies behave the same as the topologies from the $\Delta\Sigma$ toolbox, the optimized topologies are more robust in the presence of circuit noise. The performance improvement shows that the third cost function term is crucial to the performance of the final reconfigurable topologies, as it captures the possible performance tradeoffs for the topologies with respect to circuit nonidealities.

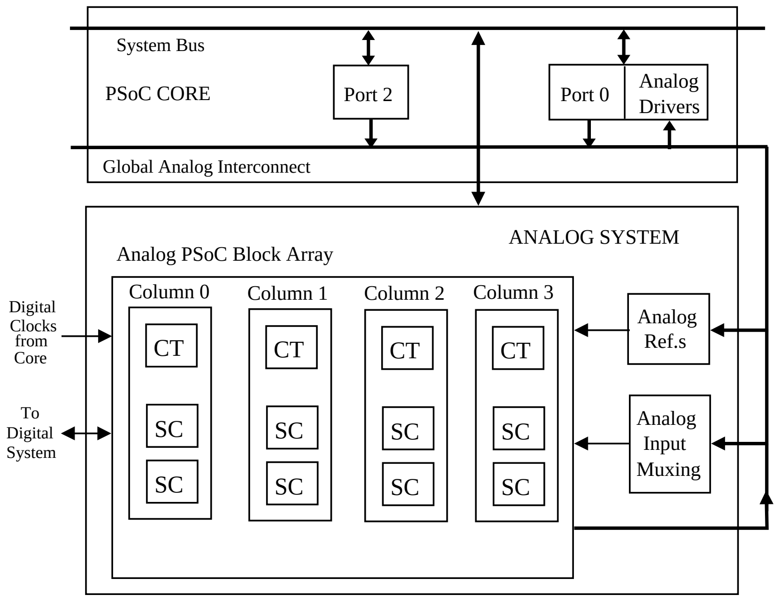


Figure 2.5: Analog subsystem of *PSoC*^{TM} [40]

2.4.2 Implementation of a reconfigurable $\Delta\Sigma$ modulator using *PSoC*^{TM} mixed-signal SoC

The *PSoC*^{TM} SoC family [40] consists of mixed-signal programmable arrays, and an on-chip microcontroller and memory devices. The top level analog architecture is shown in Figure 2.5. The analog *PSoC*^{TM} block array consists of four analog columns, each of which contains three programmable analog blocks: one continuous-time (CT) block and two switched-capacitor (SC) blocks. The analog SC blocks support $\Delta\Sigma$, successive approximation, and incremental ADC conversion, capacitor DACs, and SC filters. They have three input arrays (*ACAP*, *BCAP*, and *CCAP*) and one feedback array (*FCAP*) of binary-weighted switched-capacitors, allowing user programmability of the capacitor weights. This provides a summing capability consisting of two scaled inputs and a non-switched capacitor input. Allowable ranges for *ACAP*, *BCAP*, and *CCAP* are from $1C$ to $32C$, and for *FCAP* either $16C$ or $32C$, where C is the unit capacitance of the capacitor array [40].

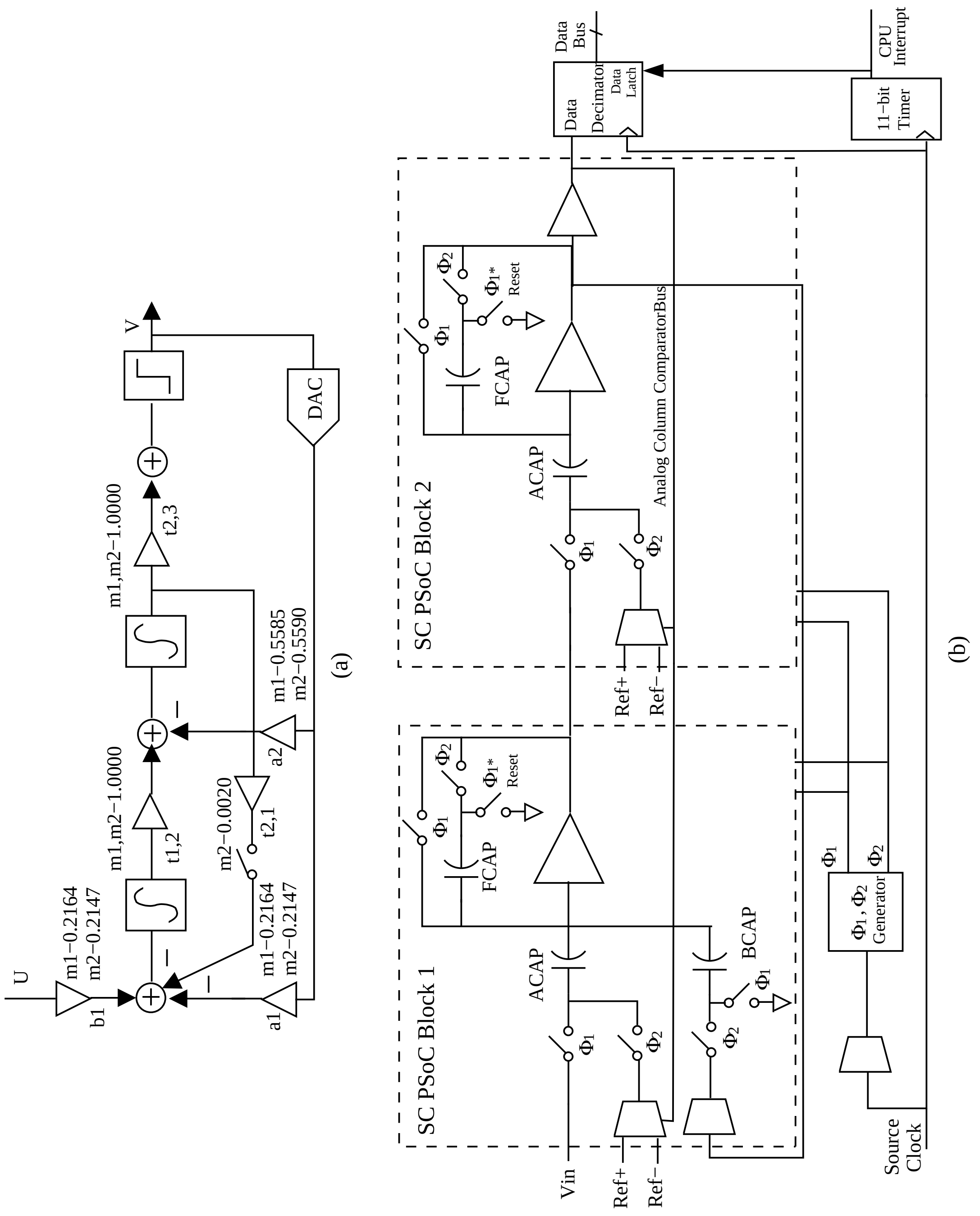


Figure 2.6: Dual-mode second-order modulator (a) topology and (b) *PSoC*^{TM} implementation

Mode	SC block 1			SC block 2		
	ACAP	BCAP	FCAP	ACAP	BCAP	FCAP
1	4C	–	16C	8C	–	16C
2	8C	1C	32C	16C	–	32C

Table 2.2: Values for the capacitor arrays.

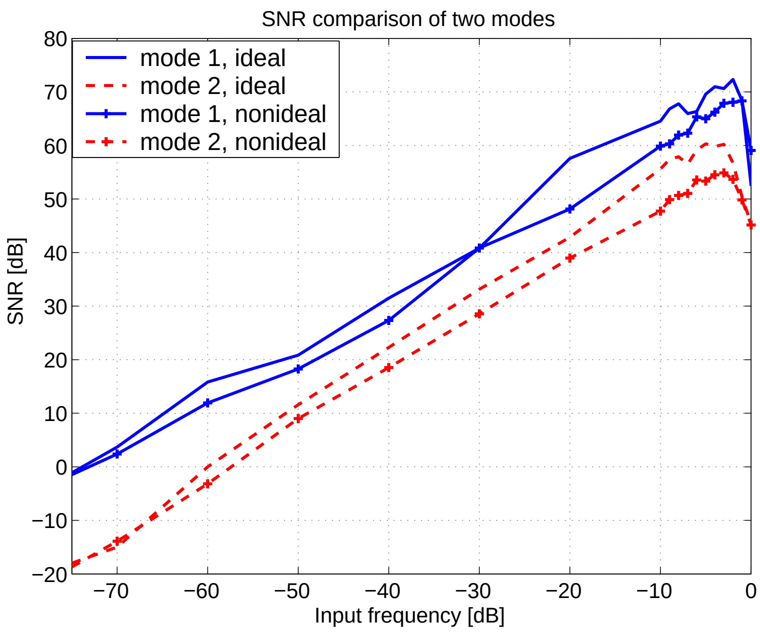
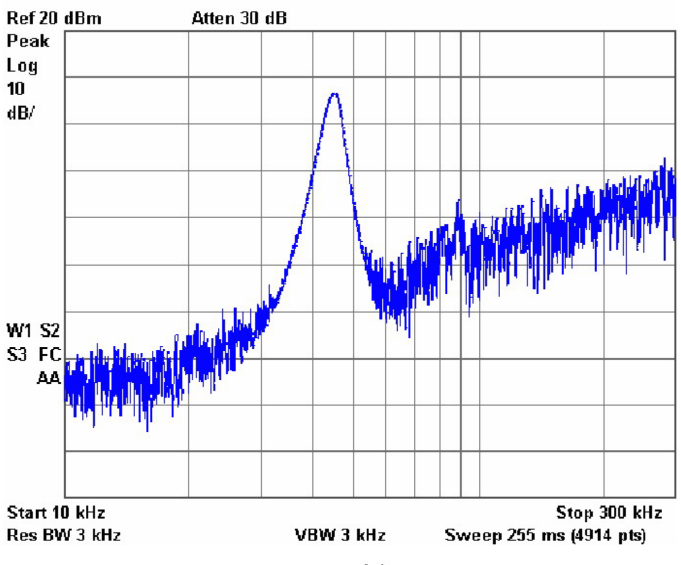
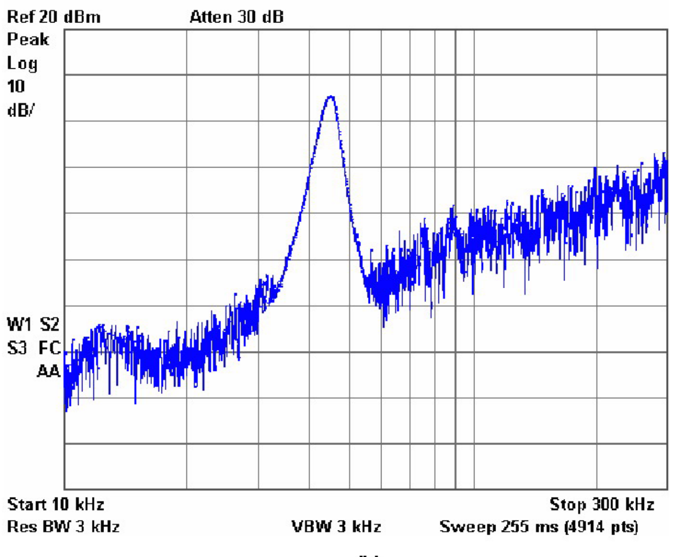


Figure 2.7: SNR comparison of the dual-mode second-order modulator



(a)



(b)

Figure 2.8: Measurement result for $PSoC^{TM}$ based implementation.

A second-order $\Delta\Sigma$ modulator can be implemented by using two SC blocks as integrators. A dual-mode second-order topology was designed with a target peak SNR for the two modes are 10 bits ($60dB$) and 8 bits ($48dB$), respectively. The two modes have five and six signal paths with OSR of 64 and 40, respectively. NTF types for the two modes are Butterworth, and Inverse Chebyshev, respectively. When the modulator is switched from one mode to another mode, four signal paths need to be reconfigured: signal path coefficients a_1, a_2, b_1 have to be modified, and one more signal path $t_{2,1}$ needs to be switched on for the second mode, as shown in Figure 2.6(a).

The implementation of the reconfigurable topology using a $PSoC^{TM}$ chip is shown in Figure 2.6(b). The coefficients (a_1, a_2, b_1) of the signal path can be modified by changing the capacitor arrays ($ACAP$ and $FCAP$). The additional signal path $t_{2,1}$, when the modulator is reconfigured, is implemented by using one more input array ($BCAP$) for the first SC block. The values of the capacitor arrays are shown in Table 2.2.

Figure 2.7 shows the simulation results achieved from Matlab. It illustrates that the peak SNR of the two reconfigurable modes equals $71dB$ and $60dB$ for the ideal modulator, respectively. If circuit nonidealities are considered, the peak SNR drops to $60dB$ and $54dB$, respectively, which can still meet the design specifications. Figure 2.8 shows the output spectrum from the measurement of the $PSoC^{TM}$ based implementation. The input is $2.1V$ peak-to-peak sinusoid wave with frequency of $45kHz$. For mode 1, the noise floor level is about $-55dB$, and the SNR is about $60dB$. For mode 2, the noise floor level is about $-47dB$, and the SNR is about $53dB$. Although the 2 curves for mode 1 have same values at $-30dB$, it has no special meaning than a coincidence that the nonideality doesn't affect the circuit's performance at this point. Also, the notch of the Inverse Chebyshev type noise shaping for the second mode is clearly shown in Figure 2.8(b). This experimental result shows that the implementation of the reconfigurable $\Delta\Sigma$ modulator meets the design specification.

2.5 Conclusions

This chapter has presented a systematic methodology for design of optimized topologies for reconfigurable single-loop switched-capacitor $\Delta\Sigma$ modulators. This methodology optimizes the topologies for minimum complexity, maximum robustness to performance degradation due to circuit nonidealities, and minimum power consumption. A case study on designing a three-mode reconfigurable $\Delta\Sigma$ modulator shows that the complexity and power saving of the generated reconfigurable modulators is about 40% and 24.2% of that of three single-mode modulators obtained with $\Delta\Sigma$ toolbox. Most importantly, the produced reconfigurable topologies are more robust to integrator leakage and gain error, circuit noise, and nonlinearity than the topologies from $\Delta\Sigma$ toolbox. The correctness of the methodology was also demonstrated by implementing a dual-mode reconfigurable topology using a $PSoC^{TM}$ reconfigurable mixed-signal SoC.

Chapter 3

Flexibility-oriented Design Methodology for Reconfigurable ΔΣ Modulators

3.1 Introduction

This chapter presents a systematic methodology for producing reconfigurable ΔΣ modulator topologies with optimized flexibility in meeting variable performance specifications. To increase their flexibility, topologies are optimized for performance attributes pertaining to ranges of values rather than being single values. Topologies are implemented on switched-capacitor, reconfigurable, mixed-signal architectures. Since configurable blocks are very valuable resources, it is extremely important that the topologies use as few blocks as possible to comply with the design specifications, rather than just worst case design. A case study illustrates the methodology for specifications from telecommunications area.

The design methodology first considers ΔΣ modulator topologies of lower order, e.g., order one or two, and identifies the maximum performance ranges that can be met with these topologies. Then, it analyzes topologies of increasingly higher order, while maximizing the performance ranges that are not already covered by the architectures of lower order. This strategy avoids unnecessarily utilizing more hardware resources for specifications that can be also met by simpler modulators. A two step process finds the maximum specification ranges that are covered by the topologies of a certain order. First, for decreasing oversampling ratios (OSRs), e.g., OSR = 128, 64, 32, 16), the signal and noise transfer functions of the modulators are computed using the Delta Sigma toolbox [46]. Then, the signal flow of the modulator topologies is computed by solving a set of mixed-integer nonlinear (MINLP) equations that are based on the computed transfer functions. The cost function minimizes the number of used programmable analog blocks and the power consumption of the topology. The MINLP procedure is a refinement of the method in [44]. As the MINLP equations assume ideal modulators, a post-optimization step fine-tunes the topology parameters while taking into account circuit nonidealities, such as jitter noise, kT/C noise, and the OpAmp white noise, finite DC gain, finite bandwidth, slew rate, and saturation [49]. This chapter demonstrates that the methodology is optimal, in the sense that it offers maximum covering of the performance ranges while minimizing the number of used programmable blocks.

The chapter is organized as follows. Section 3.2 summarizes related work. Section 3.3 introduces the reconfigurable mixed-signal architecture. Section 3.4 presents an overview of the reconfigurable AMS architecture. Section 3.5 discusses the proposed design methodology, and Section 3.6 describes a case study. Finally, conclusions are offered in Section 3.7.

3.2 Related work

Several general-purpose, reconfigurable, AMS architectures are mentioned in the literature [33, 34]. Continuous-time and switched-capacitor reconfigurable ADCs have been presented in [35, 36, 37], but no design methodology, or EDA tool, was considered. More recently the $PSoC^{TM}$ reconfigurable mixed-signal array has been offered by Cypress Inc. [38, 40] as a cost-effective solution to embedded system implementation. While several ADCs have been implemented using $PSoC^{TM}$ [41], there is no design methodology and there are no EDA tools that would allow effortless and rapid design of new ADCs. Filter synthesis methods for reconfigurable analog systems are presented in [33, 34]. Various techniques for single-mode (non-reconfigurable) $\Delta\Sigma$ ADC design have been described in [42, 43, 44]. A systematic design flow for continuous-time, reconfigurable, $\Delta\Sigma$ ADCs has been recently proposed [45]. However, the produced modulator topologies are restricted to a set of predefined performance specifications (e.g., GSM, CDMA, and UMTS communication standards). The topologies have no flexibility in addressing new performance specifications. If another specification is required then the entire process has to be repeated. This is an important limitation because flexibility ought to be one of the main strengths of reconfigurable AMS architectures.

This chapter proposes a systematic design methodology for creating flexible reconfigurable $\Delta\Sigma$ modulator topologies implemented on switched-capacitor (SC), reconfigurable AMS architectures. The work considered an AMS architecture based closely on the $PSoC^{TM}$ reconfigurable architecture. Since the number of programmable analog blocks is very limited in $PSoC^{TM}$, it is extremely important that the modulator topologies use as few blocks as possible. In contrast to the existing work, the proposed methodology generates a set of topologies that are optimized to meet performance attributes pertaining to ranges of values rather than being singular values. For example, the topologies are optimized for dynamic range (DR) requirements in the range $[DR_{min}, DR_{max}]$, and bandwidth (BW) constraints in the range $[BW_{min}, BW_{max}]$. The topologies produced can efficiently meet specifications $SP \in [DR_{min}, DR_{max}] \times [BW_{min}, BW_{max}]$ while using minimum amount of hardware. In contrast, other design methodologies would produce a single topology corresponding to the most constrained specification $< DR_{max}, BW_{max} >$. This topology is likely to be of higher order and with complex feedback and feed-forward structures [44]. The solution would obviously "waste" valuable programmable analog blocks for applications with less demanding requirements.

3.3 Reconfigurable Mixed-Signal Architecture

The reconfigurable AMS architecture under consideration is based on the $PSoC^{TM}$ mixed-signal architecture of Cypress Inc. [38, 40]. Compared to the $PSoC^{TM}$ architecture, a simplified the structure of the reconfigurable analog cells, and an increased number of configurable input connections of a cell, was employed so that $\Delta\Sigma$ topologies with complex structures could also be implemented using the architecture. This section offers an overview of the AMS architecture.

The main part of the architecture is a bidimensional array of reconfigurable switched-capacitor blocks. The array is connected to the I/O ports of the chip. The blocks are organized into m rows and n columns. Even though there is no theoretical limit for the number of rows and columns of an architecture, the number of rows and columns are actually not very high due to the lower integration densities achievable for reconfigurable AMS circuits, when compared to reconfigurable digital ICs. For example, the $PSoC^{TM}$ architecture includes, at most, three rows and four columns of configurable analog blocks [40].

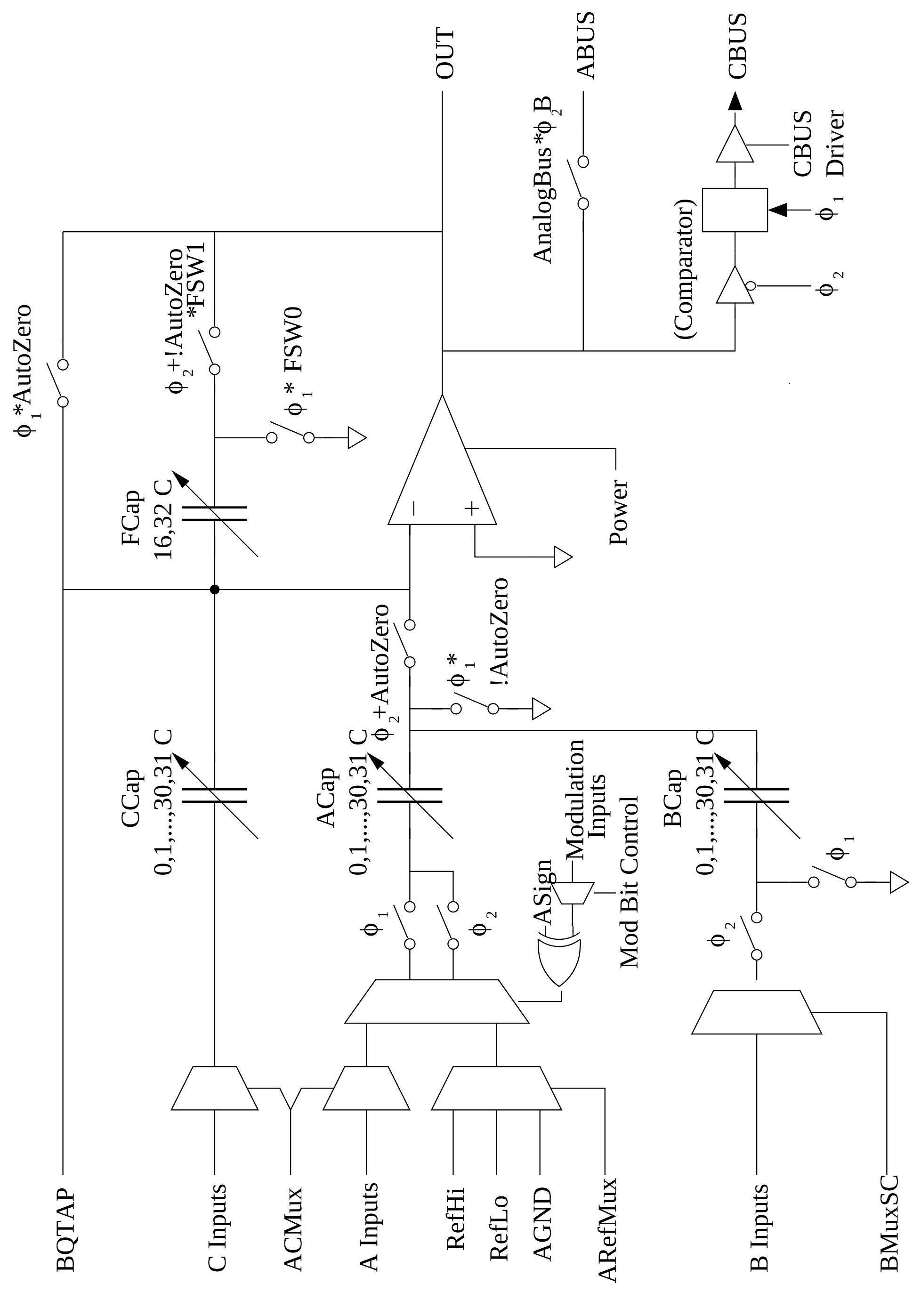


Figure 3.1: Type C reconfigurable SC block in the *PSoC*^{TM} architecture [40]

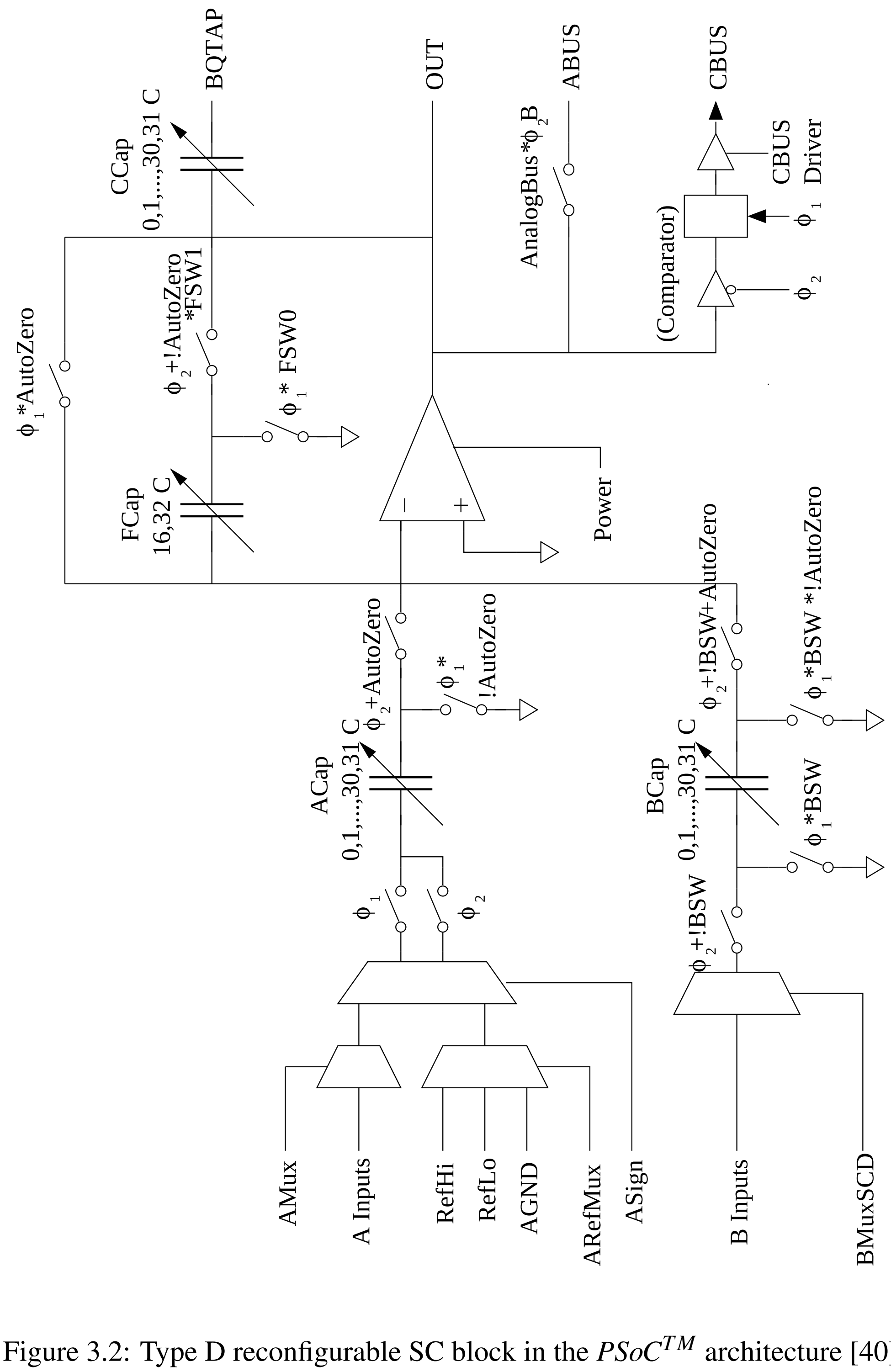


Figure 3.2: Type D reconfigurable SC block in the *PSoC*^{TM} architecture [40]

This makes the programmable analog blocks a very valuable resource that should be used conservatively in a design. Hence, minimizing the number of reconfigurable analog cells used for an implementation ought to be the main design requirement. The internal structure of type C and type D switch capacitor programmable analog blocks in $PSoC^{TM}$ is shown in Figure 3.1 and Figure 3.2.

The reconfigurable analog blocks can be connected into various $\Delta\Sigma$ topologies (netlists) through the programmable interconnect. The interconnect structure was designed, so that it offers versatile implementation of the feedback and feed-forward connections in a $\Delta\Sigma$ ADC [47, 44]. The output of each block can be connected either to the longer *row* and *column interconnects*, to implement the global feed-forward or feedback structures, or to the shorter *local interconnect* between neighboring cells. In contrast, the $PSoC^{TM}$ architecture includes only column and local interconnects [40], which might be somewhat restrictive for implementing topologies with complex feed-forward, and feedback, structures. Such architectures might offer superior DR and lower sensitivity to circuit nonidealities than architectures with few feedback structures [44].

All configurable analog blocks are identical, and their internal structure is shown in Figure 3.3. The structure is similar to that of the type C configurable $PSoC^{TM}$ blocks [40], except for the following two differences: (i) the number of programmable inputs connected to the summing node of the OpAmp was increased, and (ii) the elimination of the less used circuits, e.g., the comparator (available in each $PSoC^{TM}$ block), and the circuitry for producing interrupts. The first modification is needed for the implementation of complex high-order $\Delta\Sigma$ modulators. The second change is justified by the restriction of the AMS architecture to implementation of $\Delta\Sigma$ ADCs only, rather than a larger variety of circuits. The following are the main parts of a reconfigurable block.

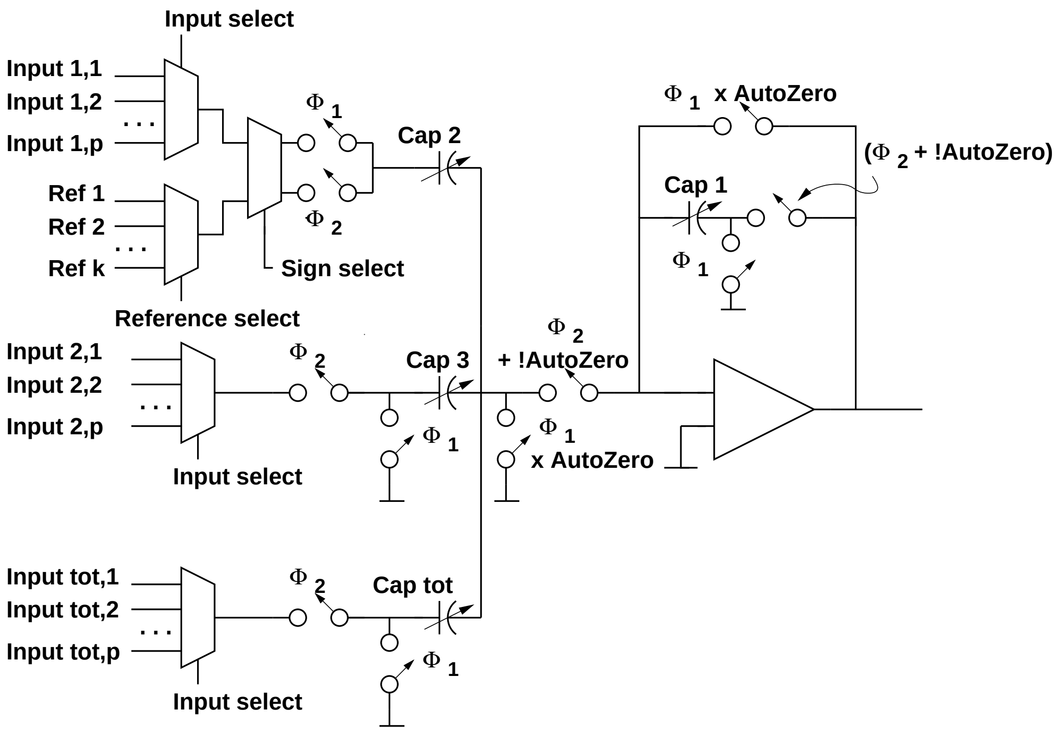


Figure 3.3: Reconfigurable SC block based on the $PSoC^{TM}$ architecture [40]

The functionality of the SC blocks is programmed by configuring the topology of the network surrounding the OpAmp. Similar to the $PSoC^{TM}$ blocks, the programmable cell can be configured to operate as a comparator, a gain stage, or an integrator. If the capacitor array Cap_1 is not connected then the circuit functions as a comparator, otherwise, the circuit is either a gain stage or an integrator. If the capacitor array Cap_1 is connected, then there is also the option of selecting whether the capacitor array should be discharged during the clock phase ϕ_1 for a gain stage, or if the capacitor is not discharged for an integrator.

In addition to the capacitor array Cap_1, each reconfigurable block has several other programmable capacitor arrays called Cap_2, Cap_3, ... and Cap_{tot}. Array Cap_2 is used for the input to the programmable cell. The rest of the capacitor arrays implement the ADC's feedback and feed-forward paths. The gain of the signal paths is set by programming the

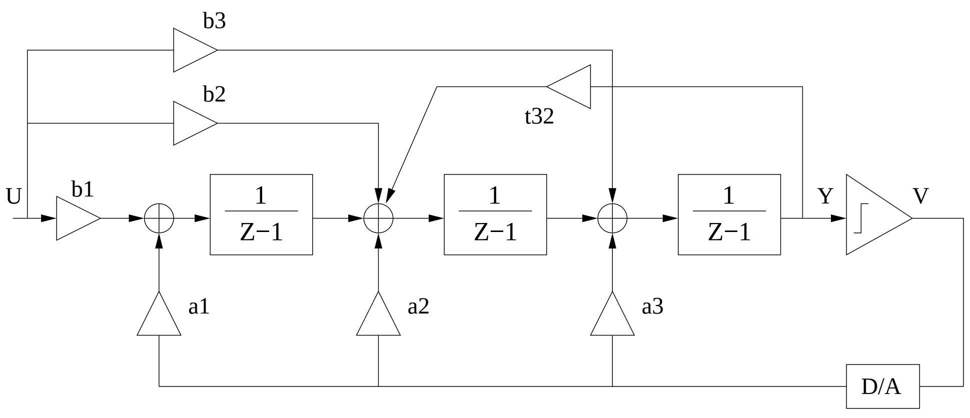


Figure 3.4: Third order ΔΣ modulator topology

ADC's feedback and feed-forward paths. The gain of the signal paths is set by programing the corresponding capacitor arrays. The value of `tot` is set by the maximum number of paths that can converge in a summing node. This value depends obviously on the order of ΔΣ for the ADCs that are implemented. For ADCs of order up to five, the value of `tot` is six. The values of the capacitor arrays can be selected from the set $\{0, \frac{1}{r}, \frac{2}{r}, \ldots, 1\}$. The value of `r` depends on the application characteristics.

Figure 3.4 shows a third order ΔΣ modulator topology. To implement this topology with the existing *PSoC*^{TM} switch capacitor architecture, signal path b3 and t32 cannot be mapped properly due to current architecture limitations, as shown in Figure 3.5. In contrast, a complete mapping of the third order topology is implemented on the proposed architecture as shown in Figure 3.6.

Section 4 shows that `r = 16` is sufficient for realizing ΔΣ ADCs of orders two and three, but `r = 128` is needed for fourth order ADCs. *PSoC*^{TM} uses `r = 32` [40].

Similar to the *PSoC*^{TM} blocks, for a positive gain, the input signal can be sampled by the clock ϕ1, or if the reference signal is selected, then it is sampled by the clock ϕ2. Also, there is the possibility of forcing or disabling the autozero capability of the SC circuits.

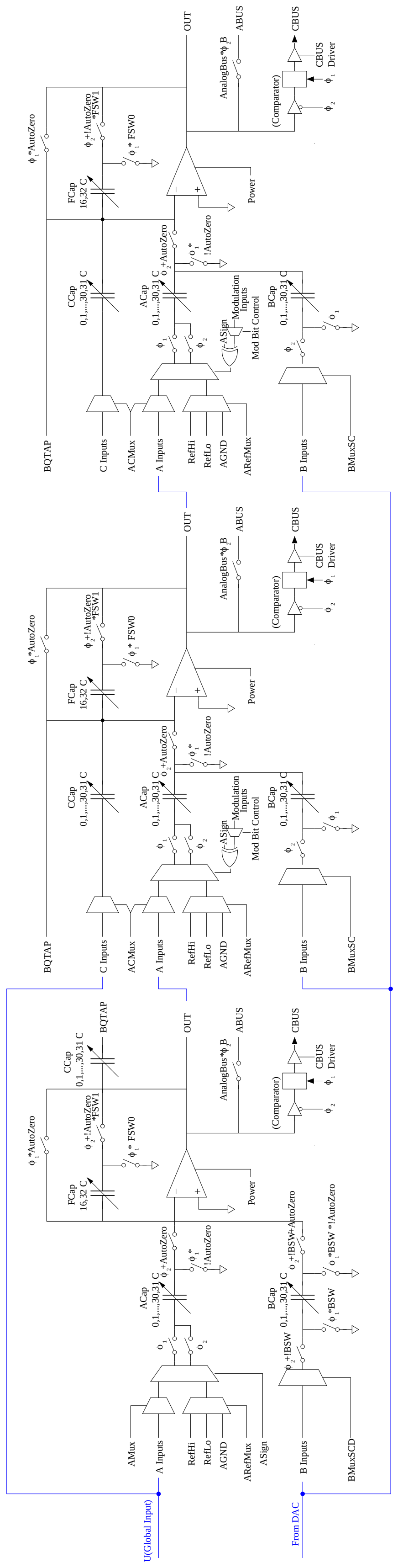


Figure 3.5: Implementation of Figure 3.4 topology on current $PSoC^{TM}$ SC architecture

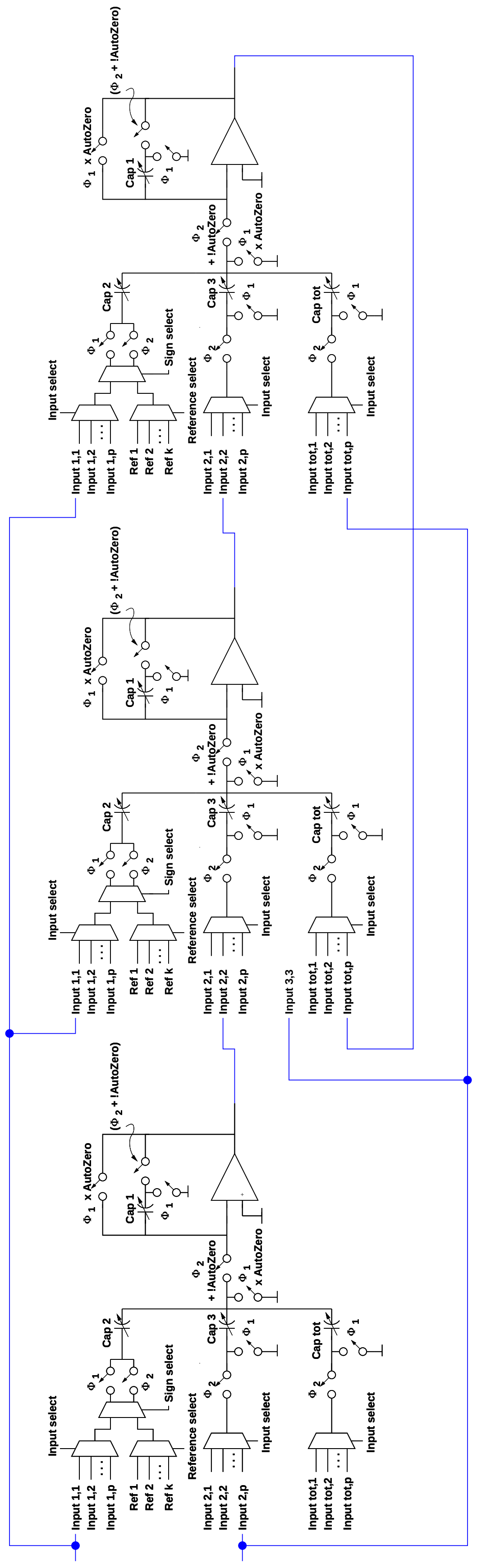


Figure 3.6: Implementation of Figure 3.4 on proposed SC architecture

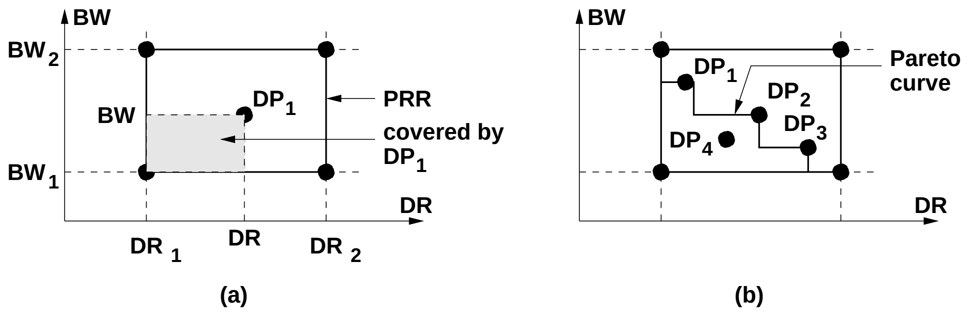


Figure 3.7: Performance requirement region (PRR)

The configurable functionality and interconnect of a SC block is defined by programming dedicated control registers. The register bits are set or open the reconfiguration switches, or select the inputs to the multiplexing circuits.

3.4 Reconfigurable Δ ADC Design Method

This section presents the main parts of the proposed methodology for reconfigurable ΔΣ modulator design: (1) the notation for expressing flexible performance requirements, (2) the technique for finding optimized modulator topologies, and (3) the overall methodology.

3.4.1 Description of flexible performance requirements

Before presenting the proposed design methodology and the related design steps, the notation used for specifying the flexible performance requirements of reconfigurable ΔΣ ADCs will be discussed. This is important to identify the set of actual (punctual) performance requirements used in the design process. For example, typical embedded control applications might require ADCs with a variable conversion accuracy (e.g., going from 6 bits to 12 bits), an adjustable peak SNR (SNR_{max}) (such as in the range 45dB to 90dB), and capable of converting signals in a bandwidth ranging from 50 kHz to 1 mHz. For these performance ranges, one has to identify the set of *basic* performance requirement values that maximally cover the ranges. Some of the requirements are hard to meet (such as precise conversion of fast signals).

Note that implementing only one $\Delta\Sigma$ modulator corresponding to the most constrained scenario is not a very efficient solution. The topology is likely to be of high order, with complex feedback and feed-forward structures, and requiring many reconfigurable analog blocks for its implementation. As explained in Section 5.2, the reconfigurable analog blocks are very valuable resources, and should be utilized with care. Utilizing less resources for simpler ADCs allows implementing more functionality on the reconfigurable AMS architecture. Also, the power consumption of complex modulators is higher, which leads to unnecessary waste of energy for more demanding applications. Third, high-order $\Delta\Sigma$ ADCs tend to be unstable [47], which limits the range of the input signals that are converted. This is justification for the observation that it is more effective to design a set of $\Delta\Sigma$ topologies (which are reconfigured for different specifications) than design the worst case topology, only.

The variable (flexible) performance requirements are expressed as closed ranges $[P_{min}, P_{max}]$. For example, the variable DR is defined as the range [50 db, 90 dB], the adjustable peak SNR_{max} as the range [50 db, 90 dB], and the bandwidth as the range [50 kHz, 2 mHz]. Figure 3.7(a) shows the resulting *performance requirements region* (PRR) - if only two requirements are considered, bandwidth and DR. The concept can be easily extended for more metrics, such as SNR_{max}, linearity, and power consumption.

The two performance ranges define a rectangular PRR. In general, the PRR becomes an n-dimensional parallelepiped, in case all n performance requirements are ranges, or a *k*-dimensional parallelepiped in an *(n-k)* discrete space, if *k* of the performance are ranges, while the rest of *(n-k)* are discrete sets. For simplicity, the present discussion refers from **this point** on only to rectangular PRRs introduced by two performance requirement ranges.

Definition: Assume that the design point DP_k is characterized by the bandwidth BW_k and the dynamic range DR_k. Then the point DP_k *covers* the rectangular PRR formed by the points $< bw, DR >$, where $bw < BW_k$ and $DR < DR_k$, as all possible requirements *bw* - *DR* are satisfied by the design point DP_k. The *hardware cost* of a point DP_k is equal to the number of reconfigurable SC blocks used to implement the point.

Figure 3.7(a) shows the PRR covered by the point DP_1. In the present case, the hardware cost of a design point is equal to the number of reconfigurable SC blocks in the implementation.

Definition: A design point DP_k is called a *basic* design point, if its covered PRR is not contained in the PRR of another design point or the union of the PRRs of a set of points with lower or equal hardware cost as the point DP_k.

The set of basic design points with the same hardware cost define the Pareto curve for that hardware cost. For example, Figure 3.7(b) shows the Pareto curve defined by the set $\{DP_1, DP_2, DP_3\}$ of basic points. Point DP_4 is not basic, unless its hardware cost is lower than that of point DP_2. Also, point DP_5 is not basic, if the individual hardware costs of points DP_1, DP_2, and DP_3 are less or the same as the cost of point DP_5.

Problem statement: The reconfigurable $\Delta\Sigma$ modulator design problem can be formulated as that of finding for *successively increasing hardware costs*, (i) the sets $SBDP_i$ of basic design points , and (ii) the corresponding $\Delta\Sigma$ modulator topologies, such that each set $SBDP_i$ covers a maximal PRR that is not also covered by the *SBDP* sets of smaller hardware cost.

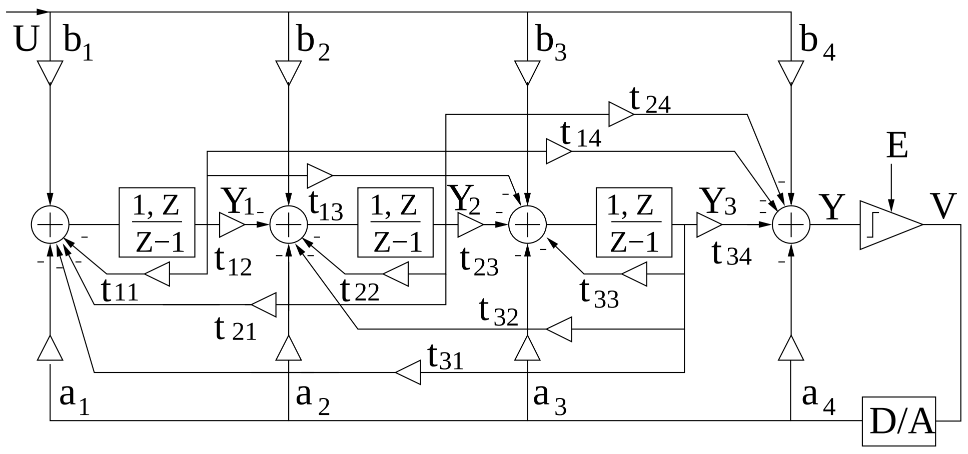


Figure 3.8: Generic 3rd order ΔΣ topology

Note that the problem statement does not require finding the sets $SBDP_i$ of minimum cardinality, thus it is not necessary to find the minimum number of basic design points that cover a PRR. The reason is that the configuration data required for implementing a modulator is minimal. It is equal to the number of control registers that have to be programmed, and is several bytes per topology.

3.4.2 Optimal ΔΣ ADC topology design

This subsection summarizes the method for synthesizing the optimal modulator topology for a given specification. The method is a refinement of the technique presented in [44]. The main parts of the method are (a) the generic topology, (b) the synthesis procedure based on MINLP, and (c) the cost function for topology synthesis.

A. Generic ΔΣ modulator topology. The crux of the ADC topology synthesis method is a generic representation that describes all the possible topologies for single-bit single-loop ΔΣ modulators. Figure 3.8 shows the representation for 3rd order ΔΣ modulators, but similar representations exist for higher order modulators also. The generic representation includes all possible feedback and feed-forward signal paths. Y_i represents the output of the i^{th} integrator, and Y is the input to the quantizer. A_i stand for the feedback coefficients from the output to the i^{th} adder, b_i are the feed-forward coefficients from the input to the i^{th} adder, and t_{ji} are the coefficients from Y_j to the i^{th} adder in the modulator. There are negative signs for all t_{ji} and a_i coefficients.

Let N be the modulator order. Then, following expressions hold, as a general rule:

$$t_{ji} \geq 0, if\ j \geq i, j = 1, ...N, i = 1, ...N+1 \tag{3.1}$$

$$t_{ji} \leq 0, if\ j < i, j = 1, ...N, i = 1, ...N+1 \tag{3.2}$$

$$a_i \geq 0, b_i \geq 0, i = 1, ...N+1 \tag{3.3}$$

There are $(N+1) \times (N+2)$ coefficients in the generic topology. It can be seen that many of the "classic" topologies [47] can be derived from the generic topology by removing some of the signal paths. Note also that the integrators could be either delayed, or delayless. For the generic topology of order N, its noise transfer function (NTF) and signal transfer function (STF) in terms of the coefficients of all signal paths are derived. It is assumed that the quantization noise E is additive white-noise[47]. For example, for a 2nd order generic $\Delta\Sigma$ modulator with delayed integrators:

$NTF_n = z^2 - z(2 - t_{11} - t_{22}) - (-1 + t_{11} + t_{12}t_{21} + t_{22} - t_{11}t_{22})$

$NTF_d = z^2(1 + a_3) - z(2 + 2a_3 - t_{11} - a_3t_{11} + a_1t_{13} - t_{22} - a_3t_{22} + a_2t_{23}) - (-1 - a_3 + t_{11} + a_3t_{11} - a_1t_{13} + t_{12}t_{21} + a_3t_{12}t_{21} - a_2t_{13}t_{21} + t_{22} + a_3t_{22} - t_{11}t_{22} - a_3t_{11}t_{22} + a_1t_{13}t_{22} - a_2t_{23} + a_2t_{11}t_{23} - a_1t_{12}t_{23})$.

B. Optimal topology generation using MINLP. By equating the symbolic TFs to the desired TFs (desired STF is assumed to be 1), $3 \times (N+1)$ equations are obtained. Obviously, there are an infinite number of solutions, considering that the number of unknowns - $(N+1) \times (N+2)$, is always larger than the number of equations - $3 \times (N+1)$. Also, in order to select any signal path in the generic topology, a corresponding binary 0/1 variable was defined to denote whether the signal path is present, or not.

For a given a cost function f, the topology synthesis problem can be formulated as:

$$\begin{gathered} minimize\ cost\ f(x_i, wx_i); \\ subject\ to : g(x_i) = 0; \\ subject\ to : h(x_i, wx_i) \leq 0; \\ subject\ to : x_i\ satisfy\ (1), wx_i \in \{0, 1\}; \end{gathered}$$

where x_i denotes any of the unknown coefficients a_i, b_i and t_{ji} defined in (1), g are the $3 \times (N+1)$ equality constraints obtained from equating the symbolic NTF and STF to the desired NTF and STF, and h are the inequality constraints relating the coefficient variables to the binary variables, so that wx_i correctly identifies whether the signal path with coefficient x_i is present, or not.

The resulting problem can be optimally solved using mixed-integer**u** nonlinear, programming (MINLP) [48]. Thus, MINLP solutions offer the best topology with respect to the cost function f. MINLP formulation is scalable, and it is easy to add additional constraints.

C. Cost function formulation. The cost function used in topology synthesis includes the following two terms: (1) one for minimizing the hardware cost of a ΔΣ modulator, and (2) one for minimizing its power consumption.

Since binary variables denote whether the corresponding signal paths are present or not, the hardware cost minimization was formulated as:

$$Minimize \sum_{i=1}^{(N+1)(N+2)} wx_i$$

Power consumption estimation was similar to [43]. The static power consumption of the OpAmp, and the dynamic power consumption of the capacitors were considered.

3.4.3 Overall design flow

The overall design flow is presented in Figure 3.9. The first step finds the minimum-order $\Delta\Sigma$ modulator topology with the least number of feedback and feed-forward paths. The topology is synthesized for the least demanding performance requirements of the PRR, the minimum bandwidth BW_{min} and the minimum dynamic range DR_{min}. The oversampling ratio is fixed to the maximum value OSR_{max}. Using the generic topology for that order, the optimal modulator topology is found for the requirements $< bw_{min}, DR_{min} >$ by solving the MINLP equations.

The resulting topology is post-optimized using a simulated annealing (SA) algorithm that attempts to improve the DR of the modulator by performing localized changes of the topology coefficients. The signal flow of the topology remains unchanged during post-optimization. The performance of each solution analyzed by SA is estimated by a MATLAB simulation of the behavioral models for SC $\Delta\Sigma$ modulators, similar to the technique by Malcovati *et al.* [49]. The models include the main circuit nonidealities, such as sampling jitter, kT/C noise, and OpAmp white noise, finite DC gain, finite bandwidth, slew rate, and saturation [49].

Assuming that the resulting design point DP_1 is characterized by the performance attributes bw_1 and DR_1, the point DP_1 is the first basic point DP found by the method.

The next step finds more basic design points with the same hardware cost as the point DP_1. Therefore, the order of the modulator is kept the same as that of the point DP_1, but the OSR is modified. Lower OSRs were then considered. For example, if the modulator for DP_1 uses $OSR = 128$, then the methodology will consider $OSR = 64$, then $OSR = 32$, and so on. Repeating the steps that produced the topology for DP_1, the topologies for the successively decreasing OSR values are synthesized. These design points are denoted as DP_2, DP_3, ..., where the point DP_i corresponds to a larger OSR value than the point DP_{i+1}. This means that the topology for the point DP_i offers a higher DR, but converts signals of lower bandwidth. Figure 3.9(b) illustrates the design points for different OSR values. Note that the sequence of points defines a *stair-case Pareto curve* in the PRR space.

It is obvious that each of the design points DP_1, DP_2, DP_3, ... are basic design points since they correspond to different bandwidth - DR trade-offs. It is not possible for the PRR covered by point DP_i to be entirely included in the PRR of another point, because all points have the same order, and are optimal for the specification for which they were synthesized.

Also, the set of design points produces a maximal covering of the PRR since each of the points was synthesized for a maximum DR. For the given set of OSR values, it is impossible to generate topologies with higher DR than those produced through MINLP [44]. This is because the MINLP algorithm produces mathematically optimal topologies.

The methodology continues by considering $\Delta\Sigma$ modulators of order higher by one. This corresponds to increasing the hardware cost of the design points by one SC block. Then, as in the previous step, the OSR is set to the maximum value, and the optimal modulator topology is synthesized. More design points are produced for successively decreasing OSR

```
(1) for modulator orders N = 1, 2, 3, ... do
(2)   for decreasing OSR values do
(3)      find the optimal modulator topology by
          solving the related MINLP equations
          for the given order N and OSR;
(4)      post-optimize the topology using SA
          guided by MATLAB simulation of the
          modulator behavioral models;
(5)   identify the stair-case Pareto curve and
          mark the PRR covered by the modulators
          of order N;
```

Figure 3.9: Overall design methodology

values, similar to the previous case. This generates a new Pareto curve.

The methodology ends either when the entire PRR is covered, when the order of the analyzed modulators becomes infeasibly large, or when the required hardware cost for the design points exceed the capability of the reconfigurable AMS architecture.

3.5 Case Study

The goal was to design reconfigurable $\Delta\Sigma$ modulators that would cover the PRR defined by the bandwidth range BW = [150kHz, 1.2mHz] and the DR range DR = [50dB, 90dB]. The PRR corresponds to typical communication standards, e.g., UMTS standard has BW = 1.2 mHz and DR = 70 dB requirements, CDMA2000 standard has BW = 615 kHz and DR = 80 dB requirements, etc. [37]. Note that the considered PRR = $[150kHz, 3mHz] \times [50dB, 90dB]$ includes not only the four standards, but also all other standards with close requirements. In addition, the topologies should cover the targeted PRR using the minimum number of reconfigurable switch-capacitor blocks. The structure of the blocks was shown

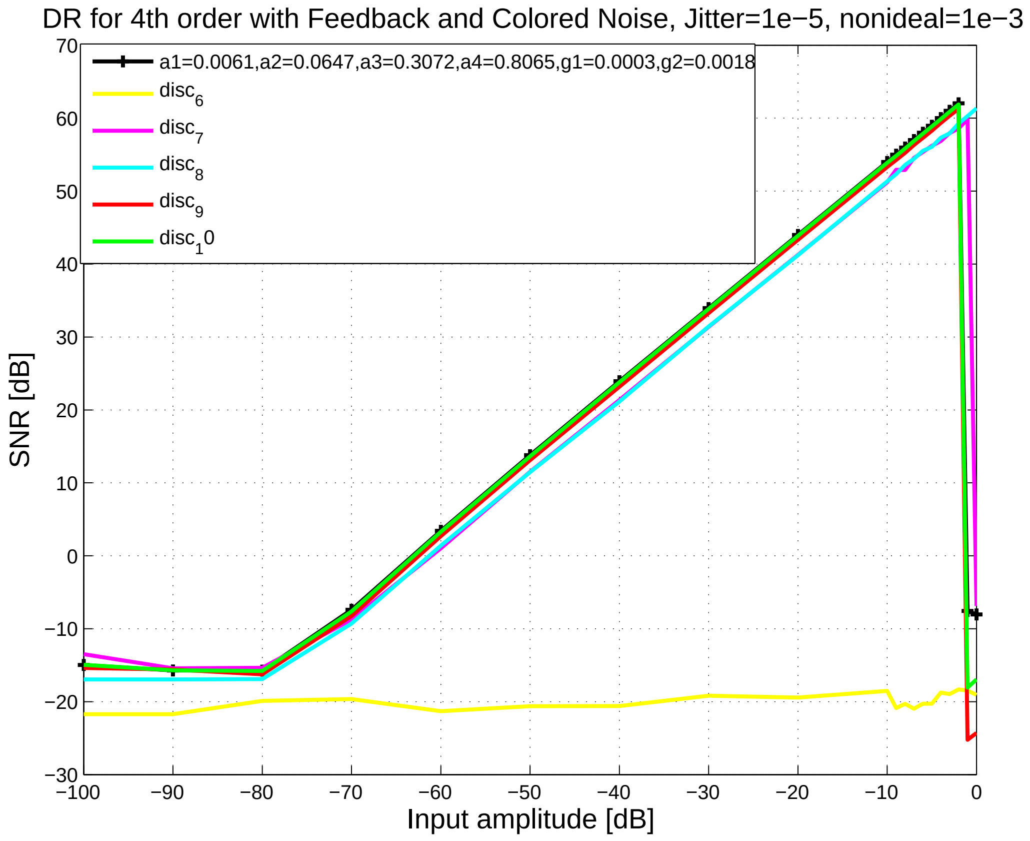


Figure 3.10: Influence of discretization on DR

in Figure 3.3.

A. Finding the discretization value for the switch-capacitor block capacitor arrays. Before synthesizing the reconfigurable topologies, a set of experiments was set-up for finding the discretization value r for the capacitor arrays. A capacitor array can implement the values $\{0, \frac{1}{r}, \frac{2}{r}, ..., 1\}$. This experiment is important because the value r determines the precision of implementing the topology coefficients on the reconfigurable switch-capacitor blocks. The larger the value of r the more precise the coefficient values can be implemented, but this increases the hardware overhead needed for programming the capacitor arrays.

A large number of topologies from the literature [47, 44] were analyzed, and simulated for different nonidealities [49]. For the fourth order modulator topology in [44], Figure 3.10 shows the resulting DR values for different discretizations $r = 2^i$, $i = 6 - 10$. For the value r = 64 (and all lesser values), the topology did not operate as a $\Delta\Sigma$ modulator. However, for values r larger than 7, the resulting DR was close enough to the ideal case. In general, for fourth order topologies, a discretization value of r = 7 is needed for achieving good DR performance. For second and third order modulators a value r = 16 provides a sufficiently good accuracy for implementing the modulator coefficients.

B. Synthesizing flexible topologies. This methodology was used to find the topologies that cover the defined PRR using a minimum number of configurable SC blocks. SA-based, post-optimization was conducted to adjust the coefficients in the presence of leakage, finite DC gain, slew rate, saturation, jitter, and colored noise. First, a set of second order modulators was produced using the MINLP-based topology synthesis procedure. Topologies were generated for the OSR values 128, 64, 32, and 16. Figure 3.11 presents the staircase, Pareto curve that resulted for the second order topologies, and the PRR covered by

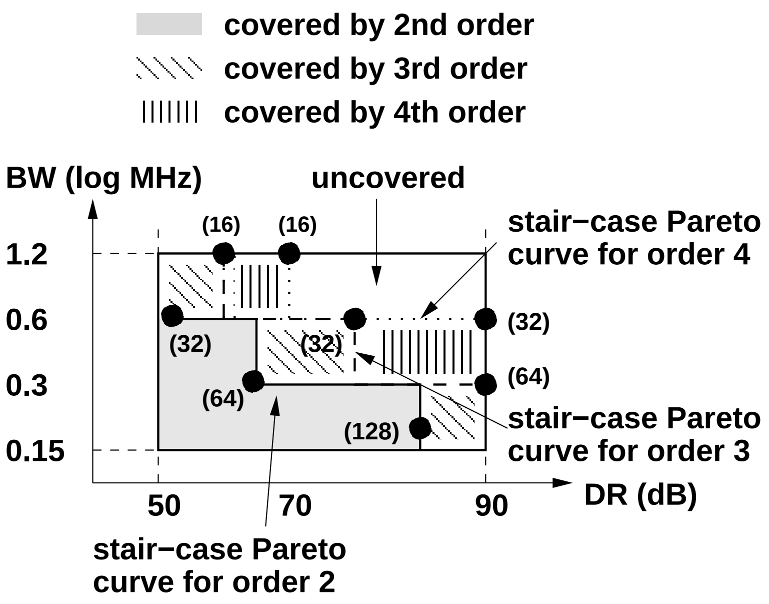


Figure 3.11: Pareto curves for different orders

these. The bullets correspond to basic design points. The corresponding OSR is shown in brackets. The architecture, for OSR = 16, did not cover any portion of the PRR, since its DR = 47 is below the lower limit of the PRR. This topology was eliminated from the set. All second order topologies used two configurable SC blocks for implementation. Then, the modulator order was increased to three, and then to four. Two new stair-case Pareto curves were produced, as shown in Figure 3.11. All third order modulators require three SC blocks, and all fourth order modulators used four blocks. One third order, and two fourth order**u** topologies were also eliminated from the set, because their covered PRRs were already covered by lower cost topologies.

The remaining 8 topologies offer a large covering of the PRR. The uncovered region corresponds to very demanding specifications that cannot be met by modulators of order lesser than 5 and with OSR up to 128. Note that only a small fraction (14.5%) of the PRR requires modulators of fourth order. Assuming a uniform probability for having all specifications in the PRR, the proposed methodology saves, in 17% of the cases, one SC block, and in about 40% of the cases, two SC blocks as compared to the straightforward solution (that implements one fourth order modulator for the worst case). Considering that all blocks have similar power consumption, this also results in a reduction in power consumption by approximately 50%, in 40% of the cases, and by 25%, in 17% of the situations.

3.6 Conclusions

This chapter has presented a systematic methodology for producing reconfigurable, $\Delta\Sigma$ modulator topologies with optimized flexibility. To increase their flexibility, topologies were optimized to meet performance attributes pertaining to ranges of values rather than singular values. Since the number of configurable SC blocks available in AMS architectures is very limited, the methodology minimizes the number of SC blocks required by a topology.

The methodology is optimal, in the sense that it offers maximum covering of the performance ranges while minimizing the number of used programmable blocks. A case study addressing specifications for telecommunications showed that compared to traditional designs, the topologies produced required fewer configurable SC blocks, and resulted in lower power consumption by 25%-50%.

Chapter 4

Dynamic Reconfiguration of Mixed-Domain Embedded Systems for Applications with Variable Performance Requirements

4.1 Introduction

Modern applications in environmental monitoring, security, robotics, military, and industrial control are based on (i) acquisition of high-level data (metadata) (video images, sound, olfactory and hyperspectral images), (ii) processing, storing, and communication of metadata, and (iii) intelligent control using higher-order semantics extracted from metadata [28, 29, 31, 39]. Systems continuously collect metadata to discover and plan new activities of interest, or to observe patterns about monitored situations. Then, the acquired knowledge is used to provide self-optimizing responses, including faster and more intelligent anticipation and reaction to critical events, superior utilization of hardware, and better surviving in harsh situations.

The performance constraints of these embedded systems are being dynamically modified, since operating conditions change continuously. Also, the hardware resources available for a task depends on the computational demands of the other tasks, since all tasks share the same resources. The dynamics of processing and performance needs does not follow any particular mathematical rule, being rather a mixture of different types of behavior, e.g., quasi-static, probabilistic, and performance constrained operation [28]. Hence, the system must continuously "comprehend" new processing demands, and accordingly customize its architecture and the dispatching of hardware resources to tasks. A naive approach would aggressively utilize high performance resources that are optimized for the worst-possible situation. However, this results in very expensive systems, likely to be unattractive in real life. Moreover, high performance processing is very energy hungry, hence unsuitable for mobile, autonomous systems.

Reconfigurable systems are attractive implementation platforms for many embedded applications due to their ability to provide low development costs and short design times, while being accessible to less experienced designers. While reconfigurable digital systems are very popular due to their capability of offering low development costs and short design time [54, 55, 56], reconfigurable mixed-domain systems (analog and digital) - in contrast, are much less studied, or employed in practical applications. This prevents a more comprehensive harvesting of the possible benefits of reconfigurable systems, since the majority of applications (e.g., embedded control and telecommunications) include significant amounts of analog signal processing. To address this limitation, research must not only focus on new reconfigurable mixed-domain architectures, but also study the related design methodologies and EDA tools. More specifically, it is essential to develop efficient techniques for designing dynamically reconfigurable systems, which include both analog and digital functional blocks.

Techniques for dynamic reconfiguration of software or digital hardware are discussed in [54, 57, 58, 56]. Several reconfigurable mixed-domain architectures are mentioned in the literature [33, 38, 34]. More recently the PSoC reconfigurable mixed-signal array has been offered by Cypress Inc. [40] as a cost-effective solution to embedded system implementation. Filter synthesis for reconfigurable analog systems is presented in [33, 34]. A systematic design flow for continuous-time, reconfigurable, $\Delta\Sigma$ ADCs has been recently proposed [45]. However, the topologies produced have no flexibility in addressing new performance specifications. Also, there is no methodology that considers the synergy between analog and digital blocks for producing more cost effective, dynamically reconfigurable designs.

This chapter proposes a methodology for designing cost-effective, dynamically reconfigurable mixed-domain implementations for metadata processing applications. To provide flexibility, performance requirements are specified as continuous ranges, or as discrete sets of values. During operation, the system switches between different design points (DPs) for its (reconfigurable) analog and digital blocks depending on the actual performance constraints, so that minimum amount of hardware is used. The proposed methodology includes two parts: (i) DP selection and (ii) system adaptation. This chapter presents two DP selection algorithms, one for performance requirements expressed as continuous ranges (e.g., bandwidth, latency, range), and one for performance defined as discrete sets (such as precisions, dynamic range). System adaptation is realized as a centralized procedure that selects for new requirements the most effective set of DPs, so that all DPs for coupled blocks can operate together (they are compatible). A case study for sound-based tracking is discussed.

The chapter has six sections. Section 2 formulates the problem, Section 3 introduces formal modeling of mixed-domain systems, and Section 4 presents the proposed algorithms. A case study is shown in Section 5, and finally, conclusions are offered.

4.2 Problem Formulation

This section discusses (a) an illustrating example for mixed-domain applications, (b) the specifics of programmable mixed-domain architectures, and (c) the dynamic reconfiguration problem in mixed-domain systems.

A. Programmable mixed-domain embedded applications. Figure 1.1 shows the structure of a sound-based tracking algorithm [29]. The system computes the position of a sound source by finding the phase difference between the audio signals received by two fixed microphones. Tracking is the main problem in applications such as environmental monitoring, unmanned autonomous vehicles (UAVs), robotics, sensor networks, and other [29, 31, 39].

For each application, the specification defines *(i)* the characteristics of inputs and outputs, *(ii)* the requirements for computing the corresponding output for each input, and *(iii)* the constraints of the implementation. For example, the tracking system specification defines input and output characteristics, such as bandwidth, range, the input noise level, and needed precision (number of bits) of the output. Processing requirements include the timing (speed) constraint for producing correct outputs for given inputs. Finally, implementation constraints include hardware cost, total power consumption, required supply voltage and clock frequencies, and number of necessary I/O pins.

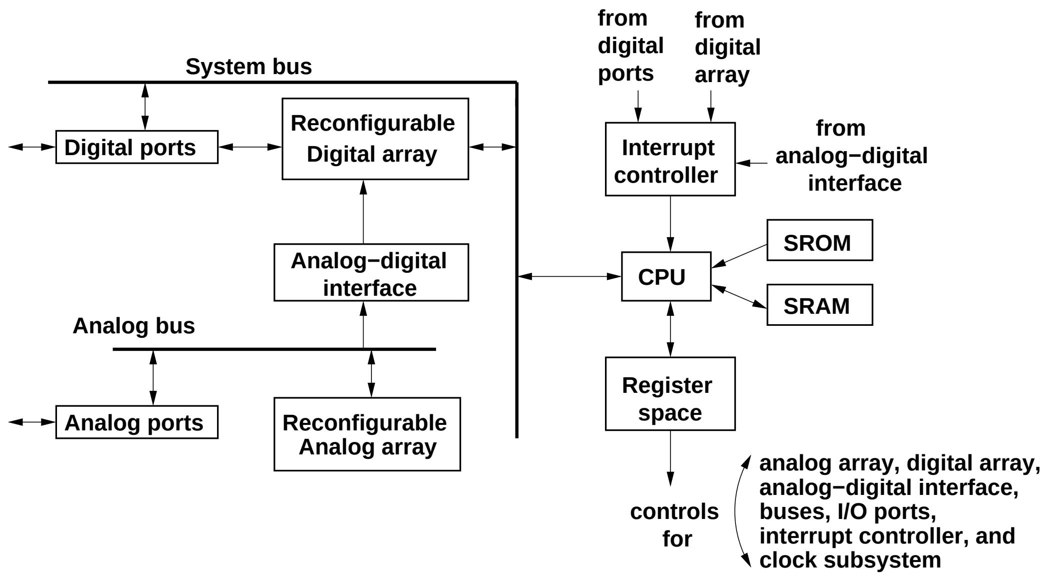

System bus
Digital ports
Reconfigurable Digital array
Analog−digital interface
Analog bus
Analog ports
Reconfigurable Analog array
from digital ports
from digital array
Interrupt controller
from analog−digital interface
SROM
CPU
SRAM
Register space
controls for
analog array, digital array, analog−digital interface, buses, I/O ports, interrupt controller, and clock subsystem

Figure 4.1: Mixed-domain architecture

Note that three categories are correlated. For example, the timing (speed) constraint of processing is related to the input bandwidth and needed output precision. The higher the bandwidth and precision, the tighter the timing constrains should be. Also, the ranges of input signals impact the gain of the gain blocks, the gain influences the transfer function of the filters, filters influence the ADC characteristics, the speed of ADCs is correlated to the FFT throughput, and so on.

Performance attributes can pertain to continuous ranges, or sets of discrete values. For example, the input signal bandwidth is in range $[BW_{MIN}, BW_{MAX}]$. Depending on the application, the actual input signal frequency has a value in this range. Similarly, the required precision (number of data bits) is in the discrete set $\{PR_1, PR_2, ..., PR_{MAX}\}$. The system precision is set to PR_{MAX} bits for highest precision applications, and to PR_1 bits for lowest precision.

B. Programmable mixed-domain embedded architecture. Figure 4.1 presents the defining aspects of the programmable mixed-domain architecture considered in this work.

Programmability is achieved in the following ways: (1) by programming the microcontroller (software), (2) by programming hardware (reconfigurable analog and digital circuits), (3) through programming the SoC interconnect, and (4) by programming the I/O ports. This architecture is largely based on Cypress' PSoC architecture [40], except that the programmable hardware is based on LUT's, similar to FPGAs.

C. Problem description (dynamic reconfiguration). Assuming that different system performance can be continuously selected from ranges or discrete sets, the design methodology must identify design points (DP) for the system blocks, so that dynamic reconfiguration offers efficient coverage of the variable performance. In addition, an adaptation algorithm must be devised to control the on-line switching between DPs. The first step is called *design point selection*, and the second step is *system adaptation design*.

The system adaptation algorithm in Figure 4.2 was considered for switching DPs whenever required by the variable performance constraint. The control strategy uses a centralized reconfiguration procedure, which receives reconfiguration requests from all blocks. For example, the application might require a higher data precision, which imposes a higher dynamic range (DR) for the ADC, higher bit width for the digital blocks, and so on. These requests are issued to the reconfiguration procedure by the blocks involved. Also, the application might request less memory usage by buffers. This can be achieved by increasing the processing speed of the digital blocks. The reconfiguration procedure inspects the DP set for each block, and then selects compatible DPs (for each block), so that the new performance requirement is met.

The total resource cost of the selected DPs must be minimal. Implementing only one DP for each block, corresponding to the most constrained scenario, is not a very efficient solution. The design is likely to be more complex than required in most situations, e.g.,

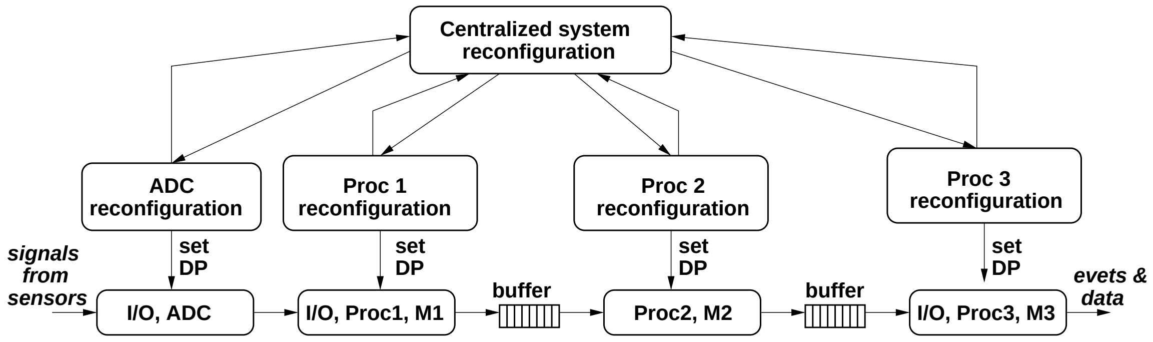


Figure 4.2: Dynamic reconfiguration of mixed-domain systems

using analog and digital filters, and ADC DPs of high order, with complex feedback and feed-forward structures, and requiring many reconfigurable analog and digital blocks for their implementation. Reconfigurable analog and digital blocks are very valuable resources, and should be utilized with care [40]. Utilizing less resources allows the implementation of more functionality of the reconfigurable architecture. Also, the power consumption of complex circuits is higher, which leads to unnecessary energy dissipation for more demanding applications. Third, complex circuits are harder to test, verify, and debug, and can be more sensitive to nonidealities, e.g., high-order ΔΣ ADCs tend to be more unstable, which limits the range of the input signals that can be converted. Tyherefore it is more effective to design a set of reconfigurable DPs (which are programmed for different performance specifications) than design for the worst case DPs, only.

4.3 Modeling for Dynamic Reconfiguration

This section describes the system description used for dynamic reconfiguration.

Mixed-domain reconfigurable systems (MDRS) are characterized by:

$$MDRS \quad = \quad PI \times PO \times Modes \times Func \tag{4.1}$$

where the PI are the primary inputs of the system, PO are the primary outputs, $Modes$ are the control signals used in selecting new DPs, and $Func$ is the description of the behavior for all operation conditions.

Specification $Func$ represents a chaining of reconfigurable blocks (called $RBlock$), as shown in Figure 4.2. Each $RBlock$ is the following quadruple:

$$RBlock \;=\; AIS \times AOS \times RModes \times PSFG \tag{4.2}$$

AIS and AOS are the inputs and outputs of a block, $RModes$ are the control signals for configuring the block, and $PSFG$ is a parameterized Signal Flow Graph expressing the multi-mode behavior of the block. $RBlock$ is shown in Figure 4.3(a). Without restricting the generality of the methodology, it can be assumed that $AIS_i \subset PI \cup AOS_k$ (blocks k have their outputs connects to the inputs of block i), and $RModes \subset Modes$.

The operation semantics of $RBlock$ are shown in Figure 4.3(b). Blocks have multi-mode operation, each mode corresponds to a different DP. The figure presents three modes (SFG_i). Transitions between modes are controlled by conditions defined over controls in set $Modes$.

Each AIS (AOS) is an attributed input (attributed output), and is defined as the Cartesian product of its attributes:

$$AIS(AOS) \;=\; A_1 \times A_2 \times \ldots \times A_k \tag{4.3}$$

A_i is an attribute with values in a range or set of discrete values. For example, attributes might correspond to signal range (e.g., $A_i \in [-1V, +1V]$), signal bandwidth (e.g., $A_j \in [20Hhz, 8kHz]$), precision (e.g., $A_p \in \{8\ bits, 9\ bits, 10\ bits\}$), and so on.

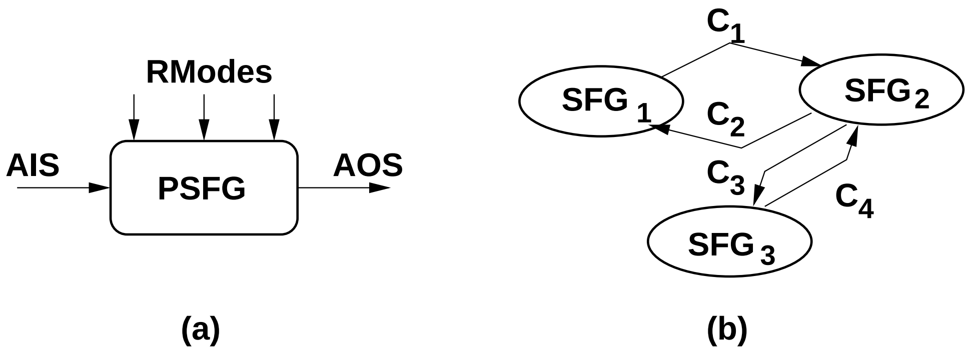


Figure 4.3: (a) *RBlock* and (b) *RBlock* execution semantics

The *RBlock* description in Figure 4.3(b) is suitable for expressing the block behavior for simulation and verification, but it less useful for DP selection since it represents blocks as a set of disjoint SFGs, while in reality DP implementations share resources (software, hardware, or memory). For example, the algorithm in Figure 4.4 includes two SFGs. One SFG uses the entire graph, and the other one employs only the left half (which is thus shared between the two SFGs).

For reconfiguration design, the functionality of *RBlock* is expressed using Parameterized SFGs ($PSFG$). A PSFG is composed of processing blocks (e.g., adders, multipliers, integrators, comparators, etc.) and switches that are interconnected with each other by signals. Switches are controlled by signals in the set $RModes$. A closed switch propagates a signal, while an open switch propagates a neutral value (e.g., zero for digital hardware or high impedance for analog circuits). Figures 4.4 and 2.1 show PSFGs for a digital DFT filter and a mixed-signal ADC.

PSFG of DFT algorithm (Figure 4.4) consists of nodes for data processing (addition and multiplication), and arcs for indicating the data flow between nodes. This is similar to the data flow graphs used in high-level synthesis. The AIS set includes all block inputs with bit width, as one of their attributes A_i. The AOS set is formed from the block output

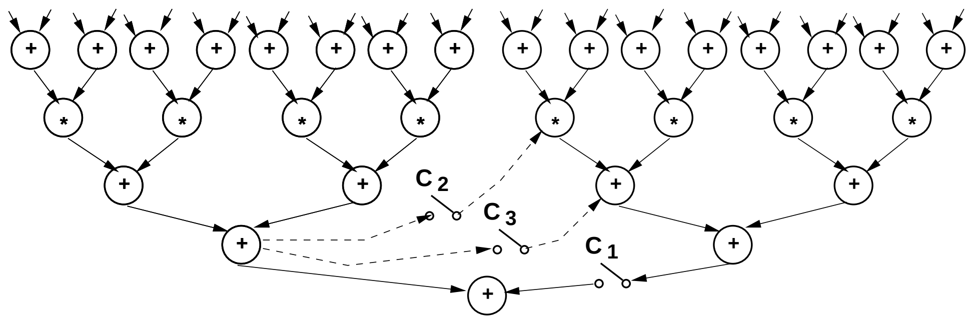


Figure 4.4: Parameterized SFG for DFT

with bit width as an attribute. Parameterization is achieved by switch C_1, which includes, or removes, the right half of the algorithm, if more processing precision or shorter throughput is needed. Additional parameterization is specified by switches C_2 and C_3 and dotted arcs. If the right half is present (switch C_1 is closed) and if switch C_2 is closed, then the related multiplication must be performed after the addition operation. Otherwise, no particular execution order is enforced. Switches C_2 and C_3 control the parallelism of the PSFG, which is useful for DP selection for different amounts of hardware resources.

PSFG of $\Delta\Sigma$ ADC (Figure 2.1) incorporates integrators (blocks I_i), gain stages (blocks a_k and t_{ij}), summing blocks, a quantizer (comparator), and a block for digital-to-analog conversion (DAC). Arcs define the signal flow between blocks. The *AIS* set includes input U with attributes such as the value range and bandwidth. The *AOS* set includes the output V with attributes such as the rate (frequency) and bit width. Parameterization is achieved by switches S_1 and S_2, that add or remove feedback links in the PSFG. Typical embedded control applications may require ADCs with a variable conversion accuracy (e.g., going from 6 bits to 12 bits), an adjustable peak signal to noise ratio - SNR (SNR_{max}) (such as in the range 45dB to 90dB), and are capable of converting signals in a bandwidth ranging from 50 kHz to 1 mHz.

This section explains (1) DP selection for performance in continuous ranges, (2) DP

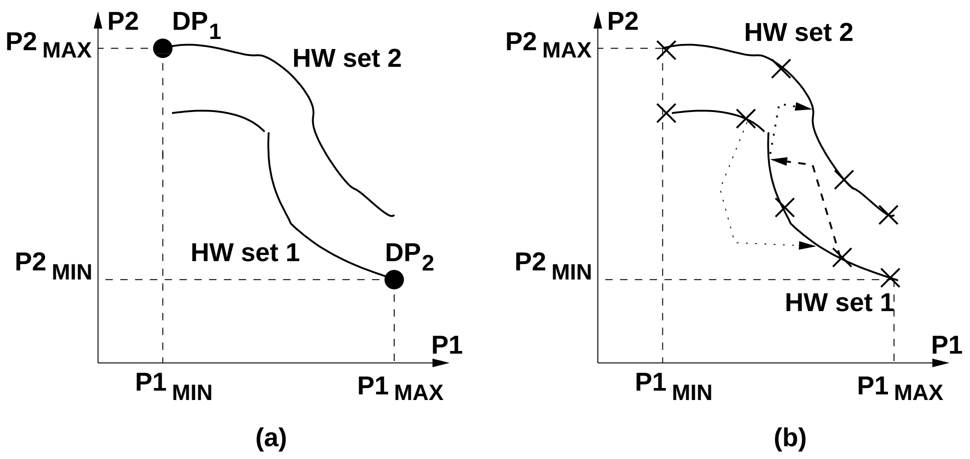


Figure 4.5: DP selection for ranges

selection for performance in discrete sets, and (3) the algorithm for system adaptation.

Variable performance is expressed as closed ranges $[P_{min}, P_{max}]$, or as discrete sets $\{P_{min}, P_2, ...P_{max}\}$. Figure 3.7(a) shows the resulting *performance requirements region* (PRR), if only two range-valued requirements are considered, e.g., bandwidth and dynamic range (DR). This concept can be easily extended for more metrics, like SNR_{max}, bit width, and power consumption.

Definition: Assuming that the design point DP_k is characterized by the bandwidth BW_k and the dynamic range DR_k. The point DP_k *covers* the rectangular PRR formed by the points $< bw, DR >$, where $bw < BW_k$ and $DR < DR_k$, since all possible requirements *bw* - *DR* are satisfied by the design point DP_k. The *hardware cost* of a point DP_k is equal to the number of reconfigurable blocks used to implement the point.

Figure 3.7(a) shows the PRR covered by the point DP_1.

Definition: A design point DP_k is called a *basic* design point, if its covered PRR is not contained in the PRR of another design point, or the union of the PRRs of a set of points with lower, or equal, hardware cost as the point DP_k.

The set of basic design points with the same hardware cost define the Pareto curve for that hardware cost. For example, Figure 3.7(b) shows the Pareto curve defined by the set $\{DP_1, DP_2, DP_3\}$ of basic points. Point DP_4 is not basic, unless its hardware cost is lower than that of point DP_2.

DP selection requires sampling the Pareto surfaces of design points so that the required performance regions are covered using minimal hardware resources. Each Pareto surface in the performance space $P_1 \times P_2 \times ... \times P_k$ consists of DPs synthesized for a fixed set of hardware resources. For example, in Figure 5.7(a), the lower curve is for the resource set `HW set 1`, and the upper curve for a different resource set `HW set 2`.

DP selection for continuous ranged performance. The execution trace of the DP selection algorithm is shown in Figure 5.7(b). It analyses the Pareto surfaces in increasing order of their hardware resource sets. Initially, N DPs are selected, so that the horizontal axis (performance $P1$) is sampled as equally as possible (e.g., the distance between consecutive points is as close as possible to $\frac{P1_{MAX} - P1_{MIN}}{N}$). N is the total number of sampling points divided by the number of hardware resource sets. The initially selected DPs are marked in the figure with 'X'. Then, for each pair of consecutive DPs, DP_i and DP_{i+1}, the following value is computed:

$$OD_{i+1} = \frac{P2(DP_{i+1}) - P2(DP_i)}{P1(DP_{i+1}) - P1(DP_i)} + \frac{P2(DP_{i+1}) - P2(DP_{i+2})}{P1(DP_{i+1}) - P1(DP_{i+2})} \quad (4.4)$$

A large OD value indicates that DP sampling is not closely tracking the DP. With the largest OD is removed, and a new DP is sampled at a position that gives the largest OD value. Figure 5.7(b) shows the "moving" of two DPs to positions with larger OD value. DP re-shuffling ends, if no more improvement is achieved through the algorithm. Then, the DP selection step is repeated for the other Pareto surfaces too, i.e. the surface for resource set `HW set 2`. After finding the DPs separately sampling each Pareto surface, the algorithm continues by re-shuffling DPs across surfaces using a cost metric that reflects the unique PRR coverage offered by a DP per its unit hardware cost.

Only non-basic DPs are considered. For each DP_i, value HC is computed, where HC is the ratio of the area covered by DP_i but not covered by DPs of the same or lower hardware cost, and the hardware cost of DP_i. Value HC reflects the unique PRR coverage obtained per unit hardware cost. DPs with low HC offer little unique covering at the expense of using additional hardware (as compared to DPs on lower surfaces). The DP with the lowest HC is removed, and instead a DP is inserted in the place that would give the highest HC. The algorithm stops, if the cost does not decrease further. Figure 5.7(b) illustrates the moving of a DP from the lower curve to the higher curve.

DP selection for discrete valued performance. This situation is simpler than the previous case of DP selection, since performance ranges do not have to be sampled. DP selection of $\Delta\Sigma$ ADCs used as an illustrative example for the algorithm. The method is presented in Figure 3.9.

The first step finds the minimum-order $\Delta\Sigma$ modulator topology with the least number of feedback and feed-forward paths. The topology is synthesized for the least demanding performance requirements of the PRR, the minimum bandwidth BW_{min} and the minimum dynamic range DR_{min}. The oversampling ratio is fixed to the maximum value OSR_{max}. The optimal modulator topology is found for the requirements $< bw_{min}, DR_{min} >$ by solving the MINLP equations [44]. The resulting topology is post-optimized using a simulated annealing (SA) algorithm that attempts to improve the DR of the modulator by performing localized changes of the topology coefficients.

Assuming that the resulting design point DP_1 is characterized by the performance attributes bw_1 and DR_1, the point DP_1 is the first basic point DP found by the method. The

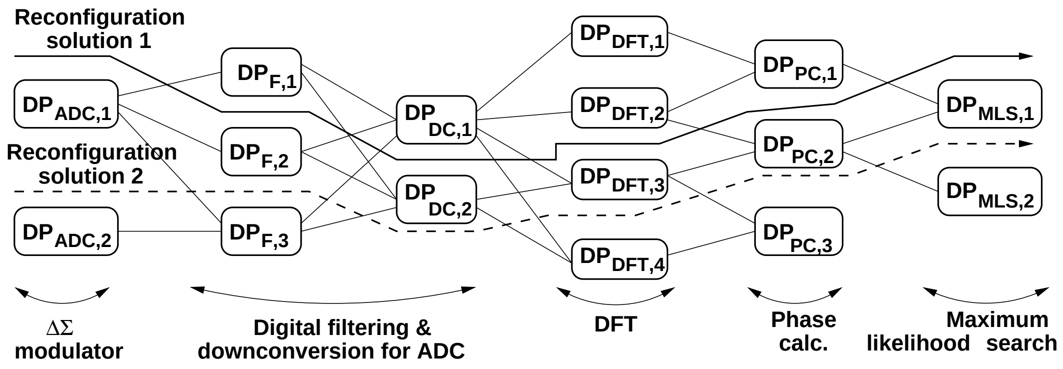


Figure 4.6: Design point compatibility graph

next step finds more basic design points with the same hardware cost as the point DP_1. Therefore, the order of the modulator is kept the same as that of the point DP_1, but the OSR is modified. Lower OSRs are considered next. For example, if the modulator for DP_1 uses $OSR = 128$, then the methodology is based on $OSR = 64$, then $OSR = 32$, and so on. Repeating the steps that produced the topology for DP_1, the topologies for the successively decreasing OSR values are synthesized. Denoting these design points as DP_2, DP_3, ..., where the point DP_i corresponds to a larger OSR value than the point DP_{i+1}, means that the topology for the point DP_i offers a higher DR, but converts signals of lower bandwidth.

The methodology continues by considering ΔΣ modulators of order higher by one. This corresponds to increasing the hardware cost of the design points by one SC block. Then, as in the previous step, the OSR is set to the maximum value, and the optimal modulator topology is synthesized. More design points are produced for successively decreasing OSR values, as in the previous case. This generates a new Pareto curve.

System adaptation. As explained in Section 4.2, system adaptation uses the centralized control scheme in Figure 4.2. The adaptation algorithm identifies one DP for each block, such that all DPs can operate together to provide the requested performance variation. Hence, a DP selected for a block must be *compatible* with the DPs found for all blocks connected to it. Otherwise, the system malfunctions, e.g., loses data, exceeds timing constraints, violates precision requirements, etc. For example, for an ADC, consider a DP with a high dynamic rate (DR), that is obtained by using a very high sampling frequency for the ADC quantizer. If the throughput of the connected digital blocks (e.g., digital filters) is too large, this DP results in data loss at the interface between the ADC and the digital blocks.

DP compatibility is checked using following two rules. (i) If output i of $RBlock_1$ is connected to input j of $RBlock_2$, the two blocks are compatible, if for all attributes A_k of signal i, the corresponding attributes A_m of signal j are superior (e.g., for frequency bandwidth, $A_k \subset A_m$). (ii) All DPs selected simultaneously for system reconfiguration must not produce infeasible solutions, such as exceeding the total number of available reconfigurable blocks and I/O pins and ports, using clock frequencies/supply voltages that are hard to generate simultaneously.

After selecting DPs for each system block, the reconfiguration methodology sets up the system's Design Point Compatibility Graph (DPCG). There is a node for each DP. Two DPs are connected by an arc, if their corresponding blocks are linked in the system, and the DPs are compatible. Figure 4.6 shows the DPCG for sound-based tracking.

The adaptation algorithm uses the DPCG to compute new reconfiguration solutions for the system "on-the-fly". For a given optimization constraint (such as number of used reconfigurable blocks), finding a new reconfiguration solution means finding the minimum cost path in the DPCG, where cost is defined by the optimization goal. This is computed using Dijkstra's algorithm.

4.4 Case Study

This case study refers to a reconfigurable implementation of sound-based tracking in Figure 1.1. The goal was to select DPs and to set-up the corresponding DPCG that would mcover the PRR defined by the bandwidth range BW = [150kHz, 1.2mHz] and the DR range DR = [50dB, 90dB] (8 bits to 16 bits precision). In addition, the topologies should cover the targeted PRR using the minimum number of reconfigurable analog and digital blocks.

A. DP selection for $\Delta\Sigma$ ADCs. This methodology was used to find the topologies that cover the defined PRR using a minimum amount of configurable analog blocks. SA-based, post-optimization was conducted to adjust the coefficients in the presence of leakage, finite DC gain, slew rate, saturation, jitter, and colored noise. First, a set of second-order modulators was produced using the MINLP-based topology synthesis procedure [45]. Topologies were generated for the OSR values: 128, 64, 32, and 16. Figure 3.11 presents the stair-case Pareto curve that resulted for the second order topologies, and the PRR covered by these. The bullets correspond to basic design points. The corresponding OSR is shown in brackets. The architecture for OSR = 16 did not cover any portion of the PRR, since its DR = 47 is below the lower limit of the PRR, the topology was eliminated from the set. All second order topologies used two configurable blocks for implementation. Then, the modulator order was increased to three, and then to four. Two new stair-case Pareto curves were produced as shown in Figure 3.11. All third-order modulators require three configurable blocks, and all fourth-order modulators use four blocks. One third-order and two-fourth order topologies were eliminated from the set because their covered PRRs were already covered by lower cost topologies.

The remaining 8 topologies offer a large covering of the PRR. The uncovered region

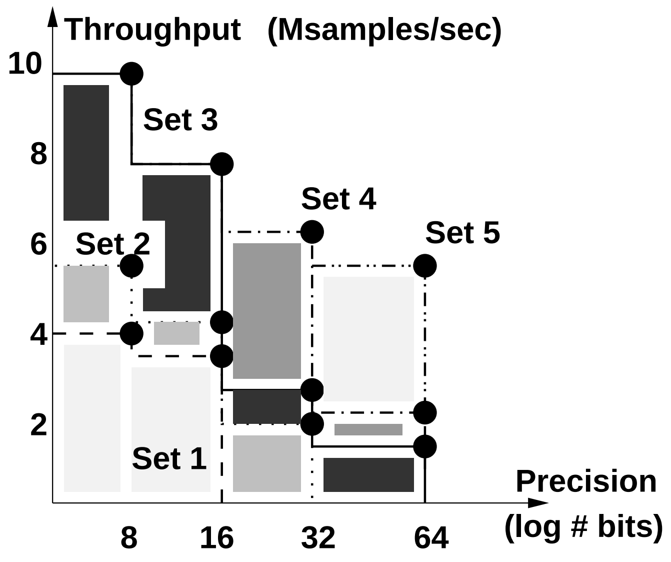


Figure 4.7: DPs for DFT

corresponds to very demanding specifications that cannot be met by modulators of order lesser than 5 and with OSR up to 128. Note that only a small fraction (14.5%) of the PRR requires modulators of fourth-order. Assuming a uniform probability for having all specifications in the PRR, the proposed methodology saves, in 17% of the cases, one configurable block, and in about 40% of the cases two configurable blocks, as compared to the straightforward solution (that implements one fourth-order modulator for the worst case). Considering that all blocks have similar power consumption, this also results in a reduction in power consumption going by approximately 50%, in 40% of the cases, and by 25%, in 17% of the situations.

B. DP selection for DFT. Figure 4.7 shows the DPs selected for the DFT in Figure 4.4. Five hardware resource sets were considered, each set being twice the size of the previous one. Set 1 is the smallest (1200 LUTs and flip-flops). For input rates less than $3 msamples/sec$ DPs between Set 4 and Set 5 should not be used. This reduces the hardware cost by a factor of approximately 4x, as compared to the worst case design (Set 5).

A simplified DPCG is shown in Figure 4.6.

4.5 Conclusions

This chapter presented a methodology for designing cost-effective, dynamically reconfigurable mixed-domain systems for metadata processing. During operation, the system switches between different designs for the analog and digital blocks depending on the actual performance needs. The chapter also proposed algorithms for DP selection and system adaptation. A case study for sound-based tracking showed that compared to worst-case design, reconfigurable topologies use fewer analog and digital blocks, and can lower power consumption in ADCs by 25%-50%.

Chapter 5

Online AMS Frontend Reconfiguration for Sensor Network Applications

5.1 Introduction

Cyber-physical systems (CPS) are envisioned to offer continuous signal sensing over broad physical areas, intelligent data processing, and comprehensive, broad-range actuation [59]. This is essential for important domains, such as infrastructure management, healthcare, environmental monitoring, and manufacturing control. After being deployed, CPS must operate reliably under a large variety of environmental conditions (e.g., temperature, noise) and performance requirements (such as bandwidth, precision, signal range, and energy/power consumption).

Networks of reconfigurable processors incorporating analog and mixed-signal (AMS) frontends appear to be a very attractive implementation platform for CPS. Reconfigurable processors can be customized online and in real time for changing requirements, and thus continuously provide performance-effective operation [54, 45]. The alternative, customized integrated circuit (IC) implementations, must be designed for the worst-case requirements. This leads to wasted resources and hence higher costs because most of the time the actual requirements are significantly milder than the worst-case conditions. Reconfigurable processors can easily support new functions, which can also help in reducing the implementation costs and reliability of the designs. Besides, complex circuits, such as high-order $\Delta\Sigma$ ADCs, are more unstable than simple structures [47]. This limits the range of the input signals that can be processed. It is more effective to dynamically switch during execution among implementations optimized for different performance needs instead of using complex, worst-case IC designs. On the negative side, reconfigurable analog and digital blocks are valuable resources that should be utilized with care [38, 40].

Design automation methods for reconfigurable digital architectures have been extensively studied, including automated placement and routing, logic and high-level synthesis, and hardware/software co-design [60]. More recently, techniques for systematic design of reconfigurable analog-to-digital converters (ADCs) have been proposed [45]. The method generates reconfigurable $\Delta\Sigma$ ADC topologies for a pre-defined set of specifications by identifying the minimal changes that can adapt the architecture to new requirements. Techniques for automated synthesis of AMS architectures have been proposed for filters and ADCs [42, 43, 44]. Various reconfigurable ADC ICs are discussed in [35, 37]. In contrast to existing work, CPS emphasize the need for flexible design, including online AMS architecture reconfiguration, as the specific requirements might be known only during operation.

This chapter presents a synthesis methods for automatically selecting the design points used for online reconfiguration of AMS architectures. The work considers that the performance requirements are unknown at design time. Instead, performance attributes can have with equal probability all values in a range. This chapter proposes criteria and algorithms for selecting design point sets that keep the obtained performance close to the requirements. The cardinality of the sets is limited so that all designs can be stored in the local memory of an AMS architecture. This chapter also defines the main characteristics of the envisioned architecture for implementing reconfigurable AMS frontends. The architecture includes both fine and coarse-grained reconfiguration in an attempt to offer both flexibility and low reconfiguration overhead. The proposed synthesis method performs two steps: (i) it synthesizes the Pareto points for different performance trade-offs, and then (ii) selects a pre-defined number of points that can cover the equally probable performance requirements in a range. The proposed synthesis procedure is performed offline because the temporal overhead required to synthesize the design points is too excessive. However, the best candidate in a set of given requirements is selected online.

This chapter proposes a new synthesis framework and metrics for selecting the designs used in dynamic reconfiguration. The concepts can be extended for other scenarios too, e.g., different performance requirements distributions (other than uniform). Also, the method is general and can be used for selecting design points for both analog and digital reconfigurable modules.

This chapter has five sections. Section 2 discusses the reconfigurable architecture. Section 3 presents the steps for finding the design points used in dynamic reconfiguration. Section 4 focuses on experiments. Finally, conclusions are offered.

5.2 Reconfigurable Mixed Analog-Digital Architecture

The suggested reconfigurable architecture comprises of a coarse-grained, reconfigurable

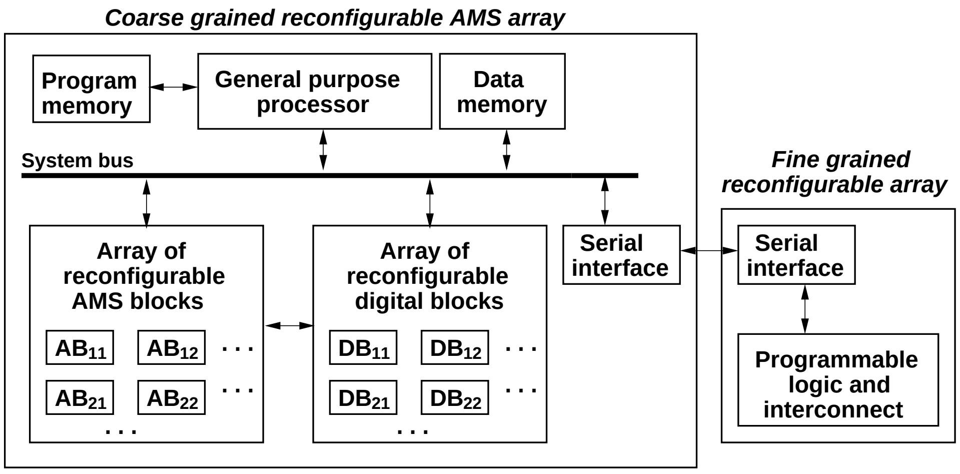

Coarse grained reconfigurable AMS array
Program memory
General purpose processor
Data memory
System bus
Array of reconfigurable AMS blocks
AB11
AB12
AB21
AB22
Array of reconfigurable digital blocks
DB11
DB12
DB21
DB22
Serial interface
Fine grained reconfigurable array
Serial interface
Programmable logic and interconnect

Figure 5.1: Coarse-grained/fine-grained reconfigurable AMS architecture

analog and mixed-signal (AMS) array interconnected by a serial interface to a fine-grained, reconfigurable digital array. Figure 5.1 illustrates the structure. Coarse-grained reconfiguration is at the module level. Fine-grained reconfiguration is at the gate level. Coarse-grained reconfiguration is more performance efficient than fine-grained reconfiguration as it utilizes less hardware overhead for reconfiguration. This is important mainly for AMS circuits, for which the extra switches and multiplexers used for reconfiguration can significantly reduce the speed and bandwidth of the circuits. However, coarse-grained reconfiguration is less flexible in implementing new functionality. The structure can be realized by interfacing a reconfigurable AMS array, like PSoC [40], with an FPGA through a serial interface, i.e. SPI or UART.

The AMS architecture operates as follows: (a) the architecture continuously monitors the real performance of the composing modules, such as data acquisition, processing, networking, and actuation, and compares it to the required performance, (b) the required performance is specified by the application, and can change dynamically as a result of events. For example, detecting a certain condition of interest might change the speed constraint of the implementation, or modify the precision requirement of data acquisition and processing. If there is significant difference between the required and real performance, then the resource management module detects which modules have to be reconfigured in order to minimize the imbalance between the two requirements. Reconfiguration can involve multiple blocks of the coarse-grained and fine-grained reconfigurable arrays.

The coarse-grained AMS array is significantly based on the PSoC mixed-signal architecture manufactured by Cypress [38, 40]. However, it was appropriate to simplify the structure of the reconfigurable analog cells, and increase the number of reconfigurable input connections of a cell, so that system topologies with more complex structures can be also synthesized, e.g., higher order $\Delta\Sigma$ ADC. The analog subsystem consists mainly of a bi-dimensional array of reconfigurable switched-capacitor blocks. The blocks are organized into m rows and n columns. Even though there is no theoretical limit for the number of rows and columns, the numbers of rows and columns are actually not very high due to the lower integration densities that are achievable for reconfigurable AMS circuits as opposed to the reconfigurable digital ICs. For example, the PSoC architecture includes, at most, three rows and four columns of reconfigurable analog blocks [40]. This makes the programmable analog cells a very valuable resource that should be used carefully in a design. Minimizing the number of reconfigurable analog cells used for implementation should be a primary design requirement. The internal structure of the reconfigurable analog cells is shown in Figure 3.3. The structure is similar to that of type `C` reconfigurable PSoC blocks [40], with the exception of the following two differences: (i) the number of programmable inputs connected to the summing node of the OpAmp is higher, and (ii) the less used circuits were eliminated, e.g., the comparator (available in each PSoC block) and the circuitry for producing interrupts. The first modification is needed for the implementation of complex high-order $\Delta\Sigma$ modulators. The second change is justified by the reconfigurable analog blocks being used mainly to implementation filters and $\Delta\Sigma$ ADCs rather than other analog circuits. More details on the architecture are presented in [62].

The functionality of the SC blocks is programmed by reconfiguring the topology of the network surrounding the OpAmp. Similar to the PSoC blocks, the programmable cell can be configured to operate as a comparator, a gain stage, or an integrator. If the capacitor array Cap_1 is not connected then the circuit functions as a comparator, otherwise, the circuit is either a gain stage or an integrator. In addition to the capacitor array Cap_1, each reconfigurable block has several other programmable capacitor arrays called Cap_2, Cap_3, ... and Cap_{tot} in the figure. Array Cap_2 is used for the input to the programmable cell. The rest of the capacitor arrays implement various feedback and feed-forward paths. The gain of the signal paths is set by programming the corresponding capacitor arrays. The value of constant `tot` is set by the maximum number of paths that can converge in a summing node. This value depends on the order of $\Delta\Sigma$ for the ADCs that are implemented. For systems of order up to five, inclusive, the value of constant `tot` is six. The values of the capacitor arrays can be selected from the set $\{0, \frac{1}{r}, \frac{2}{r}, \ldots, 1\}$. The value of `r` depends on the application characteristics. Section V shows that `r = 16` is sufficient for realizing $\Delta\Sigma$ ADCs of orders two and three, but `r = 128` is needed for fourth order ADCs. PSoC uses `r = 32` [40].

The reconfigurable analog blocks can be connected into various topologies (netlists) through the programmable interconnect. The interconnect structure implements different feedback and feed-forward connections, such as those in $\Delta\Sigma$ ADCs [47, 44]. The output of each block can be connected either to the longer row and column interconnects, to implement the global feed-forward or feedback structures, or to the shorter local interconnect

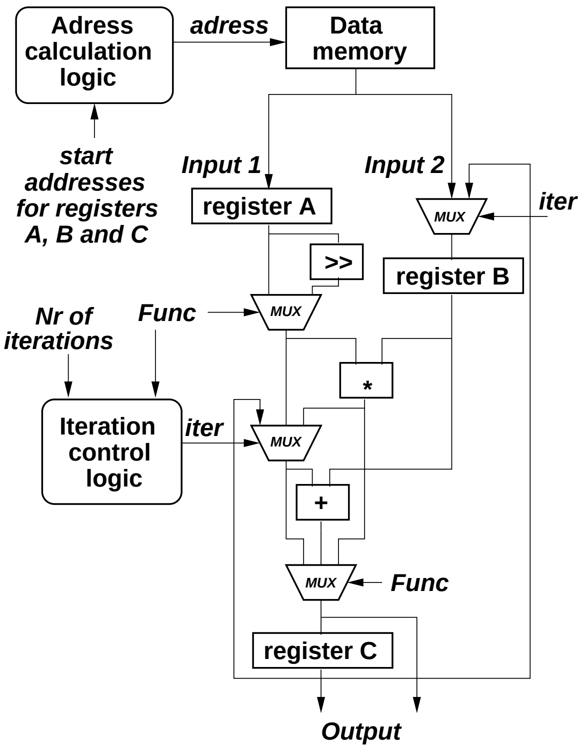


Figure 5.2: Coarse grained reconfigurable digital block

between neighboring cells. In contrast, the PSoC architecture includes only column and local interconnects [40], which might be somewhat restrictive for implementing higher-order topologies with complex feed-forward and feedback structures. Such architectures offer superior DR and lower sensitivity to circuit nonidealities than architectures with few feedback structures [44].

Figure 5.2 presents the structure of the proposed coarse-grained reconfigurable digital blocks. The data path can perform repetitive arithmetic operations (e.g., additions, subtractions, and multiplications), as well as, shift operations. Input operands are stored in registers *A* and *B*. Register *C* stores partial results, such as partial sums. The reconfigurable blocks are utilized using the following set of steps: (i) the data path is configured by programming the control registers of the block. This defines the logic for computing the

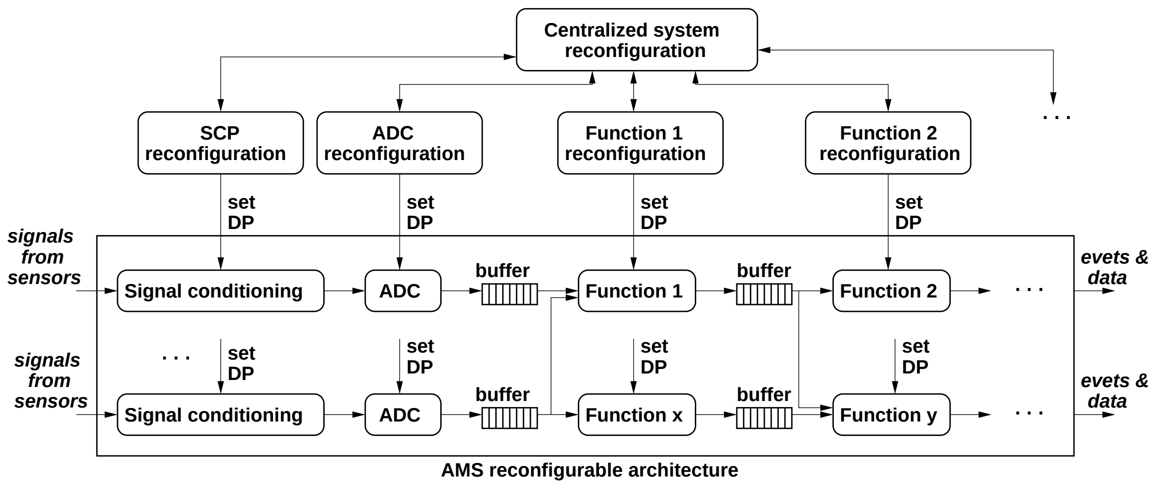


Figure 5.3: Synthesis model for overall reconfiguration of AMS system

signals *Func* that control the multiplexer circuits for reconfiguration, and the signals *iter* that control repetitive execution. This step also defines the logic for calculating the memory addresses needed to load the operands into registers *A*, *B*, and *C*. (ii) Each block reads data directly from the RAM memory by indicating the start addresses of the operands for registers *A* and *B* , as well as, the starting address for storing the results in register *C*. (iii) For iterative algorithms, each iteration starts by loading the operands from RAM, performs the computations of the data path, and updates the addresses for accessing the next set of operands from RAM.

Figure 5.4 presents the overall synthesis model for AMS reconfiguration of frontends. An AMS system includes a number of converging, and diverging, signal paths originating at the primary inputs of the system, and producing actuation signals. The analog domain includes signal conditioning paths (CPTs) and analog-to-digital conversion (ADCs). The digital domain is composed of DSP functions interconnected through data buffers, as shown in the figure. Each analog and digital module uses a specific implementation called design point (DP). DPs can be changed dynamically at execution by reconfiguring the modules.

All DPs implement the same functionality, but DPs differ by their performance and resource requirements. The reconfiguration of analog and digital modules to new DPs is controlled by the central resource management routine. The set of possible DPs is fixed for each module.

System adaptation is based on the centralized resource management scheme in Figure 5.4. Requests for system adaptation are formulated globally, if the system performance needs to be changed (e.g., system throughput, input rate, overall hardware cost, power consumption, etc.), or locally, if the performance of a block needs to be modified. The adaptation scheme selects one DP for each block in the data acquisition-processing-actuation flow, such that all DPs can operate together to provide the requested performance. A DP selected for a block must be compatible with the DPs of all blocks connected to it. Otherwise, the system malfunctions, e.g., loses data, exceeds timing constraints, violates precision requirements, and so on. For example, an ADC's DP might offer a high dynamic rate (DR) by using a very high sampling frequency. If the throughput of the connected digital blocks (e.g., digital filters) is too low, this DP results in data loss at the interface between the ADC and the digital blocks. This affects the precision of the overall processing.

5.3 Synthesis of Reconfigurable AMS

For synthesis, the frontend functionality is expressed using Parameterized Signal Flow Graphs (*PSFG*). PSFG is composed of processing blocks (e.g., adders, multipliers, integrators, comparators, etc.) and switches, that are connected with each other by signals. A closed switch propagates a signal, while an open switch propagates a neutral value (e.g., zero for digital hardware or high impedance for analog circuits). Figure 5.6 illustrates two PSFGs, *RBlock* 1 is the PSFG for a ΔΣ ADC function, and *RBlock* 2 is the PSFG for the

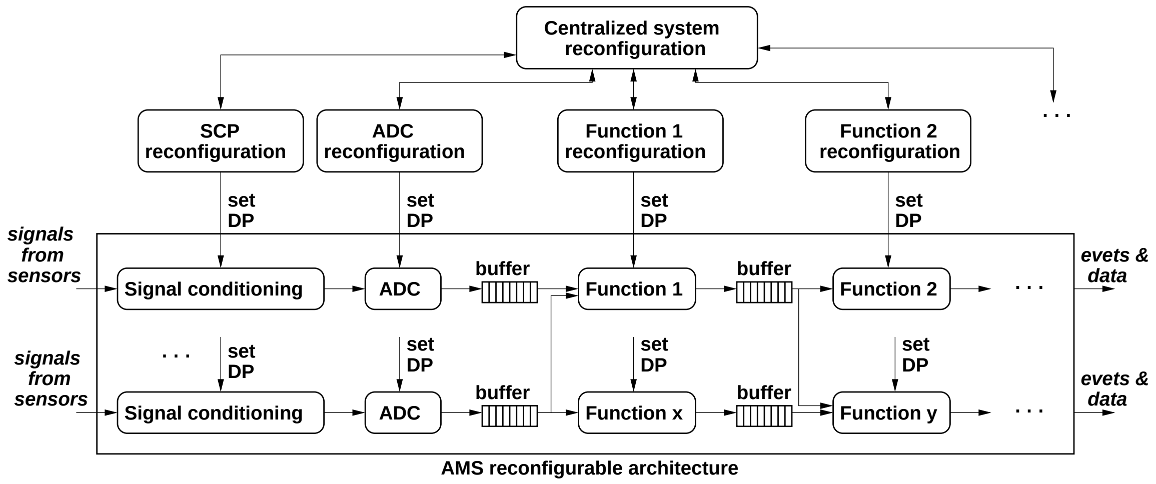


Figure 5.4: Synthesis model for overall reconfiguration of AMS system

DFT (Discrete Fourier Transform) algorithm. The PSFG of the $\Delta\Sigma$ ADC incorporates integrators (blocks I_i), gain stages (blocks a_k and t_{ij}), summing blocks, quantizer (comparator), and a block for digital-to-analog conversion (DAC). The arcs define the signal flow between blocks. Parameterization is accomplished by the use of switches S_1, S_2, ..., S_N that change the transfer function of ADC by adding, or removing, feedback paths. The parameterization of the DFT PSFG is based on switches, of the type controlled by signal C_1. The switches control the amount of SFG parallelism depending on the throughput requirement. For example, if the switch C_1 is open, then the two clusters (marked by a dotted line) must be executed in parallel. Otherwise, if the switch is closed, then the clusters are executed in sequence, but without enforcing any particular execution order on the clusters. Additional parameterization can be inserted by using more switches.

Figure 5.5 illustrates The online reconfiguration procedure is as follows. Assuming that the performance requirement P_1 changes over time as shown with bold line in the figure, this change represents a long-term, identifiable trend. In addition, short-term performance “fluctuations” are indicated by the dashed lines. The long-term change is addressed by

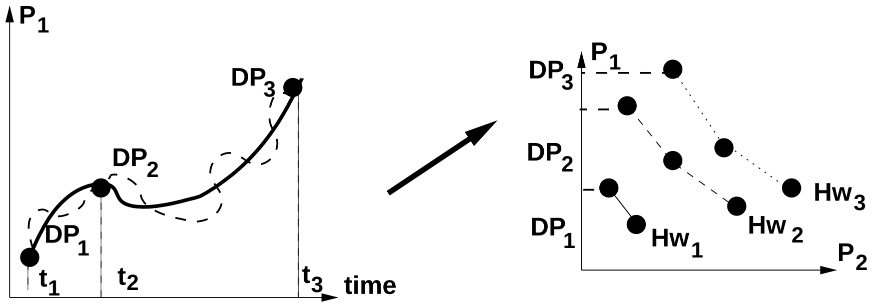


Figure 5.5: Dynamic reconfiguration

reconfiguring the architecture using implementations DP_1, DP_2, and DP_3, such as in DP_1 is used between the time moments t_1 and t_2, and so on. The three design points are design solutions that use different hardware resource sets, and implement different performance trade-offs. Short-term performance changes are addressed by changing the parameters of the design points, but without reconfiguring the hardware, such as modifying the sampling frequency of ADCs, while keeping the same topology.

Design point synthesis, for reconfiguration, samples the Pareto surfaces of the frontend modules to select DPs that cover as much as possible of the possible performance regions of an application. The analyzed Pareto surfaces include DPs that are possible using the resources of the reconfigurable AMS architecture. The number of selected DPs is less than a predefined limit that depends on the on-chip memory available for storing the DPs. Each Pareto surface for a system block in the performance space $P_1 \times P_2 \times ... \times P_k$ consists of DPs synthesized for a fixed set of hardware resources. For example, in Figure 5.7(a), the lower curve is for the resource set `HW set 1`, and the upper curve for a different resource set `HW set 2`. The DP selection problem can be formulated as follows:

$$\min \int_{\Delta T} [\alpha \sum_i \texttt{Hardware cost}_{DP[Block_i]}(\texttt{t}) + \beta \sum_k (\texttt{Performance}_k(\texttt{t}) - \texttt{Requirement}_k(\texttt{t}))^2] \texttt{dt}$$

$$\texttt{subject to:} \quad \sum_i \texttt{Hardware cost}_{DP[Block_i]}(\texttt{t}_j) \leq \texttt{Available resources}, \ \forall \texttt{t}_j \in \Delta\texttt{T}$$

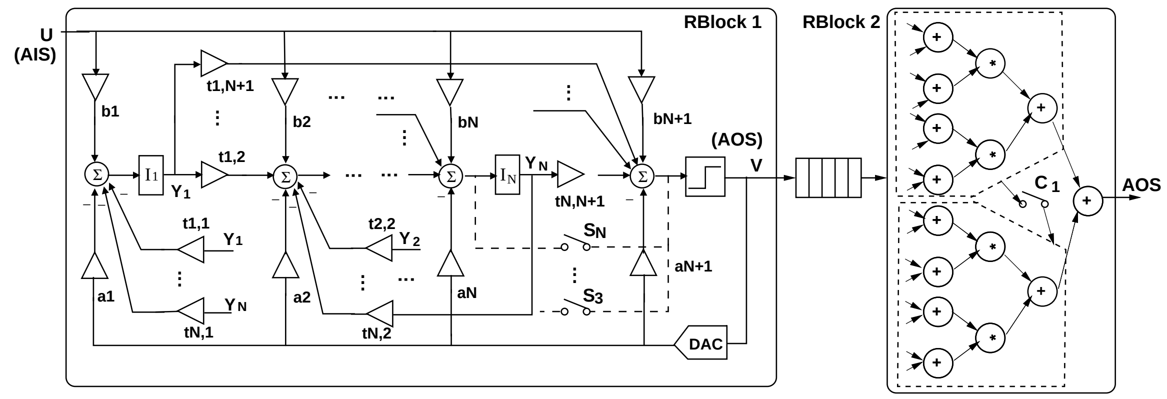


Figure 5.6: Parameterized Signal Flow Graphs

$$\texttt{subject to:}\quad \texttt{Cardinality(DP[Block}_i\texttt{])<N}_i\texttt{, }\forall\texttt{i}$$

$$\texttt{subject to:}\quad \texttt{Compatible(DP[Block}_i\texttt{],DP[Block}_j\texttt{]), }\forall\texttt{i,j}$$

The unknowns are variables $DP[Block_i]$ representing the DPs of the AMS frontend blocks. The optimization goal is to minimize a weighted sum in which the first term is the reconfigurable hardware used over time (thus, the hardware cost of the DPs selected to cover the variable performance requirements of each $Block_i$), and the second term enforces the requirement that the performance of the selected DPs match the requirements of the application as close as possible over time ΔT .

The first constraint indicates that the DPs selected at each time instance t_j require less hardware than the available resources. The next requirement states that the number of DPs selected for $Block_i$ should be less than a predefined limit N_i due to the limited on-chip memory of the AMS architecture. Finally, the DPs of the blocks in the overall implementation must be compatible with each other.

The above minimization problem is solved off-line due to its high computational requirements. The equation set is hard to solve as the functions are discontinuous and discrete. For example, functions $Hardware\ cost_{DP[Block_i]}(t)$ and $Compatible$ are discrete, and functions $Performance_k(t)$ is discontinuous. The function $Requirement_k(t)$ has different expressions depending on the nature of the application.

For CPS, we can identify three situations: (i) the most general case is in which no assumptions are made about a performance requirement P_k besides the fact that it is bounded to the range $[P_{(k,min)}, P_{(k,max)}]$. (ii) A more specific case is the one in which the values of the performance requirement P_k are uniformly distributed in the range $[P_{(k,min)}, P_{(k,max)}]$. (iii) Finally, the performance requirement values for P_k are in the range $[P_{(k,min)}, P_{(k,max)}]$, and follow a known distribution p_k. This chapter focuses on the second case. The first case is discussed in [62]. Ongoing work studies the third case.

The proposed optimization heuristics selects the set of DPs for reconfiguration by applying the following steps: First, for each block in the AMS frontend, the Pareto curves are built for increasing amounts of hardware resources. The Pareto curves correspond to the performance trade-offs involved in the application description. At this point, the complete design space of the reconfigurable system is captured. The design space is too large to be stored entirely in the on-chip memory of the AMS system, so it must be pruned to satisfy the cardinality constraint of the optimization problem. Hence, the two main steps of the synthesis procedure are: (1) DP sampling, and (2) DP pruning. The steps are discussed next.

1. DP sampling. Assuming that the values for performance requirement P_k are uniformly distributed in range $[P_{(k,min)}, P_{(k,max)}]$ the goal, for a given hardware cost, is to sample N DPs ($N > N_i$), such that the performance requirements are met as close as possible. Furthermore, assuming that the two DPs in Figure 5.7(a), DP_s and DP_{s+1}, two reconfiguration policies can be used : Use DP_s until the performance requirement for $P1$ becomes

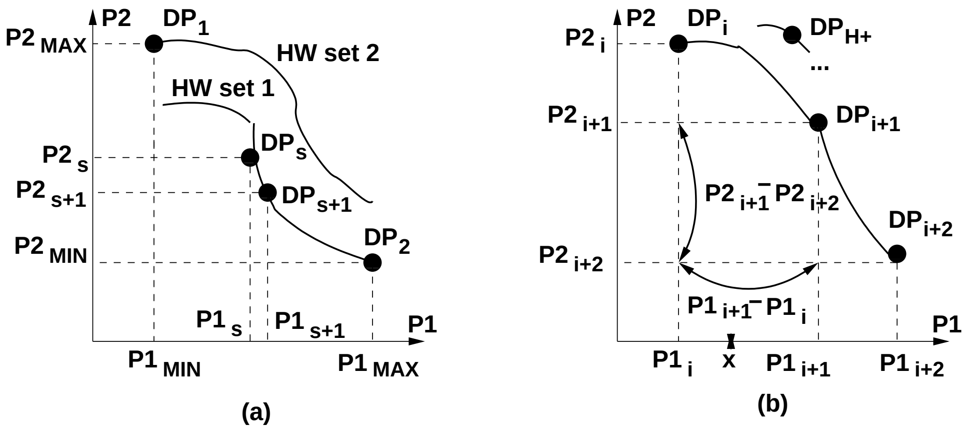


Figure 5.7: Design point selection

$P1_{DP_{s+1}}$, then switch to DP_{s+1}. This policy covers performance $P2$ but does not cover performance $P1$. The second policy uses DP_{s+1} after the requirement for $P1$ exceeds $P1_{DP_s}$. This policy meets the constraint for performance $P1$, but it does not cover performance $P2$. Thus, both policies result in performance loss (either $P1$ or $P2$) that can be estimated using the same reasoning. Consider the second policy. Assuming a uniform probability of having a performance request in the range $[P2_{s+1}, P2_s]$, the total performance loss is equal to

$$Loss_{P2}(P2_s, P2_{s+1}) = \int_{|P2_s - P2_{s+1}|} t\ dt = \frac{(P2_s - P2_{s+1})^2}{2}$$

The total loss for the N sampled points is

$$Total\ Loss_{P2} = \sum_{s \in N\ sampled\ points} \frac{(P2_s - P2_{s+1})^2}{2}$$

The total performance loss for performance $P2$ is minimized, if the N points sample equally the range $[P2_{MIN}, P2_{MAX}]$, thus the distance between two consecutive DPs is $\frac{DP2_{MAX} - DP2_{MIN}}{N-1}$. A similar reasoning identifies the optimal DP sampling for the first adaptation policy.

The DP sampling method is as follows: (a) for increasing amounts of hardware resources, the algorithm builds the Pareto curves for each system block by synthesizing the blocks for different performance trade-offs and (b) N-equally spaced design points are then sampled using the value $\sum w_i P_i$ as a distance metric where w_i is the weight assigned to performance P_i.

2. DP pruning. The pruning step eliminates DPs to reduce the cardinality of the sampled DP set to the predefined limit N_i, so that the performance loss is minimal. The pruning step first analyzes only DPs that use the same amount of hardware resources.

Figure 5.7(b) shows the removing of DP_{i+1} from the set of sampled DPs. As a result, only DP_i and DP_{i+2} are left to cover the performance ranges $[DP1_i, DP1_{i+2}]$ and $[DP2_{i+2}, DP2_i]$. If DP_{i+1} is removed then the following strategy is the best policy for dynamic reconfiguration: (i) utilize DP_{i+2} for the entire range, if $w_2(P2_i - P2_{i+2}) < w_1(P1_{i+1} - P1_i)$. Otherwise, use DP_i for the entire range. For this policy, the performance loss is $P_{Loss} = \min(P1_{i+1} - P1_i, P2_{i+1} - P2_{i+2})$. w_1 and w_2 are the weights for the two performance attributes. This reconfiguration policy is justified as follows: Assume that both DP_i and DP_{i+2} cover the ranges of performance $P1$ and $P2$. Also, assume that DP_i is used for the range $[P1_i, P1_i + x]$, and DP_{i+2} for the rest. Finding the optimal adaptation policy requires identifying the value of x, so that the performance loss due to removing DP_{i+1} is minimal. Lets assume that point x is positioned as shown in Figure 5.7(b). The performance loss is $w_1 x(P1_{i+1} - P1_i) + w_2(1-x)(P2_{i+1} - P2_{i+2}) = w_2(P2_{i+1} - P2_{i+2}) + x[w_1(P1_{i+1} - P1_i) - w_2(P2_{i+1} - P2_{i+2})]$. If $w_1(P1_{i+1} - P1_i) > w_2(P2_{i+1} - P2_{i+2})$, then the loss increases with the value of x, hence $x = 0$ minimizes the performance loss. In this case, DP_{i+2} is used for the entire range. If $w_1(P1_{i+1} - P1_i) < w_2(P2_{i+1} - P2_{i+2})$, then x should be maximum, hence DP_i is used for the entire range. For this policy, the performance loss is $P_{Loss} = \min(P1_{i+1} - P1_i, P2_{i+1} - P2_{i+2})$.

The performance loss through removing DP_{i+1} can be improved by analyzing DPs of higher hardware cost, such as DP_{H+} in Figure 5.7(b). In this case, the performance loss is $P_{Loss} = \min(P1_{i+1} - P1_i, P2_{i+1} - P2_{i+2}, P1_{i+1} - P1_{H+})$, however, more hardware resources are utilized.

Pruning a design point DP_i from the set of sampled DPs might result in pruning more DPs. For example, if DP_i is the only compatible point to DP_s, then the second DP should be also eliminated. Hence, the performance loss due to pruning DP_i is computed as $P_{Loss} = \sum_j \min(P1_{j+1} - P1_j, P2_{j+1} - P2_{j+2})$, where DP_j are all DPs eliminated, including DP_i. The pruning step eliminates $N - N_i$ DPs in increasing order of the resulting performance loss P_{Loss}.

5.4 Experiments

The ongoing work focuses on the design of a highly reconfigurable sensing node for CPS applications. Sound-based localization and classification are among the main functionalities assigned to a node. The signal processing method for sound-based localization is based on the algorithms presented in [29]. The classification algorithm is based on neural networks.

ADCs are the main building blocks of the sound-based localization frontend. The proposed DP selection methodology was utilized to identify a number of reconfigurable design points that would cover possible performance requirements, if vehicles are tracked. AMS frontend reconfiguration is useful in improving the cost-quality tradeoff of signal processing by customizing the ADC parameters to the characteristics of the application requirements and audio signature of the vehicle. For example, the bandwidth and precision requirements of the ADC can change dynamically depending on the importance of the tracked vehicle.

Table 5.1: DP selection for $\Delta\Sigma$ modulators: Case 1

Initial #	Final #	Step I	Step II	Step III	Rel.cov.(%)
10	3	62712	32395	32998	52
10	5	62712	50875	51961	82
10	7	62712	58303	59411	94
20	4	68729	42856	44858	65
20	6	68729	54903	55712	81
20	8	68729	60453	61261	89
20	10	68729	64220	64519	93
30	3	69478	30165	32998	47
30	6	69478	55013	55938	80
30	9	69478	62631	63134	90
30	12	69478	65512	66057	95
30	15	69478	67561	67915	97

Table 5.1 presents the performance of one DPs selecting case for the reconfigurable $\Delta\Sigma$ ADCs used for the frontend. This example uses 1 block, 3 hardware sets and 2 performance attribute. Initially, the Pareto sets for $\Delta\Sigma$ ADCs were produced for different bandwidth - dynamic range tradeoffs. Different first, second, and third order modulator topologies were synthesized using the method presented in [44]. Then, a number of equally-spaced DPs was selected as indicated by the DP sampling criterion in Section 5.3. Experiments were run for different number of initial points, as shown in Column one in Table 5.1. Column three shows the resulting performance coverage (in Hz $\times$ dB) of the selected DPs. Using more initial DPs obviously improves the obtained performance coverage. The final number of DPs is shown in Column two of the table. The number was varied from 10%

Table 5.2: DP selection for ΔΣ modulators: Case 2

Initial #	Final #	Step I	Step II	Step III	Rel.cov.(%)
10	3	32920	17495	17812	54
10	5	32920	26914	27594	84
10	7	32920	30713	31443	96
20	4	36026	23068	23511	65
20	6	36026	29513	29703	82
20	8	36026	32070	32338	90
20	10	36026	33834	34086	95
30	3	36423	16550	17812	49
30	6	36423	29591	29703	81
30	9	36423	33135	33413	92
30	12	36423	34585	34776	95
30	15	36423	35459	35658	98

- 50 % of the initial points. Column four indicates the performance coverage after the DP pruning step. The coverage can be improved by slightly re-positioning the remaining DPs. Column five presents the resulting coverage after this step. Finally, Column six shows the relative coverage after DP re-positioning as compared to the coverage offered by the much larger set of initial DPs. The column indicates that a coverage above 90% of the initial coverage can be obtained, if the reconfigurable node has sufficient on-chip memory to store the configuration information for ten DPs. Beyond that limit, the extra coverage provided by using more DPs is minimal. Table 5.2 shows the performance of DPs selecting case for a similar reconfigurable ΔΣ ADC example uses 1 block, 3 hardware sets and 1 performance attributes. The execution time of the algorithm was less than 0.016

seconds, for all examples. The experiments proved that by using the proposed algorithm significantly fewer points can still cover almost all of the design.

Figure 5.8 shows the DPs selected for a classifier circuit implemented using FPGAs. Device reconfiguration is performed through the selection of Pareto-optimal DPs from a design point curve. The curve contains DPs from both the single and variable bit width designs. The points correspond to different classification precision - cost tradeoffs. Note that the DP that contains 11,700 slices is not a Pareto point and can be eliminated.

5.5 Conclusions

This chapter presents a synthesis method for automatically selecting the design points used for online reconfiguration of AMS architectures. This is important for implementing the sensing frontends necessary for CPS. This work assumes that the performance requirements are unknown at the design time but have equal occurrence probability within a range. The proposed algorithm selects design points so that their overall performance remains close to any possible requirement in the range. The number of selected design points is constrained by the size of the on-chip memory for storing the points. The novel contributions of the work include the proposed synthesis framework and metrics for selecting the designs used in dynamic reconfiguration. The method is general and can be applied to both analog and digital reconfigurable modules. Experiments showed that selecting only ten DPs offers a good coverage of the possible bandwidth - precision requirements for the frontend ADCs in sound based tracking applications.

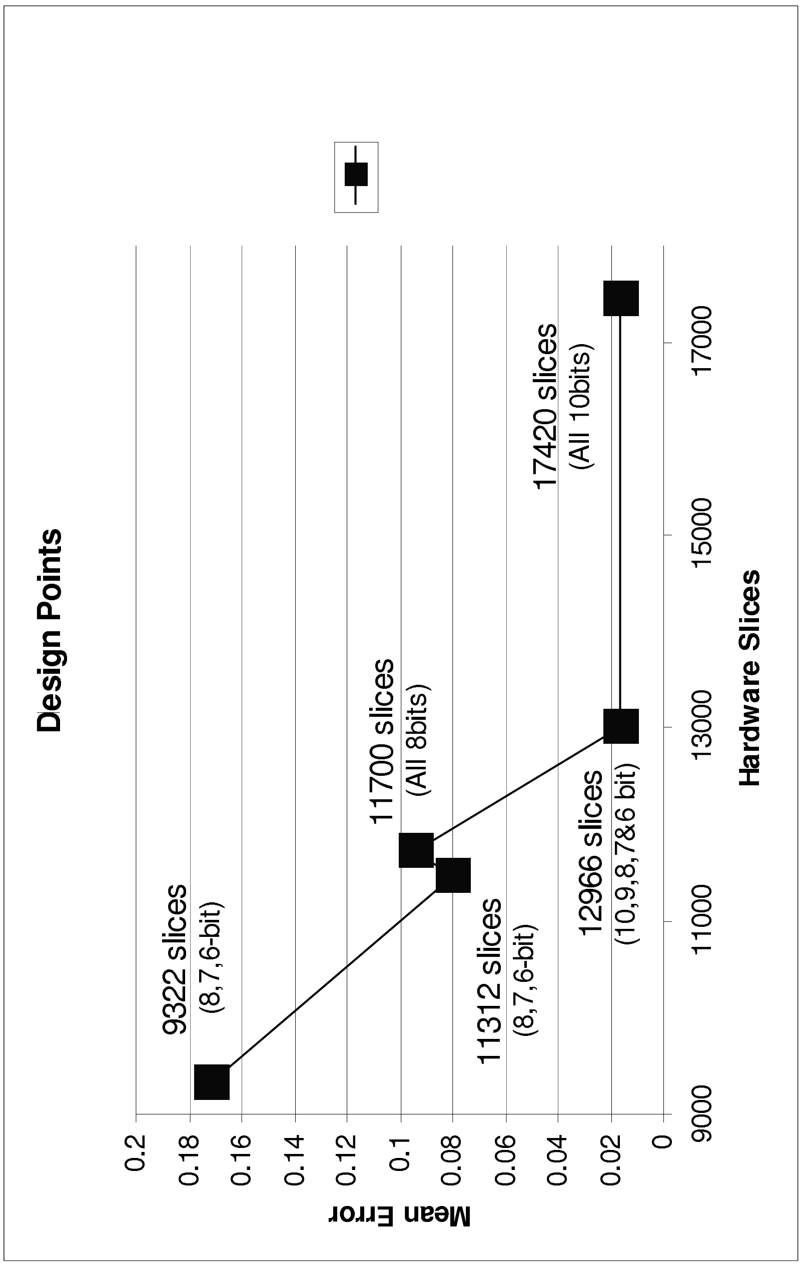


Figure 5.8: Design points for classifier circuit

Chapter 6

Conclusions

6.1 Conclusions

This thesis presents a systematic methodology for designing reconfigurable, discrete-time, $\Delta\Sigma$ modulator topologies. Topologies are optimized to minimize the complexity of the topologies, maximize the topology robustness with respect to circuit nonidealities, and minimize the total power consumption from the set of all possible topologies expressed by a generic topology.

The methodology is based on the concept of generic topology that expresses all possible feed-forward and feedback signal paths in a reconfigurable topology. The modeling of generic topologies including following nonidealities: integrator leakage and gain error, circuit noise, and circuit nonlinearity was presented. Models were used in the methodology to establish a mixed integer-nonlinear (MINLP) equation set. Equations were solved for finding optimized reconfigurable $\Delta\Sigma$ topologies. Found topologies were then refined using a *Simulink* simulation of the models with more detailed nonidealities.

A case study for designing topologies for a three-mode reconfigurable $\Delta\Sigma$ modulator showed that the proposed methodology requires less design effort, reduces power dissipation and complexity while providing more robustness than traditional methods. A reconfigurable topology implementation on a Programmable System-on-Chip (PSoC) device was also presented. By implementing a dual-mode reconfigurable DS modulator topology using PSoC reconfigurable mixed-signal SoC, the correctness of the proposed methodology has been demonstrated. Measurement results show that the modulator can meet the design specifications.

This thesis also presents a systematic methodology for producing reconfigurable $\Delta\Sigma$ modulator topologies with optimized flexibility in meeting variable performance specifications. To increase their flexibility, topologies are optimized for performance attributes pertaining to ranges of values, rather than being single values. Topologies are implemented on switched-capacitor, reconfigurable, mixed-signal architectures. As configurable blocks are very valuable resources, it is extremely important that the topologies use as few blocks as possible according to design specifications rather than just worst case design. A case study illustrates the methodology for specifications from telecommunications area. One switch capacitor block and 25% power consumption are saved in 17% of the cases, and two switch capacitor blocks and 50% power consumption are saved in 40% of the cases.

This thesis presents a synthesis method for automatically selecting the design points used for online reconfiguration of AMS architectures. This is important for implementing the sensing frontends necessary for CPS. The work assumes that the performance requirements are unknown at the design time but have equal occurrence probability within a range. The proposed algorithm selects design points so that their overall performance remains close to any possible requirement in the range. The number of selected design points is constrained by the size of the on-chip memory for storing the points. The novel contributions of the work include the proposed synthesis framework and metrics for selecting the designs used in dynamic reconfiguration. The method is general and can be applied to both analog and digital reconfigurable modules. Experiments showed that selecting only ten DPs offers a good coverage of the possible bandwidth - precision requirements for the frontend ADCs in sound based tracking applications.

Bibliography

[1] A. Doboli, and P. Eles. ”Scheduling under control dependencies for heterogeneous architectures”, *Proc. of the International Conference on Computer Design*, 1998, pp. 602-608.

[2] P. Eles, K. Kuchcinski, Z. Peng, and A. Doboli, ”System Level Hardware/Software Partitioning based on simulated annealing and tabu search”, *Design Automation for Embedded Systems*, 2, Kluwer Academic Publishers, 1997, pp. 5-22.

[3] P. Eles, A. Doboli, P. Pop, and Z. Peng, ”Scheduling with bus access optimization for distributed embedded systems”, *IEEE Transaction on VLSI*, Vol. 8, No. 5, pp. 472-491, October 2000.

[4] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al Hashimi, ”Quasi-static voltage scaling for energy minimization with time constraints”, *Proc. of the Design, Automation and Test in Europe Conference*, 2005.

[5] J. Henkel, ”A low power hardware/software partitioning approach for core-based embedded systems”, *Proc. of the Design Automation Conference*, 1999, pp. 122-127.

[6] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwareins, ”Energy-aware runtime scheduling for embedded multiprocessor socs”, *IEEE Design & Test of Computers*, 2001.

[7] T. Blickle, J. Teich, and L. Thiele, ”System-level synthesis using evolutionary algorithms”, *Journal of Design Automation for Embedded Systems*, 1998.

[8] R. Dick, and N. Jha, ”Mogac: A multiobjective genetic algorithm for the co-synthesis of hardware-software embedded systems”, *Proc. of the International Conference on Computer-Aided Design*, 1997.

[9] B. Dave, G. Lakshminarayana, and N. Jha, ”Cosyn: Hardware-software co-synthesis of heterogeneous distributed embedded systems”, *IEEE Transactions on Computer-Aided Design*, 1999.

[10] R. Ernst, ”Codesign of embedded systems: Status and trends”, *IEEE Design & Test*, 1998.

[11] S. Ishiwata and et al, ”A single-chip mpeg-2 codec based on customizable media embedded processor”, *IEEE Journal of Solid State Circuits*, 2003.

[12] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. Chase, ” Correlating instrumentation to system states: A building block for automated diagnosis and control”, *6th Symposium on Operating systems Design and Implementation (OSDI)*, 2004.

[13] Y. Diao, J. Hellerstein, S. Parekh, and J. Bigus, ”Managing web server performance with autonomous agents”, *IBM System Journal*, 2003.

[14] F. Gomez, D. Burger, and R. Miikkulainen, ”A neuroevolution method for dynamic resource allocation on a chip multiprocessor”, *Proc. of International Joint Conference on Neural Networks*, 2001.

[15] J. Wildstrom, P. Stone, E. Witchel, R. Mooney, and M. Dahlin, ”Towards self-configuring hardware for distributed computer systems”, *Proc. of the International Conference on Automatic Computing*, 2005.

[16] C. Hughes, J. Srinivasan, and S. Adve, ”Saving energy with architectural and frequency adaptations”, *Proc. of the Annual International Symposium on Microarchitecutre*, 2001.

[17] S. Kallakuri and A. Doboli, ”Energy conscious on-line archiecture adaptation for varying constraints in sensor network applications”, *Proc. of International Symposium on System Synthesis and CODES*, 2005.

[18] N. Thepayasuwan, S. Kallakuri, A. Doboli, and S. Doboli, ”Communication subsystem synthesis and analysis tool using bus architecture generation and stochastic arbitration policies”, *Proc. of IEEE Symposium on Circuits and Systems (ISCAS)*, 2005.

[19] S. Kallakuri, N. Thepayasuwan, A. Doboli, and E. Feinberg, ”A continuous time narkov decision process based system on chip buffer allocation methodology”, *Proc. of IEEE GLSVLSI Conference*, 2005.

[20] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets, ”Design and evaluation of a cross-layer adaptation framework for a mobile multimedia system”, *Proc. of SPIE/ACM Multimedia Computing and Networking Conference*, 2003.

[21] D. Sachs, S. Adve, and D. Jones, ”Cross-layer adaptive video coding to reduce energy on general-purpose processors”, *Proc. of IEEE International Conference on Image Processing*, 2003.

[22] C. Poellabauer, H. Abbasi, and K. Schwan, ”Cooperative run-time management of adaptive applications and distributed resources”, *Proc. of ACM Multimedia*, 2002.

[23] H. Zeng, C. Ellis, A. Lebeck, and A. Vahdat, ”Ecosystem: Managing energy as a first class operating system resource”, *Proc. of ASPLOS*, 2002.

[24] B. Noble, M. Satyanarayanan, D. Narayunan, J. Tilton, J. Flinn, and K. Walker, ”Agile application-aware adaptation for mobility”, *Proc. of the 16th ACM Symposium on Operating System Principles*, 1997.

[25] W. Yuan and K. Nahrstedt, ”Process group management in cross-layer adaptation”, *Proc. of the SPIE/ACM Multimedia Computing and Networking Conference*, 2004.

[26] R. Sasanka, C. Hughes, and S. Adve, ”Joint local and global hardware adaptations for energy”, *Proc. of ASPLOS Conference*, 2002.

[27] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, ”Combined voltage scaling and adaptive body biasing for low power microprocessor under dynamic workloads”, *Proc. of the International Conference on Computer-Aided Design*, pages 721-725, 2002.

[28] T. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, and V. Eruhimov, ”Computer vision workload analysis: Case study of video surveillance systems”, *Intel Technology Journal*, 9(2):109-118, 2005.

[29] D. Halupka, N. J. Mathai, P. Aarabi, A. Sheikholeslami, “Robust Sound Localization in 0.18μm CMOS”, *IEEE Trans. Signal Processing*, Vol. 53, No. 6, pp. 2243-2250, 2005.

[30] Y. Weng and A. Doboli, ”Smart sensor architecture customized for image processing applications”, *Proc. of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004.

[31] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, “Wireless Sensor Networks for Habitat Monitoring”, *Proc. ACM International Worshop on Wireless Sensor Network Applications*, 2002.

[32] Y. Weng, S. Kallakuri, A. Doboli, S. Hong, and T. Robertazzi, ”Dynamic architecture adaptation to improve scalability of sensor networks: A case study for a smart sensor for face recognition”, *Proc. of Real Time System Symposium*, 2004.

[33] S. Ganesan, R. Vemuri, “Technology Mapping and Retargeting for Field-Programmable Analog Arrays”, *Proc. DATE*, pp. 58-65, 2000.

[34] H. Wang, S. Vrudhula, “Behavioral Synthesis of Field Programmable Analog Array Circuits”, *ACM TODAES*, Vol. 7, Issue 4, pp. 563-604, 2002.

[35] K. Gulati, H.-S. Lee, “A Low-Power Reconfigurable Analog-to-Digital Converter”, *Journal Solid-State Circuits*, Vol. 36, No. 12, pp. 1900–1911, Dec. 2001.

[36] M. Miller, C. Petrie , “A Multibit Sigma-Delta ADC for multimode Receivers”, *Journal Solid-State Circuits*, Vol. 38, No. 3, pp. 475–482, March 2003.

[37] R. Van Veldhoven, “A Triple-Mode Continuous-Time ΣΔ Modulator With Switched-Capacitor Feedback DAC for a GSM-EDGE/CDMA2000/UMTS Receiver”, *JSSC*, Vol. 38, No. 12, pp. 2069–2076, Dec. 2003.

[38] M. Mar, E. Blom, “An Architecture for a Configurable Mixed-Signal Device”, *Journal Solid-State Circuits*, Vol. 38, No. 3, pp. 565-567, 2003.

[39] R. S. Woodd-Walker, J. L. Watkinsa, A. S. Brierleyb, “Identification of Southern Ocean acoustic targets using aggregation backscatter and shape characteristics”, *ICES Journal of Marine Science*, Vol. 60, No. 3, pp. 641-649, 2003.

[40] “PSoC Mixed Signal Array”, *Document No. PSoC TRM 1.21*, Cypress Semiconductor Corporation, 2005.

[41] “DelSig8 v3.2, 8 Bit Delta Sigma ADC”, *Application Note*, Cypress Semiconductor Corporation, Oct. 3 2005.

[42] K. Francken, G. Gielen, “A High-Level Simulation and Synthesis Environment for ΔΣ Modulator”, *IEEE Trans. CADICS*, Vol. 22, No. 8, pp. 1049–1061, 2003.

[43] F. Medeiro, A. Perez-Verdu, A. Rodriguez-Vazquez, “Top-Down Design of High-Performance Sigma-Delta Modulators”, *Kluwer*, 1999.

[44] H. Tang, A. Doboli, “High-Level Synthesis of ΔΣ Modulator Topologies Optimized for Complexity, Sensitivity and Power Consumption”, *IEEE Trans. CADICS*, Vol. 25, No. 3, pp. 597-607, 2006.

[45] Y. Wei, A. Doboli, “Systematic Methodology for Designing Reconfigurable ΔΣ Modulator Topologies for Multimode Communication Systems”, *Proc. DATE*, 2006.

[46] R. Schreier, "The Delta-Sigma Toolbox 6.0", http://www.mathworks.com/matlabcentral/fileexchange, Dec. 2004.

[47] S. Norsworthy, R. Schreier, G. Temes, "Delta-Sigma Data Converters. Theory, Design, and Simulation", *IEEE Press*, 1997.

[48] R. Fletcher, "www-neos.mcs.anl.gov/neos / solver / MINCO:MINLP - AMPL".

[49] P. Malcovati, et. al., "Behavioral Modeling of Switched-Capacitor Sigma-Delta Modulators", *IEEE Trans. C. & S. - I*, Vol. 50, No. 3, pp. 352-364, 2003.

[50] H. Tang, Y. Wei, and A. Doboli, "MINLP Based Topology Synthesis for Delta-Sigma Modulators Optimized for Signal Path Complexity, Sensitivity and Power Consumption", Proc. DATE, pp. 264–269, 2005.

[51] G. Gielen, K. Francken, E. Martens, and M. Vogels, "An Analytical Integration Method for the Simulation of Continuous-Time $\Delta\Sigma$ Modulators", *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 23, No. 3, pp. 389–399, March 2004.

[52] R. Schreier, "The Delta-Sigma Toolbox 5.2", www.mathworks.com/support/ftp/controlssv5.shtml, Nov. 1999.

[53] http://www-neos.mcs.anl.gov/neos/solvers/minco:MINLP /AMPL.html

[54] S. Hauck, T.W. Fry, J. Kao, "The Chimaera Reconfigurable Functional Unit", *Proc. IEEE Symposium on FCCM*, pp. 87-96, 1997.

[55] B. Miramond, J. Delsome, "Design Space Exploration of Dynamically Reconfigurable Architectures", *Proc. DATE*, pp. 366-371, 2005.

[56] M. Writhlin, “Disc: The Dynamic Instruction Set Computer. FPGAs for Fast Board Developement and Reconfigurable Computing”, *Proc. SPIE*, pages 92-103, 1995.

[57] G. Stitt, F. Vahid, “Hardware/Software Partitioning of Software Binaries”, *Proc. ICCAD*, 2002, pp. 164-170.

[58] G. Stitt, R. Lysecky, F. Vahid, “Dynamic Hardware/Software Partitioning: A First Approach”, *Proc. DAC*, 2003, pp. 250-255.

[59] E. Lee, “Cyber physical systems: Design challenges”, *Technical Report, No. UCB/EECS-2008-8, University of California at Berkeley*, 2008.

[60] S. Hauck, A. Dehon, “Reconfigurable Computing”, *Morgan Kaufmann*, 2008.

[61] Y. Wei, P. Sun, A. Doboli, ”Systematic Methodology for Reconfigurable Switched-Capacitor Delta Sigma Modulator Design”, *IEEE International SOC Conference (SOCC’06)*, 2006.

[62] P. Sun, Y. Wei, A. Doboli, “Flexibility-oriented Design Methodology for Reconfigurable Delta Sigma Modulators”, *Proc. DATE Conference*, 2007.

[63] P. Sun, Y. Zhao, M. Gilberti, A. Doboli, D. Curiac, D. Pescaru, ”Dynamic Reconfiguration of Mixed-Domain Embedded Systems for Applications with Variable Performance Requirements”, *Adaptive Hardware and Systems (AHS)*, 2008.

[64] P. Sun, C. Ferent, M. Gilberti, A. Doboli, ”Online AMS Frontend Reconfiguration for Sensor Network Applications”, *European Conference on Circuit Theory and Design (ECCTD’09)*, 2009.