

Copyright © 1995, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**AUTOMATIC SYNTHESIS OF CMOS
ALGORITHMIC ANALOG-TO-DIGITAL
CONVERTER**

by

Gani Jusuf

Memorandum No. UCB/ERL M95/27

21 April 1995

**AUTOMATIC SYNTHESIS OF CMOS
ALGORITHMIC ANALOG-TO-DIGITAL
CONVERTER**

by

Gani Jusuf

Memorandum No. UCB/ERL M95/27

21 April 1995

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

**Automatic Synthesis of CMOS Algorithmic
Analog-to-Digital Converter**

Copyright 1993

by

Gani Jusuf

Automatic Synthesis of CMOS Algorithmic Analog-to-Digital Converter

by

Gani Jusuf

Doctor of Philosophy in
Electrical Engineering and Computer Sciences
University of California at Berkeley

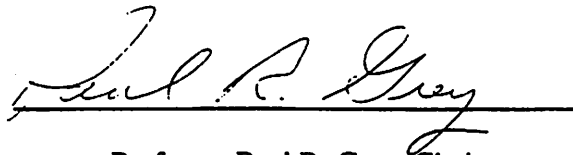
ABSTRACT

The steady decrease in technological feature size is allowing increasing levels of integration in analog/digital interface functions. These functions consist of analog as well as digital circuits. While the turn around time for an all digital IC chip is very short due to the maturity of digital IC computer-aided design (CAD) tools over the last ten years, most analog circuits have to be designed manually due to the lack of analog IC CAD tools. As a result, analog circuit design becomes the bottleneck in the design of mixed signal processing chips. One common analog function in a mixed signal processing chip is an analog-to-digital conversion (ADC) function. This function recurs frequently but with varying performance requirements. The objective of this research is to study the design methodology of a compilation program capable of synthesizing ADC's with a broad range of sampling rates and resolution, and silicon area and performance comparable with the manual approach.

The automatic compilation of the ADC function is a difficult problem mainly because ADC techniques span such a wide spectrum of performance, with radically different implementations being optimum for different ranges of conversion range, resolution, and power dissipation. We will show that a proper choice of the ADC architectures and the incorporation of many analog circuit design techniques will simplify the synthesis proce-

dure tremendously. Moreover, in order to speed up the device sizing, hierarchical optimization procedure and behavioral simulation are implemented into the ADC module generation steps.

As a result of this study, a new improved algorithmic ADC without the need of high precision comparators has been developed. This type of ADC lends itself to automatic generation due to its modularity, simplicity, small area consumption, moderate speed, low power dissipation, and single parameter trim capability that can be added at high resolution. Furthermore, a performance-driven CMOS ADC module generator, CADICS based on design rules and spice parameters has been developed. CADICS takes a set of input files and generates the complete ADC netlist, layout, and performance summary. A prototype of the automatically generated ADC has also been fabricated and tested.

A handwritten signature in cursive script, reading "Paul R. Gray", is written over a solid horizontal line.

Professor Paul R. Gray, Chair

**To my mother
and Caroline**

Acknowledgements

No words could describe my appreciation for the support, guidance, encouragement, patience, and understanding given by Professor Paul R. Gray throughout this work and my years at Berkeley. I would also like to thank my other advisor, Professor Alberto Sangiovanni-Vincentelli for his valuable advice in both research and academic matters.

I would gratefully acknowledge all of my fellow students who have made my many years at Berkeley more enjoyable. From the old generation of graduate students, Steve Lewis, Sehat Sutarja, Beomsup Kim, and Yuh-Min Lin provided me with a lot of support when I started the research. Of the current generation, I have to thank Timothy Hu, Greg Uehara, Cormac Conroy, Dave Cline, Weijie Yun, Ken Nishimura, and Robert Neff for their help in maintaining a good balance between work and leisure. Many thanks have to be given to Ed Liu, Henry Chang, Umakanta Choudhury, Enrico Malavasi, and Eduardo Charbon for their support throughout this project. There are so many more people to thank without them this research would not have been possible.

This research was supported by SRC under contract number SRC-92-DC-008 and DARPA under contract number N00039-87-C-0182. Their support is gratefully acknowledged.

Finally, I would like to thank my wife, Caroline for encouraging and giving me moral support for all these years. My greatest thanks of all has to be given to my mother, without her I won't be where I am today.

TABLE OF CONTENTS

CHAPTER 1	Introduction	1
1.1	Background and Motivation	1
1.2	Integrated Circuit Design Automation	2
1.3	Thesis Contribution	6
1.4	Thesis Organization	7
CHAPTER 2	Automatic Generation of Analog Blocks	9
2.1	Introduction	9
2.2	Why Do Module Generation?	10
2.3	Existing Analog Module Generators	13
2.3.1	Operational-Amplifier Module Generator.	13
2.3.2	Switched-Capacitor Filter Module Generator	14
2.4	Analog-To-Digital Conversion Module Generator	17
2.4.1	Overall Goals	17
2.4.2	Functionality	19
2.4.3	Characterization	19
2.4.3.1	Differential Non-Linearity (DNL)	23
2.4.3.2	Integral Non-Linearity (INL)	23
2.4.3.3	Gain Error	25
2.4.3.4	Offset Error	25
2.4.4	Signal-To-Noise Ratio	26

TABLE OF CONTENTS

- 2.4.5 Previous Work 28
 - 2.4.5.1 Analog Cell Library 28
 - 2.4.5.2 Performance-Driven Based on Standard Cells 28
- 2.4.6 Methods of Attack 30
- 2.5 Summary 31

CHAPTER 3 ADC Architectures Suitable for Automatic Compilation 33

- 3.1 Introduction 33
- 3.2 A/D Conversion Architectures 34
 - 3.2.1 Serial A/D Conversion Technique 34
 - 3.2.2 Successive Approximation A/D Conversion Technique 34
 - 3.2.3 Parallel A/D Conversion Technique 37
 - 3.2.4 Pipelined A/D Conversion Technique 38
 - 3.2.5 Oversampled A/D Conversion Technique 40
- 3.3 Architecture Comparison 42
- 3.4 Desired Properties 45
- 3.5 An Improved Algorithmic A/D Converter 46
 - 3.5.1 Algorithmic A/D Converter 47
 - 3.5.2 Speed Improvement Solution 50
 - 3.5.3 Comparator Offset Solution 55
 - 3.5.3.1 Two-Comparator Scheme and Digital Error Correction 57
 - 3.5.4 Proposed 1-Bit/Cycle Algorithmic A/D Converter 62
- 3.6 Summary 64

CHAPTER 4 ADC Circuit Generation 67

- 4.1 Introduction 67
- 4.2 Architectural Selection 67
- 4.3 Circuit Synthesis 71
 - 4.3.1 Optimization Approaches 72
 - 4.3.1.1 Global Optimization Approach 72
 - 4.3.1.2 Hierarchical Optimization Approach 72
- 4.4 Implementation 75
 - 4.4.1 Inputs and Outputs 75
 - 4.4.2 Optimization 77
 - 4.4.3 Netlist Module Generator 83
 - 4.4.3.1 Mapping Function 84

TABLE OF CONTENTS

4.4.3.2	Generating Operational Amplifiers	91
4.4.3.3	Generating Comparators	92
4.4.3.4	Generating Switch and Capacitor Sizes	93
4.4.3.5	Generating Digital Circuitry	93
4.4.4	Evaluator	93
4.4.4.1	Behavioral Simulator	94
4.4.4.2	Floorplan Optimization	100
4.4.5	Scheduler	101
4.4.6	Quality of the circuit synthesis	101
4.5	Summary	102
CHAPTER 5	ADC Layout Generation	105
5.1	Introduction	105
5.2	Approach	105
5.2.1	Flat Layout Procedure	106
5.2.2	Hierarchical Constraint-Driven Layout	107
5.2.2.1	Circuit Partitioning	108
5.2.2.2	Constraint Generation	108
5.2.2.3	Device Generation, Placement, and Routing	110
5.3	Implementation	110
5.3.1	Circuit Partitioning and Floorplanning	110
5.3.1.1	Structured-Netlist	112
5.3.2	Floorplanning	112
5.3.2.1	Slicing Structure	113
5.3.2.2	Layout Style	115
5.3.3	Floorplan Optimization	117
5.3.3.1	Device Structure and Shape	117
5.3.3.2	Modified Stockmeyer's Algorithm	120
5.3.3.3	Channel Area Estimation	127
5.3.4	Data Structure	129
5.3.5	Routing	129
5.4	Summary	132
CHAPTER 6	Examples and Test Results	133
6.1	Introduction	133
6.2	Examples	133
6.2.1	Manually Designed Example	133
6.2.2	Automatically Generated Examples	135
6.3	Test Set-up	141
6.4	Test Results	143

TABLE OF CONTENTS

6.5	Summary	147
CHAPTER 7	Conclusion	149
7.1	Summary of Research Results	149
7.1.1	An Improved Algorithmic A/D Conversion Architecture	149
7.1.2	ADC Module Generation	150
7.2	Future Work	153
BIBLIOGRAPHY		155
APPENDIX A : Building Blocks		163
A.1	Introduction	163
A.2	Sample and Hold Block	163
A.2.1	S/H Circuit	163
A.2.2	Switching Sequence	167
A.2.3	Transfer Function	168
A.2.4	Noise Source	170
A.2.5	Switch Size Calculation	172
A.2.6	Amplifier	181
A.3	Gain of Two Block	181
A.3.1	2X Circuit	181
A.3.2	Switching Scheme	182
A.3.3	Transfer Function	183
A.3.4	Noise Source	185
A.4	Amplifier	187
A.4.1	Amplifier Specifications	187
A.4.2	Basic Op-amp Topology	192
A.4.3	Common Mode Feedback Circuit	197
A.4.4	Biasing Circuit	198
A.4.5	Auto-Zero Circuit	200
A.4.6	Noise Reduction	202
A.5	Comparator Blocks	203
A.6	Digital Circuits	205
A.7	Trim Array	207

CHAPTER 1 Introduction

1.1 Background and Motivation

Over the last two decades technology feature size has decreased allowing increasing levels of integration in analog/digital (A/D) interface functions. These functions are usually mixed analog-digital integrated circuits containing both analog and digital circuitry. Very high levels of integration have been achieved in mixed-signal IC's intended for high-volume manufacturers in dedicated high-volume applications, resulting in very low cost of manufacturing. These examples include telephony components such as modems and voice coder/decoders, interface components for local area networks and so forth. This cost reduction through higher level of integration has been more difficult to achieve in components with functionality specific to systems that will be manufactured in low to moderate volume applications. The sophisticated automatic synthesis tools that allow fast turn-around implementation of high-integration components in purely digital functions, such as gate array, standard cell, and field programmable logic arrays (FPLA's), do not as yet exist for analog and mixed signal components. One of the objective of the research described here is to make a contribution to that goal.

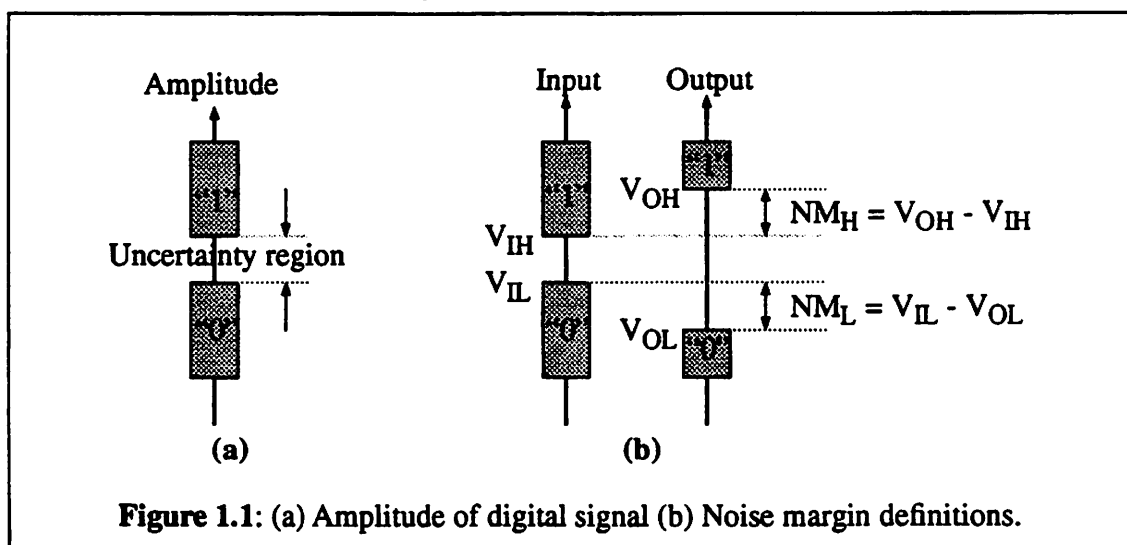
There are several factors that complicate semi-custom analog and mixed signal design. Because of rapid evolution in technology, design techniques and tools must adapt to a rapidly changing technology base. Unfortunately, most analog IC products presently have to be designed manually by expert analog IC designers due to the lack of analog CAD tools. As a result, analog circuit design has become the bottleneck in the design of mixed analog/digital interface chips. This has seriously hampered the effort to reduce the product development time mentioned earlier. In the next section, a brief discussion of why digital CAD tools progress much faster than its analog counterparts will be presented. The progress in semi-custom applications is limited more by the lack of suitable analog CAD tools such as mixed signal simulator, design capture, synthesis tools for commonly used blocks, place and route, layout extraction, verification, and testing.

One common analog function in a mixed signal chips such a modems is an analog to digital conversion (ADC) block. This function recurs frequently but with varying performance requirements [CHEC78], [TOWN80], [OHAR87]. Efforts to automate the design of A/D conversion functions have been reported previously [ALLE86], [HELM87]; however, there are some shortcomings in the implementation of these tools which will be described in Chapter 2. Since the ADC block is one of the critical block in the design of mixed-signal chips, we are motivated to study the approach and implementation of an ADC synthesis tool so that it can be used to improve or speed up the turn around time of mixed signal chips

1.2 Integrated Circuit Design Automation

Integrated circ: design can be divided into two different classes: digital and analog circuits. The distinction is mainly due to their differences in signal representations, sensitivity, and performance trade-off.

In digital circuits, signals are represented by a stream of discrete voltage levels which is usually represented in terms of binary numbers "1" or "0". These binary levels are usually representing a range of continuous voltages as shown in Figure 1.1(a). Typically we would like to minimize the uncertainty region shown in Figure 1.1(a) so that the allowable disturbance voltages (*noise margins*) at the input of a digital block can be maximized. Figure 1.1(b) shows how the noise margins are defined based on the range of the input voltages (V_{IL} , maximum input voltage for logic "0" and V_{IH} , minimum input voltage for logic "1") and output voltages (V_{OL} , maximum output voltage for logic "0" and V_{OH} , minimum output voltage for logic "1"). In digital circuit, signal disturbance at the input of a block does not get forwarded to the output as it does in analog circuit. As a result, a substantial level of signal disturbances from noise coupling, power supply fluctuation, and random noise will not degrade the digital signal level as it passes a digital block. Furthermore, having these discrete signal representations allows digital circuits to have a standardized interface levels, which in turn gives a standard power supply requirement. As a result, digital circuits have only a *small set* of performance measures that need to be optimized such as power consumption, gate delay, power-delay product, and area.



In analog circuits, signals are represented by continuous voltages, currents or charges as a function of time. Depending on the application, analog input signals can have a range from a few millivolts to several tens or even hundreds of volts. Signal disturbances which are tolerable in one application can be devastating in another. Moreover, power supply requirement will also vary depending on applications. Unlike digital circuits, analog circuits have no standardized interface levels or standard power supply requirements. As a result, analog circuits have *a large set* of performance measures that need to be optimized such as bandwidth, power dissipation, gain, offset, dynamic range, area, noise, supply rejection, overall accuracy, and many others. These performance measures of analog circuits depend more heavily on details of device behavior than do those of digital circuits.

With the distinctions mentioned above, we can now discuss the design automation of integration circuits which is a subset of Electronic Design Automation (EDA). The EDA effort started in early 1950's when digital computers were just becoming popular [GRAH53]. Since then, EDA has made tremendous progress and branched out into many different specialized fields over the last forty years. One of these specialized fields is integrated circuit design automation (ICDA). For the last 15 years, ICDA has evolved into two different fields: CAD for digital IC's and CAD for analog IC's. Digital IC CAD has advanced at a much faster rate than its analog counterpart for several reasons:

1. **Complexity:** Digital circuits are formed from a smaller number of primitives and are described by a much smaller number of performance parameters than analog circuits. Hence they are easier to design in many ways than analog ones since digital circuits have far fewer optimization constraints.

2. **Number of generic primitive blocks:** In digital domain there exist simple and generic block functions from which many higher level and complex digital

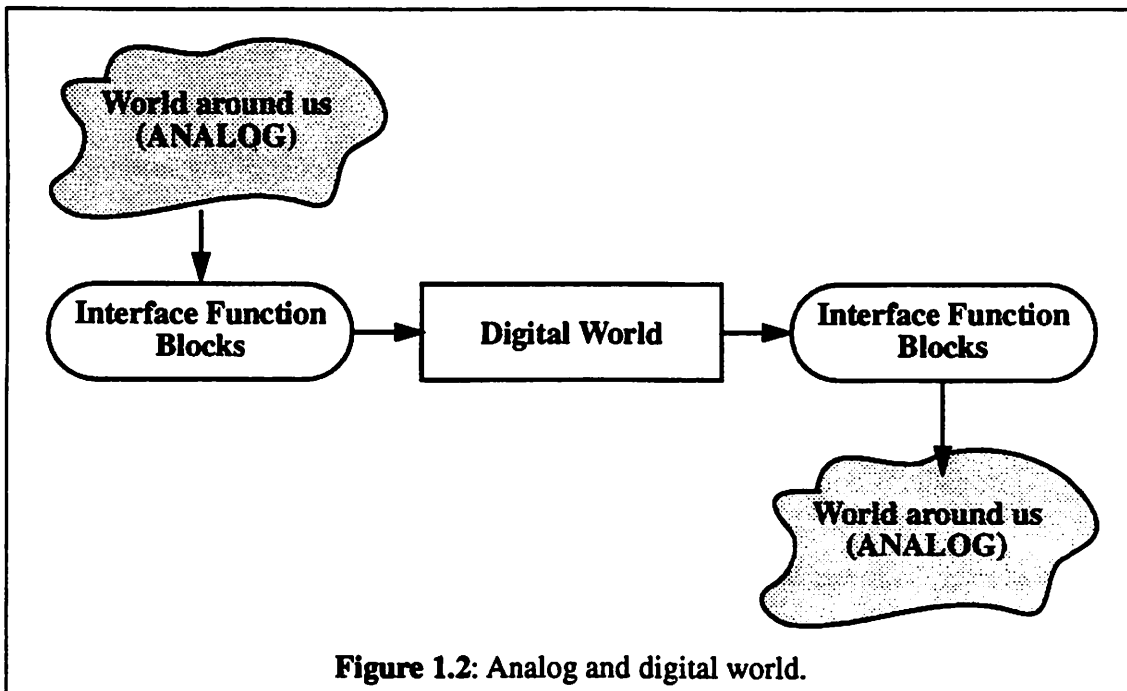
blocks can be implemented. Examples of these simple generic blocks are NAND and NOR gates. For analog circuits, such a simple and generic block does not exist.

- 3. Size of User Community Base:** Measured economically by sales, purely digital integrated circuits are about 5 times larger than the mixed signal and analog applications. Furthermore, the digital domain is dominated by CMOS implementations of logic functions operating on 5 volts, with speed and logic density being the main performance parameters of interest. In the analog domain, the user base is highly fragmented into different technologies (CMOS, Bipolar, BiCMOS), various performance domains (high frequency, low power, high precision, high power, etc.), and supply voltage (30V, 12V, 5V, 3V, 1.5V to name the most important standards). As a result, the cost of the research and development of digital computer-aided design software can be amortized over a much larger base of potential users. On the other hand, the cost of the research and development of analog computer-aided design software, which tends to be specific to one technology, supply voltage, and performance domain, becomes very expensive because of the small user community.

Recently, there has been a surge of interest in the field of analog ICDA both from universities as well as commercial laboratories. With constant scaling in technology feature size, an important motivation drive in the 90's is to increase the level of integration in IC chips to reduce cost. This means that the inclusion of both analog as well as digital circuits into a single chip is a must. For this to be successful, improved maturity of EDA for analog integrated circuits is a necessary condition. With this impetus, hopefully, analog ICDA will progress at a much faster rate.

1.3 Thesis Contribution

With the steady decrease in the cost of designing digital VLSI and the advancement of digital signal processing (DSP), more and more analog building block IC's will be replaced by more highly integrated mixed signal application specific integrated circuits commonly known as ASIC's. Some have suggested that analog circuit design will die as an IC implementation methodology and digital circuit design will take over. This belief is probably true in many IC applications where signal processing functions can be common to digital circuits. However, the world around us is still analog and anything interacts with environment needs some kind of interface blocks as shown Figure 1.2.



In order to make use of advancements in digital VLSI, we need to convert analog data into its digital representation. Thus analog data has to be interfaced with special converters in order to be processed digitally. Borrowing an example from [GRAY87] on the

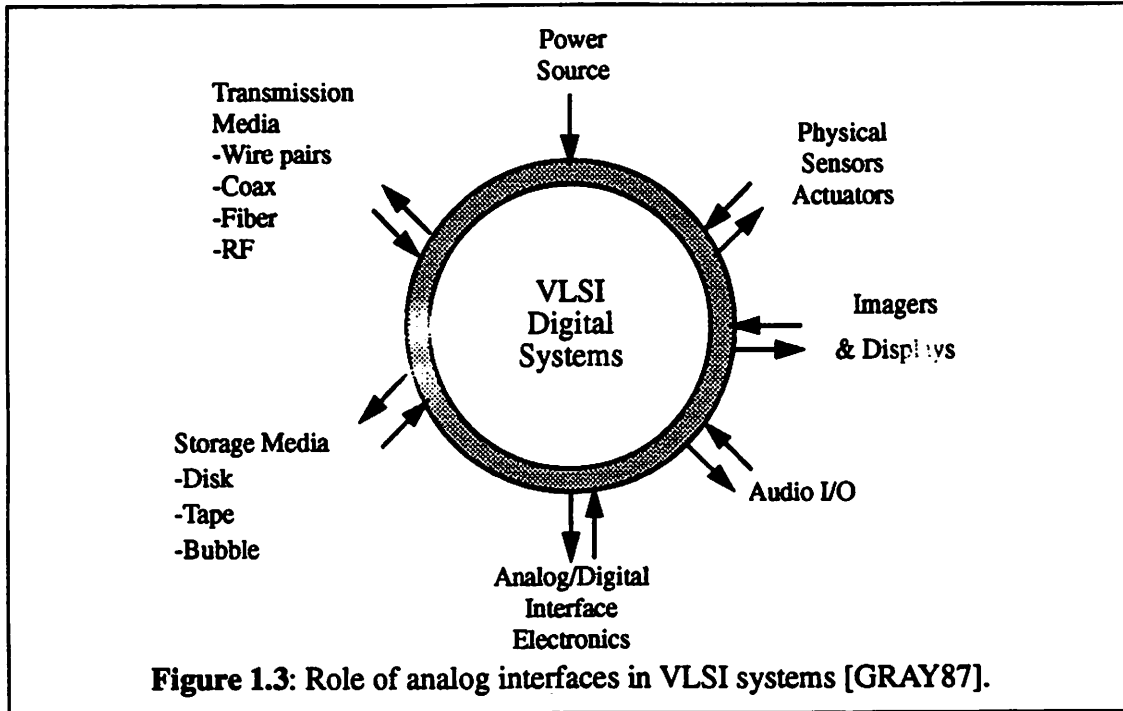
role of analog interfaces in VLSI systems, a VLSI digital system is like an egg and the analog interfaces is the eggshell as shown in Figure 1.3. As the egg is getting bigger (which is analogous to the advancement of digital VLSI system), the eggshell is getting thinner (which is analogous to more and more analog block functions being replaced by digital circuits). But the egg will never hatch! Thus the analog interface functions will remain as an important role in VLSI systems.

One of the most common blocks used in analog-digital mixed-signal IC's is an Analog-to-digital converter (ADC). Efforts to create better ADC blocks are actively pursued both in academic as well as industry sectors. In this thesis, we will present a new improved algorithmic ADC architecture that is capable of converting analog signal at a higher rate than previously reported algorithmic ADC architecture. We will also introduce a two-comparator scheme that allows us to use low-precision or simple comparators to do the comparison without affecting the overall performance of the A/D converter.

In the area of analog CAD, this thesis will present a methodology to approach and implement a module generation program for such a general function as A/D converters. We will discuss in detail how we implement a technology-independent ADC module generator, CADICS [JUSU90] that optimizes performance, silicon area, and power dissipation so that the resulting ADC's are comparable with a manual design. We will also show at the end that the methodology can also be applied for implementing a general mixed signal design automation system.

1.4 Thesis Organization

In the remaining of this thesis, the study and implementation of a performance-driven ADC module generator will be presented. In Chapter 2, a brief discussion of exist-



ing analog block module generators and a detailed discussion of ADC blocks, previous work in ADC module generation, and methods of attack will be presented. In Chapter 3, an architectural study of different A/D conversion techniques will be presented. Comparisons in architecture, performances, trade-off, and complexities will be discussed in detail. An improved algorithmic ADC architecture will then be presented. Detailed implementations of the circuit and layout generations of the ADC module generator will be covered in Chapter 4 and 5 respectively. In Chapter 6, some examples and experimental results will be presented. In Chapter 7, a summary of research and a brief discussion as to what the future direction will be discussed. Finally, an in depth discussion of the circuit design techniques for each functional block of the ADC is presented in the Appendix A.

CHAPTER 2 Automatic Generation of Analog Blocks

2.1 Introduction

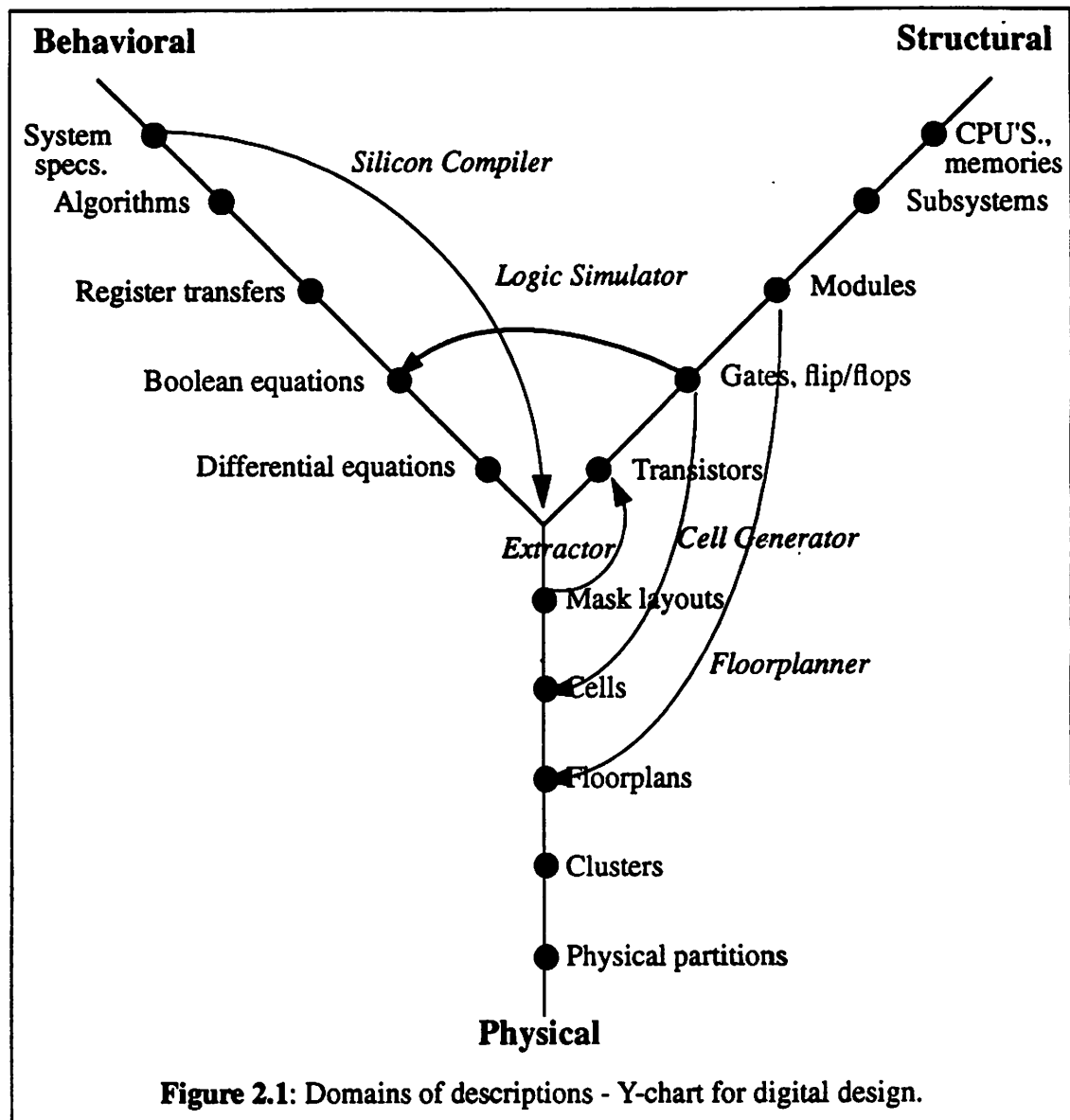
Automatic generation of analog blocks has become *very desirable* as system level designers try to shorten the turn-around time of designing mixed signal chips. Unfortunately, the lack of such tools forces the designers to utilize off the shelf components or pre-designed analog blocks that are optimized to certain applications. Efforts to come up with a general analog design framework that is capable of generating complete analog block layouts from high level specifications are currently being carried out [HARJ89], [DEGR87], [CHAN92], [GIEL92]. In this chapter, we will first discuss why module generation of specific analog functions is a good starting point to the realization of a general automatic analog design framework. Examples of existing analog block module generators will then be presented. The goals of analog-to-digital conversion (ADC) module generator developed will then be described.

2.2 Why Do Module Generation?

As mentioned in Chapter 1, the implementation of analog CAD tools, in general, is more difficult than the implementation of digital CAD tools, as is the analog design problem. Ideally we would like to have a general analog circuit framework similar to that of its digital counterparts. However, analog circuits in general require more design freedom in order to meet performance objectives. They often exploit the full spectrum of capabilities exhibited by individual devices. In analog circuits, the individual devices often have different sizes and electrical characteristics. These devices require optimization of different performance requirements. The importance of each performance requirement will depend on circuit applications.

In digital design methodology, there are multiple abstraction levels that can be represented by a tri-partite representation of design (commonly known as the Y-chart [GAJS86]) as shown Figure 2.1. This unified model of design representation separates the structural, behavioral, and physical domains. The level of abstraction decreases as one moves toward the center vertex. The procedures to go from one level of abstraction to the next one are clearly defined since they can be structured on to several levels of hierarchy.

For analog design, there are few acceptable abstraction levels such as functional and behavioral. The rest of abstraction levels are not clearly defined. The procedures to move from one point to the next point in a domain will be different from one analog function to the next. Therefore, to come up with a general design methodology that can generate all possible analog blocks is very difficult if not impossible. Instead, a more realistic goal would be to have an analog design framework or mixed signal design framework that can generate different analog or mixed signal blocks. This kind of framework is basically an integration of all different CAD tools written specifically for analog or mixed signal blocks. An example of such a framework is currently being developed at U.C. Berkeley



[CHAN92].

While it is sometimes possible to share CAD tools between digital and analog portions of a circuit, such as design rule checkers and extractors, many CAD tools must be designed for use primarily on analog circuits. Many existing analog CAD tools are the extension of the digital tools. Generic tools such as router, placement, and compactor that

can handle analog blocks are becoming hot research topic in university sectors. Examples of analog routers are constrained-driven channel router, ART [CHOU90] and constrained-driven maze router, ROAD [MALA90]. These routers are also capable of routing fully differential blocks that have become a common analog circuit design technique. Analog placement [CHAR92] and compaction [COHN91], [FELT92] tools have also been reported recently. All these tools are still developing to reach maturity so that they can be used as the generic tools in an automatic as well as manual block generation step.

Other possible tools in this mixed signal design framework would be module generators of different analog functions. It will be very desirable if we would be able to come up with a general design methodology for automatic generation of *all* analog blocks without the need of implementing different module generators for different analog blocks. Unfortunately, as it has been discussed in the previous chapter, analog circuit design is the most knowledge intensive process of IC design tasks. The quality of a design will depend a lot on the expertise level of circuit designers. This expertise is usually obtained over a long period of time from the experience of doing numerous circuit designs. In order to implement a general analog design framework, we need to be able to *capture* and *formalize* the existing design knowledge entirely into the program database. This job is an extremely difficult if not impossible task to do.

The automatic generation of known analog block functions seems to be a more feasible goal to do since capturing and formalizing design knowledge of a specific analog function is simpler and easier to do. As the need to automatically generate other analog functions rise, their module generators can be implemented. In the next section, existing module generators of two commonly used analog functions: switched-capacitor filter (SCF) and operational amplifier (op-amp) will be described.

2.3 Existing Analog Module Generators

2.3.1 Operational-Amplifier Module Generator.

The operational amplifier is one of the basic building block used in many analog functions. Many efforts in the automatic generation of this block have been reported OASYS [HARJ87], IDAC-ILAC [DEGR87], OPASYN [KOH90]. All the module generators start with a given set of specifications for the op-amp and generate the device sizes.

The OASYS system is implemented using an expert system that decomposes a given set of op-amp specifications into several sub-modules' specifications for input stages, output stages, or biasing circuits. These sub-specifications are then used to search for the most suitable pre-defined sub-modules such as current sources, differential pairs, or individual transistors. When the sub-modules are found and their specifications are met, an op-amp will then be constructed by joining all the chosen sub-modules. This op-amp will then be synthesized to get the op-amp specifications and its device sizes.

The IDAC-ILAC and OPASYN systems have similar implementation approach. Both systems generate device sizes of the pre-stored op-amp topologies by optimizing using the analytical equations. Since OPASYN will be used later as a low-level synthesis tool for generating op-amps and comparators in our ADC module generator, we will elaborate on it more.

A typical set of input specifications for OPASYN would be open loop gain, power dissipation, unity gain frequency, phase margin, and slew rate. The outputs of the op-amp synthesis would be the op-amp netlist, layout, and performance summary. A simplified block diagram of the procedures used in OPASYN is shown in Figure 2.2. OPASYN first carries out a sequence of operations such as topology selection and numerical optimization to determine the device sizes. A netlist and performance summary will then be gener-

ated. The device sizes are then used to generate layout of the op-amp. Currently, the topology database of the netlist generator consists of an output buffer, a comparator, a single-ended basic two stage and a fully differential folded cascode op-amps. The layout generation takes the device sizes information generated by the circuit generator and outputs the complete layout of the block. Currently, the OPASYN layout generator does not support *fully differential* folded cascode op-amp layouts since specific routines for this op-amp topology need to be implemented. Efforts to enhance the layout generation capability of OPASYN are being done. As a result, the layout generator of OPASYN will not be used as a low level block layout generator in the ADC module generator, CADICS. The layout generation will be described in detail in Chapter 5. An example of a basic two-stage op-amp performance summary and layout are shown in Figure 2.3(a) and (b) respectively.

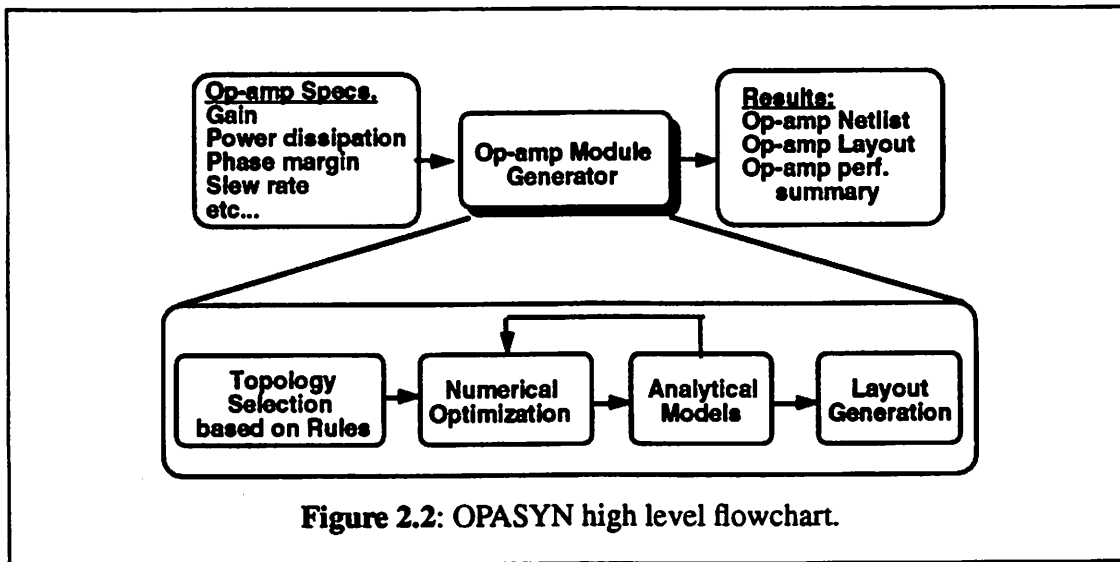
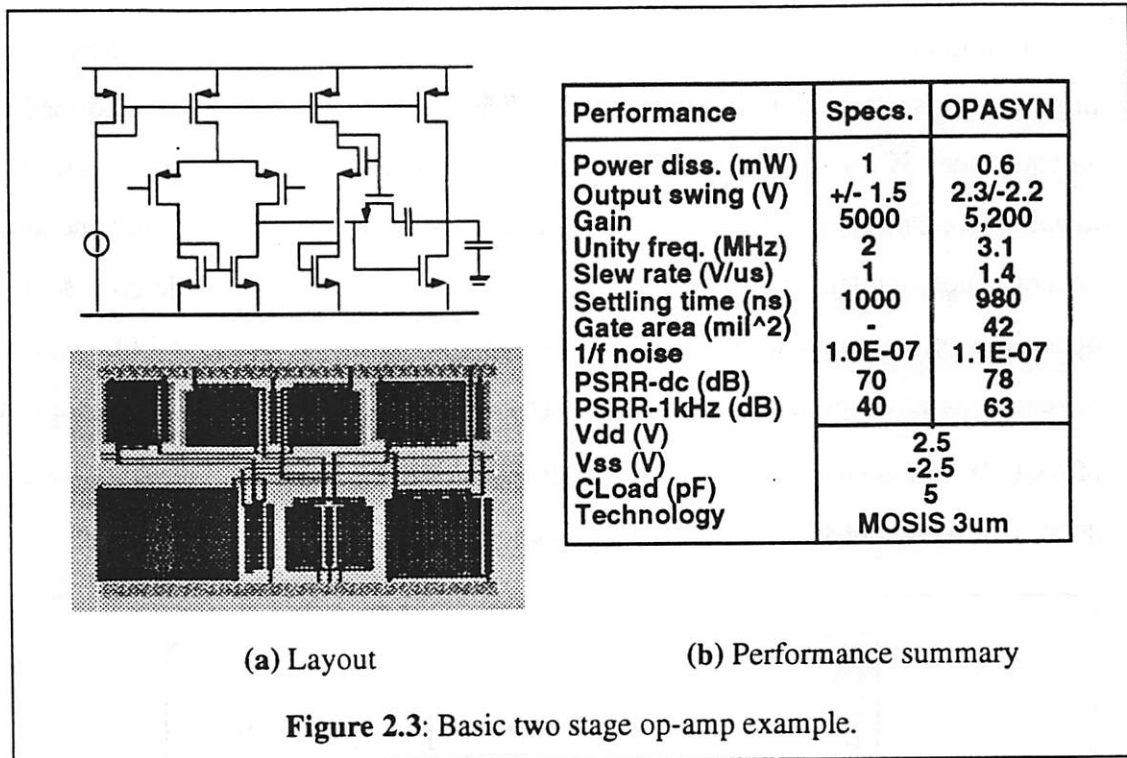


Figure 2.2: OPASYN high level flowchart.

2.3.2 Switched-Capacitor Filter Module Generator

Filtering is one of the most common block used to precede or proceed a telecommunications or audio systems. For example, an anti-aliasing filter is usually used to filter



out of band signal in the front end of an A/D converter to prevent aliasing. A decimation filter is used to filter the out of band noise preceding a sigma-delta modulator. One type of filter implementation is an analog sampled data system implemented in MOS technology commonly known as switched-capacitor filter (SCF). It is called SCF because the building block is a switched-capacitor integrator.

Many SCF synthesis programs [SANC85], [EATO87], [TRON87], [LUCA87], [YAGU86] have been reported in the literature. The popularity of SCF synthesis program is partially due to the fact that a straight-forward method or algorithm exists to synthesize an SCF from a given set of specifications. One of SCF module generators discussed here is ADORE [YAGU86] developed at U.C. Berkeley. ADORE is capable of generating switched-capacitor filter layout from a given set of specifications such as filter type, order, and frequency response.

The overall steps of the module generation are as follows. Given a set of specifications of a filter, a continuous inductor-capacitor (LC) ladder filter is generated by using a filter synthesis program, FILSYN [SZEN77]. The resulting filter is then transformed into an equivalent SCF network. Capacitor ratio evaluation and dynamic range scaling maximization are carried out. The resulting netlist is then passed on to the layout generator to generate the complete layout of the SCF. The layout generator will undergo a series of operations such as generating switches, obtain op-amp module from the cell library, ordering capacitor array, placing all modules, and routing all channels. The high level flowchart of ADORE is shown in Figure 2.4. A 10th order Chebychev bandpass filter layout and frequency response are shown in Figure 2.5(a) and (b) respectively.

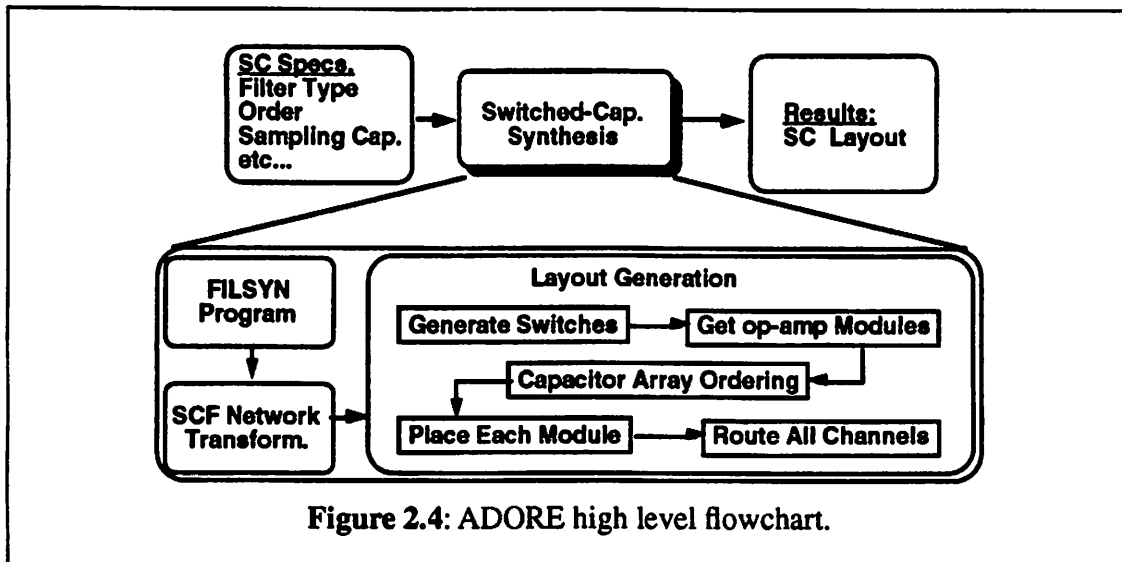
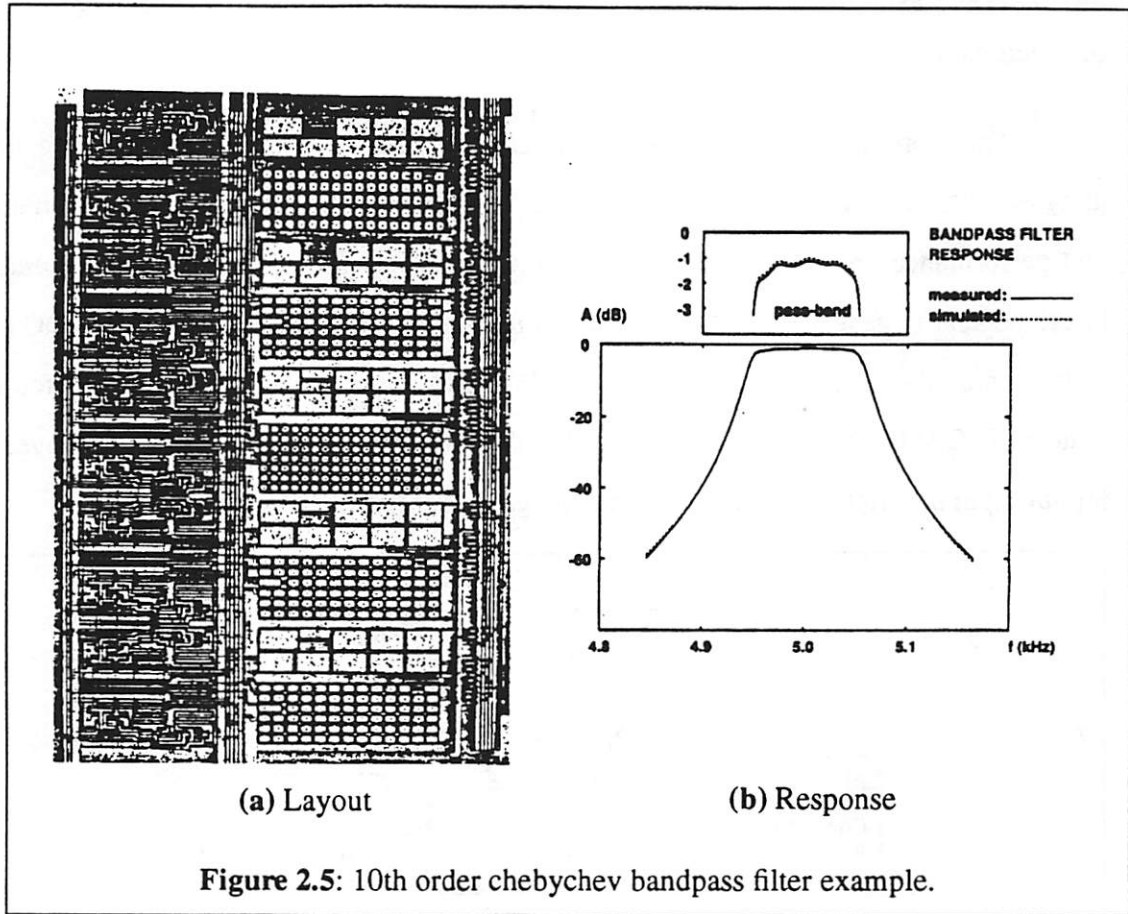


Figure 2.4: ADORE high level flowchart.



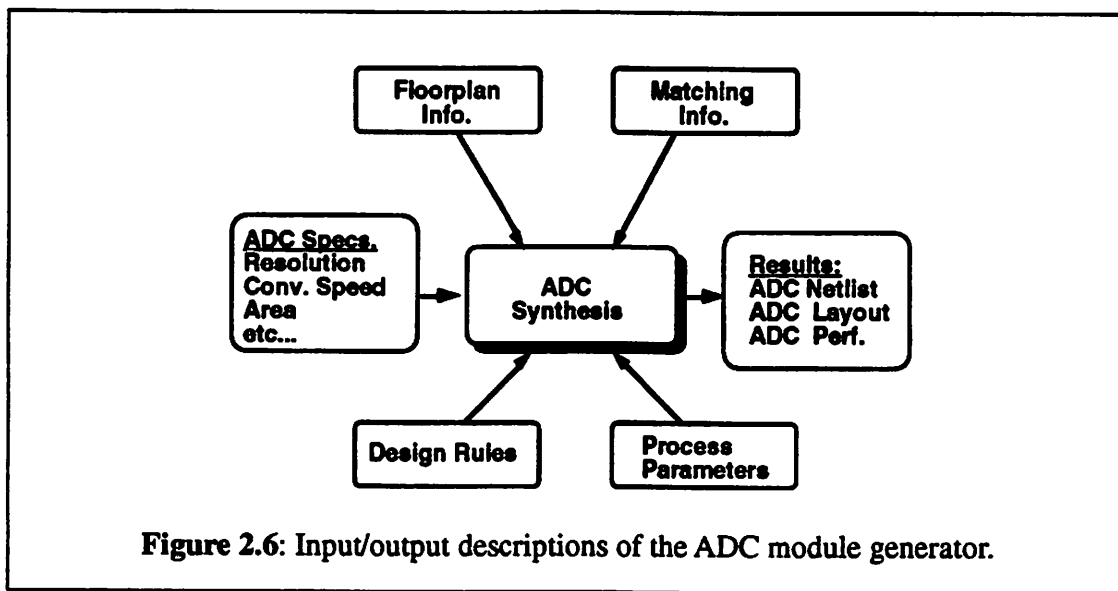
2.4 Analog-To-Digital Conversion Module Generator

2.4.1 Overall Goals

Analog to digital conversion function recurs frequently in mixed analog-digital chips with varying performance requirements depending on the applications [CHEC78], [TOWN80], [OHAR87]. Due to the non-existence of highly optimized ADC module generators, this analog block can become the bottleneck of the turn-around time for the overall design of the chips. Just like the op-amp and SCF module generators, we would like to

automatically generate A/D conversion block netlist and layout optimized to the given set of specifications.

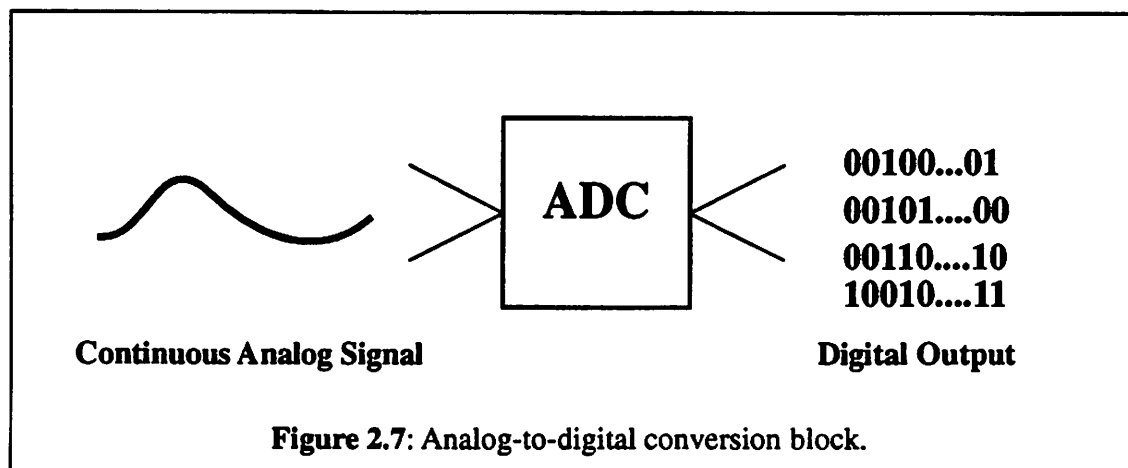
Therefore, it is very desirable to have an ADC module generator capable of generating A/D converters with a broad range of sampling rates and resolution, and silicon area and performance comparable with a manual approach. Moreover, the module generator has to be able to generate ADC's for different design rules. Finally, the resulting ADC has to be compatible with digital environment. We have developed a CMOS ADC module generator, CADICS [JUSU90] that fulfills all of the above requirements. The overall input/output descriptions of the ADC module generator is shown in Figure 2.6.



In the remainder of this chapter, a brief description of ADC function and characterization (mainly static ones) will be reviewed. Many terms defined here will be used throughout the remainder of this thesis. Finally, an overview of existing ADC module generators and our implementation methods will be discussed.

2.4.2 Functionality

An N-bit 100kHz A/D converter is an electrical device which converts or encodes a continuous analog signal with a granularity of $\frac{1}{2^N}$ into a digital representation with a finite number of binary bits, N as shown in Figure 2.7. The higher the number of bits, the more accurate the analog input signal can be represented. As the unit indicates 100 kHz specifies the rate at which the A/D converter performs the conversion. The most common performance specifications of an A/D converter are *resolution*, *speed*, *linearity*, and *signal-to-noise ratio* (SNR).



2.4.3 Characterization

The evaluation of the performance of an A/D converter is very crucial both from the circuit design stand point as well as the requirements dictated by a particular application of the device. One of the most common tests an A/D converter undergoes is the measurement of its transfer characteristic with respect to a ramp input signal as shown in Figure 2.8 (i.e. analog voltage in - digital code out). For every analog input signal level, there exists a corresponding binary representation. Because of the finite number of bits,

there is a finite granularity in the digital approximation d_0, d_1, \dots, d_{N-1} of the actual analog input signal V_{IN} .

$$V_{IN} = V_{FS} \left(\frac{d_0}{2^0} + \frac{d_1}{2^1} + \dots + \frac{d_{N-1}}{2^{N-1}} \right) + \epsilon \quad (\text{EQ 2.1})$$

Where V_{FS} = full scale input signal = Max. $\|\pm V_{IN}\|$

$d_i = i^{\text{th}}$ digit

ϵ = deviation of digital output from V_{IN}

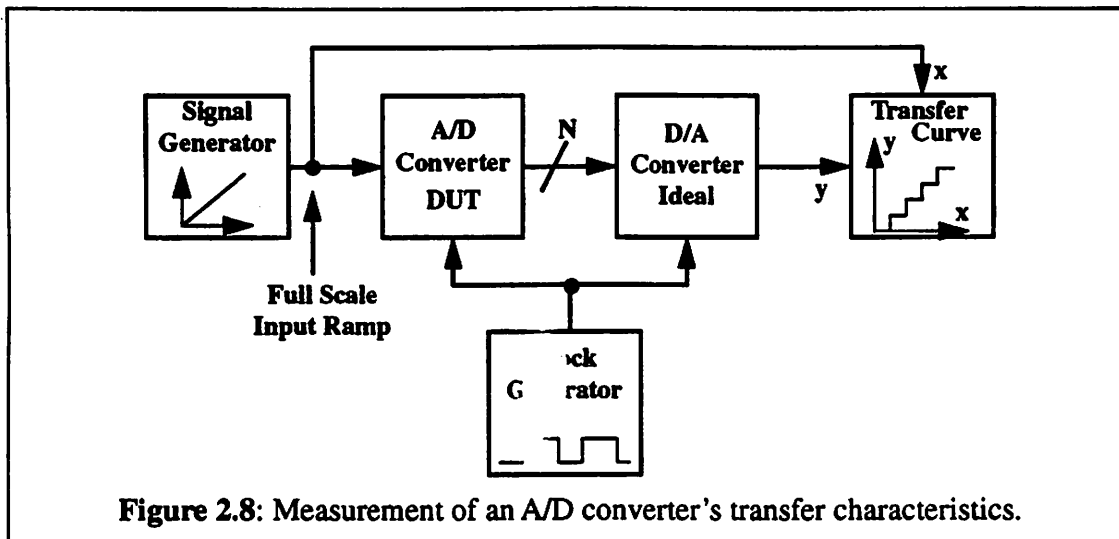
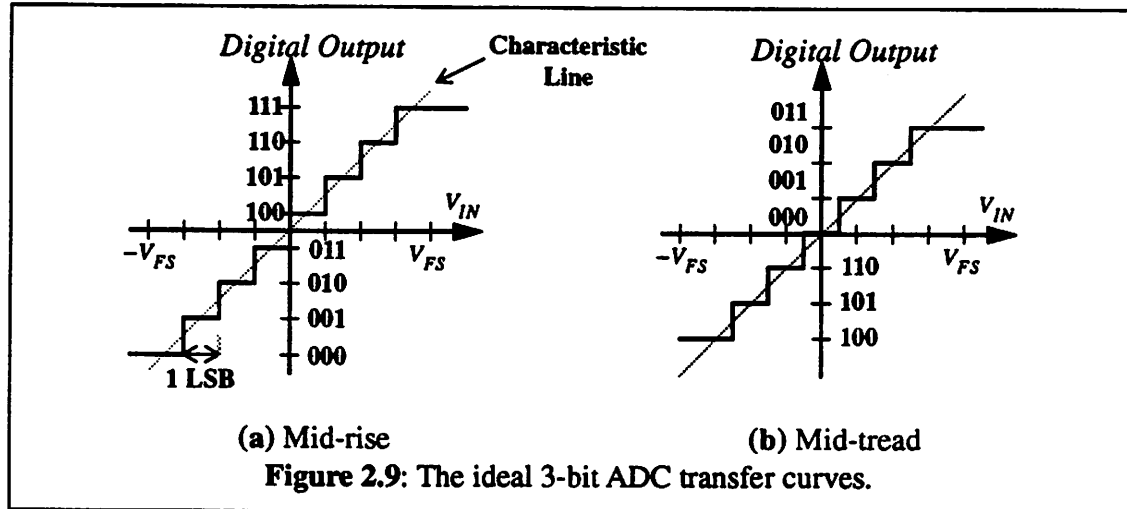


Figure 2.8: Measurement of an A/D converter's transfer characteristics.

Quantization error is unavoidable when quantizing a continuous signal into a digital domain. The transfer curve shown in Figure 2.9(a) clearly shows each transition point of a continuous analog input signal corresponding to each digital output code change. The transfer curve looks like a stair case with steps at the analog transition points. This is due to the fact that digital output of a N-bit converter has a finite number of representations (2^N distinct digital codes); whereas, the analog signal is continuous and hence has an infinite number of levels. Two possible *ideal* transfer curves of a simple 3-bit A/D converter

are shown in Figure 2.9.



The difference between the two depends on where the origin intersection is. The transfer curve shown in Figure 2.9(a) is referred to as *mid-rise* because the origin intersection occurs in the middle of a vertical transition between 2 different digital codes. On the other hand, the transfer curve shown in Figure 2.9(b) is referred to as *mid-tread* because it intersects the origin in the middle of the tread representing the digital code for zero volts input. Unlike a mid-rise case which has 2^N digital output codes, a mid-tread case has only $2^N - 1$ possible codes. The rest of the discussion will use the mid-rise transfer curve as an example.

In reality the transfer curve is affected by many non-idealities contributed by non-ideal functional blocks. As a result, the transfer curve will have non-uniformly spaced steps as illustrated in Figure 2.10(b). Let 1 least significant bit (LSB) be the smallest quantization step for an N-bit conversion.

Let the input voltage range be between $-V_{FS}$ and V_{FS} .

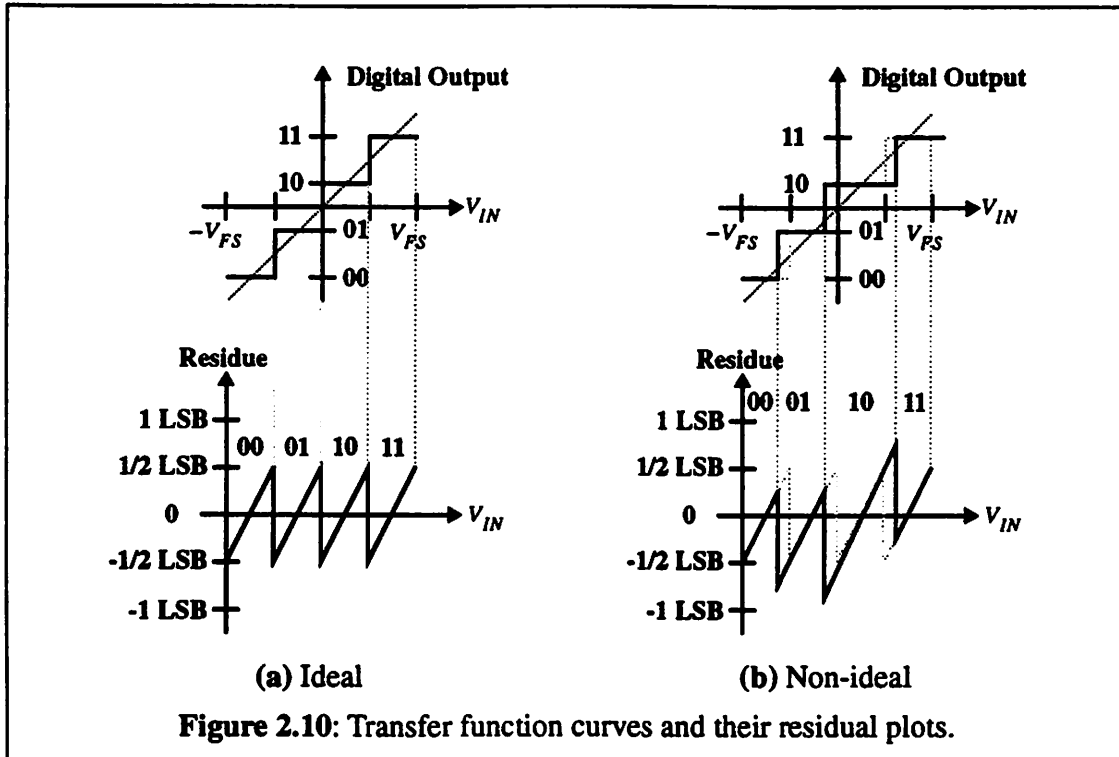
$$1 \text{ LSB} = \frac{2V_{FS}}{2^N} \quad (\text{EQ 2.2})$$

Then the residual voltage can be written as

$$\text{Residue} = \frac{(V_{IN} - V_{digital})}{1\text{LSB}} \quad (\text{EQ 2.3})$$

$$V_{digital} = 2V_{FS} \left(\frac{d_0}{2} + \frac{d_1}{2^2} + \dots + \frac{d_{N-1}}{2^N} \right) \quad (\text{EQ 2.4})$$

and plotted in Figure 2.10. Notice that the residual voltage for an ideal transfer curve fluctuates between $\pm \frac{1}{2}$ LSB (Figure 2.10(a)); whereas, the non-ideal one fluctuates between ± 1 LSB for this particular transfer characteristic (Figure 2.10(b)).



2.4.3.1 Differential Non-Linearity (DNL)

As mentioned earlier, the voltage difference between two consecutive transition voltages is the width of a quantization step. For an ideal transfer curve, the quantization steps are all equal to at least $\pm\frac{1}{2}$ LSB or 1LSB in an absolute magnitude. For an actual transfer function curve as shown in Figure 2.11, the quantization steps are not uniform at all. As a measure of how much the actual quantization step width varies from the ideal step width of 1 LSB, differential non-linearity (DNL) is defined.

$$\text{DNL}(k) = x(k) - 1\text{LSB} \quad (\text{EQ 2.5})$$

Where k = the k^{th} digital code

$$k = 1, 2, \dots, 2^N - 1$$

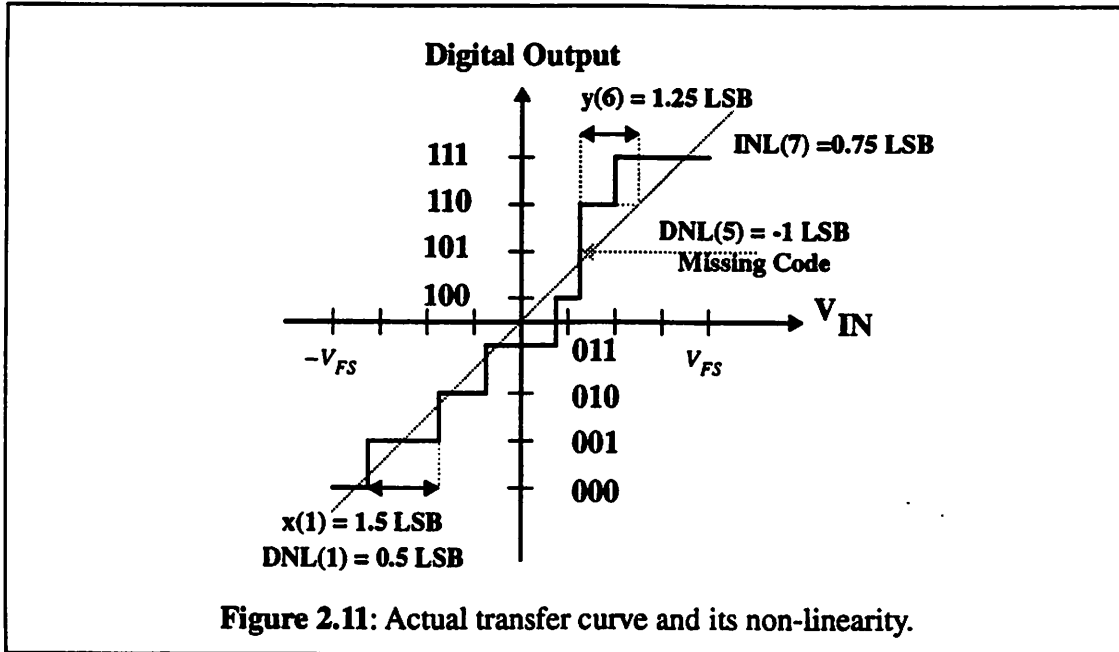
$x(k)$ = the actual step width of k^{th} digital code

$$\text{DNL}(k) = 0 \text{ for } k = 0 \text{ and } (2^N - 1)$$

If $x(k)$ is equal to zero, the corresponding DNL is going to be -1LSB . When this occurs, we have a *missing code*. Figure 2.11 shows that digital code 101 is skipped and therefore, it is *missing*. For illustration purposes, the DNL plot of transfer function curve shown in Figure 2.11 is shown in Figure 2.12(a).

2.4.3.2 Integral Non-Linearity (INL)

Integral non-linearity (INL) for each digital code is defined as the transition voltage between a code and the next code minus the input voltage corresponding to this code obtained from the characteristic line. It can be shown [LEWI87b] that



$$\text{INL}(x) = \sum_{i=1}^{k-1} \text{DNL}(i) \quad (\text{EQ 2.6})$$

Where $k = 1, 2, \dots, 2^N - 1$

$$\text{INL}(0) = 0$$

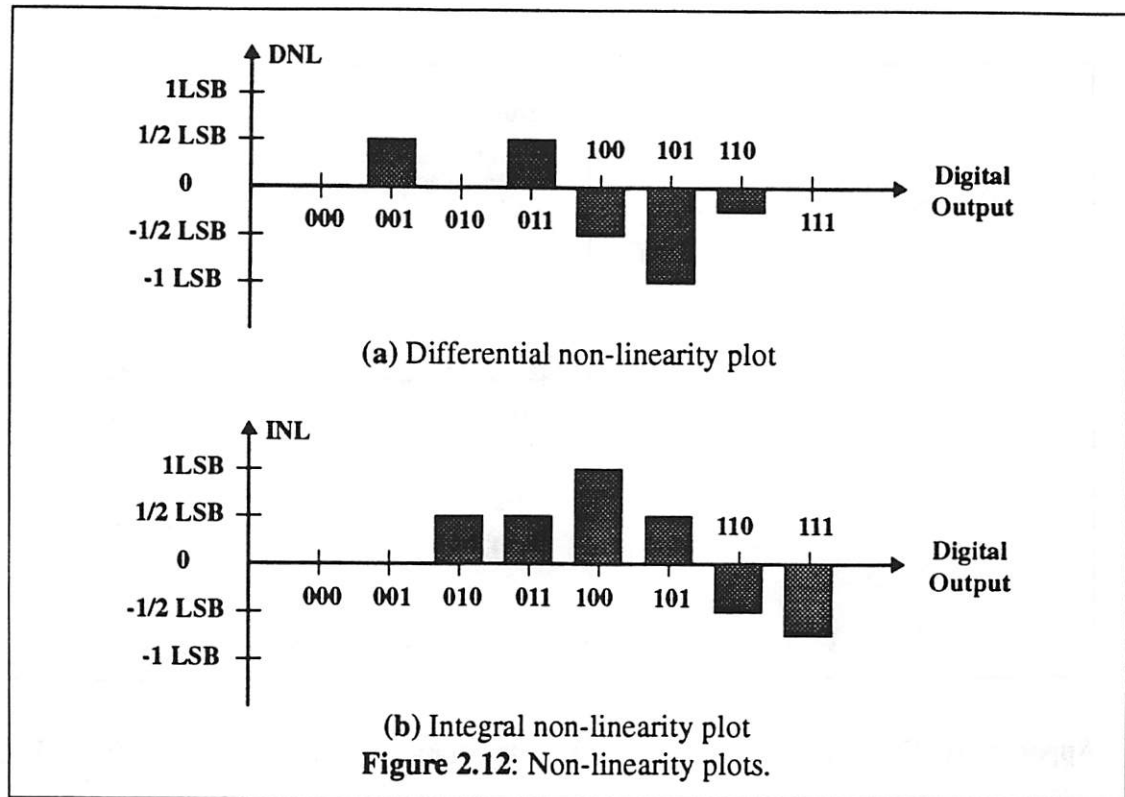
INL can also be defined as the deviation of the ideal intersection point on the characteristic line to the actual point, as shown in Figure 2.11.

$$\text{INL}(k+1) \approx y(k) - \frac{1}{2} \text{LSB} \quad (\text{EQ 2.7})$$

Where $y(k)$ = the horizontal distance from the beginning

of code k to the ideal characteristic line

Figure 2.12(b) shows the INL plot of the respective DNL plot in Figure 2.12(a) according to EQ 2.6. The overall linearity of an A/D converter depends on the maximum absolute



magnitude of DNL and INL.

2.4.3.3 Gain Error

Gain error occurs when a *gain* block in an A/D converter does not multiply the signal by the ideal value. This gain error will affect all the codes by the same percentage. As a result, the resulting digital code center line slope differs from the ideal one. If the gain is bigger/smaller than it is supposed to be, the slope is bigger/smaller than the ideal one as shown in Figure 2.13.

2.4.3.4 Offset Error

Offset error can be viewed as having an additional input voltage source in the front of an A/D converter. A detailed explanation of this error will be described in Chapter 3 and

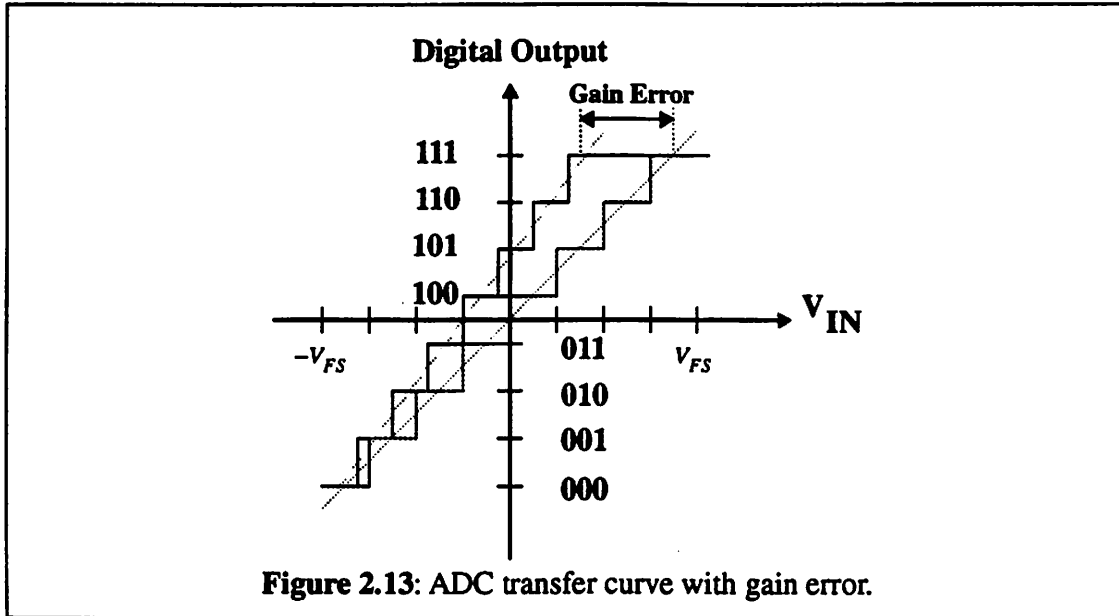
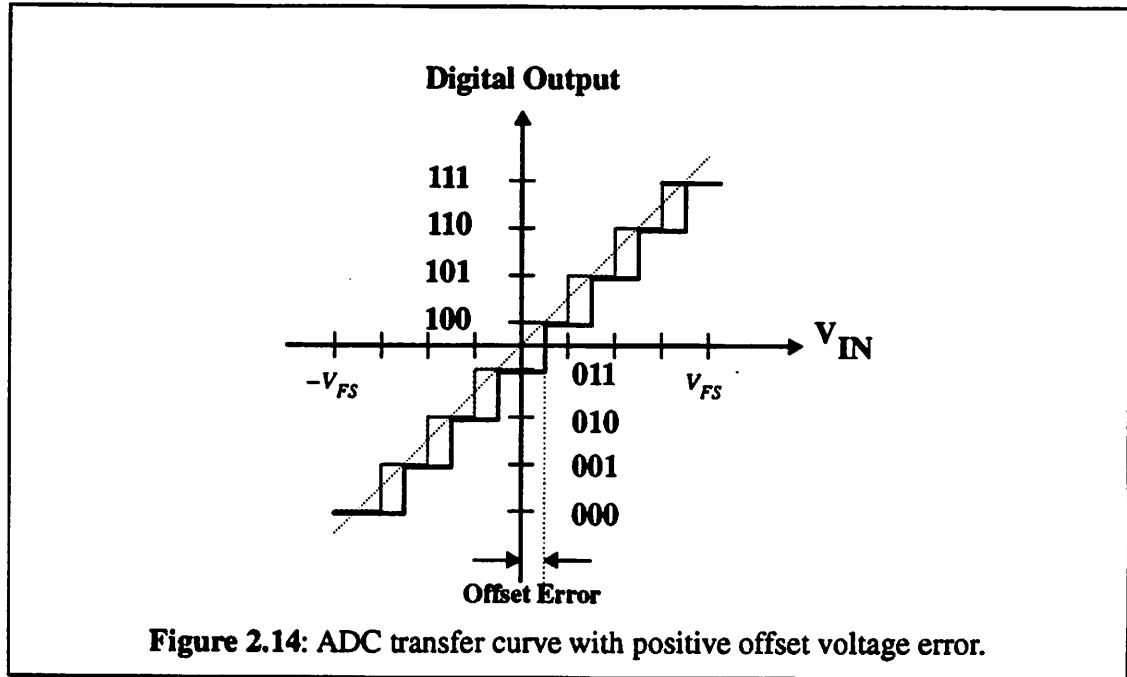


Figure 2.13: ADC transfer curve with gain error.

Appendix A. Thus, for an input voltage V_{IN} , the converter will treat it as if $V_{IN} + V_{offset}$ were its input. Thus plotting the transfer curve with respect to the actual input voltage V_{IN} , the entire curve gets shifted to the right or left (depending on the sign of the offset voltage) by the exact amount of the magnitude of the offset. As an illustration, Figure 2.14 shows the effect of a positive offset on the transfer curve shown in Figure 2.9(a). Notice that the linearity and monotonicity of the A/D converter are not affected by the offset.

2.4.4 Signal-To-Noise Ratio

Signal-to-noise ratio (SNR) falls into the dynamic characterization category. It is an important measurement for communication system performance, even though it is not always the most accurate standard for performance comparison between two systems. A typical SNR test configuration is shown in Figure 2.15. The SNR is the ratio of the mean square peak-to-peak amplitude of a sinusoidal input signal to the smallest quantization step width. If the full scale peak-to-peak amplitude is $2V_{FS}$ then the theoretical limit of



maximum SNR for ideal A/D converter with zero DNL and INL can be found by the following equations [RABI76]:

$$\text{Max SNR} = \left[\frac{\frac{V_{FS}^2}{2}}{\frac{(1\text{LSB})^2}{12}} \right] \quad (\text{EQ 2.8})$$

$$\text{Max SNR} = 1.5 (2^{2N}) \quad (\text{EQ 2.9})$$

Or in a simplified form:

$$\text{Max SNR} = 6.02N + 1.76\text{dB} \quad (\text{EQ 2.10})$$

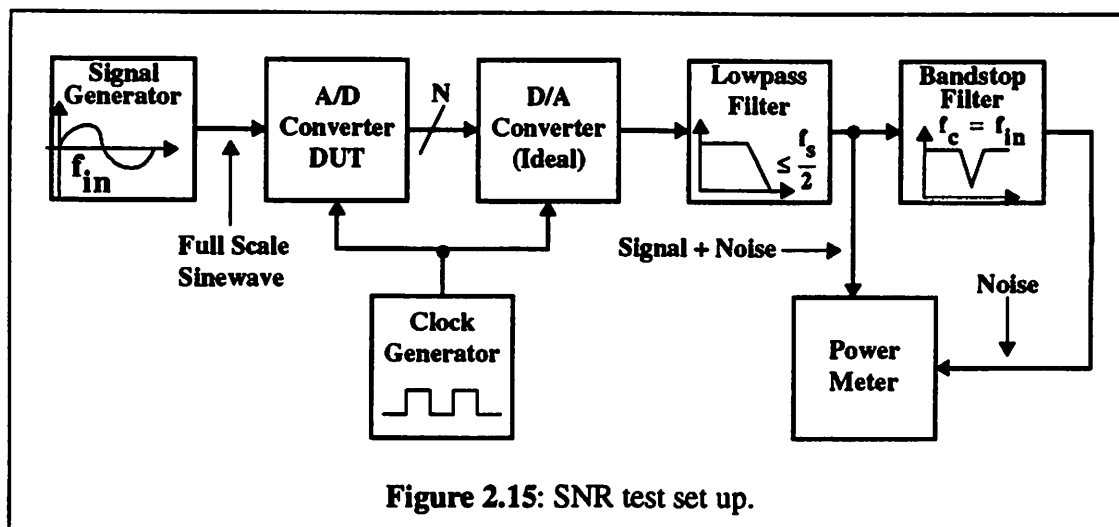


Figure 2.15: SNR test set up.

2.4.5 Previous Work

2.4.5.1 Analog Cell Library

In many mixed-signal chips today, standard off-the-shelf analog cells are often used in order to provide the analog functions needed. Unfortunately, these pre-designed, hard-coded cells do not offer flexibility of requirements and technological evolution. Cell libraries are usually optimized to a specific performance for a particular technology. As the technology feature size and process parameters change, these cells have to be regenerated again. To redesign the entire analog cell libraries will be very hard, tedious, error-prone, time consuming, and last but not least, it will require expert analog designers.

2.4.5.2 Performance-Driven Based on Standard Cells

The idea of analog-to-digital conversion module generation is not new. Allen and Barton [ALLE86] introduced a silicon compiler for successive approximation (SA) A/D and D/A converters. The basic approach of the program is a generalization of *standard cell* approaches as in digital block design automation by incorporating a family of analog standard cells. The program takes a set of input specifications such as type of block (ADC or

DAC), number of bits, maximum tolerable differential non-linearity (DNL), and maximum tolerable integral non-linearity (INL). It then goes through a topological selection algorithm by first calculating non-linearity of each converter topology and then selecting what combination of resistors and capacitors is best to realize the input specifications while minimizing the passive component area. The non-linearity equations assume a worst case design. If the specifications cannot be met, it will give the user the best linearity available. If there exists a topology that meets the specifications, a layout will be generated. The layout procedure is based on digital placement and routing methods which uses fixed height of standard cells as well as parameterized cells into a predefined structure.

An analog silicon compiler was developed as a part of the CONCORDE system [HELM87]. The compiler is capable of generating successive approximation and serial A/D converters. The basic approach of this compiler is to use *pre-designed analog blocks* such as op-amps and comparators. No effort was made to optimize performance of the resulting A/D converters since no optimization was carried out. The layout procedure used many digital tools as such the SSTARtm state machine compiler to generate the control logic. Special routines were developed to size the capacitor arrays and switches.

The two A/D conversion module generators described above have similarity in that they were implemented using *standard cell* approach. ADC module generators based on this approach will also have a limited range of sampling rate and resolution. Furthermore, area and power cannot be optimized with respect to the given specifications. As a result, the A/D converters generated will not be competitive with manual design. In other words, a performance-driven module generator based on based standard cells is directed toward *reducing* design cycle at the expense of *area* and *performance*.

2.4.6 Methods of Attack

Unlike digital circuits, analog circuits in general require more design freedom in order to be applied effectively. They often exploit the full spectrum of capabilities exhibited by individual devices. In an analog circuit, the individual devices often have different sizes and electrical characteristics. These devices require optimization of different performance requirements. The importance of each performance requirement will depend on circuit applications.

For this reason alone, a *performance-driven* module generator based on *design rules* and *spice parameters* is a better approach. This type of module generators can be implemented using an expert system approach or a parameterized schematic of known architecture approach. Examples of expert system implementation for simple analog blocks such as operational amplifiers (op-amps) and comparators have been presented in earlier sections. Unfortunately, A/D converter is a much more complex function than op-amps or comparators. There is no clear way as how to set the rules for this implementation so that the resulting A/D converters are of any use at all.

A parameterized schematic of known architecture implementation, on the other hand, is a better approach because it can generate highly-effective circuits based on known architectures for complex analog functions in a flexible environment by optimizing the device sizes with respect to various performance measures. [KOH90] and [DEGR87] have demonstrated the implementation of op-amps and other simple blocks.

Unlike switched-capacitor filter synthesis [YAGU86], [LUCA87], ADC synthesis does not have a *well-defined* procedure. The module generation of A/D conversion functions is a particularly difficult problem because of the wide spectrum of resolutions and sampling rates encountered in real applications, the trade-off between area versus power dissipation, and component matching requirements for different A/D conversion architec-

tures. Our approach for the module generation of A/D converters is to focus on *techniques* that allow design of circuits with efficiency comparable to the ones manually designed by skilled designers.

Because of the wide range of parameter spreads in analog integrated circuits, analog designers over the years have developed circuits which cancel out the first order variations in key parameters. This means, however that the analog circuits then become sensitive functions of second order variations of such parameters, for example the capacitor ratio matching in a gain of two block. We use many analog circuit design techniques (such as auto-zeroing circuits [DEGR85], capacitor trimming array [OHAR87], fully differential architecture, or even specifically designed 2-comparator scheme [JUSU90b]) to further reduce or overcome these second order effects so that the module generation procedures can be *simplified*.

The overall implementation flowchart of our A/D conversion module generator is shown in Figure 2.15. The module generation will be divided into three major tasks: *architecture selection*, *netlist generation*, and *layout generation*. The flowchart shows other supporting inputs needed to carry out the job. Details explanation will be presented in Chapter 3, 4, and 5 of this thesis.

2.5 Summary

In summary, analog block generation is the bottleneck of the turn around time of mixed signal chips. While analog standard cell libraries will give short term solution, they do not offer flexibility to the change in technology feature size. On the other hand, performance-driven analog block module generator based on design rules and spice parameters will provide a long term solution. Since ADC functions are the most common analog

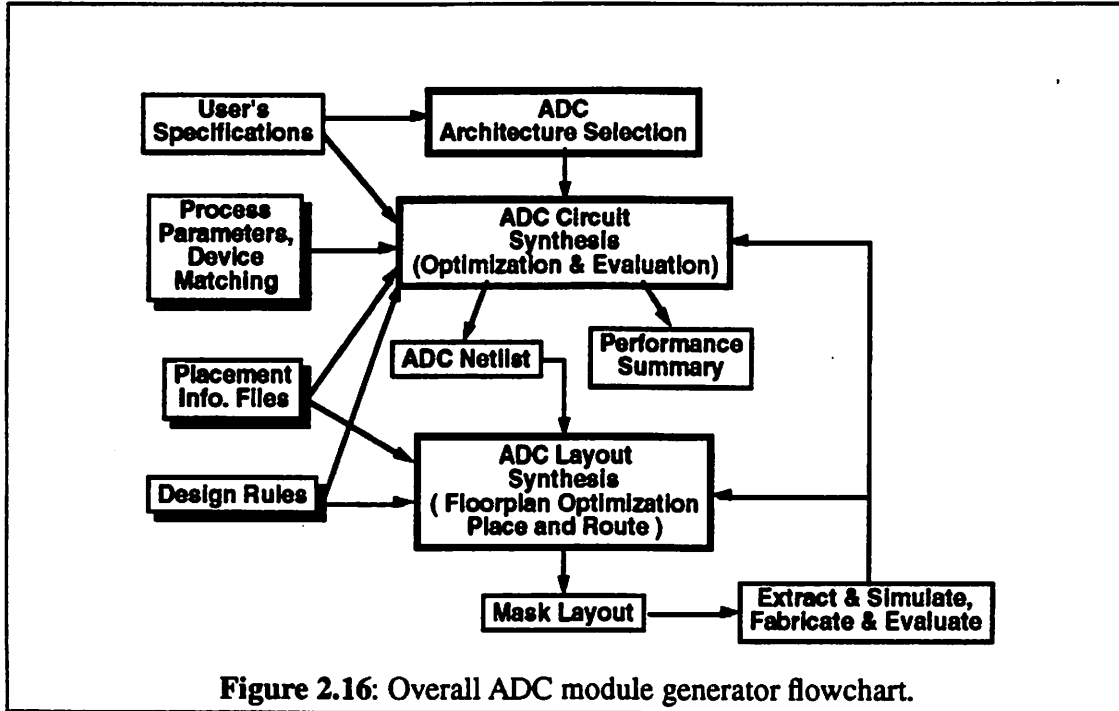


Figure 2.16: Overall ADC module generator flowchart.

blocks in mixed signal chips, we chose to study the implementation of ADC module generator.

CHAPTER 3 ADC Architectures Suitable for Automatic Compilation

3.1 Introduction

For A/D conversion functions, the realm of required performance maps into four distinct classes of realizations spanning the conversion rate spectrum (integrating, oversampling, successive approximation, and flash/pipeline architectures) and two distinct regions of accuracy or linearity (less than 10 bits and greater than 10 bits). Each of these requires a different implementation in order to be reasonably competitive with hand-crafted designs in terms of power dissipation and silicon area. Therefore, if *not carefully planned*, the generation of A/D conversion module generators will require the generation of *an extremely large number* of such module generators.

In the next section, five different ADC architectures (serial, successive approximation, flash, pipelined, and oversampling) will be briefly discussed. We will then point out the desired properties of ADC architecture from the module generation point of view. Finally, we will present an improved algorithmic A/D converter specifically designed to be implemented into our ADC module generator.

3.2 A/D Conversion Architectures

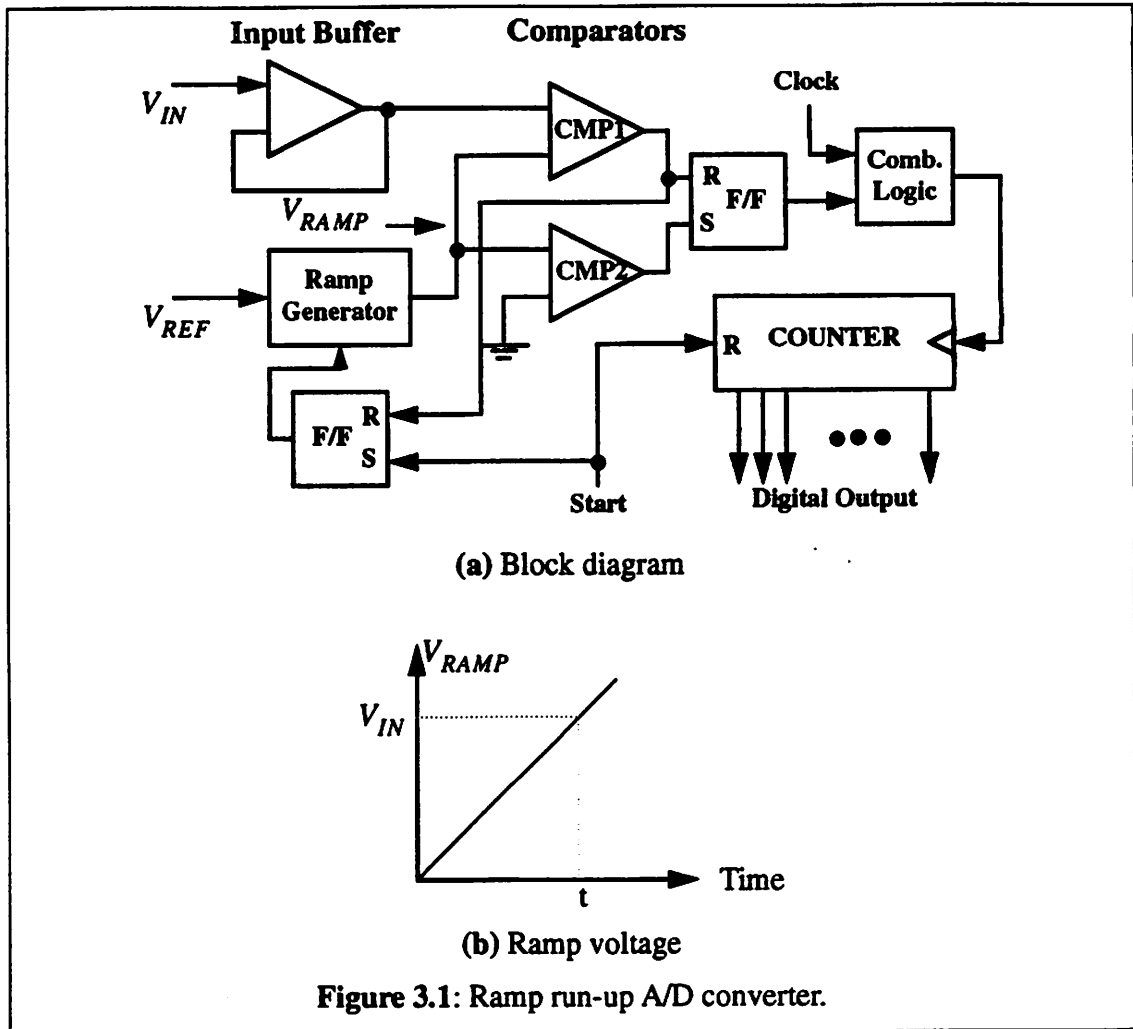
3.2.1 Serial A/D Conversion Technique

The serial A/D conversion technique has been used for many years. There are many different implementations of this technique such as dual slope, quad slope, charge balancing, voltage-frequency conversion, and others. Just to illustrate the idea, a ramp run-up implementation is shown in Figure 3.1. The basic operation of this implementation is as followed. When the *start* signal is high indicating the start of conversion, the ramp generator will start generating a ramp voltage V_{RAMP} . V_{RAMP} is then being compared to the input voltage, V_{IN} and the ground using two comparators: CMP1 and CMP2. As long as V_{RAMP} is larger than V_{IN} the counter will keep on incrementing its value. When V_{RAMP} is equal to or a little larger than V_{IN} at time t , the counter will stop incrementing and the ramp generator will stop ramping. The content of the counter will then be the digital representation of the analog input voltage.

The advantages of serial A/D conversion technique are that the implementation is very simple, the results are inherently monotonic, and linearity is independent of component matching. Unfortunately, this technique is very slow. As the number of resolution increases, the number of clock cycles per conversion increases exponentially as 2^N , where N is the number of bits desired. As a result, this technique is usually used in low frequency (10 to 100 Hz) applications such as general purpose digital volt meters (DMV's) and very low cost instruments.

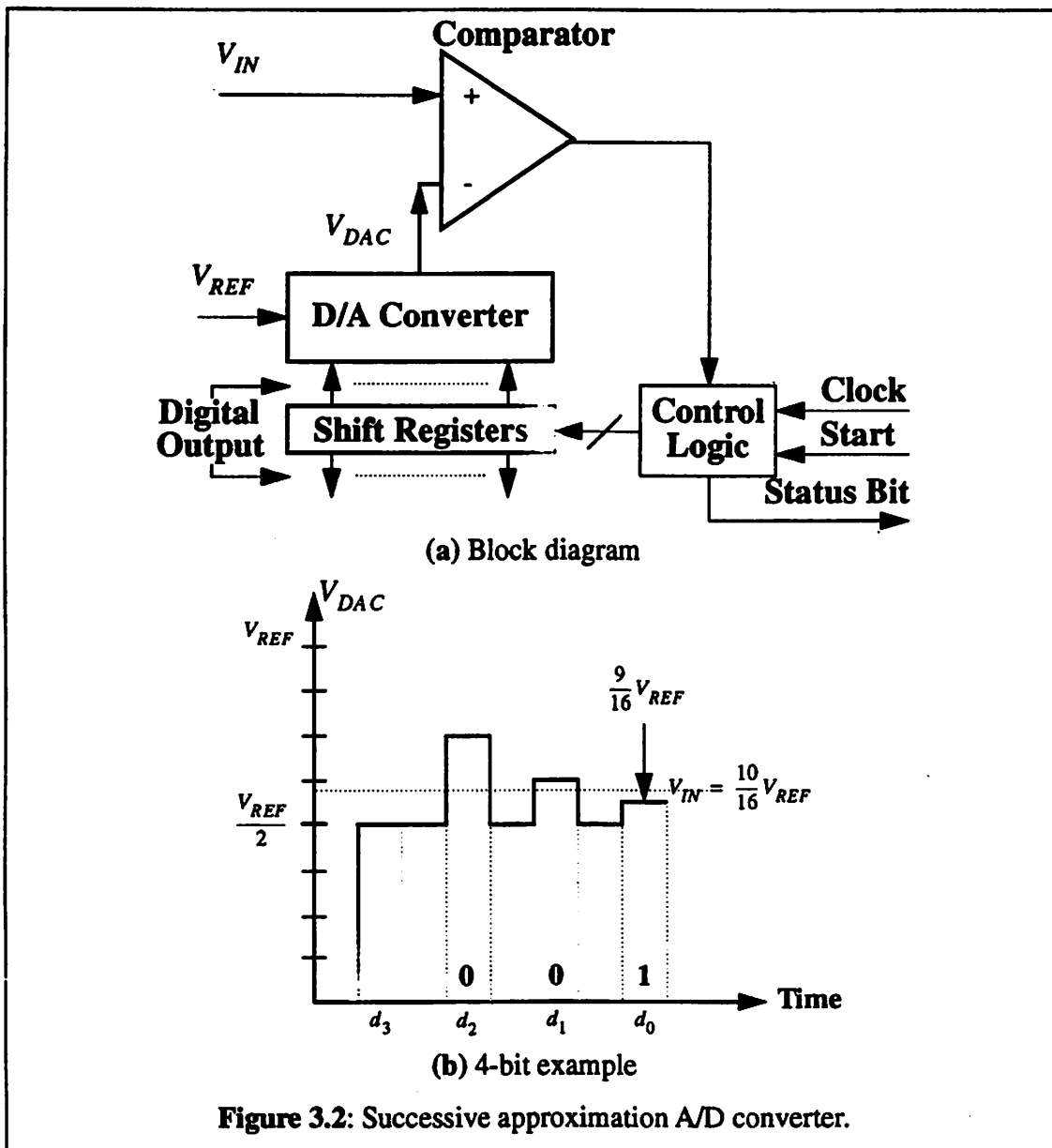
3.2.2 Successive Approximation A/D Conversion Technique

The successive approximation (SA) A/D conversion technique is one of the most commonly used technique to implement A/D converters often used to interface analog sig-



nals at voiceband and above with computer peripherals such as modems. The basic operation is to do binary search through 2^N voltage levels by using a precision digital to analog converter (DAC). A simple SA A/D converter consists several basic building blocks (Figure 3.2(a)) such as comparator, digital to analog converter (DAC), control logic, and shift registers for the output codes.

At the start of a conversion, all the registers will be clear and the output of the DAC will have zero voltage. At the first cycle, the most significant bit will be determined



first by comparing the input voltage V_{IN} to V_{DAC} . If V_{IN} is larger than V_{DAC} then the comparator output will output a “1” or else “0”. The output of the comparator is then fed into the control logic to determine if the digital output is zero or one. This digital output gets stored into the shift register which generates the digital code needed for the DAC to

do the binary search. At every iteration, 1 bit of digital code is obtained. Thus the operation is repeated N times to get N bits. Figure 3.2(b) illustrates what the DAC output level during the conversion for a 4-bit example.

This type of converter can give fairly high resolution and moderate conversion speed (10-100 kHz) since the conversion time is fixed and linearly increases as the resolution requirement increases. Moreover, the speed is independent of input voltage and previous result. In the example above, we are assuming that the input voltage V_{IN} varies very slowly and noise disturbance is low to cause significant error in the conversion. If this is not true, then a sample and hold function is needed to sample and hold the input voltage until the conversion is completed.

As mentioned above, the advantages of this particular A/D conversion technique are that it converts in N clock period where N is the number of desired bits and its building block is very simple. Unfortunately, the accuracy, linearity, and speed are affected by the DAC, V_{REF} , and comparator. For N -bit resolution, the DAC requires component to match to $\frac{1}{2^N}$ and the comparator offset voltage cannot exceed $\frac{V_{REF}}{2^{N+1}}$ for $\frac{1}{2}$ LSB linearity. Examples of this technique are the traditional DAC-based [MCCR75] and algorithmic [OHAR87] A/D converters.

3.2.3 Parallel A/D Conversion Technique

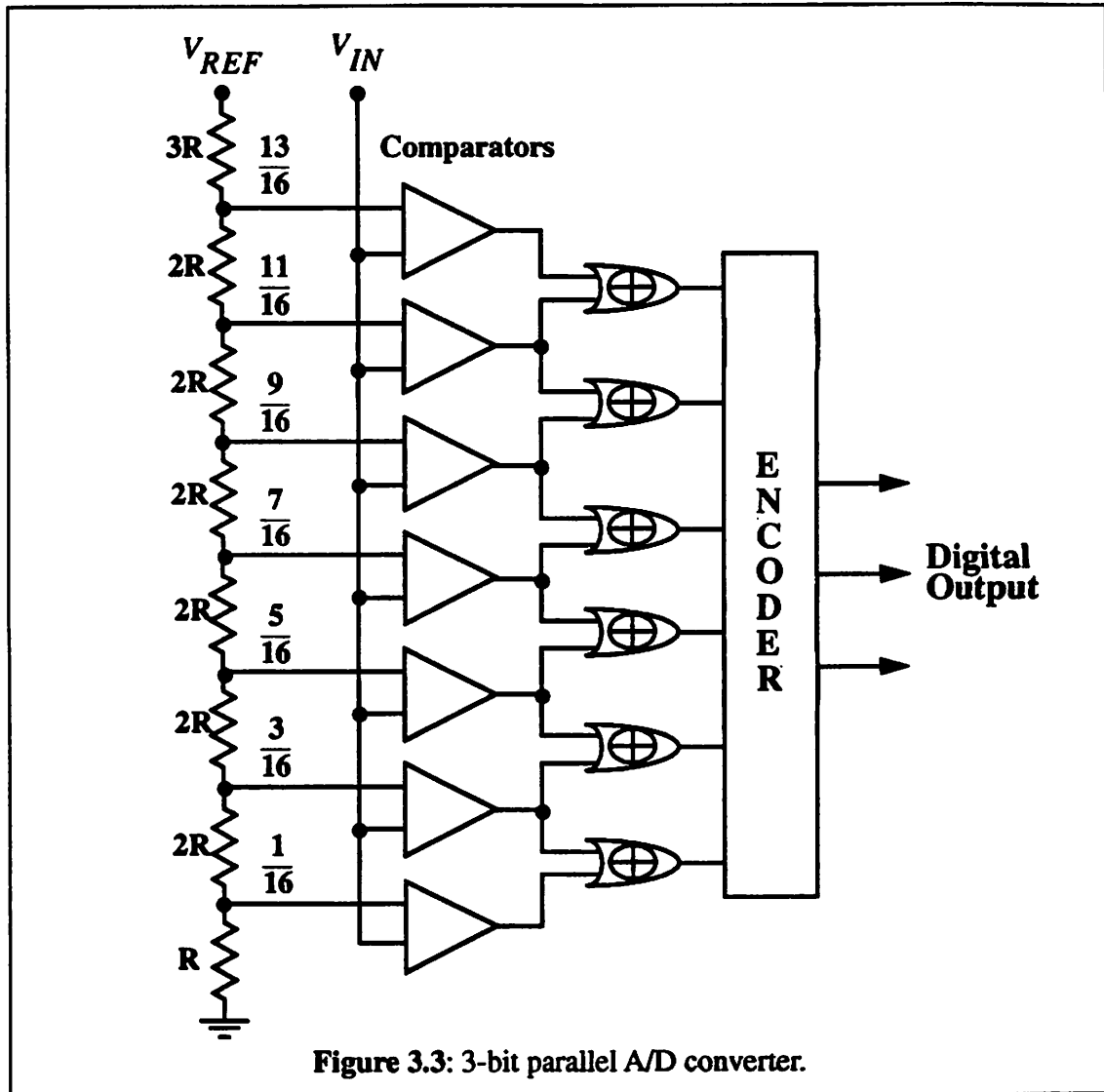
The parallel A/D converter, commonly known as flash A/D converter, is capable of converting in the shortest among different A/D conversion techniques. This type of converters is used mainly in video and radar processing with low resolution (up to 9 bits) and very high speed (10 MHz - 4 GHz). Figure 3.3 shows a 3-bit flash ADC block diagram with a resistive-DAC. The basic operation is as follows. An input voltage V_{IN} will be

compared to a bank of comparators that are tapped to a resistive-DAC for their threshold voltages. For a 3-bit resolution, usually it needs approximately $2^3 - 1$ comparators as shown in Figure 3.3. The outputs of the comparators are then got encoded to get 3 bit digital output codes.

The obvious advantage of this approach is that no clock is needed to do the conversion and the conversion speed is limited by how fast can the comparators and digital circuitry switch. Unfortunately, this type of implementation is very costly. As the number of resolution increases the hardware requirement and area increases exponentially. For example for a 9-bit A/D converter will need $2^9 - 1$ comparators with offset voltage of less than $\frac{V_{REF}}{2^{10}}$ and resistor string to match to 0.1%. Over the years different implementations have been proposed to reduce the area penalty of this type of converter such as two-step flash [DOER88] and pipelining several stages of fewer bit of flash converter to obtain the desired resolution [LEWI87].

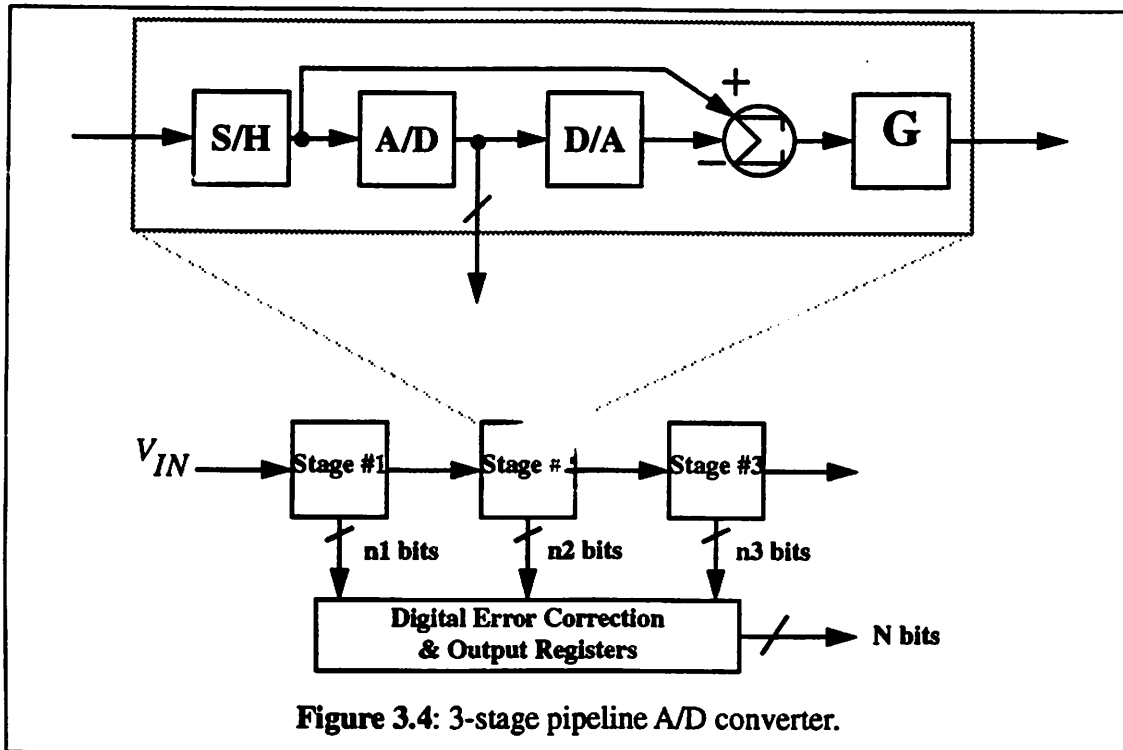
3.2.4 Pipelined Conversion Technique

The pipelined A/D conversion technique employs multi stages of A/D converter block to do the conversion concurrently. Each stage consists of a sample and hold block (S/H), an A/D converter, DAC, summer (Σ), and a gain block (G) as shown in Figure 3.4. The input voltage V_{IN} is sampled onto the S/H and is held while the first stage is doing the conversion of $n1$ bit digital output. The resulting digital output is then passed onto the DAC to generate a quantized voltage of the input signal which will be subtracted from the held input signal. The residual is then amplified and passed on to the next stage. While the next stage is converting this residual voltage, the first stage is sampling a new input voltage. The resulting digital output from all stages will be accumulated and passed



out together. Therefore, each conversion can be done within one clock period. As a result, the conversion speed is much faster than the SA and serial A/D conversion techniques and almost as fast as the flash A/D conversion technique.

The big advantage of pipelining is that much less hardware is needed to achieve the same throughput as flash. This is true when the number of bit is greater than 6 bit or else the overhead cost of doing pipeline stages is not worthy. Another advantage is that



digital correction and self-calibration can be applied so that high precision comparators are not required. It turns out that depending on the number of bits desired, there is an optimum number of bits per stage [LEWI87b]. The major disadvantage is that it requires fairly fast analog inter-stage processing and a sample and hold function to hold the analog signal.

3.2.5 Oversampled A/D Conversion Technique

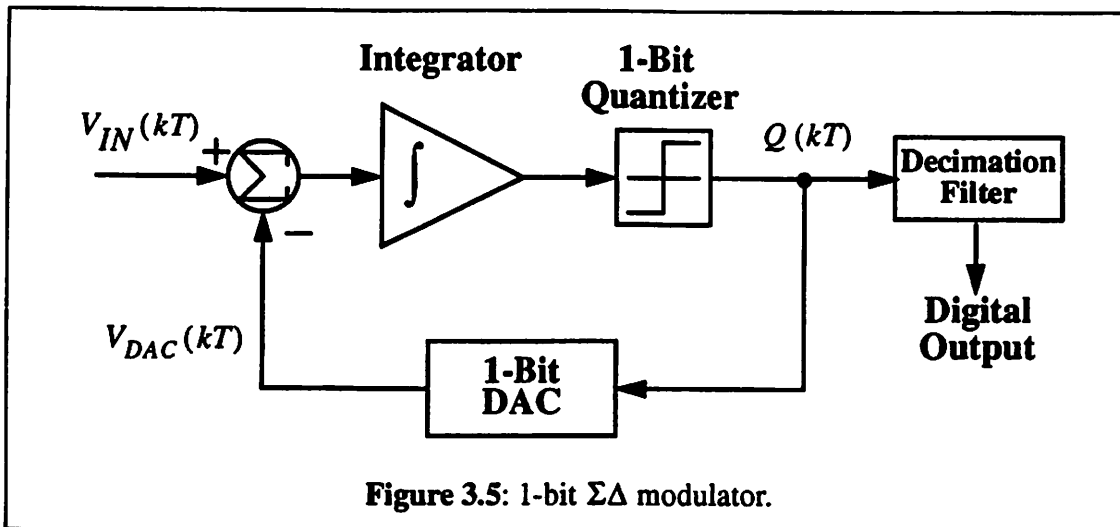
Unlike the A/D conversion techniques described earlier, oversampled A/D converters operate at non-Nyquist sampling rate (i.e. sampling at well above the Nyquist rate). The ratio of the sampling frequency to the Nyquist rate is commonly called *oversampling ratio*. The basic idea of this technique is to quantize an analog signal coarsely at a sampling rate much higher than the Nyquist rate of the signal. A filter in a feedback configuration is arranged to spread out the quantization noise in the entire frequency band. The

resulting coarse digital output is then passed through a digital processing block (decimation filter) to obtain more accurate digital representation of the given analog signal and therefore, filtering the out of band quantization noise.

One implementation of oversampled A/D converter is a 1-bit sigma-delta modulator shown in Figure 3.5. It consists of a summer, an integrator, a 1-bit quantizer, and a 1-bit D/A converter in a feedback loop and a decimation filter. The input voltage $V_{IN}(kT)$ is subtracted by the quantized analog voltage $V_{DAC}(kT)$ and then integrated by the integrator. The integrated difference is then quantized by a comparator to give a logic “1” or “0”. This digital output is then fed back into the DAC to generate the new DAC voltage. If the DAC is an ideal DAC, the error at the input to the integrator would be equal to the quantization error. Since the modulator is running at a much higher frequency than the Nyquist rate, the quantization error will depend on the oversampling ratio. Thus, the resolution of oversampled A/D converter will depend on the oversampling ratio. For example, sampling at two times the Nyquist rate will increase the SNR by 3 dB which translates roughly to an increase of $\frac{1}{2}$ LSB in overall accuracy [CAND81]. The final digital output can be obtained by feeding the digital output of the 1-bit quantizer into a decimation filter.

The big advantage of this approach is that component matching can be relaxed while maintaining linearity since a simple comparator can be used as the quantizer. The anti-aliasing filter needed at the front-end of the modulator can be implemented much cheaper since the transition band of the filter can be much wider than the one for Nyquist rate A/D converters. Furthermore, the majority of the building blocks can be implemented digitally so that integrating both the converter and the digital signal processing on to a single chip is very easy and cheap in CMOS technology.

Unfortunately, this technique has a lower throughput and consumes more area (due

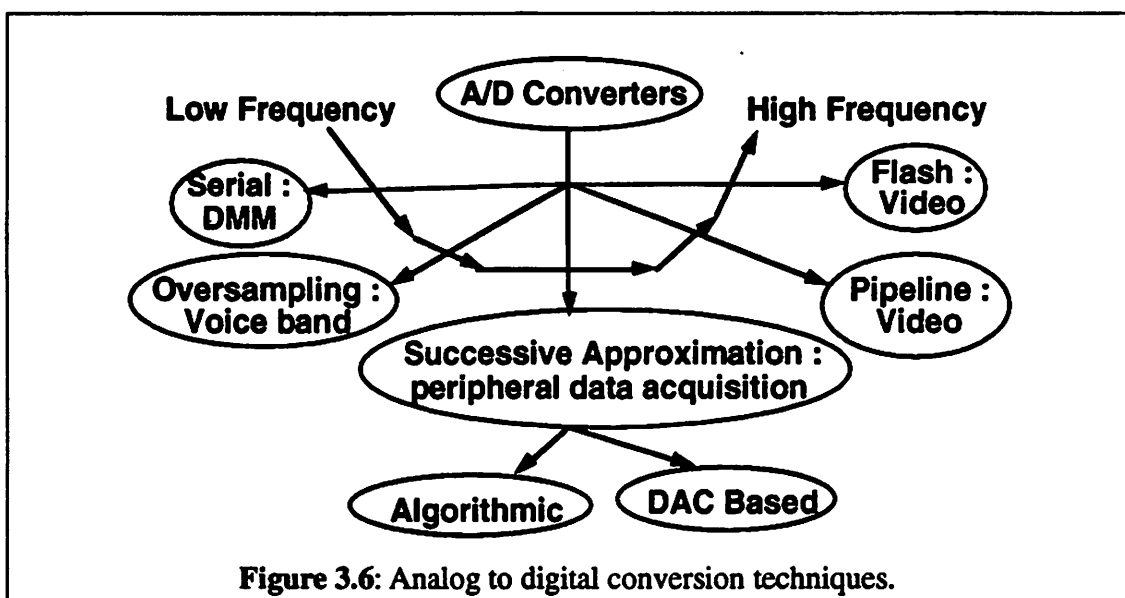


to extra digital processing block) than the successive approximation (SA) approach. As the conversion speed is getting faster, the sampling clock requirement becomes extremely high making it difficult to generate. There have been a great interest from the academics as well as commercial laboratories to produce high resolution A/D converters using this particular technique. Effort to improve the quantization noise reduction in the baseband in order to minimize oversampling ratio has led to using higher order modulators such as 2nd [CAND85], 3rd [MATS87], 4th [OPTE91], or even 5th order $\Sigma\Delta$ modulators. As technology feature size scales down, these higher order implementations become more economical due to higher density bringing this type of converter performance approaches those of SA.

3.3 Architecture Comparison

The five A/D conversion techniques can be summarized as follows. For a low frequency spectrum, the serial technique is very suitable. This technique is usually used for very low speed applications such as the digital multimeter (DMM). For low to mid fre-

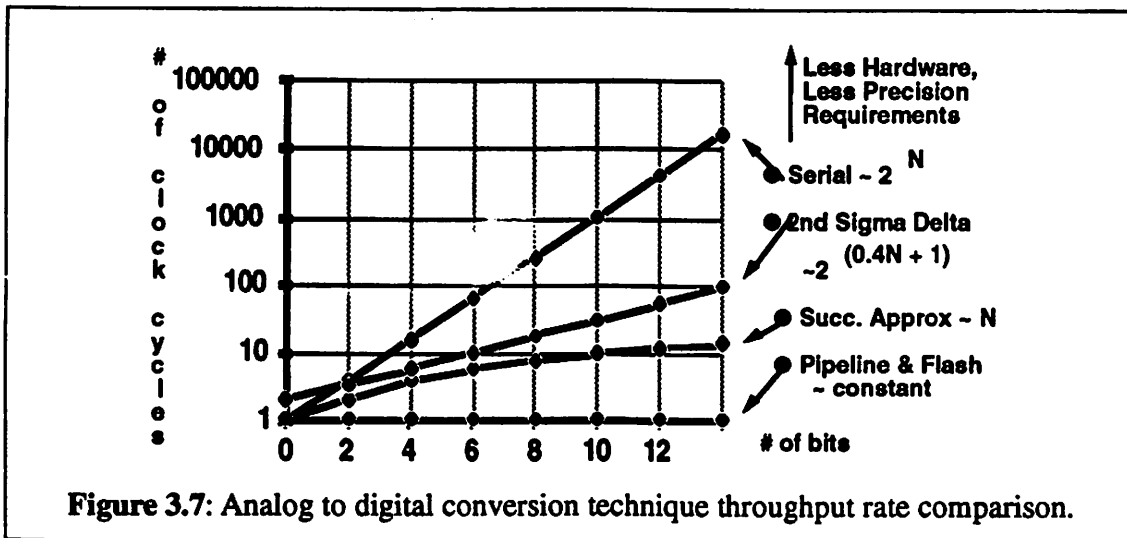
frequency range, A/D converters are usually implemented using oversampled or successive approximation techniques. These types of A/D converters are usually used for voiceband and peripheral data acquisition products. At the very high frequency range, two techniques: pipeline and flash are commonly used. These types of A/D converters are very suitable for video application products. The summary of different ADC techniques and their possible applications are shown in Figure 3.6.



The requirements of each technique are different in that each one has its own advantages over the other. For example, if we compare the number of bits versus the number of clock cycles needed to do the conversion, the serial A/D conversion technique requires the most number of clock cycles as the resolution increases. The number of clock cycles needed increases exponentially with exponential constant of N where N is the number of bits. For a second order sigma delta modulator, the number of clock cycles per conversion can be approximated to be $2^{(0.4N+1)}$. For successive approximation technique, the number of clock cycles needed grows linearly as the number of resolution increases.

Finally, for pipelined and flash techniques, the number of clock cycles stays constant independent of the number of resolution. Figure 3.7 compares the number of clock cycles needed versus the number of resolution for the five A/D conversion techniques.

Referring back to Figure 3.7, the pipelined and flash A/D conversion techniques are capable of the fastest conversion and the serial one is the slowest. However, there is trade-off among these techniques, in that the pipelined and flash techniques require high precision and more hardware requirements than the successive approximation technique. As we move from the bottom of the chart, the serial technique requires the least hardware and the least precision requirements; whereas, the flash technique requires the most hardware and the highest precision of them all. In other words, all the techniques mentioned above complement each other. Therefore, in order to make a complete ADC module generator, one has to cover all possible spectrum of frequency and resolution.



3.4 Desired Properties

As mentioned before, the number of required A/D conversion module generators can be reduced by careful selection of the architectural approach. For example, the use of binary-weighted capacitor approach [ALLE86] requires a complex self-calibration controller be added to the module for resolution above 10 bits. The technique also does not scale gracefully to video speed or to very high resolution. In contrast, the use of oversampled techniques [HAUS85], [BOSE88] at the low speed end, digitally corrected algorithmic technique in the mid-range [OHAR87], [JUSU90b], and the pipelined technique at the high speed end [LEWI87] potentially allows spanning of the entire speed spectrum with basic elements based on the same basic differential switched-capacitor integrator or amplifier block.

The research described here is aimed at the implementation of A/D converters in the *mid-range* frequency. We chose to study an algorithmic implementation and specifically have developed an improved version of it that *lends* itself to *automatic generation*. Details of the improved algorithmic A/D converter will be presented in the next section. The primary advantages of this particular converter implementation are:

- It is very area-efficient compared to other ADC approaches
- The trade-off between conversion rate and power can be made in a relatively straight forward way by varying the bias currents and device sizes in the active circuitry (such as comparators and operational amplifiers). The same comment applies to area since area and power couple very closely.

- The trade-off between resolution and area can be made in a straight forward way by scaling up/down the capacitor sizes to achieve the required $\frac{kT}{C}$ noise level and by adding the auto-calibration module if required.
- The linearity of the converter is dependent on only one parameter that can be easily adapted to satisfy requirements above 10 bits by adding a small module for self-calibration, poly fuse trim or EPROM trim.
- The technique uses a small amount of total capacitance which means that in single-poly technology, metal-poly capacitors can be used without a devastating area impact.
- It has an integral sample and hold function.
- The particular type of digital correction used allows the converter to achieve inherent monotonicity at high resolution without the need of trimming.

The properties mentioned above prove to be very important in that they simplify the implementation of the circuit generation presented in Chapter 4.

3.5 An Improved Algorithmic A/D Converter

In the rest of this chapter, we will discuss briefly the operation of an algorithmic A/D converter, its limitations, and solutions leading to an improved algorithmic A/D converter that is capable of converting at higher clock rate and insensitive to comparator offset voltages. Detailed discussion of the circuit implementation of the entire A/D converter can be found in Appendix A.

3.5.1 Algorithmic A/D Converter

The algorithmic A/D conversion technique is one way of implementing a successive approximation approach. The basic concept of an algorithmic A/D converter is to repetitively perform an operation on the signal in which the signal is compared to a set of references (usually one or two), and from that a coarse quantization decision is made. Then the analog voltage corresponding to that encoded level is subtracted from the signal, and the remainder is amplified and held. That remainder then becomes the new input signal to the block, and the entire operation is performed over again on the remainder. That process is repeated repetitively (*algorithmically*) until the A/D conversion is complete. Usually one bit is resolved on each pass through the process. A simplified algorithmic ADC block diagram shown in Figure 3.8 consists of a sample and hold (S/H) function, a gain of two (2X) block, a summer, a comparator, and digital circuitry to store the result as well as controlling the switches. The operation of the A/D converter is as follows:

At the first iteration, an analog input voltage V_{IN} is sampled onto the S/H block by connecting switch S_1 to the input node. The sampled voltage is then held by the S/H block and multiplied by two by the 2X block. The multiplied voltage V_M is then compared to the reference voltage V_{REF} . If V_M is larger than V_{REF} , the comparator will output a logic "1" and switch S_2 will be connected to V_{REF} subtracting V_M by the same amount. If V_M is less than V_{REF} , the comparator will output logic "0", switch S_2 will be connected to ground, and V_M will be passed without any subtraction. The resulting voltage V_e is then recirculated again through the same procedure until the desired number of bits is obtained. The entire conversion can be represented analytically as follows:

$$\text{Let } V_{\epsilon, i} = V_{M, i} - d_i V_{REF} \quad (\text{EQ 3.1})$$

Where $V_{\epsilon, i}$ = the residual voltage after the $(i+1)^{\text{th}}$ iteration

$V_{M, i}$ = the amplified voltage V_M after the $(i+1)^{\text{th}}$ iteration

d_i = the i^{th} digital output of the conversion (d_0 = MSB)

V_{REF} = the reference voltage

$$1^{\text{st}} \text{ iteration: } V_{M, 0} = 2V_{IN} \quad (\text{EQ 3.2})$$

$$V_{\epsilon, 0} = V_{M, 0} - d_0 V_{REF} = 2V_{IN} - d_0 V_{REF} \quad (\text{EQ 3.3})$$

$$2^{\text{nd}} \text{ iteration: } V_{M, 1} = 2V_{\epsilon, 0} = 2^2 V_{IN} - 2d_0 V_{REF} \quad (\text{EQ 3.4})$$

$$V_{\epsilon, 1} = V_{M, 1} - d_1 V_{REF} = 2^2 V_{IN} - [2d_0 + d_1] V_{REF} \quad (\text{EQ 3.5})$$

...

$$N^{\text{th}} \text{ iteration: } V_{M, N-1} = 2^N V_{\epsilon, N-2} \quad (\text{EQ 3.6})$$

$$V_{\epsilon, N-1} = 2^N V_{IN} - [2^{N-1} d_0 + 2^{N-2} d_1 + \dots + 2d_{N-2} + d_{N-1}] V_{REF} \quad (\text{EQ 3.7})$$

Dividing EQ. 3.7 by 2^N and solve for V_{IN} , we obtain:

$$V_{IN} = \left[\frac{d_0}{2} + \frac{d_1}{2^2} + \frac{d_2}{2^3} + \dots + \frac{d_{N-1}}{2^N} \right] V_{REF} + \epsilon_N \quad (\text{EQ 3.8})$$

Where $\epsilon_N = \frac{V_{\epsilon, N-1}}{2^N}$ = quantization error

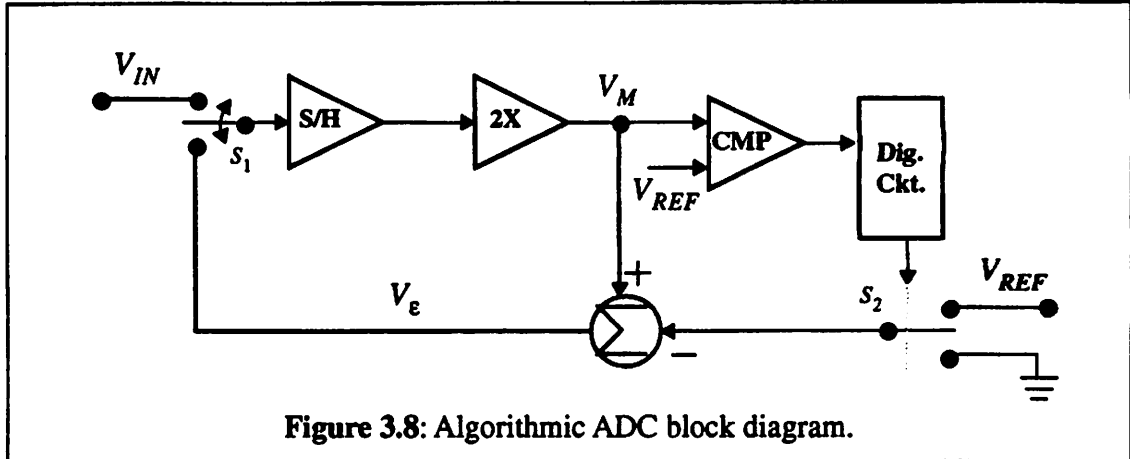


Figure 3.8: Algorithmic ADC block diagram.

From EQ. 3.8, we can obtain the lower and upper limits for the quantization error:

$$-\frac{V_{REF}}{2^{N+1}} < \epsilon_N < \frac{V_{REF}}{2^{N+1}} \quad (\text{EQ 3.9})$$

As long as ϵ_N is within this limit, the conversion result is guaranteed to have no missing code. In summary, EQ. 3.8 describes how an analog signal is represented in an algorithmic A/D conversion.

In spite of having many advantages as mentioned before, this particular implementations have many limitations. These limitations and their possible solutions are tabulated in TABLE 3-1. Two of the limitations: *moderate speed* and *comparator offset voltage* have question marks on the entries representing no acceptable solutions. In the rest of this section, we will discuss in great detail how to improve the conversion speed and how to avoid using high-precision comparators. Detailed derivations of all other limitations and solutions are presented in Appendix A.

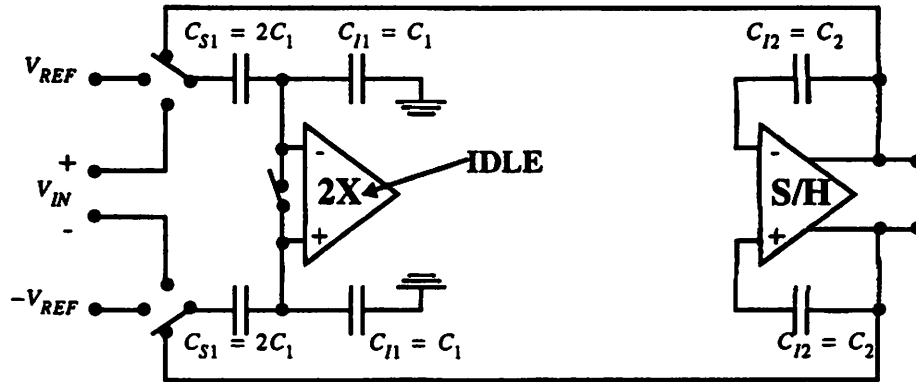
TABLE 3-1 Algorithmic ADC limitations and solutions.

Limitations	Solutions
Need an accurate gain of two: -> finite op-amp gain -> capacitor ratio matching	Folded cascode or telescopic architectures Trim array or calibration
Moderate speed (3 clock cycles/bit)	???
Sensitive to loop offset: -> op-amps -> switches -> comparators	Auto-zero circuit CMOS switches and fully differential arch. ???

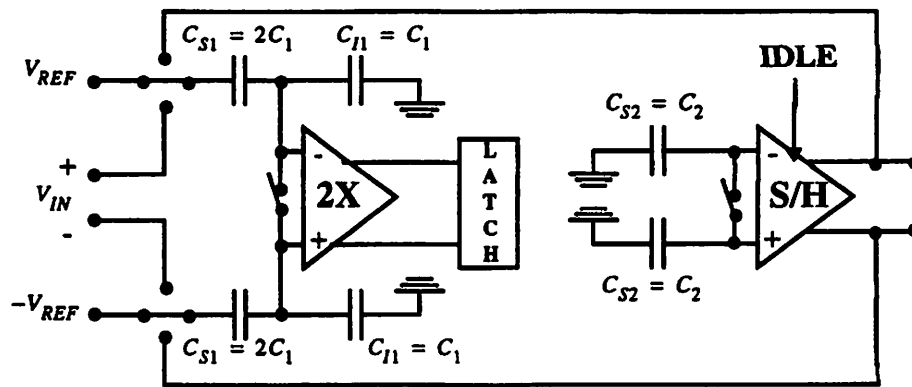
3.5.2 Speed Improvement Solution

All of algorithmic A/D converters previously reported in the literature [LI84], [SHIH85], [ONOD88], [OHAR87] require on average 3 clock cycles or more to resolve 1 bit. As a result, this type of implementation is capable of resolving signals of only moderate frequency. Let us analyze the conversion sequence of a typical algorithmic converter as shown in Figure 3.9. During voltage acquisition Figure 3.9(a), input voltage V_{IN} at the very first cycle or residual voltage V_{RES} in the remaining cycles is sampled on to the sampling capacitor C_{S1} . During this period, the op-amp of the gain of two is idling and the S/H block is holding the residual voltage. In the second phase (Figure 3.9(b)), the 2X block is used as a pre-amplifier for the latch and a decision bit is generated. At this time, the S/H is being reset by discharging its sampling capacitor C_{S2} . Notice that the op-amp of the S/H block is idling during this period. Finally, in the third period the residual voltage is generated and sampled on to C_{S2} for the next bit conversion as shown in Figure 3.9(c). The three operations get repeated N times for N resolution.

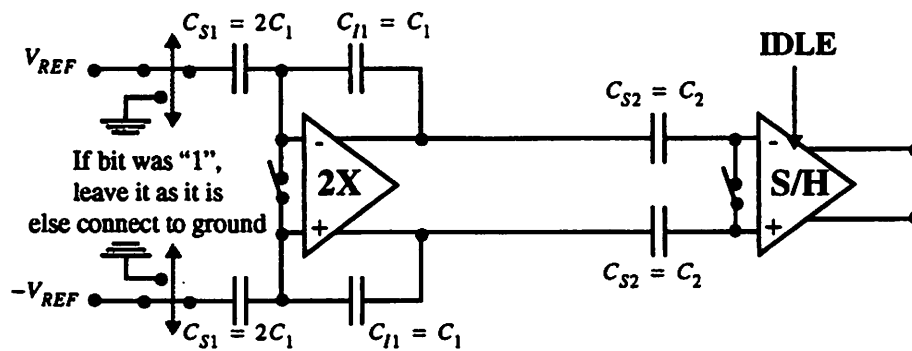
From Figure 3.9, we observe that some of the functions are idling during the conversion. By simply adding an extra S/H block, all the blocks can be fully utilized during



(a) Acquiring residual/input voltage mode



(b) Making bit decision mode



(c) Generating Residual voltage

Figure 3.9: Clocking schemes of a conventional algorithmic A/D Converter.

the conversion cycle. The idea is basically to time multiplex two S/H blocks so that the 2X block will never idle during the conversion. Shown in Figure 3.10 is a simplified version of how these blocks are connected to realize the proposed algorithmic A/D converter. The detailed clocking schemes of the proposed algorithmic A/D converter is shown in Figure 3.11. The operation of this proposed ADC is as follows:

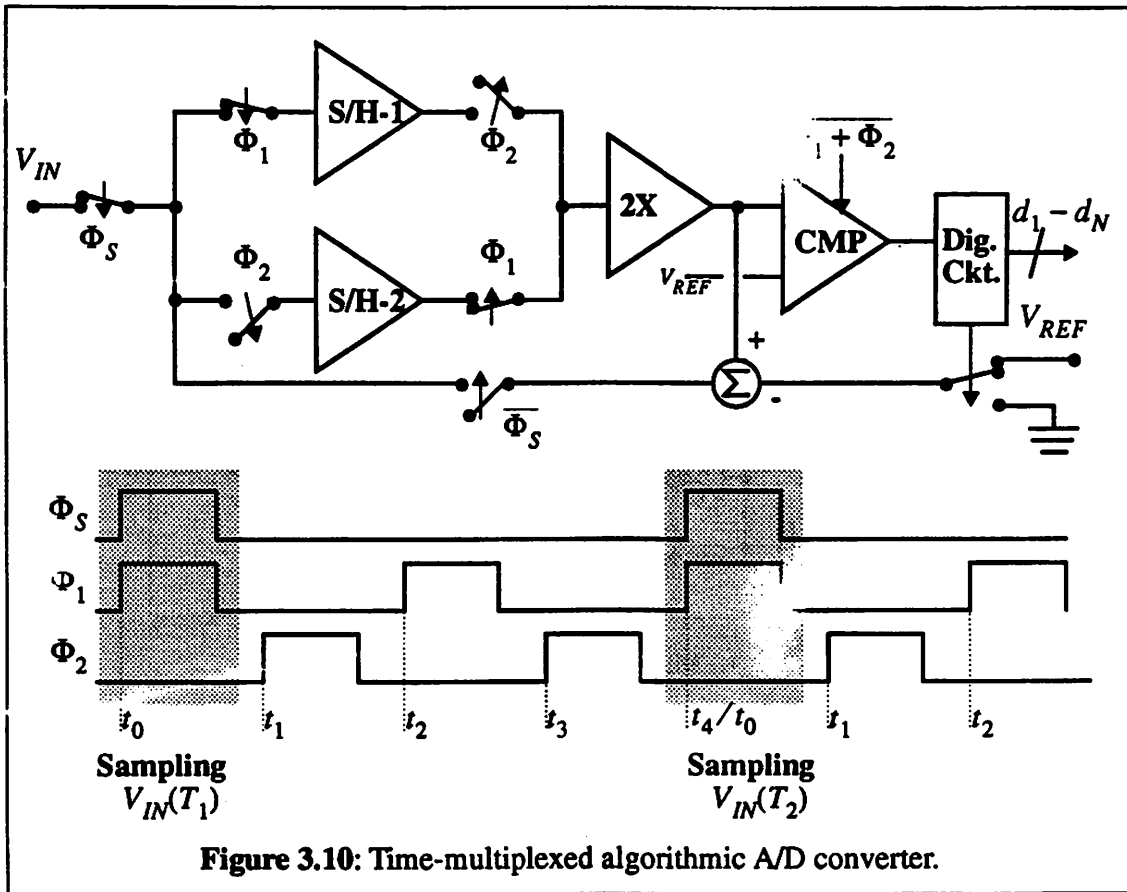


Figure 3.10: Time-multiplexed algorithmic A/D converter.

- **Mode 1:** $\Phi_S = \text{HIGH}$ in the first cycle and LOW for the rest of the conversion. $\Phi_1 = \text{HIGH}$ and $\Phi_2 = \text{LOW}$. During this period as illustrated in Figure 3.11(a), the residual voltage $V_{e,i}$ is sampled on to the S/H-1 block, while the S/H-2 and 2X blocks are in the hold and multiplication modes respectively

for $V_{e,i-1}$. The reference voltage V_{REF} will be added, subtracted or passed depending on the bit decision from the previous conversion. For the very first cycle, the S/H-1 block will sample V_{IN} , while the rest of the blocks are in the initialization or reset mode.

- **Mode 2:** $\Phi_S = \Phi_1 = \Phi_2 = \text{LOW}$. During in this period illustrated in Figure 3.11(b), the amplified residual voltage $V_{e,i-1}$ is compared with V_{REF} to determine the current bit. The result will be stored in a register and used as control signals for the reference subtraction circuits.
- **Mode 3:** $\Phi_S = \Phi_1 = \text{LOW}$ and $\Phi_2 = \text{HIGH}$. During this period illustrated in Figure 3.11(c), all the blocks experience the same tasks as during mode 1 except that S/H-1 is doing the holding for $V_{e,i}$ and S/H-2 is doing the sampling for $V_{e,i+1}$
- **Mode 4:** $\Phi_S = \Phi_1 = \Phi_2 = \text{LOW}$. This period is exactly the same as mode 2.

The conversion continue by repeating modes 1 to 3 until the desired number of bits is obtained. Notice that there is no block that is idle through out the conversion cycles. At the positive edge of every clock (either Φ_1 or Φ_2), one bit will be resolved. Thus the proposed algorithmic A/D converter is capable of resolving 2 bit per one clock cycle of Φ_1 or Φ_2 . We would like to call it a 1-bit/cycle algorithmic A/D converter because in order to generate two non-overlap clock Φ_1 and Φ_2 of period T, two periods of the master clock are needed. Figure 3.12 shows the complete time division of this 1-bit/cycle algorithmic

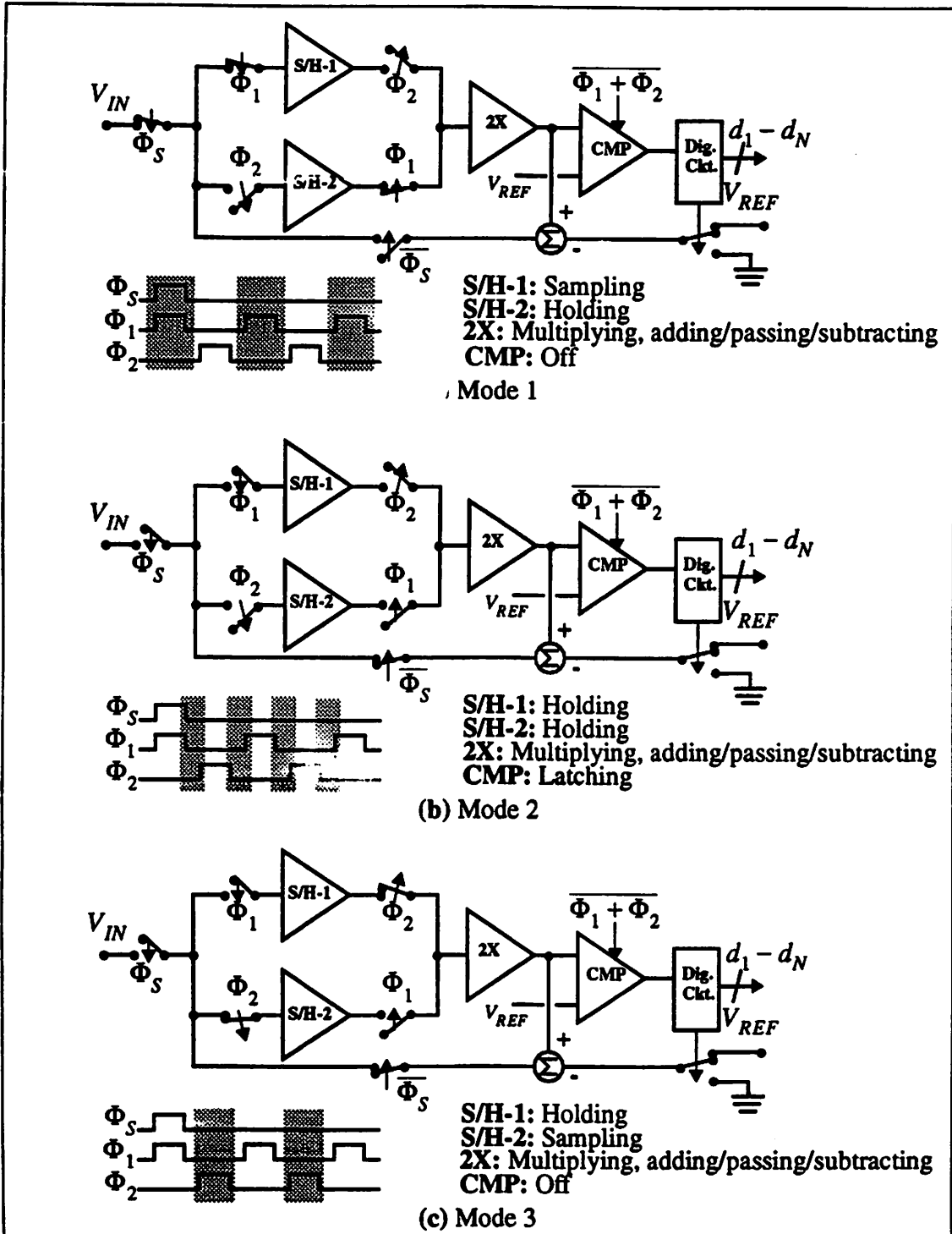
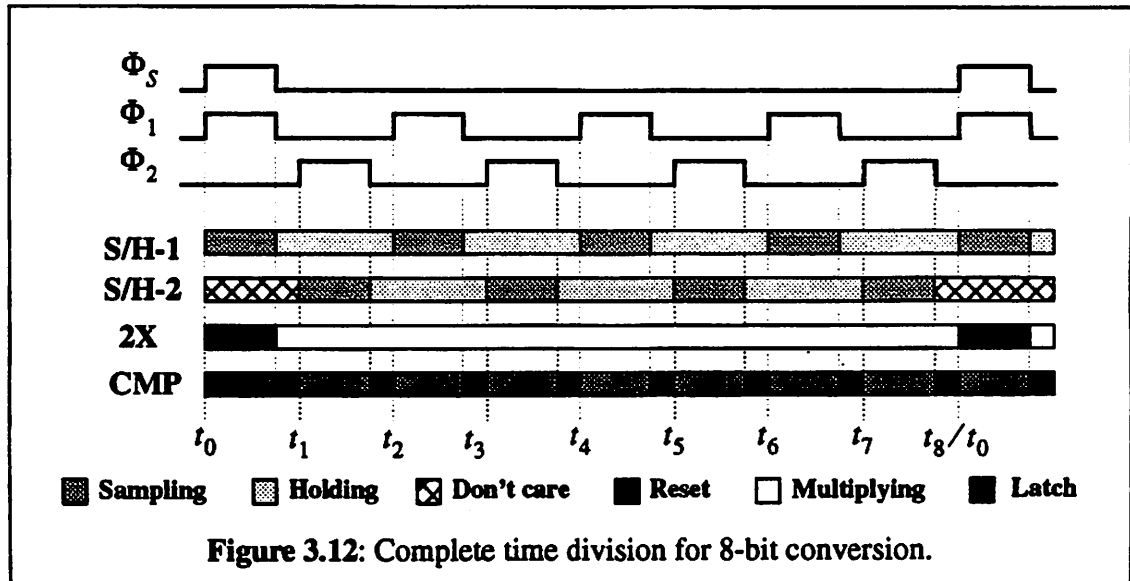


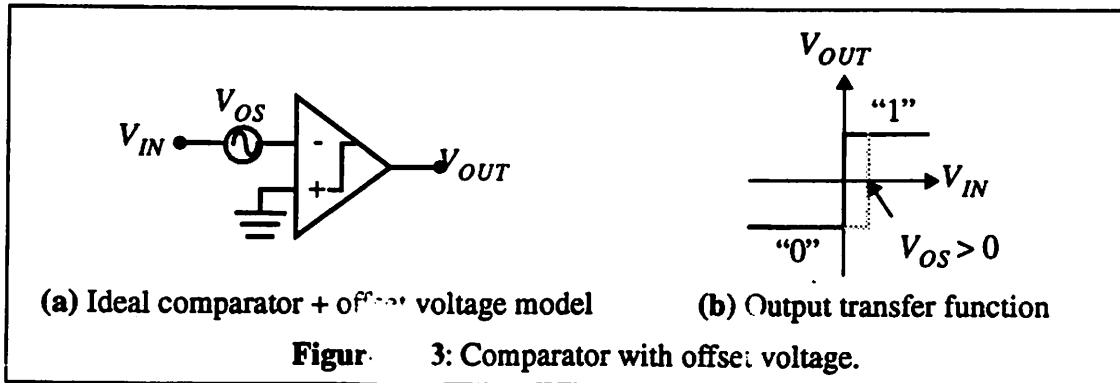
Figure 3.11: Clocking schemes of the proposed algorithmic A/D converter.

A/D converter converting an input signal into 8 digital bits. We can now fill in the solution of moderate speed limitation in TABLE 3-1 with this improved algorithmic ADC architecture.



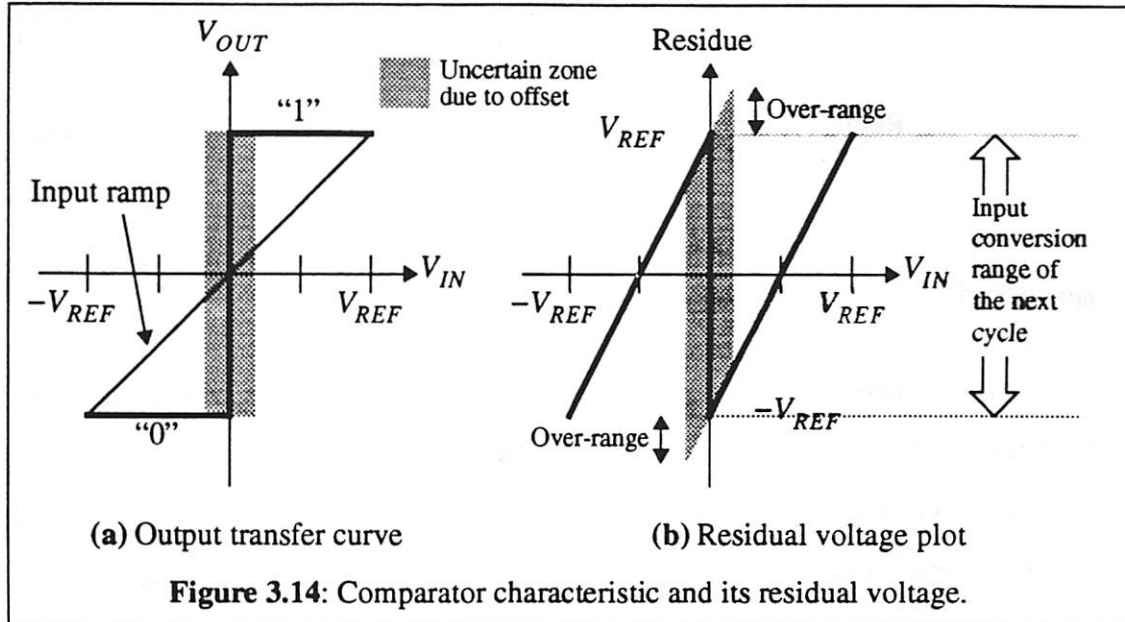
3.5.3 Comparator Offset Solution

The overall linearity of an algorithmic A/D converter depends not only on the gain of two accuracy but also on the performance of the comparator used. Offset voltage of comparator which is due to device mismatch and process variation in particular affects directly to the overall linearity of the converter. A comparator with an offset voltage V_{OS} can be modelled as an ideal comparator in series with a voltage source of V_{OS} as shown in Figure 3.13(a). The transfer characteristic of the comparator output with respect to the given input voltage V_{IN} is shown in Figure 3.13(b). For ideal comparator, the transfer curve is a solid line with zero crossing at $V_{IN} = 0$. If $V_{OS} > 0$ the zero crossing will shift to right as shown in Figure 3.13(b) by a dashed-line.



If the offset voltage of the comparator exceeds 1 LSB of the converter, a wrong digital output will be obtained. Figure 3.14 illustrates the effect of offset voltage to the residual voltage. If the comparator is ideal, ramping the input voltage V_{IN} from $-V_{REF}$ to V_{REF} will yield a residual plot shown by solid line in Figure 3.14(b). Let us assume that the comparator has finite offset voltage either positive or negative then the output transfer curve will be shifted to the right or left as shown by the shaded region in Figure 3.14(a). In turn, the zero crossing of the residual will also get shifted to the right or left (represented by shaded region in Figure 3.14(b)). As a result, the residual voltage will exceed the input conversion range of the next conversion cycle. If the amount of over-range voltage exceeds 1 LSB, it will saturate the next bit conversion and produce missing codes.

There are several ways to solve the comparator offset error. The first one is to design comparators with small enough offset [SOO85], [SAUL82], [ALLS82]. Often such a comparator requires complicated circuitry and consumes fairly large area. The second one is to store the comparator offset in a capacitor before the comparison cycle [DOER88], [YUKA85]. The solutions mentioned above are suitable for a manual design in which one just has to design a comparator for a particular set of specifications. Since our goal is to generate a wide range of A/D converters, it will be very desirable if we can use a simple comparator circuit for different specifications so that we can simplify the



implementation of automatic ADC circuit generation. We have designed a 2-comparator scheme with digital error correction circuit to enable us to use low precision comparators making it very desirable for automatic as well as manual design of A/D converters.

3.5.3.1 Two-Comparator Scheme and Digital Error Correction

As mentioned earlier, the residual voltage may saturate the next bit conversion and produce missing codes if it is too large. But if the conversion range of the next stage is increased to allow larger residual voltages, the results are decoded and the errors are corrected, comparator offset requirements can be relaxed or even ignored. This concept is called digital error correction [LEWI87] and was first introduced by [HORN72].

Borrowing this idea, we propose a two-comparator scheme with digital error correction to take care of the comparator offset. With a one-comparator scheme, the residual voltage will fall into 2 regions and will have characteristic curve according to these two equations.

$$\text{Region 1: if } V_{IN} \geq 0 \text{ then residue} = 2 \left(V_{IN} - \frac{V_{REF}}{2} \right) \quad (\text{EQ 3.10})$$

$$\text{Region 2: if } V_{IN} < 0 \text{ then residue} = 2 \left(V_{IN} + \frac{V_{REF}}{2} \right) \quad (\text{EQ 3.11})$$

Thus, any offset voltage from this comparator will saturate the input conversion of the next cycle.

With a two-comparator scheme, for every cycle, one and a half bits are resolved and only one bit is stored. The residual voltage, thus, has *three* possible regions to reside instead of two. Figure 3.15 illustrates how the two comparators are connected and the respective transfer function characteristic with $\pm V_T$ as their threshold voltages.

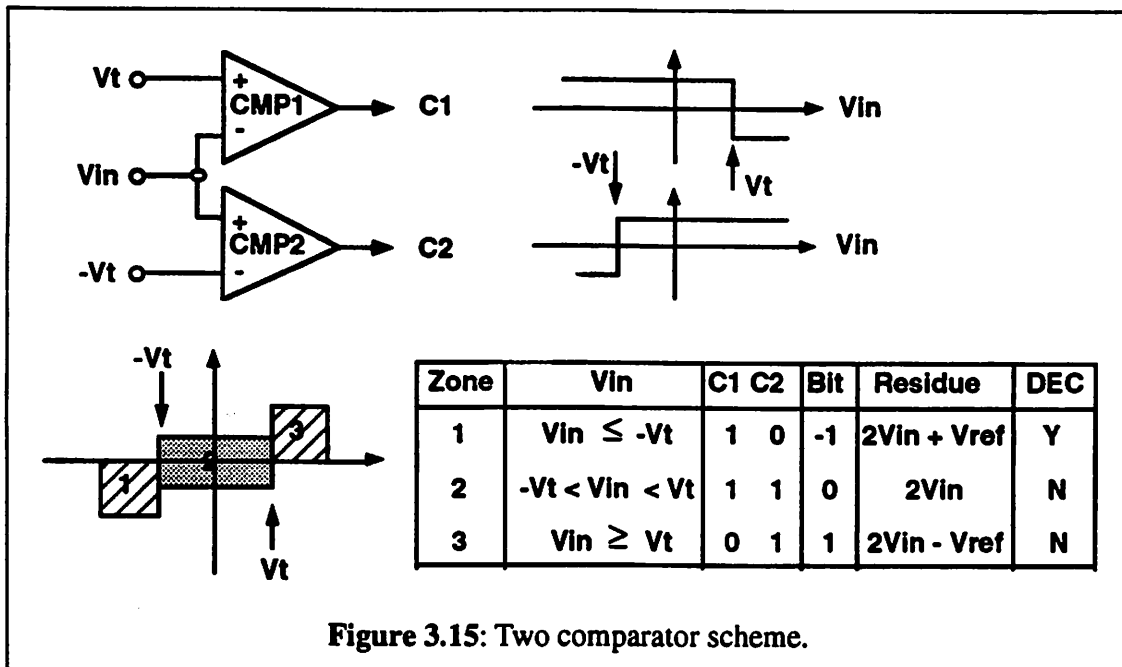


Figure 3.15: Two comparator scheme.

$$\text{Region 1: if } V_{IN} \leq V_T \text{ then residue} = 2V_{IN} + V_{REF} \quad (\text{EQ 3.12})$$

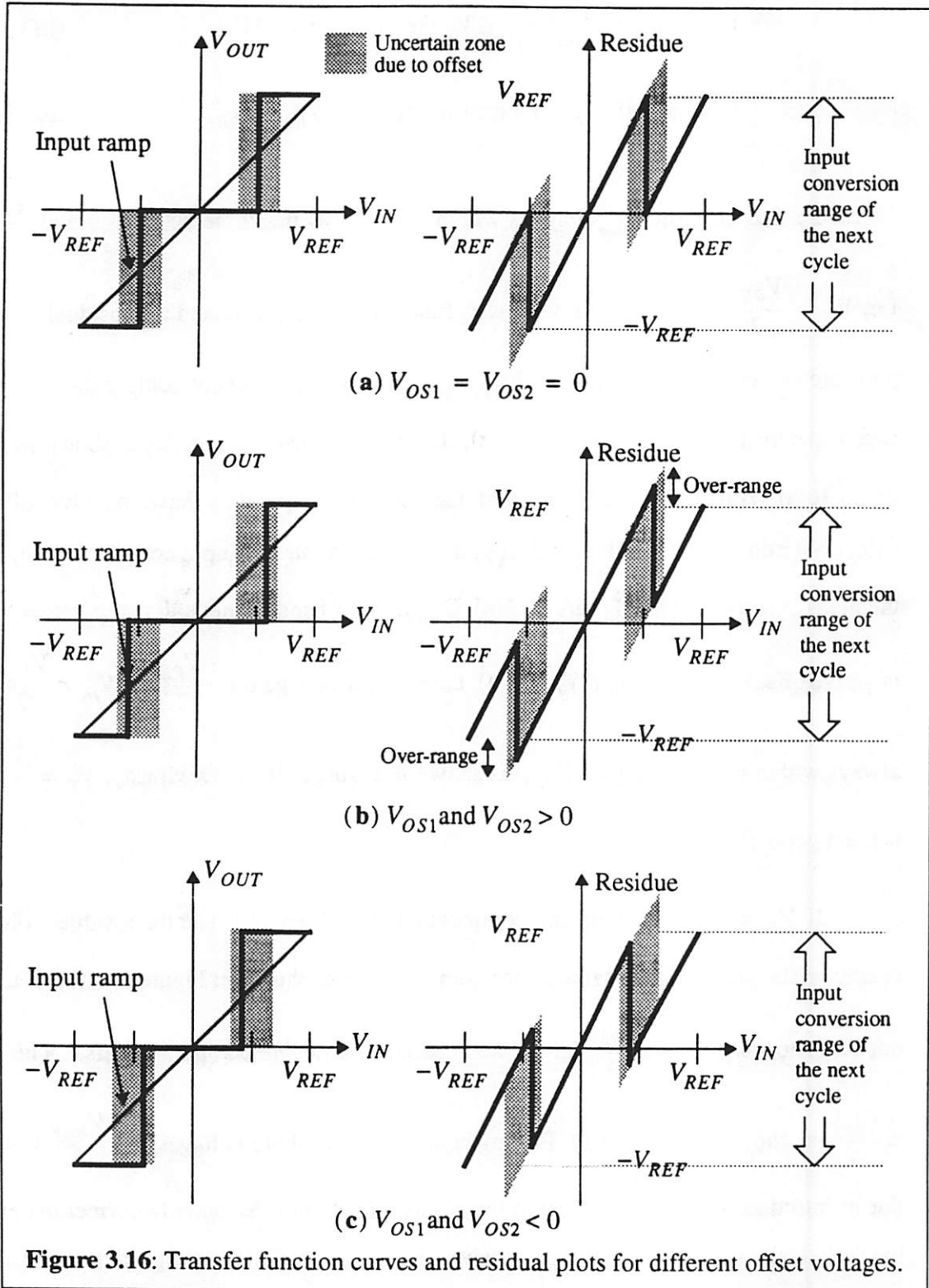
$$\text{Region 2: if } -V_T \leq V_{IN} \leq V_T \text{ then residue} = 2V_{IN} \quad (\text{EQ 3.13})$$

$$\text{Region 3: if } V_{IN} \geq V_T \text{ then residue} = 2V_{IN} - V_{REF} \quad (\text{EQ 3.14})$$

To ensure that the residual voltage is within $\pm V_{REF}$, V_T has to be between 0 and $\frac{V_{REF}}{2}$.

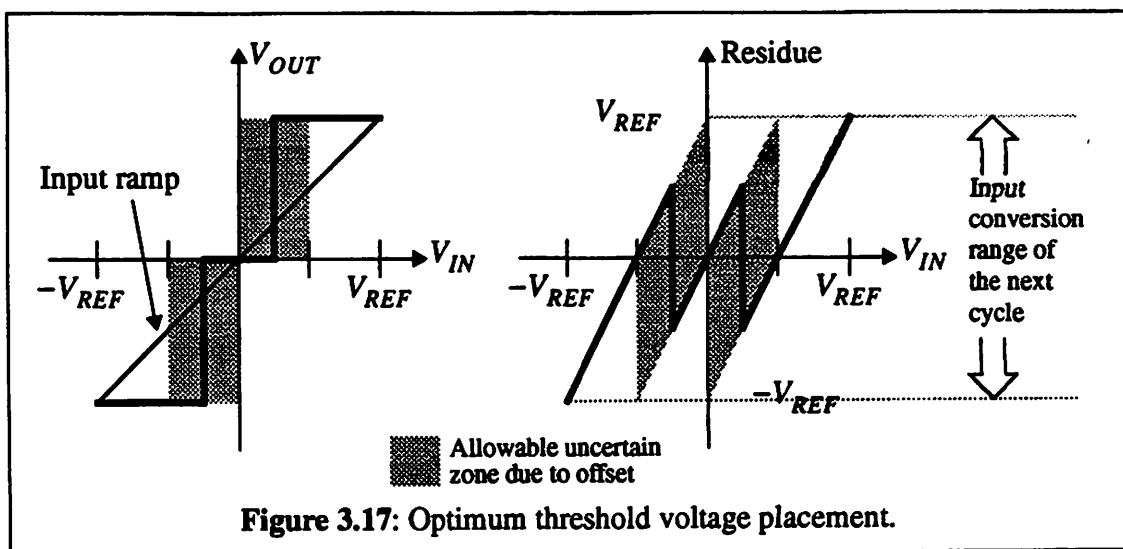
Let $V_T = \frac{V_{REF}}{2}$. The comparator transfer function characteristic and the residual voltage plots are shown in Figure 3.16. Let V_{OS1} and V_{OS2} be the offsets of comparators one and two respectively. For $V_{OS1} = V_{OS2} = 0$, the ideal residual voltage plot is shown in Figure 3.16(a). However, if the first and the second comparators have positive offsets ($V_{OS1} > 0$ and $V_{OS2} > 0$), the residual voltage will saturate the input conversion range of the next cycle as shown in Figure 3.16(b). On the other hand, if the both comparators have negative offsets ($V_{OS1} < 0$ and $V_{OS2} < 0$), the residual voltage for $-\frac{V_{REF}}{2} < V_{IN} < \frac{V_{REF}}{2}$ is always within the boundary of V_{REF} as shown in Figure 3.16(c). Obviously, $V_T = \frac{V_{REF}}{2}$ is not a good choice.

If $V_T = 0$, then having two comparators are redundant, since the residual voltage characteristic is exactly the same as one comparator case shown in Figure 3.14(b). It turns out that choosing $V_T = \frac{V_{REF}}{4}$ gives the *maximum allowable* comparator offset which is $\pm \frac{V_{REF}}{4}$ as shown in Figure 3.17. Placing comparator threshold voltages at $\pm \frac{V_{REF}}{4}$ is similar to introduce *intentional offsets* in the comparators and subsequently correct the error by digital circuit. If the comparators have offsets, the residual voltage will always be



within the input conversion range of the next cycle as long as the offsets are within

$$\frac{-V_{REF}}{4} \leq V_{OS1}, V_{OS2} \leq \frac{V_{REF}}{4}$$



By having such a large range of allowable comparator offsets, no high precision comparators are needed with this 2-comparator scheme. The only penalty is the requirement of an additional digital error correction circuit to correct the error; fortunately, this digital error correction circuit is very simple. It will be shown in Appendix A that the digital error correction circuit consists only of simple adders and multiplexers. The conversion algorithm for the N-bit conversion with digital error correction is as follows:

- At the beginning of the conversion, the sign is always assumed positive i.e. $d_0 = 1$ ($d_0 = \text{MSB}$)
- The rest of the conversion is to resolve the remaining N-1 bits. Each subsequent cycle, the comparator outputs, C1C2, are decoded to obtain the current

bit, reference subtraction control signals, and correction information. The decoding and correcting decisions are as follows:

- If V_{IN} is in region 3 i.e. $C1C2 = 01$, then the current bit is "1".
- If V_{IN} is in region 2 i.e. $C1C2 = 11$, then the current bit is "0".
- If V_{IN} is in region 1 i.e. $C1C2 = 10$, then the current bit is "1" but the earlier conversion bits are 1 bit too large. Therefore, the earlier conversion bits need to be subtracted by one.

To verify the 2-comparator scheme, two examples of a 3-bit conversion will be derived.

Let $N = 3$ bits, $V_{REF} = 2$ volts. $1 \text{ LSB} = \frac{2V_{REF}}{2^N} = 0.5$ volts and comparator thresholds

are at $\frac{\pm V_{REF}}{4}$. The first example has an input voltage $V_{IN} = \frac{3}{16} V_{REF}$. The expected digital

output code and the conversion cycles are illustrated in Figure 3.18(a). Notice that in this example no digital error correction is needed. The second example has an input voltage

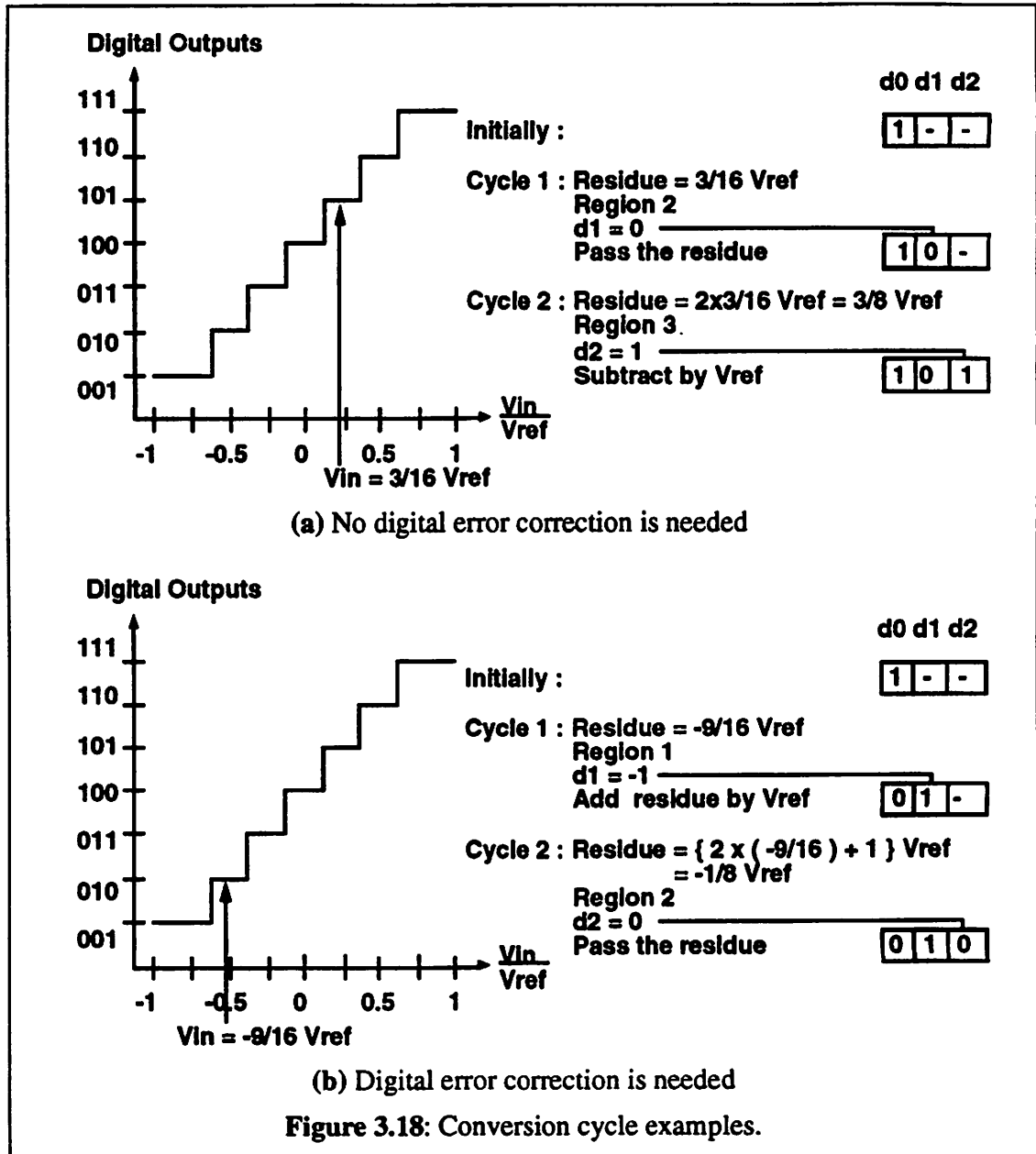
$V_{IN} = \frac{-9}{16} V_{REF}$. The expected digital output code and the conversion cycles are

shown in Figure 3.18(b). For this particular example, digital error correction is needed.

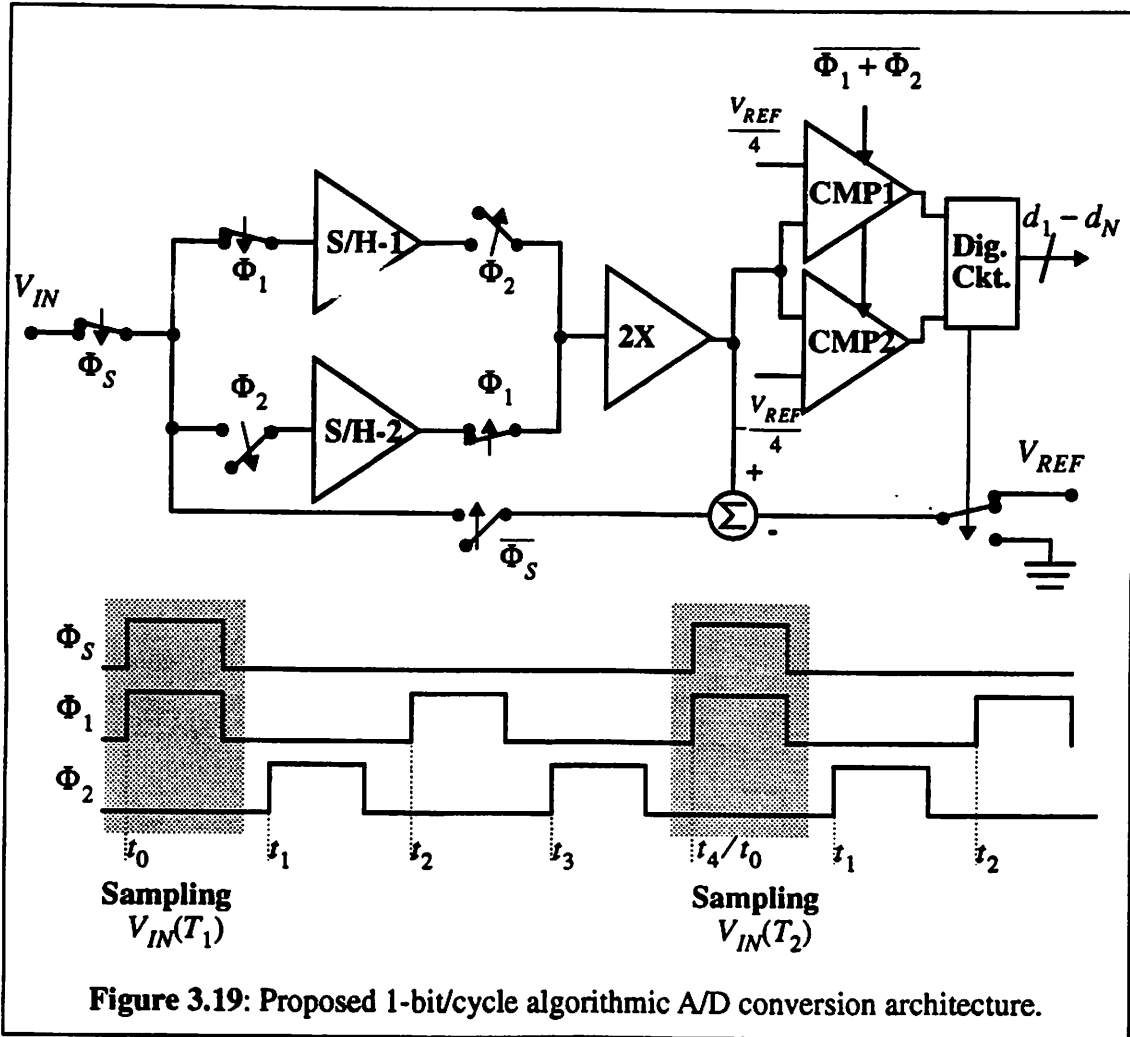
Clearly, the resulting digital outputs of the two examples above agree with the expected digital output codes.

3.5.4 Proposed 1-Bit/Cycle Algorithmic A/D Converter

Combining the new algorithmic conversion concept and the 2-comparator scheme introduced in section 3.5.2 and section 3.5.3, a *1-bit/cycle algorithmic A/D converter without high precision comparators* is proposed. This improved architecture will not only yield a higher conversion rate but will also allow the designer to use simple comparators. The block diagram of this proposed architecture is shown in Figure 3.19. The detailed implementation issues of each functional block can be found in Appendix A. We can now



fill in the question mark entries of TABLE 3-1 with the solutions described above. TABLE 3-2 tabulates a complete algorithmic ADC limitations and solutions.



3.6 Summary

In this chapter, a brief discussion about architectural study has been presented. Different A/D conversion architectures with their advantages and disadvantages are compared. In order to limit the number of module generators needed, careful selection of A/D conversion architecture is very crucial. As the first implementation, we have focused on an algorithmic A/D conversion implementation. A new improved algorithmic A/D converter has been implemented in order to broaden the spectrum resolution as well as conversion

rate. This algorithmic A/D converter is capable of converting signal at three times faster than the conventional one. With the use of two-comparator scheme, low precision comparator block can be used for a wide range of resolution and conversion rates. With these extra properties, the improved algorithmic A/D converter fits nicely for the automatic module generation. In the next two chapter, a detailed discussion of automatic netlist and layout generation will be presented.

TABLE 3-2 Complete limitations and solutions of algorithmic A/D converter.

Limitations	Solutions
Need an accurate gain of two: -> finite op-amp gain -> capacitor ratio matching	Folded cascode, telescopic, etc. arch. Trim array or calibration
Moderate speed (3 clock cycles/bit)	Time multiplexed 2 S/H blocks
Sensitive to loop offset: -> op-amps -> switches -> comparators	Auto-zero circuit CMOS switches and fully differential arch. 2-comparator scheme + digital error correction

ADC Architectures Suitable for Automatic Compilation

CHAPTER 4 ADC Circuit Generation

4.1 Introduction

In section 2.4.4, we have discussed that the parameterized schematic of known architecture circuit generation is a better approach because it can generate highly-effective circuits based on known architectures for complex analog functions through optimization of device sizes with respect to various performance measures. In this chapter, the automatic generation of ADC netlist generation based on this approach will be discussed. The overall flowchart of the module generation is again shown in Figure 4.1. The discussion will be formatted as follows. First the architectural selection process and different circuit synthesis approaches will be discussed. The implementation of the chosen circuit synthesis approach will then be presented in depth.

4.2 Architectural Selection

In section 3.4, we have listed the desired properties for an ADC technique to be considered a candidate of a building block in a module generator. We also mentioned that

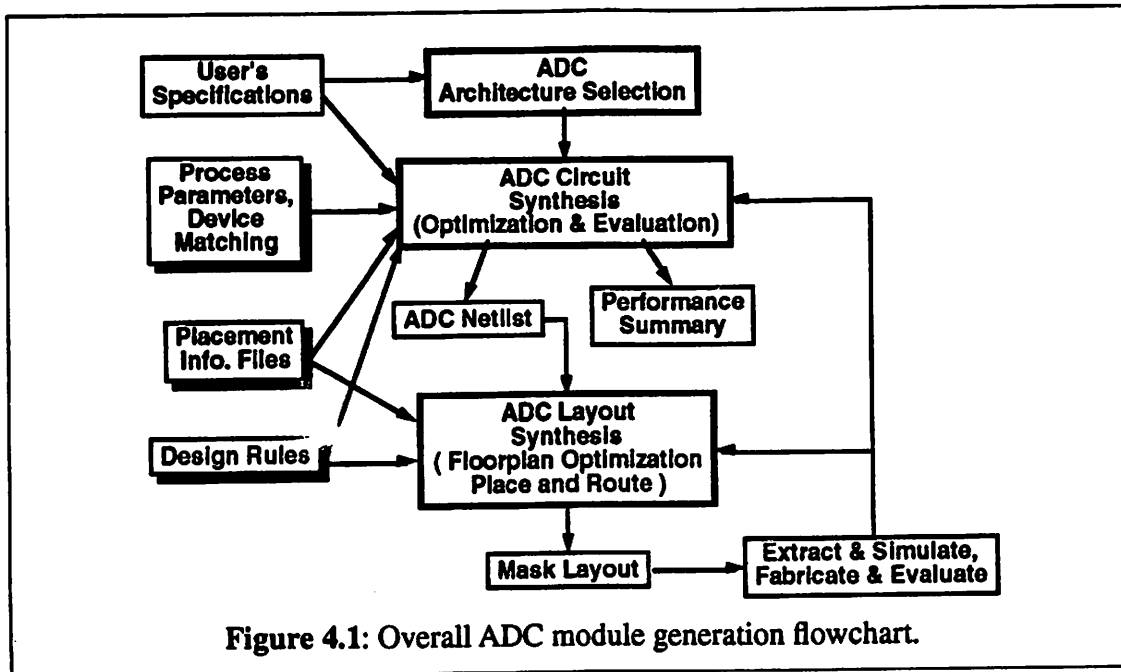


Figure 4.1: Overall ADC module generation flowchart.

as the *first step* we decided to concentrate on an ADC implementation in the mid frequency range. We have chosen and specifically designed an algorithmic ADC as our first building block in the module generator. Since this is the only available ADC topology, architectural selection routine has not been implemented. In order to complete the discussion, we will describe what could and should be carried out during this selection process.

The main function of an architectural selection step is to determine which ADC architecture is the most suitable to be generated first for a given set of specifications. This step is carried out in order to speed up the ADC generation without having to try out every possible ADC block in the database. This selection process can be done by pruning through sets of rules, previous runs of the module generator, exhaustive search through all existing ADC architectures in the database, or combination of all the above.

Let us assume that we have 5 different ADC architectures (serial, sigma delta, cyclic, pipelined, and flash) stored in the database. From the most common ADC specifi-

cations: resolution, conversion frequency, area, and power dissipation, a selection tree can be built. Based on resolution, a set of sub-trees can be obtained by classifying the 5 ADC's into low (< 6 bits), medium ($6 \leq \text{resolution} < 12$ bits), and high (≥ 12 bits). For each of this branch, 3 sets of similar trees can be also be built based on speed, power dissipation, and area. As an illustration, a selection tree for the 5 ADC's is shown in Figure 4.2. The low, medium, and high specification cut-off can be chosen for a specific CMOS technology by generating different ADC's with different specifications.

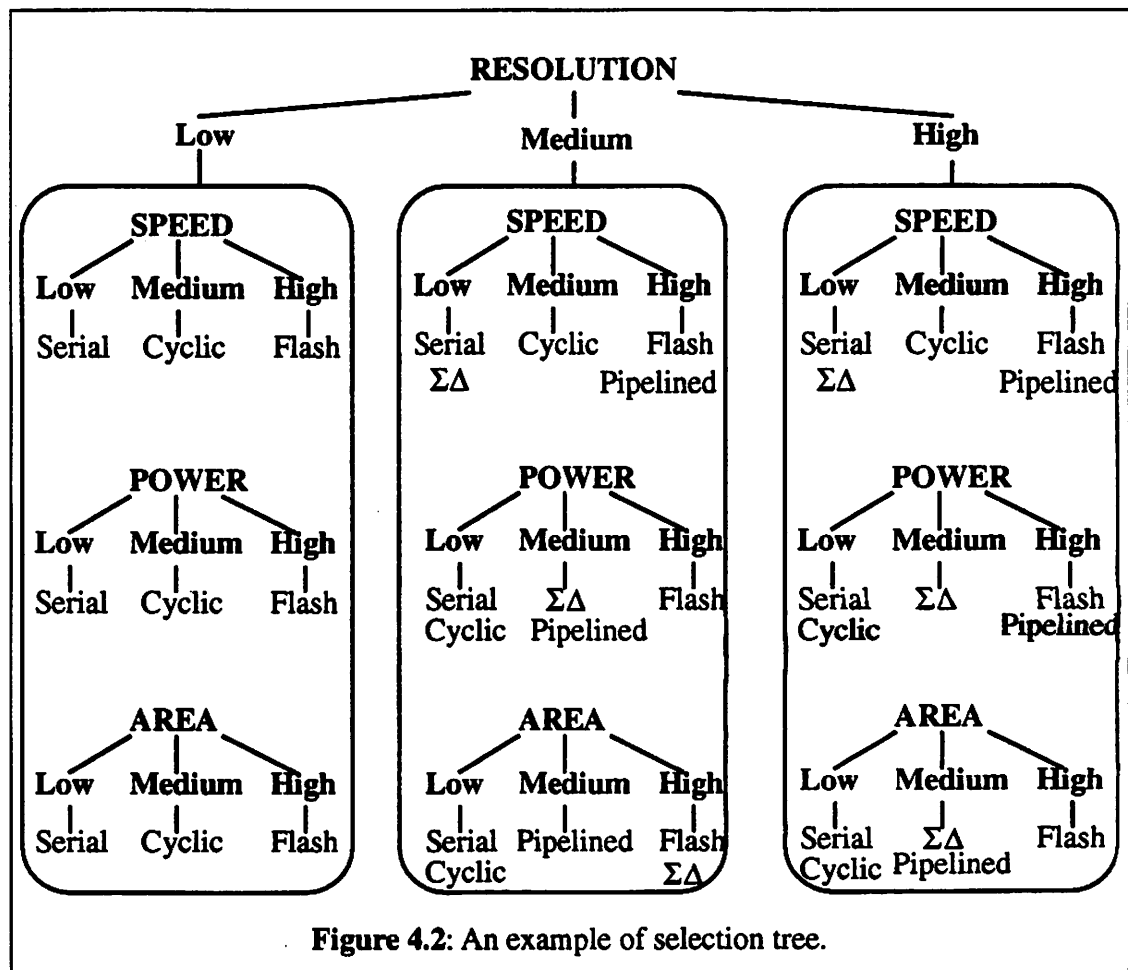


Figure 4.2: An example of selection tree.

For example, given the four specifications, we can narrow the ADC architecture choices by traversing the tree.

Example 1: a 12-bit, low-power, low-speed, and small area ADC is requested.

12-bit (High resolution) -> serial, cyclic, sigma delta, pipelined

Low-power -> serial, cyclic

Low-speed -> serial, sigma delta

Small area -> serial, cyclic

Selection rank: serial(3), cyclic(2), sigma-delta(1)

Example 2: a 8-bit, medium-power, high-speed, small area ADC

8-bit (Med. resolution): -> serial, cyclic, sigma delta, pipelined, flash

Medium-power -> sigma-delta, pipelined

High-speed -> flash, pipelined

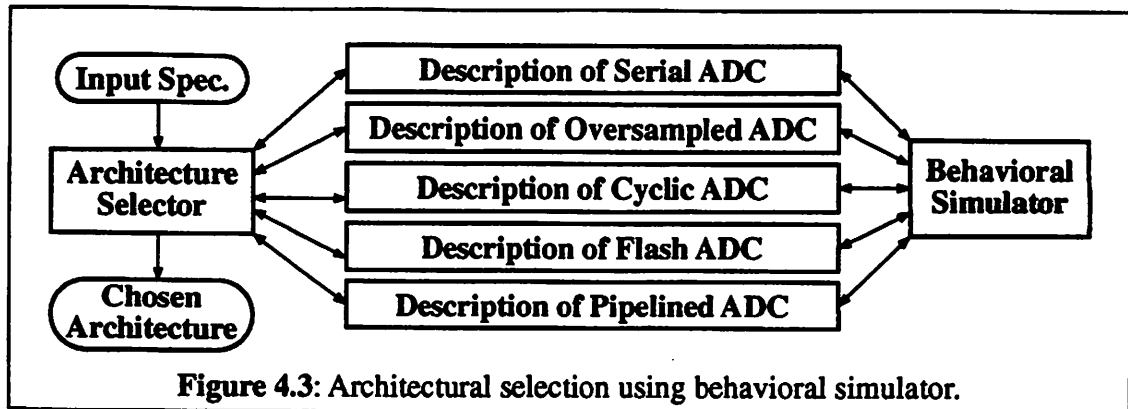
Small-area -> serial, cyclic

Selection rank: pipelined(2), cyclic(1), sigma-delta(1), flash(1), serial(1).

For the first and second examples, serial and pipelined ADC's respectively seems to be the *best candidates* to be generated *first*.

Another way to do the selection process is by using simulators such as behavioral simulator. But complex processes need to be included with this procedure in order to extract information from the simulation results. A priori information regarding the behav-

ior of subblock used inside each ADC has to be known or characterized in order to predict the feasibility of the given specifications. The selection process using behavioral simulator is illustrated in Figure 4.3.



From the discussion above, doing an exhausted search throughout the database might not be bad idea since the selection rules are very difficult to set and they have to be changed or adjusted as the process parameters change. A combination of set of rules and behavioral simulator will yield the best solution.

4.3 Circuit Synthesis

The netlist generation commonly known as circuit synthesis takes as its inputs a set of specifications for an ADC block such as resolution, conversion speed, maximum input signal frequency, maximum integral non-linearity (INL), total power dissipation, supply and reference voltages, total area, layout aspect ratio, and technology file. In addition to these specifications, statistical information for capacitor and transistor matching can also be provided. The outputs of the circuit synthesis will be the entire circuit netlist and the performance summary of the ADC block. The core of the circuit synthesis which is effectively device size generation can be implemented using global or hierarchical opti-

mization approaches. The following two sections will discuss briefly as why we chose one over the other.

4.3.1 Optimization Approaches

4.3.1.1 Global Optimization Approach

A global optimization approach attempts to optimize an ADC circuit as a whole at once. An example of this approach, DELIGHT.SPICE [NYE88] uses a circuit simulator SPICE coupled with an optimization engine. This approach was shown to be fairly effective for simple circuits. But this is a CPU intensive approach and it relies on the circuit simulator to guarantee convergence. For some ADC specifications which require a large number of circuit simulations to get a data point, this method will take weeks or even months to run (assuming no DC convergence problem in the circuit simulator).

Another example of this approach is to formulate the entire block performances in a set of closed-form equations as a function of electrical characteristics of individual devices. This type of implementation is known as an optimization based on a standard schematic approach [KOH90], [HARJ89], [DEGR87]. For a simple block such as op-amps, input buffers, or comparators, this method is quite efficient because almost all of the behavior of the block have the closed analytical equations. Unfortunately, for complex analog blocks such as ADC's, it is very difficult to derive many of specifications in closed form equations. Some specifications such as INL and DNL do not have closed form equations. Therefore, this straight forward approach cannot be applied for complex analog blocks. A modified version of this approach described in the next section will solve this problem.

4.3.1.2 Hierarchical Optimization Approach

ADC blocks are fairly complex analog circuits. An attempt to optimize the design

of this block as a whole will be a very difficult task to do. If we examine the overall block diagram of typical ADC blocks, we can easily recognize that an ADC block can be broken into several different functional blocks in which each block performs a certain task and has certain requirements depending on the overall specifications. These functional blocks are called subblocks. For the ADC architecture proposed in Chapter 3, these subblocks are sample/hold (S/H), gain of two (2X), comparators (CMP), and digital circuitry.

For a given set of ADC specifications, there is a mapping from the overall specifications to the subblocks' specifications. In order to ease the optimization of the design, each subblock can be optimized individually to meet the mapping specifications. The resulting performances of each subblock can then be combined to be used as evaluation variables for the overall performances. If the overall performances are not met, a new set of subblocks' specifications will be generated. Therefore, effectively we will have several optimizations within an optimization. This type of optimization is what we call a *hierarchical optimization* approach. A simple flowchart of a 2-level hierarchical optimization approach is shown in Figure 4.4.

Referring to Figure 4.4, for a given set of input specifications, the circuit synthesis will undergo a sequence of operations. First, the ADC specifications and their relative priorities will be mapped into subblocks' specifications and their respective relative priorities. This mapping function can be realized by a set of rules given by experience analog designers or running a behavioral simulation of the ADC circuit or a combination of both. Then the specifications and priorities of each subblock are fed into the local optimizer to generate the device sizes. This local optimizer can be a low-level synthesis block such as OPASYN [KOH90] and LAGER [BROD90] to generate op-amps and digital circuitry respectively, or a user can specify a certain library cell with its performance summary. This step allows flexibility for a user to use other desirable tools. The results of each local

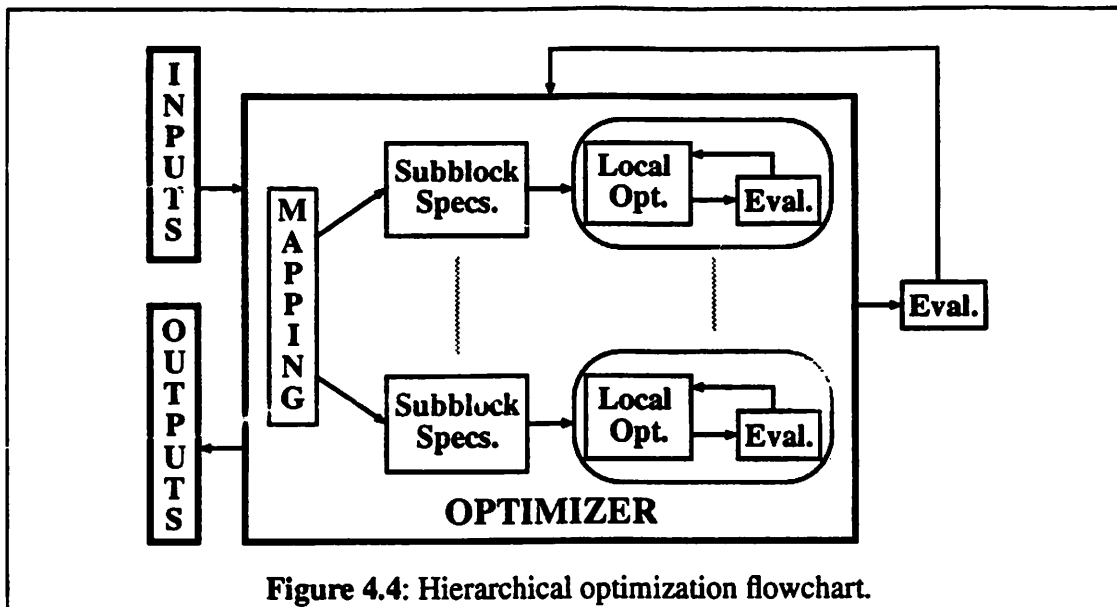


Figure 4.4: Hierarchical optimization flowchart.

optimization are then fed into the evaluation routine to do global verification. This evaluation routine can be a monte carlo simulation, worst case design, circuit simulator, or behavioral simulator. Because the global evaluation step will be carried out repeatedly during the outer loop of the optimization phase, we would like to be able to evaluate the results very quickly. To do monte carlo simulation or spice simulation will take a very long time to do. The behavioral simulation, on the other hand, generates performance predictions in a very short time. This behavioral simulator will be explained in great details in section 4.4.4.1.

To speed up the overall optimization process, design parameters for the ADC block have to be chosen carefully. If all the input specifications are used as design parameters, it will take a very long time for the optimization to find the solution points. We ought to find a set of design parameters that directly affects the performance of the ADC blocks. For the ADC architecture proposed in Chapter 3, we can find that *total power consumption* and *total area* directly dictate the block performance. Of course, there are other factors that will affect the performance of the ADC such as types of op-amp used, device matching, or

process parameters. These factors always exist even if we design the ADC's manually.

In order to make sure that the resulting ADC circuits are those of practical ones, we have to set a limit for the two design parameters. In this case, we have to set the upper limit values for them. Obviously, the objective now is to minimize the power consumption as well as the total area needed such that the given specifications are met. This type of optimization is commonly known as one-sided limit or inequality type constraints. The optimization algorithm implementation and cost function selection will be discussed later in section 4.4.2.

The circuit synthesis approach just described consumes very little time. Moreover, it has the expansion capability of using other available low-level synthesis tools to optimize low-level subblocks and of allowing incorporation of standard cell blocks if desired. Keep in mind that this hierarchical optimization concept can also be applied to other analog or mixed signal function module generations as well.

4.4 Implementation

Figure 4.5 shows the flowchart of circuit synthesis based on the hierarchical optimization approach. Detailed discussion of each block will be presented next.

4.4.1 Inputs and Outputs

The inputs to the circuit generator are a set of ADC specifications and their relative priorities, spice process parameters, device matching information, design rules, and placement information files. This section will summarize briefly what each input file consists of.

The ADC specification file consists of 2 types of inputs: *static* and *variables* specifications. Static specifications such as supply voltages (± 2.5 volts or ± 5.0 volts), technol-

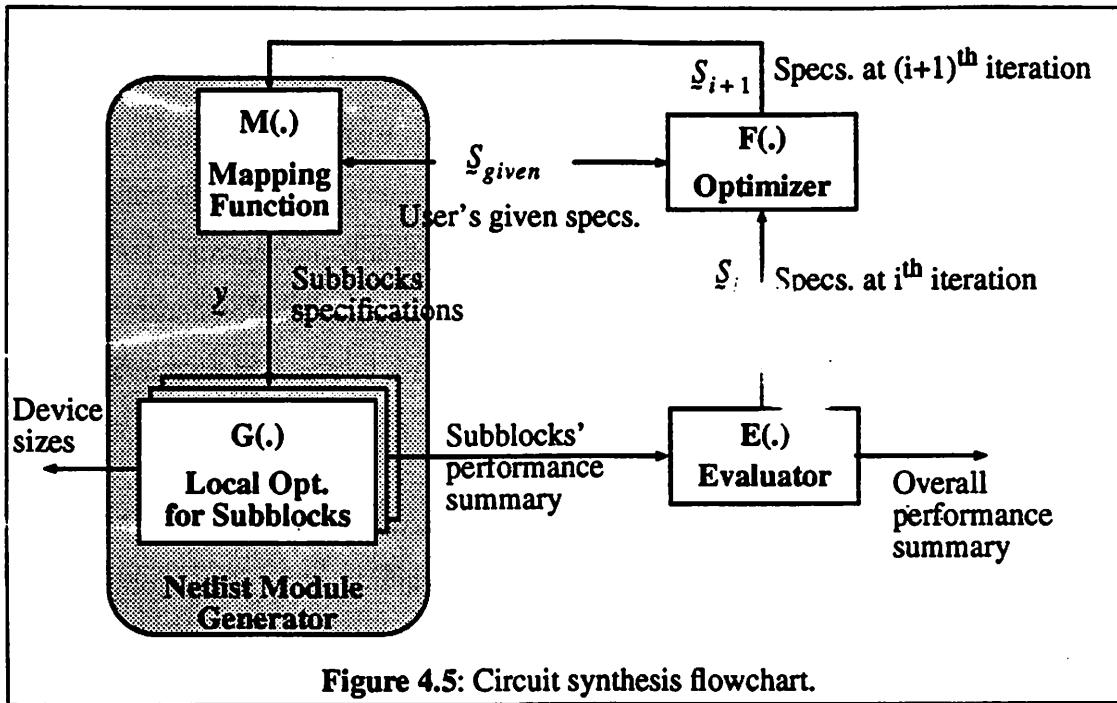


Figure 4.5: Circuit synthesis flowchart.

ogy types (MOSIS 2 μm or 3 μm CMOS process), and maximum input signal frequency (default to half the sampling rate) are used as the conditions in which the remaining specifications are optimized around. The specifications would always be the same throughout the optimization. On the other hand, variable specifications such as resolution, conversion speed, silicon area, power dissipation, differential non-linearity, integral non-linearity, aspect ratio, maximum reference voltage magnitude will be optimized according to the priorities given at the beginning the circuit generation.

The spice process parameters are basically used as device parameter values during the calculation of individual block performances. These parameters are technology and vendor dependent. The device matching values are the information regarding how accurate the matching of two identical devices. Currently, only two values are provided into this file: capacitor ratio accuracy and transistor matching. These values will also be technology and vendor dependent. Device matching information is usually not available from

the vendor. Thus characterization process needs to be carried out in order to gather these information. Due to the unavailability of the matching information, these values are currently estimated roughly for the technologies used. A 3σ value is used as the matching accuracy for the ratio of two 1 picofarad capacitors. Ideally, a table of 3σ values for different capacitor values should be available. As a result, when matching ratio of capacitor value other than 1 picofarad is needed, the 3σ value will be extrapolated from the existing one using statistical formula. More discussion on the capacitor scaling will be discussed later in this chapter.

Design rules and floorplan information files are needed in order to estimate the actual silicon area during the circuit generation more accurately. These two files are used mainly by the floorplan optimizer. Design rules are provided by the vendor depending on the technology. The floorplan information file is a file that stores the information of the relative placement of all the blocks. Usually this file has been optimized to a specific block architecture. Modification to this file is allowed but not recommended. A detailed discussion of this file will be presented in Chapter 5.

4.4.2 Optimization

We mentioned earlier that reducing the number of design parameters is a very important step in shortening the optimization process. Besides carefully choosing the design parameters, selecting the cost function to evaluate the resulting specifications is very important as well.

The optimization type for the outer loop is a non-linear optimization with one-sided constraints or bounded constraints. We can specify the optimization in the form:

$$\text{minimize } f(x) \tag{EQ 4.1}$$

$$\text{subject to } \underline{x} \in \Omega \quad (\text{EQ 4.2})$$

where f = a real-value function

Ω = the feasible set and a subset of R^n

\underline{x} = the chosen design parameters

The outer loop of the optimization was implemented using an *iterative descent* algorithm: *steepest descent*. This algorithm is one of the simplest and easiest to implement. The basic idea is that the algorithm generates a series of points from the preceding points *iteratively* and the cost or objective function decreases in value when it is evaluated directly or indirectly using the generated points. The process to find the optimum points is what we usually call *line searching*. One very popular and simple line search is search by Golden Section [LUEN84]. This type of line search is very suitable for function that has boundary points and has unknown degree of smoothness even though it converges linearly with convergence ratio of 0.618.

The algorithm of the steepest descent algorithm for two design parameters can be summarized as follows:

Let $\underline{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ be the design variables and $\underline{x}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ be the initial points.

Define dx and dy as small steps on x and y axes.

$$\begin{bmatrix} \nabla_x f \\ \nabla_y f \end{bmatrix} = \begin{bmatrix} \left. \frac{f(x_0 + dx) - f(x_0)}{dx} \right|_{y=y_0} \\ \left. \frac{f(y_0 + dy) - f(y_0)}{dy} \right|_{x=x_0} \end{bmatrix} \quad (\text{EQ 4.3})$$

$$\nabla f(\underline{x}_0) = \begin{bmatrix} \frac{\nabla_x f}{\|\nabla f\|} dx \\ \frac{\nabla_y f}{\|\nabla f\|} dy \end{bmatrix} \quad (\text{EQ 4.4})$$

where $\|\nabla f\| = \sqrt{\nabla_x f^2 + \nabla_y f^2}$

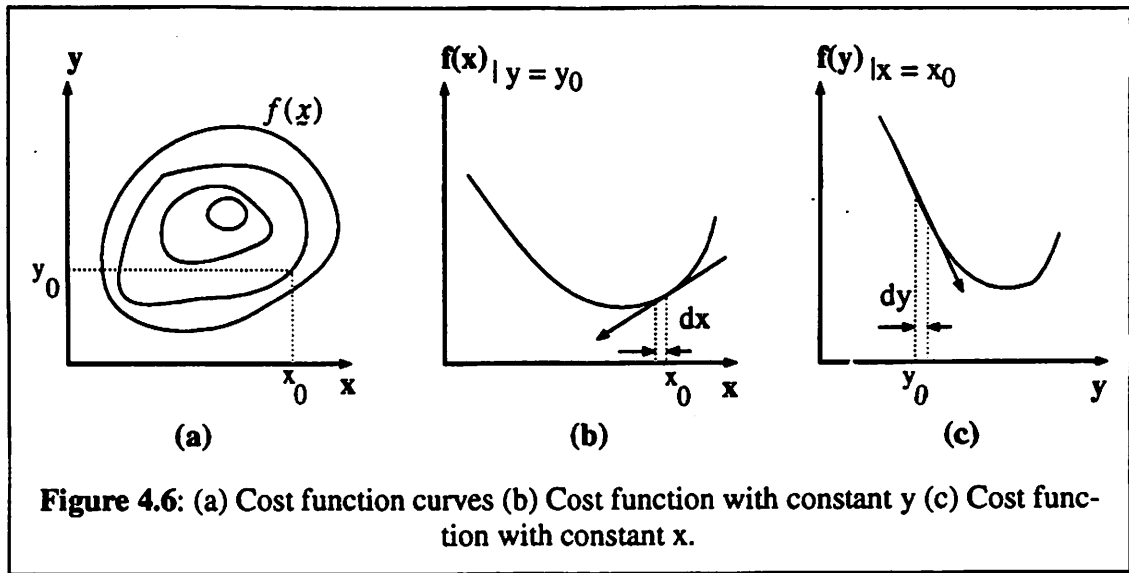


Figure 4.6: (a) Cost function curves (b) Cost function with constant y (c) Cost function with constant x .

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \cdot \nabla f(\underline{x}_k) \quad (\text{EQ 4.5})$$

$$0 \leq \alpha_k \leq \infty \quad (\text{EQ 4.6})$$

The solution set: \underline{x} where $\nabla f(\underline{x}) = 0$ (EQ 4.7)

Figure 4.5 illustrates the equations just described. We have pointed out that the conversion rate of the steepest descent is *linear*. The convergence ratio can be formulated in terms of the eigenvalues of the cost function [LUEN84].

$$R = \left[\frac{A-a}{A+a} \right]^2 \quad (\text{EQ 4.8})$$

where a = the smallest eigenvalue of f and $a > 0$

A = the largest eigenvalue of f and $A > 0$

The cost function at the k^{th} iteration for our circuit synthesis is chosen to be the sum of exponential functions:

$$f(x_k) = \sum_{i=1}^n \exp(d_i) \quad (\text{EQ 4.9})$$

$$d_i = x_{i,k} - x_{i, \text{given}} \text{ for minimizing specification} \quad (\text{EQ 4.10})$$

$$d_i = x_{i, \text{given}} - x_{i,k} \text{ for maximizing specification} \quad (\text{EQ 4.11})$$

where $x_{i,k}$ = the i^{th} specification at the k^{th} iteration

$x_{i, \text{given}}$ = the i^{th} given specification

n = the total number of specifications

If $n = 3$, the Hessian of f is:

$$\nabla^2 f = \begin{bmatrix} \exp(d_1) & 0 & 0 \\ 0 & \exp(d_2) & 0 \\ 0 & 0 & \exp(d_3) \end{bmatrix} \quad (\text{EQ 4.12})$$

Since the specifications have different unit values, the largest eigenvalue will be much larger than the smallest eigenvalue. As a result the convergence ratio will be roughly unity and the speed of the steepest descent will be very slow.

In order to avoid this problem, we introduce *scaling* to the variables, d so that they are roughly of the same magnitudes. The specifications are normalized to their given specifications:

$$d_i = \frac{x_{i,k} - x_{i,given}}{x_{i,given}} \text{ for minimizing specification} \quad (\text{EQ 4.13})$$

$$d_i = \frac{x_{i,given} - x_{i,k}}{x_{i,given}} \text{ for maximizing specification} \quad (\text{EQ 4.14})$$

In order to incorporate the specifications' priorities, weight factor can also be introduced to the cost function by multiplying the normalized exponential constant with the weight factor. Default values for this weight factor should be 1. For specifications that are high on priority list, weight factor greater than unity can be applied. However, there should be a reasonable limit as how big the weight factor can be since the exponential function is very sensitive to its exponential constant.

The steepest descent algorithm is also used in OPASYN [KOH90]. Thus, we try to reuse the existing optimization code of OPASYN with some modifications. OPASYN uses an enhanced version of steepest descent with a golden section line search. A history heuristic intended to detect the steepness of the gradient is also added to speed up the search. For detailed information of the algorithm, please refer to [KOH89]. Figure 4.7 summarizes the steepest descent algorithm used in OPASYN as well as our circuit synthesis.

The overall implementation of the outer optimization loop can be summarized in a pseudo-C routine shown in Figure 4.8. At the beginning of the circuit generation, all the necessary input files will be read in. A set of starting points for the optimization will then be selected. A user can select the starting points or use the default values. Using these starting points, the optimization of the ADC circuit generation is executed until the requested specifications are satisfied or the number of optimization cycles exceeds the maximum number of iterations. Within the optimization loop, *netlist module generator* routine will be called, the current cost function will be calculated and if necessary a new

Inputs:

$f(x)$ with $x_{min} \leq x \leq x_{max}$

A starting point x_s

Outputs:

A local optimal point x_{opt}

Algorithm:

Set λ to INIT-LAMBDA (=5.0)

Set η to GOLD-SECTION (=0.8)

Set STEEP-HISTORY and VALLEY-HISTORY to 0

$x_0 = x_s$

while ($\|\nabla f\| > \text{END-FACTOR}$ or stop \neq true) do {

 if (STEEP-HISTORY \geq 10) then $\lambda = \lambda \times 3$

 if (VALLEY-HISTORY \geq 5) then $\lambda = \lambda \times 0.3$

 Get $\nabla f(x_0)$

$x_1 = x_0 - \lambda \nabla f(x_0)$

$x_2 = x_0 - \eta \lambda \nabla f(x_0)$

 if ($f(x_1) \leq f(x_2)$) then do {

 STEEP-VALLEY = STEEP-VALLEY + 1

 VALLEY-HISTORY = 0

 } until ($f(x_1) \leq f(x_2)$) {

$x_1 = x_2$

$\eta = \eta^2$

 if ($\eta \leq \text{ETA-LIMIT}$) then stop = true

 VALLEY-HISTORY = VALLEY-HISTORY + 1

 STEEP-HISTORY = 0

$x_2 = x_0 - \eta \lambda \nabla f(x_0)$

 } end else

$x_0 = x_1$

} end while

Return $x_{opt} = x_0$

Figure 4.7: An enhanced steepest descent algorithm.

set of points will be determined.

```
optimization()
{
    readAllNecessaryInputFiles();
    selectStartingPoints();
    while (not_optimized || within_iteration_limit) do {
        netlistModuleGenerator();
        calculateCostFunction();
        determineNextPoints();
    } /* end of while */
} /* end of optimization() */
```

Figure 4.8: A pseudo-C routine of the overall optimization.

4.4.3 Netlist Module Generator

In order to coordinate the process of generating all the necessary subblocks' requirements, calling appropriate tools, generating results, and many others, a customized routine for each ADC architecture needs to be implemented. This routine is called *netlist module generator*. The content of the netlist module generator (NMG) will be architecture specific. Thus the number of NMG's will depend on the number of ADC topologies stored in the database. Many analog design knowledge are incorporated into writing this routine. The remaining of this section will briefly discuss the flow of the NMG for the algorithmic ADC proposed in Chapter 3.

Inside the NMG, several tasks will be executed in order to generate ADC netlist based on the current set of points in the optimization loop. These tasks are the mapping of ADC specifications into subblock specifications, generating op-amps, comparators, switches, capacitors, and digital circuitry, and finally evaluating the overall ADC block using a behavioral simulator. Some of these tasks will be discussed in the following sec-

tions. The simplified pseudo-C routine of the algorithmic ADC NMG is shown in Figure 4.9.

```

netlistModuleGenerator()
{
    mapping();
    generateOpamp();
    generateComp();
    generateSwitchCaps();
    generateDigCkts();
    callBehSim();
    generateADCNetlist();
} /* end of netlistModuleGenerator() */

```

Figure 4.9: A simplified pseudo-C routine of netlist module generator.

4.4.3.1 Mapping Function

Before we implement a mapping function for a particular ADC architecture, we have to understand the effect of different non-idealities to the overall ADC performances. Examples of these non-idealities are limited op-amp gain, offset voltages from op-amps and comparators, or capacitor matching accuracy. Monte carlo simulations of the improved algorithmic ADC described in Chapter 3 were carried out separately in order to find out which subblock is the critical one or what combination of offset voltages yields the worst performances. With the help of the behavioral simulator described later in this chapter, we are able to understand the sensitivity of the ADC block with respect to op-amp gains and offset voltages, capacitor matching, and comparator offset voltages. Many of these requirements and mapping equations are derived analytically in Appendix A.

Referring back to Figure 4.5, the mapping function takes two different sets of specifications: user's given specifications, S_{given} and current specifications at $(i+1)^{th}$ itera-

tion, \mathcal{S}_{i+1} . The basic function of the mapping function is to map or to generate all the required subblocks' specifications from either \mathcal{S}_{given} or \mathcal{S}_{i+1} . The *simplified* pseudo-C routine of the mapping function is shown in Figure 4.10 and the explanation of it is to follow.

```
mapping()
{
    if (iteration == 0) {
        genMinReq(givenSpecs);
        genRelPrior();
    } /* end of if */
    genSubblockReq(currentSpecs);
} /* end of mapping() */

genMinReq(givenSpecs)
{
    initLocalSpecs();
    genMinOpampGain();
    genMaxOpampOffset();
    genMaxCompOffset();
    genMinCapacitor();
} /* end of genMinReq() */

genSubblockReq(currentSpecs)
{
    genOpampSpecs();
    genCompSpecs();
    genDigSpecs()
} /* end of genSubblockReq() */
```

Figure 4.10: A pseudo-C routine of the mapping function.

At the very first iteration, the mapping function uses \mathcal{S}_{given} to generate all the extreme requirements necessary for the critical blocks (op-amps, comparators, and capac-

itor sizes) in order to achieve the requested specifications. These extreme requirements are the minimum op-amp gain needed, maximum op-amp offset voltages, minimum capacitor size allowed, or maximum comparator offset voltages. These limits are set independently by assuming that the other factors are non-existence and the linearity specifications for the ADC are *twice* better than the requested ones. All limits except minimum capacitor requirement are obtained using the behavioral simulator, ADSIM described in section 4.4.4.

The minimum capacitor requirement is determined by selecting the larger of the two capacitor values obtained from $\frac{kT}{C}$ noise limitation and capacitor ratio matching extrapolation for the requested specifications.

$\frac{kT}{C}$ Noise Limitation: The minimum capacitor size due to this limitation is determined by using the total input referred mean square noise derived in Appendix A. The minimum capacitor allowed is determined by making sure the input referred noise voltage is less than a quarter LSB of the ADC block.

$$\bar{v}_{input}^2 = M \frac{kT}{C} \leq (0.25 \text{ LSB})^2 \quad (\text{EQ 4.15})$$

$$M \frac{kT}{C} \leq \left[\frac{2V_{REF_MIN}}{2^{N+2}} \right]^2 \quad (\text{EQ 4.16})$$

$$C \geq M kT \left[\frac{2^{N+2}}{2V_{REF_MIN}} \right]^2 \quad (\text{EQ 4.17})$$

Where N = the number of bits

V_{REF_MIN} = the minimum reference voltage requested

M = a constant that is a function of the feedback factor of a block

C = the minimum capacitor allowed.

For $N = 9$, $V_{REF_MIN} = 1$ volt, $M = 2.5$, and $T = 27$ °C, $C_{MIN} = 43$ femto-farads.

Thus, for low resolution ADC specifications, the $\frac{kT}{C}$ noise is negligible.

Capacitor Matching Ratio Extrapolation: The capacitor used in the ADC module generator is a SiO₂ capacitor type. The capacitor generator is capable of generating poly to poly or metal to poly capacitors for arbitrary values and aspect ratios. Further capacitor layout generation discussion can be found in Chapter 5. A SiO₂ capacitor is chosen because it has the smallest temperature coefficient (+25 ppm/°C [McCR75]) among different types of capacitors. Moreover, the process technology has matured so much that the oxide thickness can be made fairly uniform. In order to reduce the fringing effect, a grounded guard ring is added to enclose the capacitor as shown in Figure 4.11. The capacitor is placed on the top of a well that is connected to the quiet analog ground so that the noise coupling from the substrate can be minimized. Furthermore, to reduce the undercut effect on different capacitor sizes, an undercut insensitive geometry capacitor is used. As an example, if we try to match 2 capacitors of value 2 picofarads and 1 picofarad, 2 1-picofarad capacitors are used to realize the 2-picofarad capacitor.

Thus the ratio capacitor error between two capacitors can be approximated to depend on the matching of two equal value capacitors. Due to the unavailability of matching information for capacitor ratio of different values, the 1 pico-

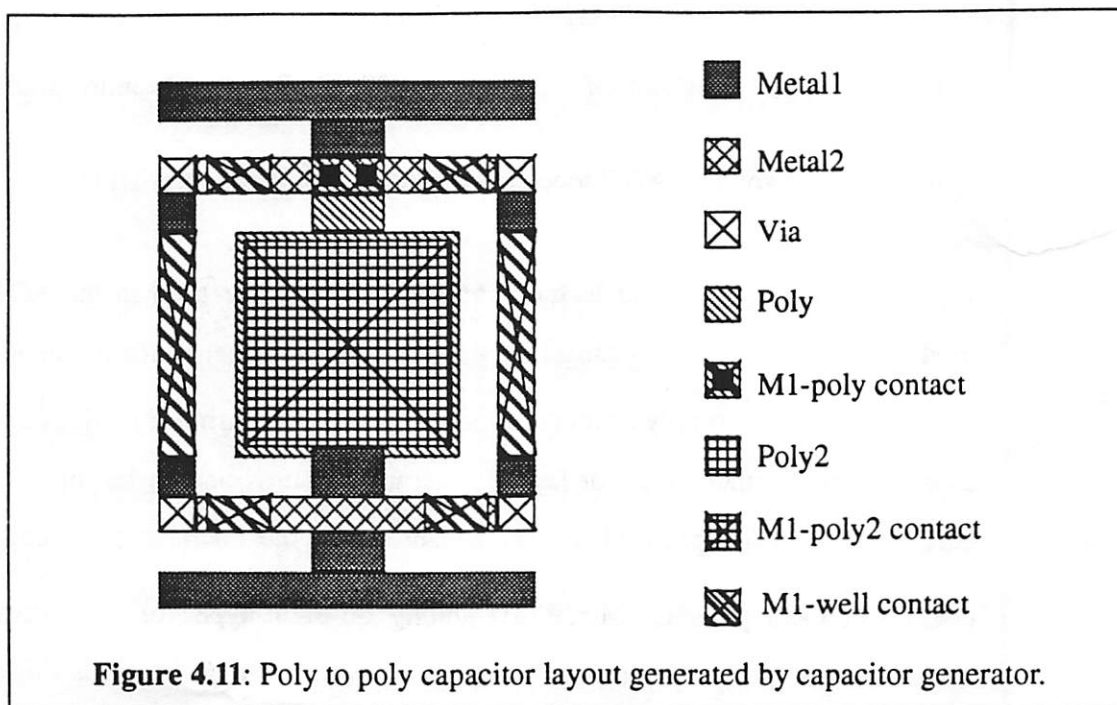


Figure 4.11: Poly to poly capacitor layout generated by capacitor generator.

farad capacitor ratio 3σ mismatch value is used to extrapolate different capacitor mismatch values. Let C_1 and C_2 be two 1-picofarad capacitors.

$$C = \frac{C_1 + C_2}{2} \quad (\text{EQ 4.18})$$

$$\Delta C = C_1 - C_2 \quad (\text{EQ 4.19})$$

$$C_1 = C + \frac{\Delta C}{2} \quad (\text{EQ 4.20})$$

$$C_2 = C - \frac{\Delta C}{2} \quad (\text{EQ 4.21})$$

$$\frac{C_1}{C_2} = \frac{C + \frac{\Delta C}{2}}{C - \frac{\Delta C}{2}} = \frac{1 + \frac{\Delta C}{2C}}{1 - \frac{\Delta C}{2C}} \quad (\text{EQ 4.22})$$

Implementation

$$\frac{1}{1-x} \approx 1+x \text{ for } x \ll 1 \quad (\text{EQ 4.23})$$

$$\frac{C_1}{C_2} \approx \left(1 + \frac{\Delta C}{2C}\right)^2 = 1 + \frac{\Delta C}{C} + \left(\frac{\Delta C}{2C}\right)^2 \quad (\text{EQ 4.24})$$

$$\frac{C_1}{C_2} \approx 1 + \frac{\Delta C}{C} \quad (\text{EQ 4.25})$$

The matching requirement of two equal capacitors for N -bit resolution ADC has to be within half LSB of the ADC.

$$\frac{\Delta C}{C} \leq \frac{1}{2^{N+1}} \quad (\text{EQ 4.26})$$

$$N \leq 3.32 \log \left(\frac{\Delta C}{C}\right)^{-1} - 1 \quad (\text{EQ 4.27})$$

For $\frac{\Delta C}{C} = 0.001$, $N \leq 9$ bits. If different capacitor value ratio is required, the 3σ mismatch value can be scaled accordingly. Thus the ratio mismatch of two A -picofarad capacitors can be found to be:

$$3\sigma = \frac{1}{\sqrt{A}} \frac{\Delta C}{C} \quad (\text{EQ 4.28})$$

where 3σ = the ratio mismatch value of the scaled capacitors

A = the scaled factor

$\frac{\Delta C}{C}$ = the given ratio mismatch value of two 1-picofarad capacitors.

From EQ 4.26 and EQ 4.28, we can then determine the minimum capacitor allowed for a given mismatch information, $\frac{\Delta C}{C}$ and resolution, N .

$$\frac{1}{\sqrt{A}} \frac{\Delta C}{C} \leq \frac{1}{2^{N+1}} \quad (\text{EQ 4.29})$$

$$A \geq \left[2^{N+1} \frac{\Delta C}{C} \right]^2 \quad (\text{EQ 4.30})$$

For $N = 10$ and $\frac{\Delta C}{C} = 0.1\%$, $A \geq 4.194$. Thus, the smallest unit capacitor allowed is 4.194 picofarads.

In the remaining of the iterations, the current specifications, S_{i+1} are used to generate all subblocks' requirements such as op-amps, comparators, and digital circuitry. This process is carried out using analytical formulas that map input specifications to subblocks' specifications. Some mapping requirements are obtained using behavioral simulator as mentioned earlier. All the equations used can be found in the Appendix A. As examples, the unity gain frequency, ω_u , output swing- V_{OUT} , power dissipation - PWR , and area - $AREA$ of the op-amp can be obtained by the following equations:

$$\omega_u = \left[\frac{A_V}{\frac{3P_{\Phi} (1 + A_V f)}{2.8N}} \right] \quad (\text{EQ 4.31})$$

$$V_{OUT} = V_{REF_{MIN}} \quad (\text{EQ 4.32})$$

$$PWR = \frac{P_C}{k_P} \quad (\text{EQ 4.33})$$

$$AREA = \frac{A_C}{k_A} \quad (\text{EQ 4.34})$$

where A_V = the minimum open loop gain of the op-amp obtained from the behavioral simulator

P_{Φ_s} = the duration of sampling pulse width extracted from the conversion speed and resolution specifications

f = the feedback factor of the closed-loop block

N = the desired number of bits

V_{REF_MIN} = the given minimum voltage reference

P_C = the current total power value

k_P = a constant equating total power to individual op-amp power

A_C = the current total area value

k_A = a constant equating total area to individual op-amp area.

The resulting subblocks' specifications will then be used to generate the low-level blocks.

4.4.3.2 Generating Operational Amplifiers

Op-amp generation is being carried out by an op-amp synthesis, OPASYN described in Chapter 2. A fully differential folded cascode op-amp described in Appendix A has been added into the OPASYN topology database since a fully differential architecture is used throughout the ADC circuit. The specifications are generated by the mapping function and then fed into OPASYN. The simplified pseudo-C routine of the routine is shown in Figure 4.12. Within the op-amp generation routine, OPASYN is called several times in an attempt to choose the optimum value of unit capacitor with the sample/hold and gain of two blocks. The reason why OPASYN is called several times is that the load capacitor of the requested op-amp is not optimized yet. Basically, the loading capacitor

value for the op-amp is set by multiplying a known constant to the allowable unit capacitor, C_S chosen from the two determining factors: $\frac{kT}{C}$ noise and ratio mismatch. Since the allowable unit capacitor is usually fairly small, the resulting op-amp input capacitance, C_{IN} will be compared to the allowable unit capacitor, C_S . If C_{IN} is bigger than C_S , a new value for C_S will be adjusted so that the attenuation at the input of the block is minimized. The value of C_S cannot be increased to much because it will eventually affect the speed of the op-amp. All of these are carried within `checkCsVsCl` procedure. In order to avoid infinite loop for some specifications, a maximum iteration limit is set.

```
generateOpamp()
{
    initLoadingCap();
    while (not_done && (iteration < MAX_ITER_LIMIT)) {
        callOpasyn();
        checkCsVsCl();
    } /* end of while */
} /* end of generateOpamp() */
```

Figure 4.12: A simplified pseudo-C routine of op-amp generation.

4.4.3.3 Generating Comparators

Comparator generation is fairly simple because the two-comparator scheme proposed in Chapter 3 is used in the ADC block. The maximum allowable comparator offset voltages are $\pm \frac{V_{REF}}{4}$ volts. If $V_{REF} = 1$ volt, the comparator offset voltage magnitude can be as large as 250 mV. Thus, this simple comparator can be used for a wide spectrum of resolution and sampling rates. In order to simplify the comparator generation, a simple comparator topology described in Appendix A has been added into the OPASYN topology

database. The comparator specifications generated by the mapping function can then be fed into OPASYN to generate the comparators.

4.4.3.4 Generating Switch and Capacitor Sizes

The generation of switch and capacitor sizes are done after the op-amps and comparators are generated. The final allowable unit capacitance, C_S obtained during op-amp generation will then be used as the value as the unit capacitor for many capacitors such as sampling, integrating, and reference capacitors. All switch sizes can then be also determined. Detailed derivation of the equations used in the routine can be found in Appendix A.

4.4.3.5 Generating Digital Circuitry

For digital circuitry, a standard cell approach is used since it is the least critical of all and the standard cell library exists for the supported technologies. The only variable block of the digital circuitry is the digital error correction register. It increases linearly as the number of bits increases. The logic diagram of all digital circuitry used can be found in Appendix A.

4.4.4 Evaluator

Evaluation of the optimization is carried out not only by the built in functions within the optimization topology but also by using *behavioral simulator* and *floorplan optimizer*. For specifications such as power dissipation and maximum reference voltages are easily calculated. For specifications such as conversion rate, resolution, differential non-linearity, integral non-linearity, and gain error are obtained by using behavioral simulator. Total silicon area is estimated by passing the netlist to the floorplan optimizer. The following sections will discuss the behavioral simulator and floorplan optimizer implementations.

4.4.4.1 Behavioral Simulator

We have argued that evaluating the performance of complex analog blocks using circuit simulator such as SPICE is not very practical because not only it will take very long time to execute a simulation but also it might have convergence problems. Moreover, the evaluation will be executed many times during the optimization cycles. The time needed to do the worst case DNL/INL simulation for an N-bit ADC proposed in Chapter 3 can be approximated to be

$$\text{Time} = S \cdot 2^N \cdot \log(A^{-1}) \quad (\text{EQ 4.35})$$

where N = the number of bits

S = time needed by SPICE to simulate an N-bit A/D conversion

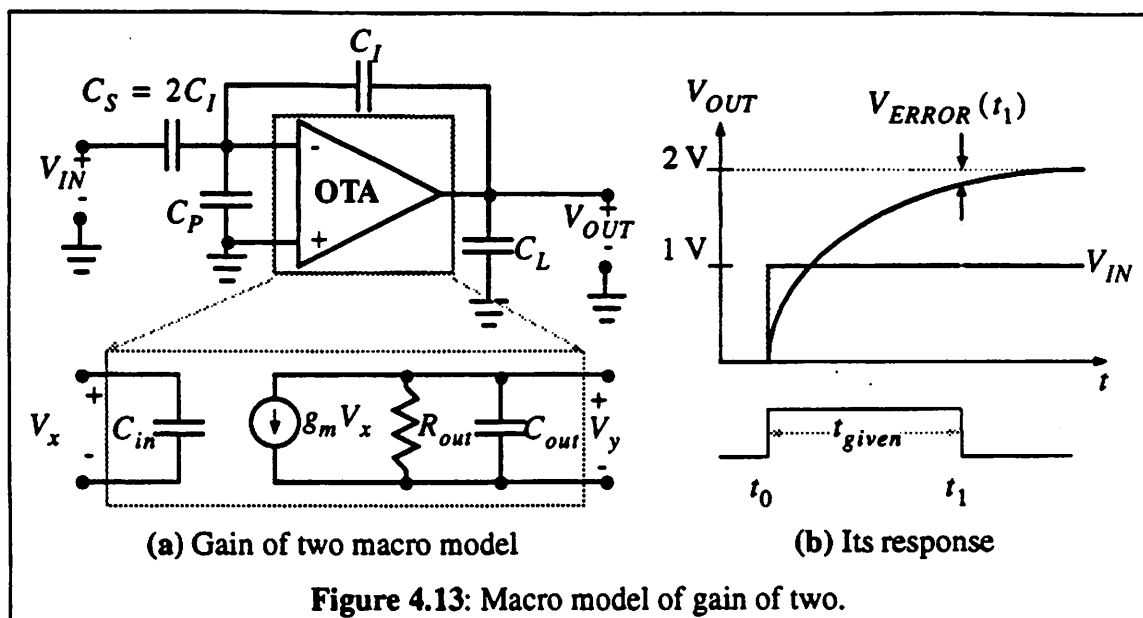
A = the requested DNL-INL precision

For $N = 9$ bits, $A = 0.01$, and $S = 0.5$ hour, the time needed is 1701 hours. Of course, the actual value of S is much longer than the given number. The time needed to convert an analog signal through a 9-bit new-improved algorithmic ADC using SPICE on DEC5000 is found to be around 19 hours. For these reasons, we have decided to implement a behavioral simulator for the ADC evaluation so that we can speed up the optimization cycles.

Behavioral simulation is an effort to predict the performance of a system by macro-modelling each functional block by its analytical model. An example of a functional block is an operational amplifier. For the folded cascode op-amp described in Appendix A, we can ignore the higher order poles and assume that the op-amp has a single-pole transfer function. From the low-level synthesis tool, OPASYN, we can extract information regarding the small signal parameter values of the op-amps such as input capacitance- C_{in} , transconductance- g_m , output resistor- R_{out} , output capacitance- C_{out} .

These parameters are then used to estimate the response of blocks such as sample/

hold and gain of two blocks. Figure 4.13 illustrates how the gain of two block is macro-modelled. For a given input step voltage at time t_0 , we can estimate what the output voltage would be at time t based on an analytical model of the block.



$$V_{OUT}(t) = - \left[\frac{C_S}{\frac{C_S + C_I + C_P + C_{in}}{g_m R_{out}} + C_I} \right] \left[1 - e^{-\frac{(t-t_0)}{\tau_{fb}}} \right] V_{IN}(t_0) \quad (\text{EQ 4.36})$$

$$\tau_{fb} = \frac{R_{out} (C_I + C_L + C_{out})}{1 + g_m R_{out} \left(\frac{C_I}{C_S + C_I + C_P + C_{in}} \right)} \quad (\text{EQ 4.37})$$

where C_S = sampling capacitor

C_I = integrating capacitor

C_P = total parasitic capacitor on the summing node

C_{in} = input capacitor of the op-amp

C_L = load capacitor at the output node.

g_m = transconductance of the op-amp

C_{out} = output capacitance of the op-amp

R_{out} = output resistance of the op-amp

For a given time- t_{given} , we can then approximate what the error voltage at the output would. This information can then be used as one of the variables to predict the resolution of the given time which effectively is the pulse width of the sampling clock of the ADC block.

$$V_{ERROR}(t) = \frac{C_S}{C_I} V_{IN}(t_o) - V_{OUT}(t) \quad (\text{EQ 4.38})$$

For the ADC proposed in Chapter 3, we are only interested in the values of node voltages at the end of pulse width, i.e. at the end of the sampling or holding time. As a result, we only have to know the starting voltages, time duration, and the voltages at the end of the time duration. Since the time duration or pulse width is known throughout the conversion, we can model the ADC by using a *discrete event time model*.

For the proposed 1-cycle algorithmic A/D conversion architecture that is used as the building block for the module generator, we only need to model the time when Φ_1 , Φ_2 , and Φ_S goes from high to low and low to high. These transition points are shown in Figure 4.14 as $t_0, t_1, t_2, t_3, t_4, \dots, t_8$ for a 4-bit A/D conversion example. Due to the cyclic nature of the ADC architecture, only 4 distinct time points: t_0, t_1, t_2, t_3 that needs to be coded with minor addition on t_0 and t_1 .

By carefully analyzing the behavior of the ADC in between these time points

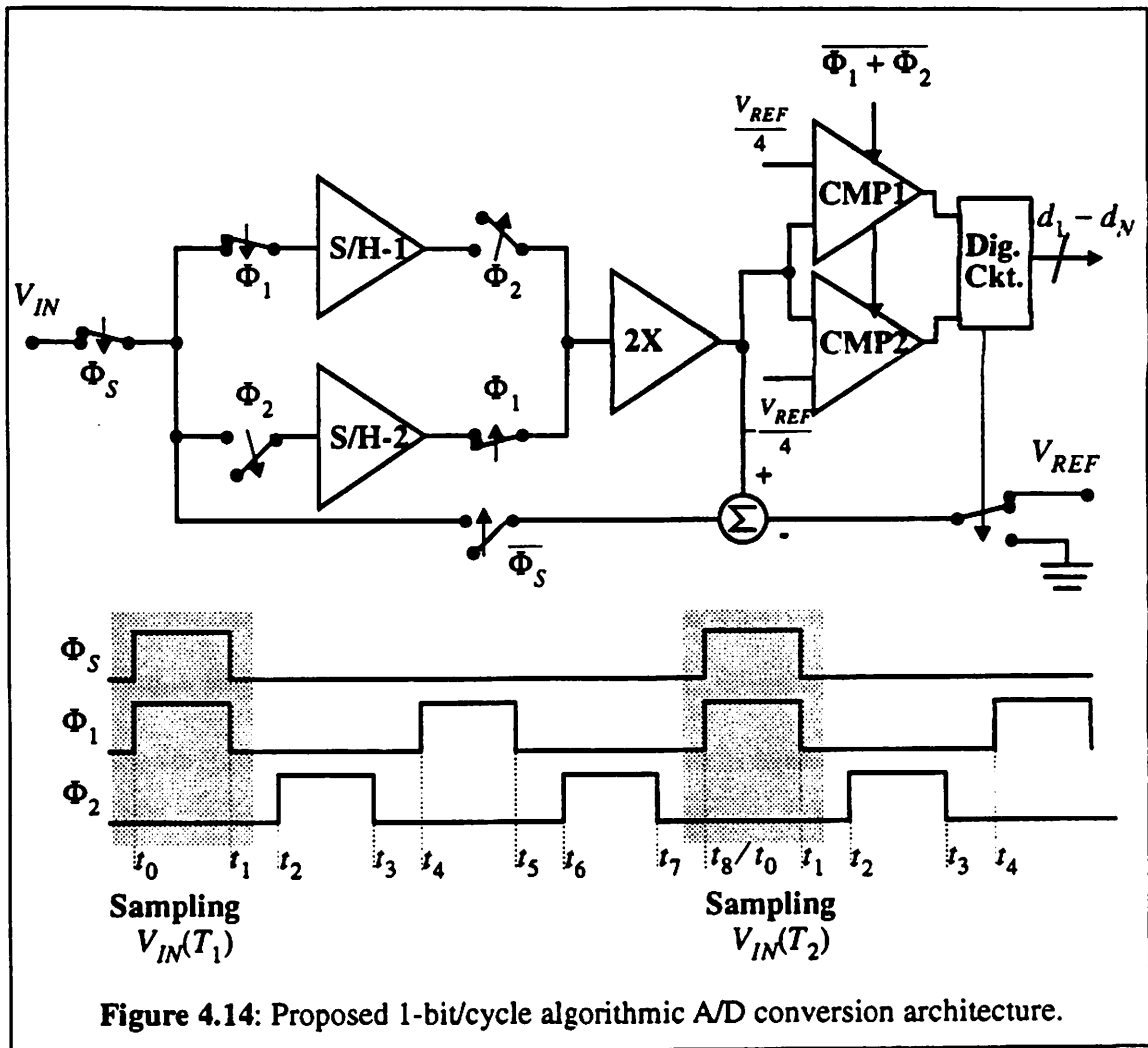
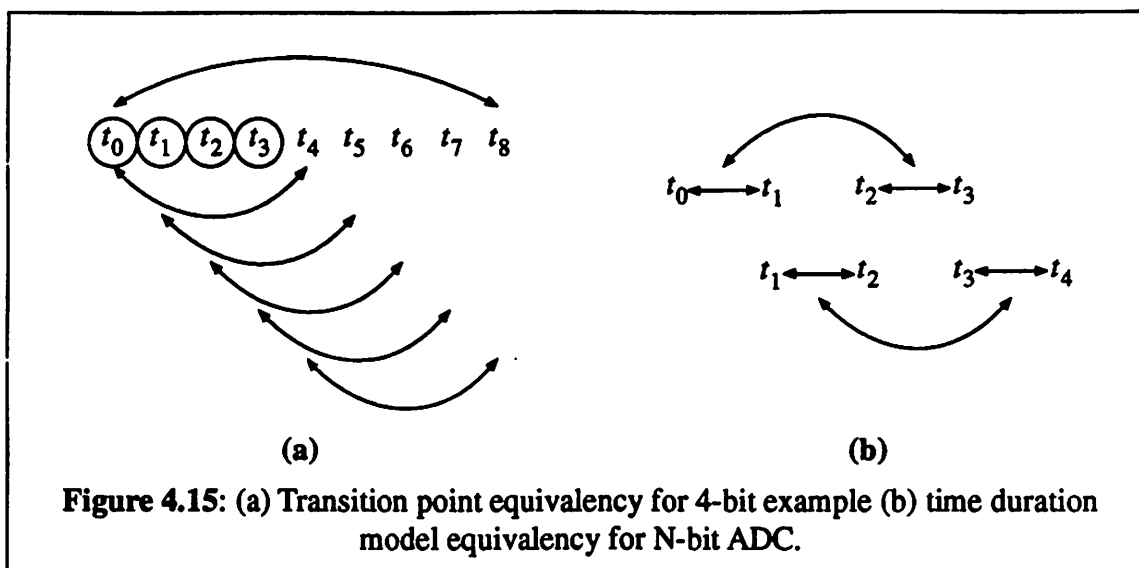


Figure 4.14: Proposed 1-bit/cycle algorithmic A/D conversion architecture.

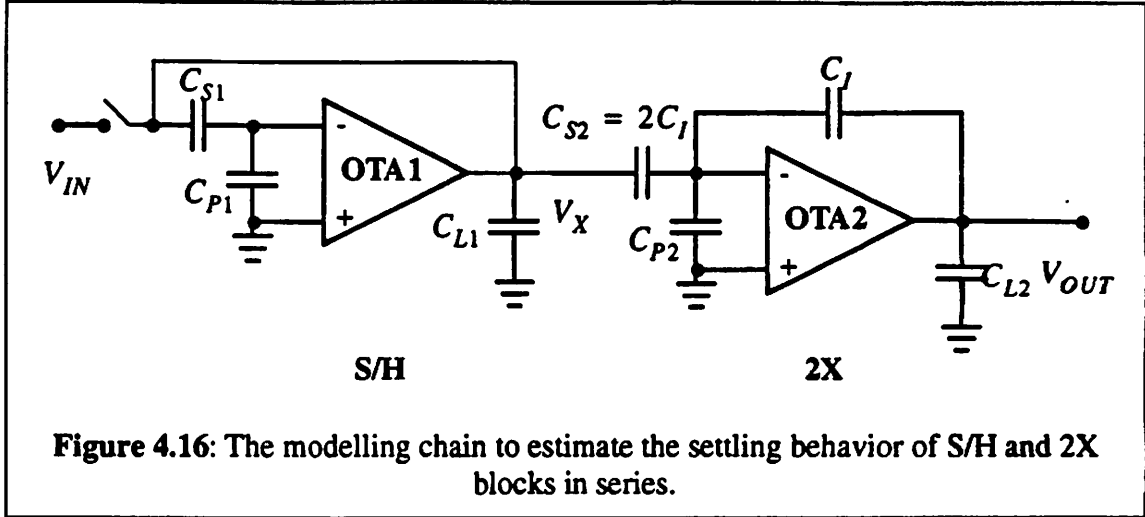
(please refer to Chapter 3 for detailed operation of the ADC), the behavioral model can be further simplified. The behavioral model between $t_0 - t_1$ time period is *equivalent* to the one between $t_2 - t_3$ time period. The same is true between $t_1 - t_2$ and $t_3 - t_4$. As a result, the number of behavioral modelling between time period is reduced by half and the implementation of the simulator for the ADC is greatly simplified. The equivalencies of transition points and time duration model are shown in Figure 4.15.

Critical equations for complex chain of blocks need to be derived and verified



against the actual circuits to make sure that they behave as they are supposed to be. Referring back to Figure 4.14, when Φ_1 is low and Φ_2 is high or between time t_2 and t_3 , S/H-1 is holding the sampled voltage from the previous cycle, 2X is multiplying the output voltage of S/H-1, and the S/H-2 is sampling the output voltage of 2X. Since the sampling of a voltage only involves the sampling capacitor and the switch, we can treat this operation as a load capacitor at the output node of 2X to simplify the discussion. Detailed discussion of the circuit implementation of the entire ADC block can be found in Appendix A. The entire operation during the period $t_2 - t_3$ can be modeled as shown in Figure 4.16.

It is not trivial to model the behavioral model for two consecutive op-amps in closed loop configurations accurately. The usual approach would be to approximate the settling time of these two blocks to be the sum of each block. But this is not true, because as the S/H block is in the hold mode, V_X starts slewing, and at the same time the 2X block is doing the multiplication of this slewing signal. Let us assume that the two op-amps are the same and have a single pole transfer function. If we apply a step function at the input voltage, V_{IN} during the sampling mode, the voltage is then held and multiplied by 2. We



can then derive the following equations.

$$\frac{V_X}{V_{IN}}(s) = \frac{A_{SH}}{1 + s\tau_{SH}} \quad (\text{EQ 4.39})$$

$$\frac{V_{OUT}}{V_X}(s) = \frac{A_{2X}}{1 + s\tau_{2X}} \quad (\text{EQ 4.40})$$

where A_i = the closed loop gain of the i -block

τ_i = the closed loop output time constant of the i -block

$$\frac{V_{OUT}}{V_{IN}}(s) = \frac{V_X}{V_{IN}}(s) \cdot \frac{V_{OUT}}{V_X}(s) \quad (\text{EQ 4.41})$$

Applying the Laplace transform to EQ 4.41, and solve for V_{OUT} , we get:

$$V_{OUT}(s) = A_{SH}A_{2X} \left(\frac{1}{s} - \frac{\beta\tau_{SH}}{1 + s\tau_{SH}} - \frac{(1 - \beta)\tau_{2X}}{1 + s\tau_{2X}} \right) \quad (\text{EQ 4.42})$$

$$\text{where } \beta = \frac{1}{1 - \frac{\tau_{SH}}{\tau_{2X}}}$$

If we transform EQ 4.42 back to time domain, we obtain:

$$V_{OUT}(t) = A_{SH}A_{2X} \left(1 - \beta e^{-\frac{t}{\tau_{SH}}} - (1 - \beta) e^{-\frac{t}{\tau_{2X}}} \right) V_{IN}(t) \quad (\text{EQ 4.43})$$

To verify the validity of EQ 4.43, a spice simulation of the two blocks and a behavioral simulation based on the equation above are compared assuming the two op-amps are the same and the unit capacitors are of the same values. The value of τ_{2X} can then be approximated to be $1.6\tau_{SH}$. The results of the spice simulation and behavioral simulator track very closely as shown in Figure 4.17.

In order to accurately model the two blocks, process variation effects such as capacitor mismatch, offset voltages from op-amps and comparators, sampling circuit finite bandwidth, and many other are taken into account in the behavioral simulator. For detailed implementation, please refer to CADICS users' manual [JUSU93].

4.4.4.2 Floorplan Optimization

Besides evaluating the performances using behavioral simulator, we would like to be able to predict or estimate what the total area of the final layout would be. This is very important since area is one of the chosen design parameters. We will defer the discussion of the floorplan optimization to the next chapter when we discuss the layout generation aspect of the module generation. We will explain why we choose to do fixed floorplan rather than arbitrary floorplan and how we do the floorplan optimization.

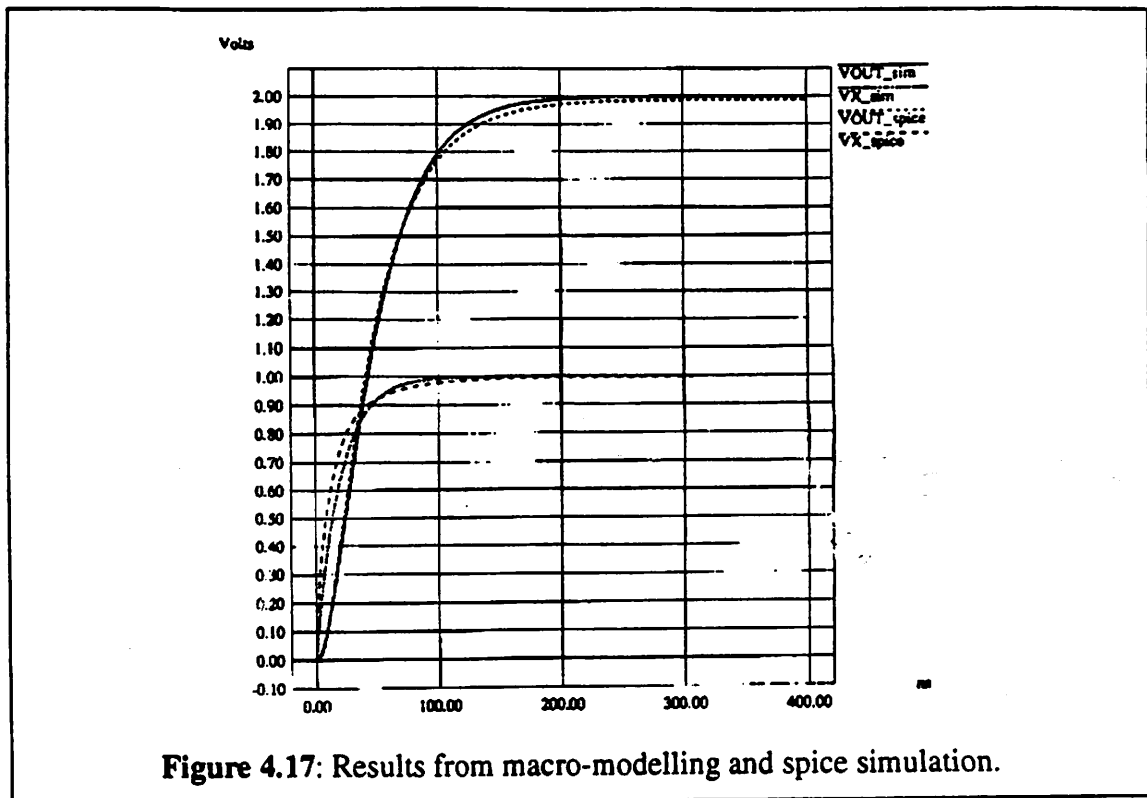


Figure 4.17: Results from macro-modelling and spice simulation.

4.4.5 Scheduler

All the functions discussed in the earlier section can then be summarized in a flowchart shown in Figure 4.18. We call this flowchart as a *customized routine* since it is specific to a particular technology. In this flowchart, the capacitor size selected is checked if its matching is met. If it is not, then a trim-array needs to be added. Currently, the trim-array generator has not been implemented. In Appendix A, a suggested trim array and how to calibrate these capacitors are described.

4.4.6 Quality of the circuit synthesis

The quality of the circuit synthesis will depend on how accurate the analytical equations and the behavioral models describe the circuits. The same problem will also appear in a manual design. The generated ADC netlist is not intended to be used as is but

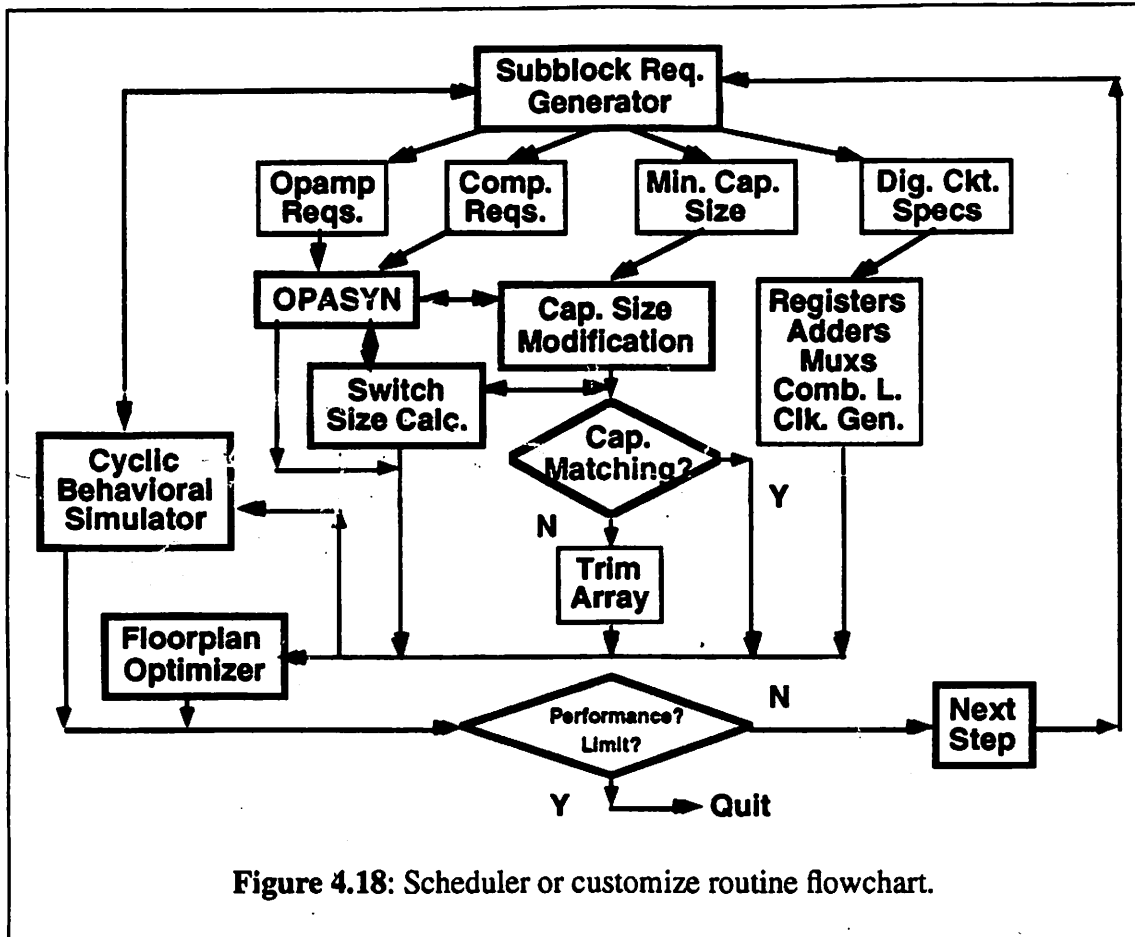


Figure 4.18: Scheduler or customize routine flowchart.

further simulation using different spice files for slow-slow and fast-fast models are recommended. Besides the above factors, other effects such as op-amp and comparator topologies, device matching, and process technology also play important roles in determining the quality of the circuit synthesis.

4.5 Summary

In this chapter, we have presented the approach and implementation of the netlist generation of ADC module generation. A method of doing the architectural selection has been proposed. A hierarchical optimization is introduced to do the device sizing. This type of optimization allows the use of other low level synthesis tools to generate the subblocks.

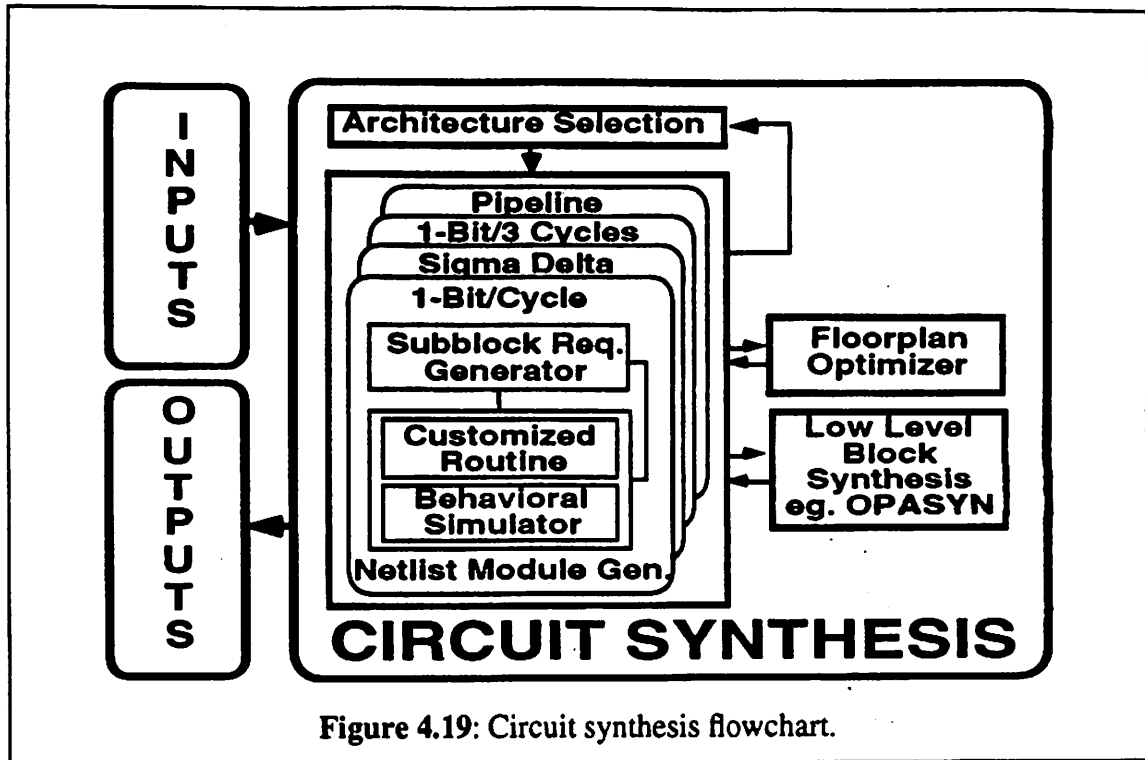


Figure 4.19: Circuit synthesis flowchart.

In order to predict the performances of the generated ADC circuit, a floorplan optimizer and a behavioral simulator are added. The behavioral simulator is a discrete time event simulator based on macro-modelling each functional subblock of the ADC by its analytical model. The outputs of the circuit synthesis are the complete netlist and performance summary of the ADC. The methodology introduced here can be extended to do performance-driven hierarchical top-down design of other complex analog blocks as well as mixed analog-digital systems. Currently, effort to do just this is underway at Berkeley [CHAN92]. The outline of the implementation can be summarized as follows:

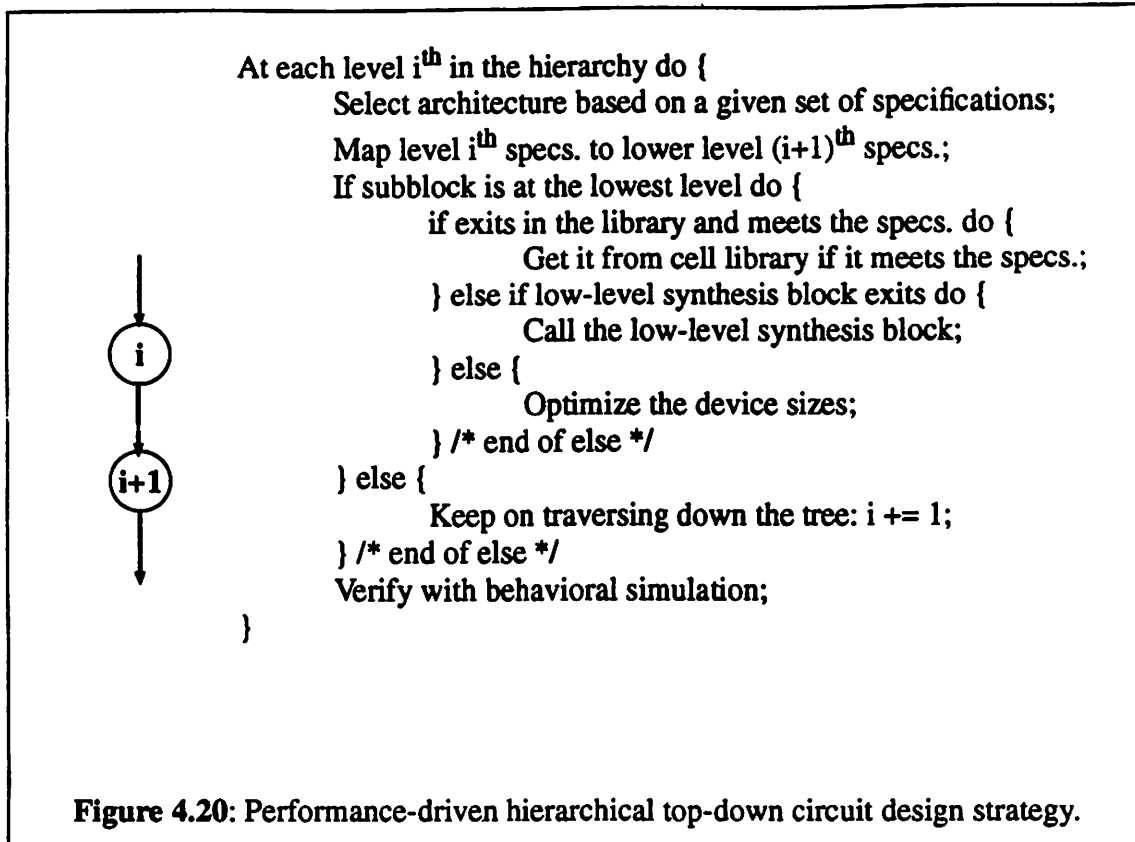


Figure 4.20: Performance-driven hierarchical top-down circuit design strategy.

CHAPTER 5 ADC Layout Generation

5.1 Introduction

In this chapter, a discussion regarding the approach and implementation of the ADC layout generation based on a spice netlist generated by the circuit synthesis will be presented. The layout generation takes the same set of input files as the circuit synthesis with two additional input files: an ADC netlist generated by the circuit synthesis and device structure information file as a result of floorplan optimization. The final output will be a complete layout of the block. The overall flowchart of ADC module generation is shown again in Figure 5.1.

5.2 Approach

Layout generation of analog blocks is not as simple as the one for digital blocks. Many CAD tools written for digital blocks cannot be applied directly to generate analog blocks. In chapter 1, we have discussed why analog CAD tools are not as matured as the digital ones. Many analog CAD tools for automatic layout generation are currently being

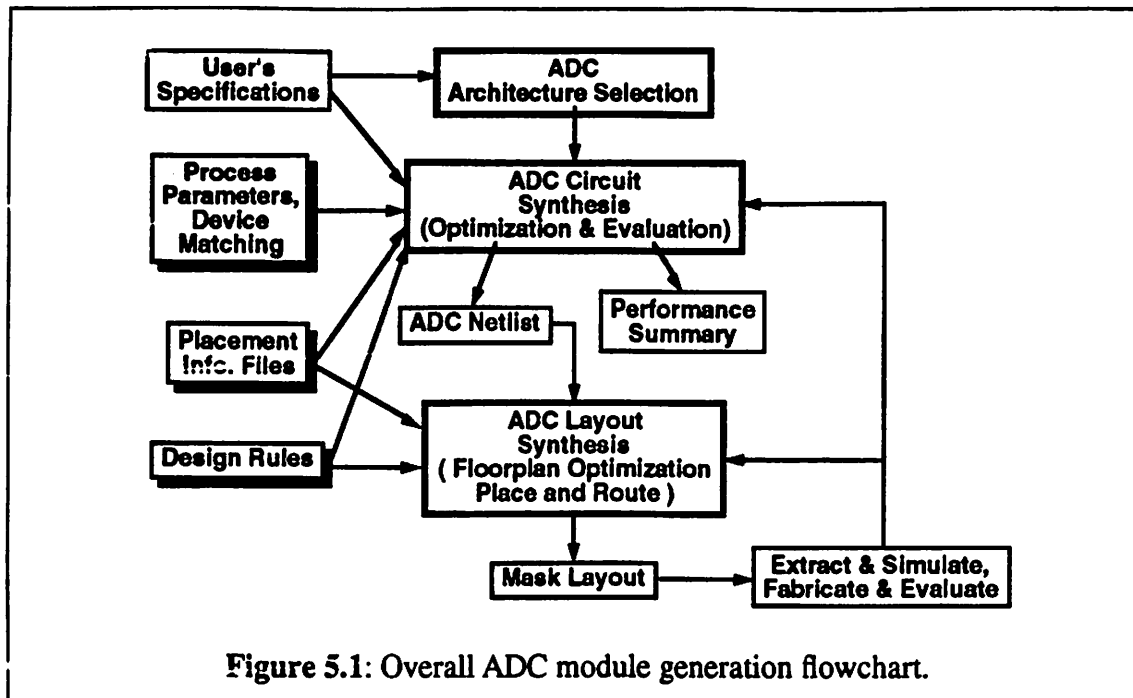


Figure 5.1: Overall ADC module generation flowchart.

implemented and not readily available to be used within our layout generator. When this project was started many of the tools were not available; as a result, the methodology of the layout generation chosen was adapted to suit each issue in doing the implementation of the automatic layout generation. The following two sections discuss two possible approaches for doing automatic layout generation: *flat* and *hierarchical constraint-driven layout procedures*.

5.2.1 Flat Layout Procedure

One common approach for doing automatic layout generation is to do a *flat layout approach*. The basic procedures of this approach is shown in Figure 5.2. For a given netlist or connectivity information, it will first generate all the necessary devices such as transistors and capacitors. All the generated devices are then placed *randomly* and *simultaneously* using a placement program. The placed cells are then routed and compacted. The layout generation steps just described have one objective: *minimize the total silicon area*

of the final layout. Unfortunately, analog circuits depend heavily on *fine details* of layouts. Careless action while doing the layout generation will degrade the performance of the final circuits regardless how superior the performance of the circuit itself. Therefore, the flat approach just discussed is not suitable for automatic generation of analog blocks.



Figure 5.2: A flat approach layout generation.

5.2.2 Hierarchical Constraint-Driven Layout

Another approach would be to do a *hierarchical constraint-driven layout procedure* as shown in Figure 5.3. The basic idea for this approach is that for a given circuit netlist, the layout generator will undergo a sequence of operations before producing final layout.

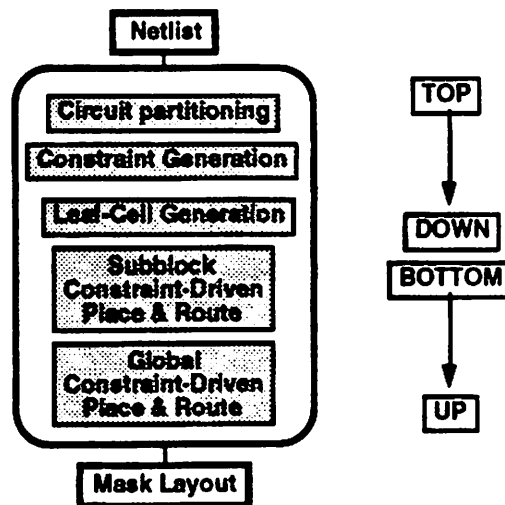


Figure 5.3: Hierarchical constraint-driven layout procedure.

5.2.2.1 Circuit Partitioning

A circuit partitioning step is carried out in order to group devices into functional blocks so that sensitive analog blocks can be grouped together and be separated from the noisy digital blocks. Ideally, the partitioning will be carried out automatically until the block consists of low-level devices. A pseudo-C routine for this step is shown in Figure 5.4(a). Unfortunately, to write a circuit partitioning program that can do this job is almost impossible to do. Therefore, this step is usually done manually. Once we have established the block partition or hierarchy, the layout generation can proceed. In the next section, a *structured-netlist* will be introduced to represent the circuit partitioning of complex analog or mixed analog-digital blocks.

5.2.2.2 Constraint Generation

Once the circuit has been partitioned, constraint generation for these blocks will be carried-out to gather information among the blocks such as coupling between sensitive signals and allowable parasitic capacitances on the interconnects. The constraint generation is done recursively from the lowest level block and up. At each level of the hierarchy, specifications are passed on to the next level down to be used as the input of the next level blocks. At the lowest level, the constraint is generated by maximizing the constraint and it was then passed up to the next level of the hierarchy so that all constraints from the lower level blocks are included in the higher blocks. The constraint generation can be done recursively from the lowest level block and up. A pseudo-C routine to do hierarchical constraint generation is shown in Figure 5.4(b).

While the pseudo code of the hierarchical constraint generation looks very simple, the actual implementation of it is very difficult and is a subject of extensive research project. The methodology and implementation to do constraint generation for a simple block such as op-amps has been reported[CHOU93]. Effort to extend the capability of this


```
circuitPartitioning(block)
{
    if (block!= device_level) {
        partitionBlock(block, &number_of_subblock);
        for (i=1; i<= number_of_subblock; i++) {
            circuitPartitioning(subblocki); /* recursive call to itself */
        } /* end of for */
    } /* end of if */
} /* end of cktPartitioning() */
```

(a)

```
constraintGeneration(block, specifications, constraint)
{
    constraint = NULL;
    if (block != lowest_level) {
        mapSpecification(block, specifications);
        for (i=1; i <= number_of_subblock; i++) {
            constraintGeneration(subblocki, specificationsi, constrainti);
            constraint += constrainti;
        } /* end of for */
    } /* end of if */
    constraint += maximizeConstraint(block, specifications);
} /* end of constraintGeneration() */
```

(b)

Figure 5.4: Pseudo-C routines for (a) circuit partitioning (b) constraint generation.

constraint generator is currently being studied at U.C. Berkeley [CHAN92], [CHAR92].

5.2.2.3 Device Generation, Placement, and Routing

Once all the constraints are generated, it will then proceed to do *automatic floor-planning, area optimization.*, and device generation. The devices are then placed using a constraint-driven placement [CHAR92] and routed using constraint-driven router [CHOU93], [MALA90] to build subblocks. It will then hierarchically traverse up to complete the place and route of the entire block in the same manner. At each level of the hierarchy, the generated constraint is used to drive the placement and router program. To optimize the final layout even further, compaction can be executed at each level of the hierarchy or on the final layout.

This layout generation procedure just described is commonly known as *top-down* and *bottom-up* layout procedure. In the remaining of this chapter will discuss a simplified version of this hierarchical layout procedure to ease the implementation of the layout generation and to produce as good and dense quality of layout as a manual approach.

5.3 Implementation

In this section, implementation of the layout generator that takes a spice-like netlist as its input and generates the complete layout of the block is presented. First, individual topics relating to the implementation will be discussed in great details. At the end, the overall algorithm of the implemented layout generator will be shown and a summary will be given.

5.3.1 Circuit Partitioning and Floorplanning

Ideally we would like to use automatic circuit partitioning program to do the circuit partitioning, but such a tool does not exist. Automatic circuit partitioning program is

not an easy program to write since it needs to capture many variety of analog blocks for the tool to be useful. For this reason, we chose to do customized circuit partitioning i.e. the ADC block is manually partitioned into subblocks. This step is very easy to do since we have identified and broken the ADC into its functional subblocks when generating its circuit netlist. Figure 5.5(a) illustrates how the ADC block proposed in Chapter 3 can be partitioned. The figure only shows the partitioning at one level of the hierarchy. Further partitioning can be done for each blocks until the lowest level of the hierarchy is reached. Figure 5.6(a) shows how SH2 block is partitioned into two blocks: SC and OTA. In turns, OTA consists of many transistors.

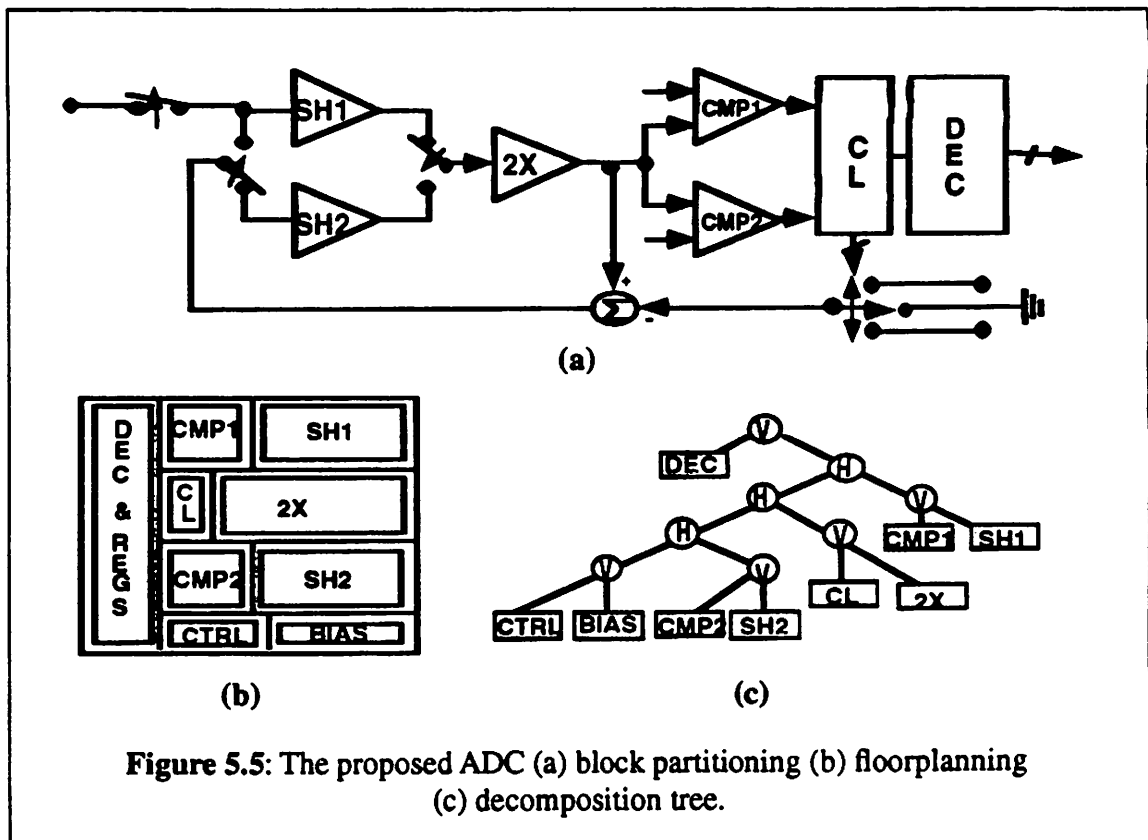


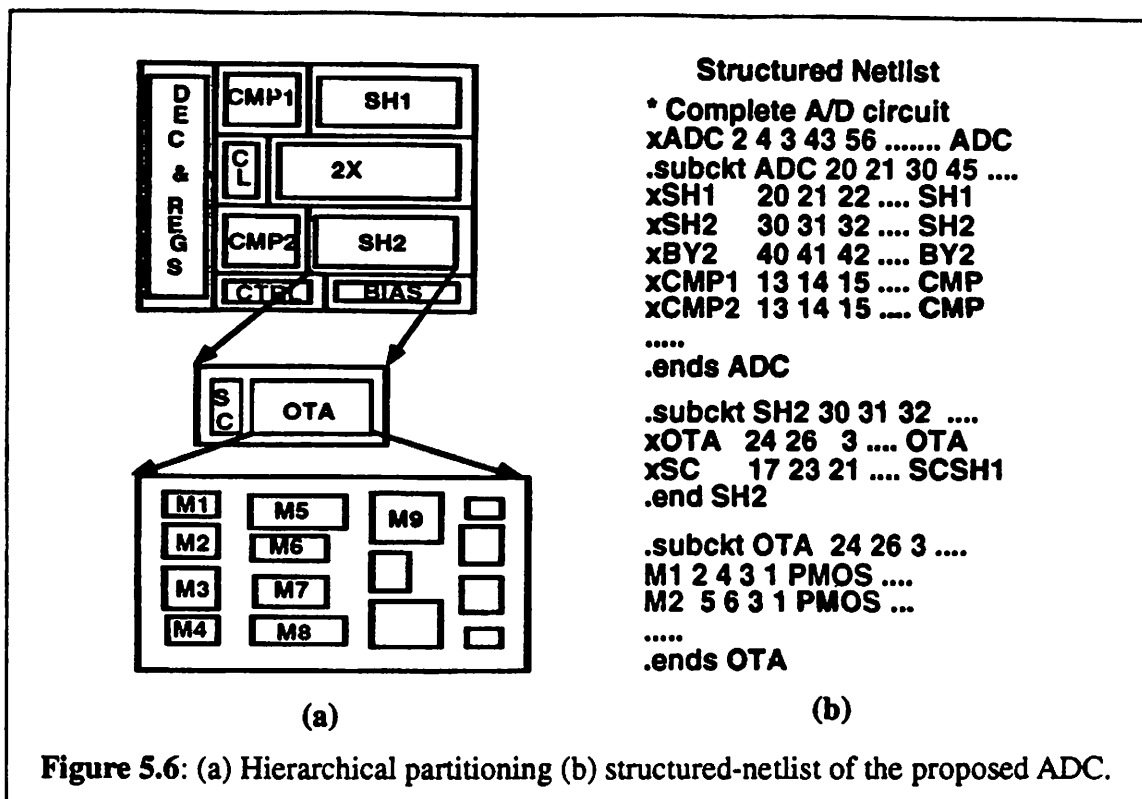
Figure 5.5: The proposed ADC (a) block partitioning (b) floorplanning (c) decomposition tree.

5.3.1.1 Structured-Netlist

To ease the partitioning process, we decided to generate a *structured-netlist* that reflects accurately the block partitioning of the ADC block. The structured-netlist is basically a complete spice-like netlist generated by the circuit synthesis by defining each sub-block using *subcircuit definition* (.SUBCKT) available in SPICE. An example of a structured-netlist of the proposed ADC is shown in Figure 5.6(b). At the top level, the spice netlist calls a subcircuit: ADC. Subckt ADC in turn calls a bunch of subcircuit such SH1, SH2, BY2, and others. Traversing down the subcircuit call for SH2, we can then find that it calls two other subcircuits: OTA and SCSH1. And finally, at the lowest level, OTA subcircuit lists the actual device sizes. This style of circuit netlisting turns out to be very useful since one can look at the netlist and be able to find out how a block is being partitioned. Moreover, it will also simplify the required data structure of the layout generation program (as it will be shown later in this chapter). Figure 5.6 illustrates details of circuit partitioning of the ADC block that is reflected by the structured-netlist.

5.3.2 Floorplanning

Due to the lack of automatic placement tool for analog blocks, we have decided to have a *fixed-floorplan* layout. We have studied the optimum floorplan of the proposed ADC by manually designed and laid-out the complete ADC in order to study its layout morphology as well as verifying the functionality of the ADC circuitry. A *slicing structure* floorplan design is chosen here because it is *well-suited* for hierarchical layout, *easy* to do the routing, *compact* representation of the floorplan information, and moreover *algorithm exists* for optimal shape and orientation of modules. The circuit partitioning and floorplanning for the proposed ADC block is shown in Figure 5.5(a) and (b). Notice that all the analog blocks: SH1, SH2, 2X, and BIAS are gathered on right side of the layout away from the noisy digital blocks.



5.3.2.1 Slicing Structure

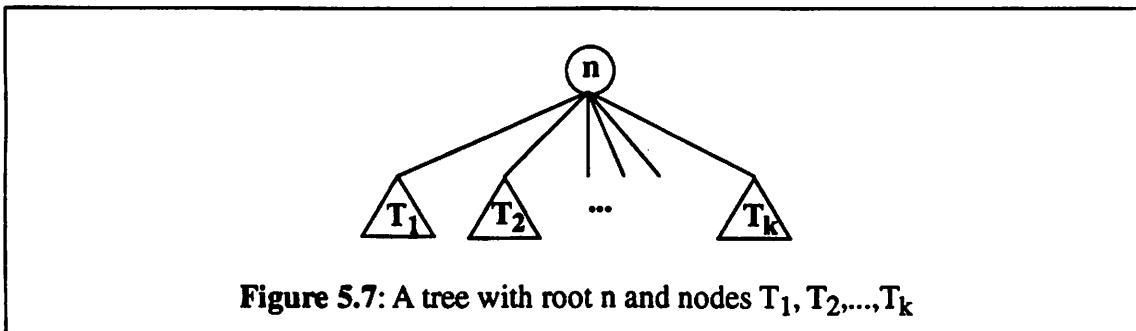
One of the advantages of a slicing structure floorplan is that information can be stored in a simple data structure of a binary tree or commonly known as *decomposition tree*. A decomposition tree for the floorplan of the higher level ADC block is shown in Figure 5.5(c). Each node of the decomposition tree can be assigned *V* for *vertical* or *H* for *horizontal* to indicate how the region is sliced. The *left* and *right* children of a node represents the *left* and *right* slices of a vertical cut respectively or the *bottom* and *top* slices of a horizontal cut respectively. By convention *the first slice* will always be *the left-most* and *the top-most* so that the resulting tree will be in *one-to-one* correspondence with the actual slicing structure of the block. Moreover, other information such as the width and height of the enclosing rectangle can also be stored there.

The top node of the tree which is commonly called *root* represents the top level of

the hierarchy of the block. At the end of each branch of the tree, there will be a *leaf node* which contains a single module such as a transistor or a capacitor. As a result, all the nodes that are not leaf nodes will have two children attached to them.

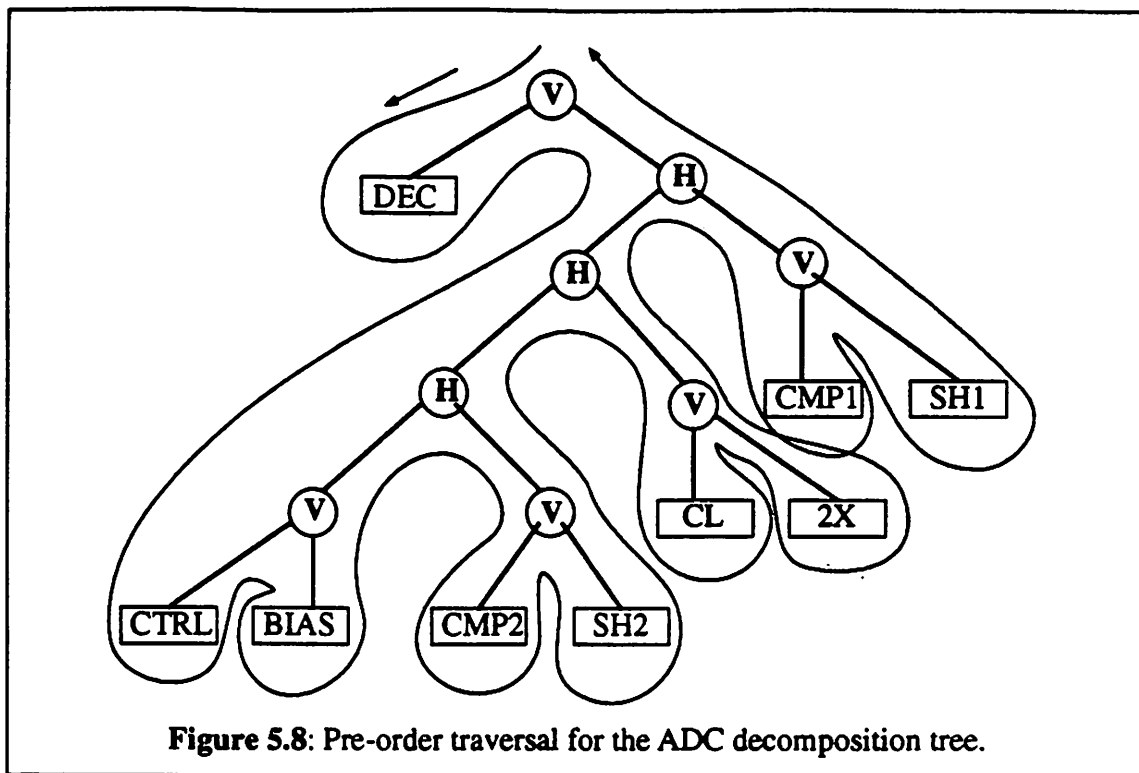
In order to allow user's to have the flexibility of changing device placement, placement information files are made available to be modified. This floorplan files will then be parts of the inputs of the layout synthesis as shown in Figure 5.1. These files consist of block names at higher level of hierarchy and device names at the lowest level of the hierarchy. A *pre-order listing* or *pre-order traversal* is used to stored the data in the files. Looking at Figure 5.7, pre-order listing can be defined as follows:

Definition: The *pre-order listing* of the nodes of T is the root n of T followed by the nodes of T_1 in pre-order, then the nodes of T_2 in pre-order, and so on, up to the nodes of T_k in pre-order.



In other words, as we traverse down the tree from the top node, we list a node the *first time* we pass it. Referring to Figure 5.5(c), the pre-order traversal can be found by travelling through the left child first recursively and then to the right child or simply draw a path on the outer edges of the tree and list the node as the first time we pass it (Figure 5.8). The floorplan file for the tree would be:

V DEC H H H V CTRL BIAS V CMP2 SH2 V CL 2X V CMP1 SH1



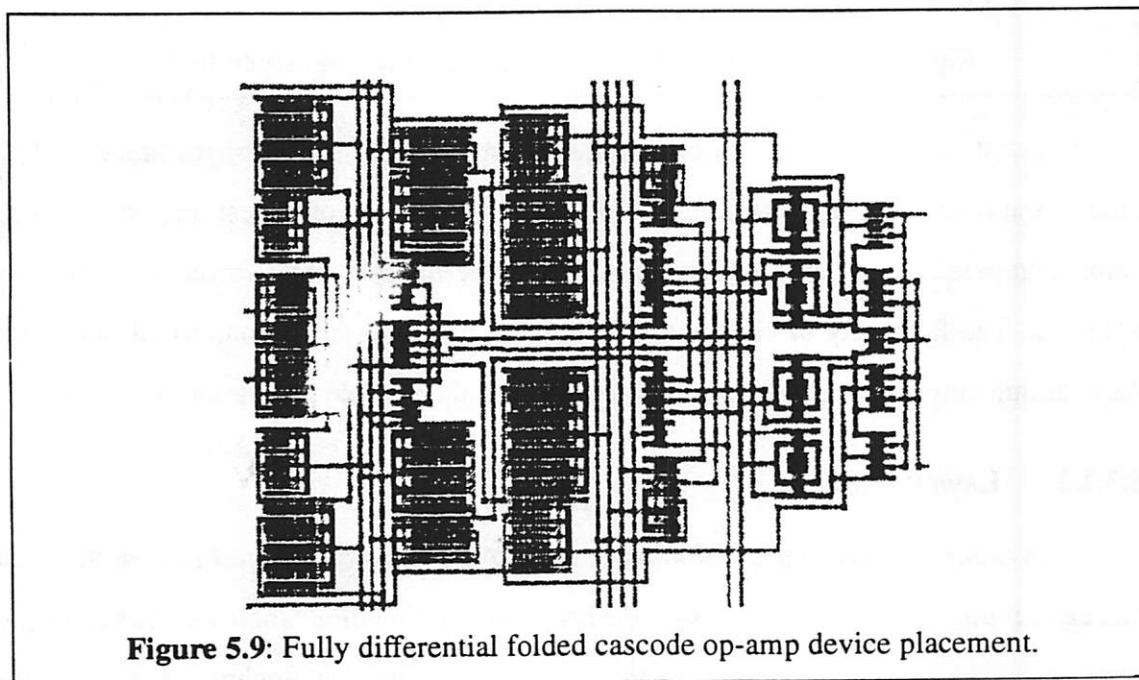
In spite of the freedom to change the floorplan file, one is not recommended to modify the floorplan files of high level blocks other than those of lowest ones such as op-amps, comparators, switches and capacitors since the floorplan files stored now are very optimum. The flexibility of changing the floorplan files will enable one to add different ADC architectures with different floorplannings into the module generation program.

5.3.2.2 Layout Style

In order to improve the performance of the ADC, *fully differential circuits* are used throughout the analog path. As a result, the layout generation of all analog blocks has to be done as symmetrically as possible to preserve this property. One technique that is commonly employed is *mirroring*. The basic idea is to layout the top half of the block and then mirror it to get the other half. This technique is very simple conceptually if it is done manually. Sometime a single block has multiple symmetry axis for its devices. Method to han-

Multiple symmetry axes by using a virtual axis can be found in [MALA91]. For our layout style, we assume we only have a single symmetry axis. Since we have adopted a fixed floorplan layout, we just have to ensure that the devices are placed symmetrically along an axis.

An example of the circuit placement of a folded cascode op-amp used in the ADC is shown in Figure 5.9 respectively. The symmetry axis is chosen to be the horizontal one. The layout style that we adopt here is to place all transistors with their gates parallel to the axis so that drain or source abutment between two transistors can be carried out if possible. The advantages of source or drain abutment of two transistors is to reduce the drain or source to substrate capacitance of the joint node.



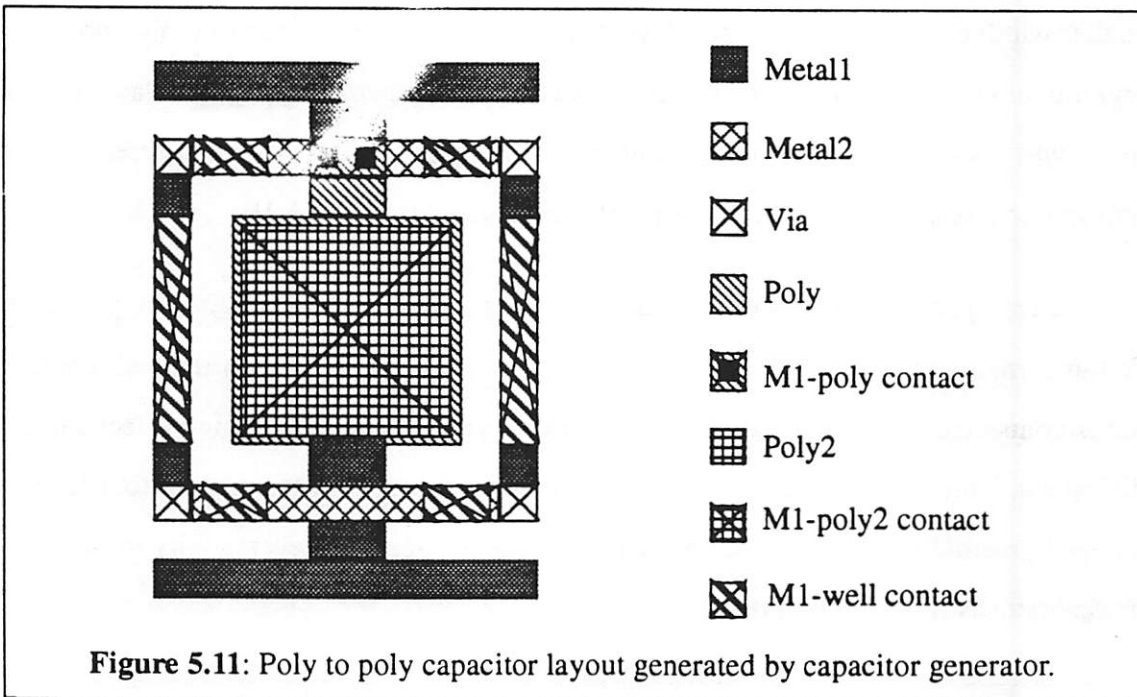
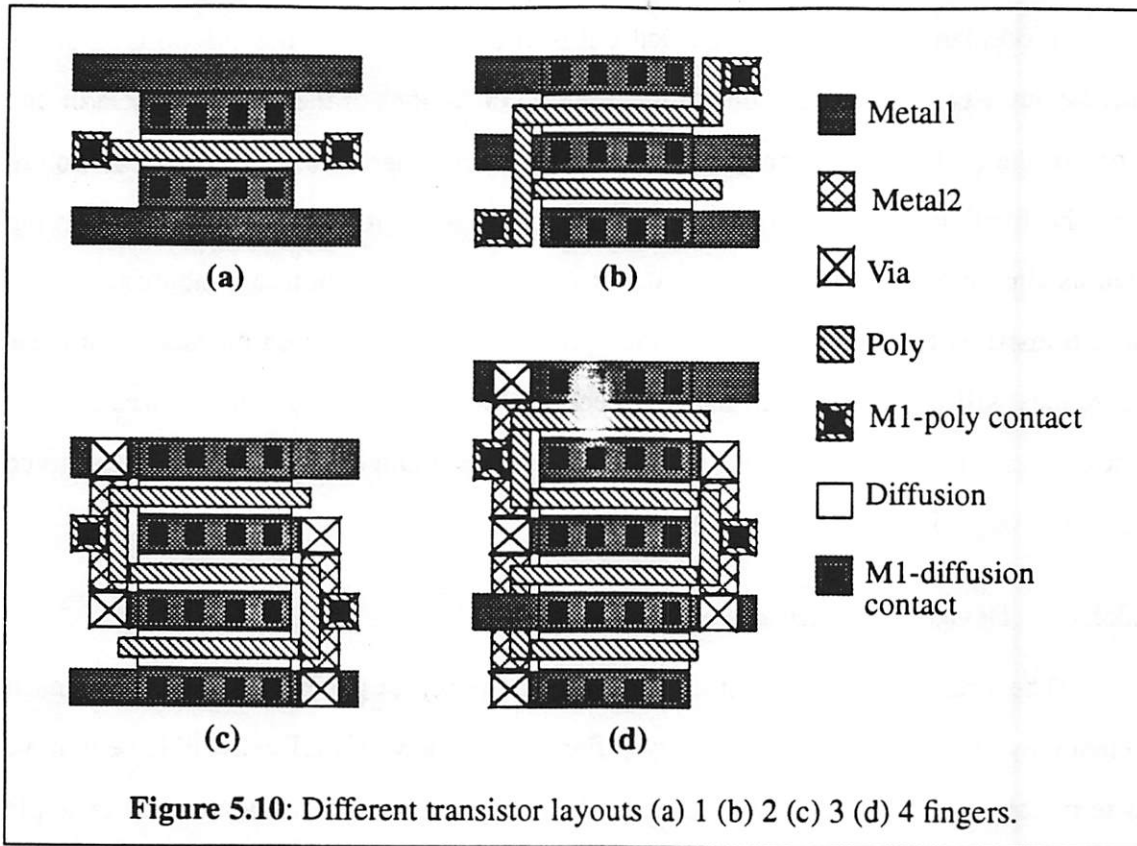
5.3.3 Floorplan Optimization

Floorplan optimization is carried out in order to optimize the silicon area of the final layout. Recall that area is one of the input specifications of the module generator and is one of the design parameters for the circuit synthesis. Therefore, accurate prediction of what the final layout area will be desirable. Techniques to reduce the area consumption such as slicing a transistor into several transistors of smaller width and abutting two or more transistors are incorporated into the optimization step. Detailed discussion of these techniques will be presented in the next section. Moreover, an algorithm introduced by Stockmeyer[STOC83] will be used to determine the optimum shape of the modules given their relative positions as a slicing structure.

5.3.3.1 Device Structure and Shape

The structure and shape of devices used in the layout generator need to be chosen so that they fit into the chosen layout style. For instance, a MOSFET cell will have to have its terminals accessible from both sides of the channel as shown in the op-amp example (Figure 5.9). Moreover, if necessary a transistor will be sliced into several transistors of smaller width to get a much compact layout. The number of slices is commonly known as *finger number*. Thus a transistor generator has to be able to generate transistor layout for a given type, width, length, and *finger number*. Examples of transistors with different finger numbers generated by the transistor generator are shown in Figure 5.10.

For capacitor layout, we have chosen to create a low-noise capacitor by placing a well underneath the bottom plate. The capacitor is then enclosed within the well contact that is connected to a quiet analog supply of the circuit so that the fringing effect can be minimized. Three types of capacitors can be generated based on top and bottom layers: *poly-poly*, *metall-poly*, and *metal2-metal1*. A layout of capacitor generated by the capacitor generator is shown in Figure 5.11.



The well/substrate contact layout is very simple and therefore will be omitted here. A summary of the device generator headers are summarized in Figure 5.12.

createCapacitor() generates capacitor given:

name: capacitor name (if it is not given then a generic name will be used

ex. CAPPN.WxH -> CAP Poly-Poly on N-well of WxH size.)

value: capacitor value in Farads

ratio: capacitor aspect ratio (width/height)

top: type of top layer ex. POLY2, MET1, or MET2

bottom: type of bottom layer ex. POLY1 or MET1

wellType: type of well ex. NWELL or PWELL

unitVal: unit value of 1um X 1um capacitor value

octTech: OCT technology file ex. *dpmos* or *scmos*

(a)

createTransistor() generates transistor given:

mType: transistor type ex. NMOS or PMOS

gateWidth: total gate width in lambda unit

gateLength: length in lambda

finger: number of transistor slices

transName: transistor name

octTech: OCT technology file ex. *dpmos* or *scmos*

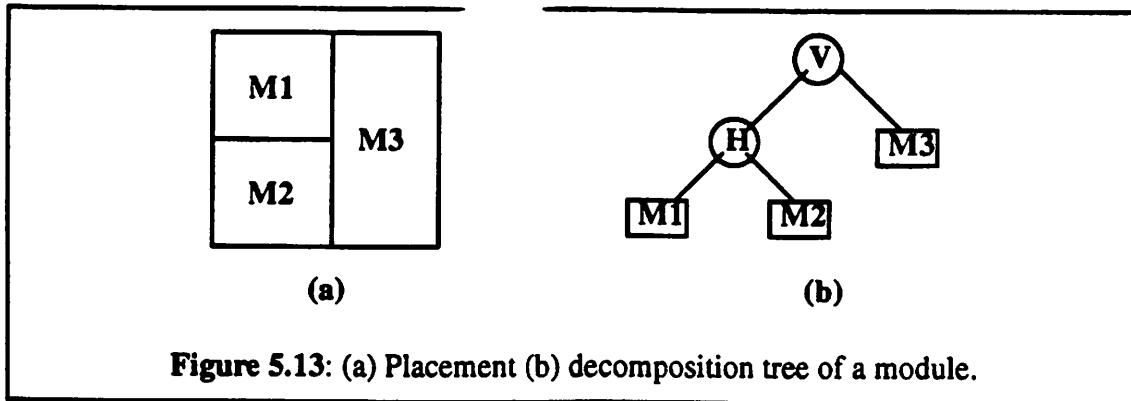
(b)

Figure 5.12: (a) Capacitor (b)transistor generator headers.

5.3.3.2 Modified Stockmeyer's Algorithm

Stockmeyer's algorithm for determining the optimal shape and orientation of modules with slicing structure placement is very well known. This operation can be carried out in polynomial time with the worst case complexity of $O(n^2)$ where n is the number of modules. In 1987 Otten[OTTEN87] defined a *shape constraint relation* of modules in the x-y plane for the shape determination. We will explain by giving an example of how the algorithm is used in the floorplan optimization step.

Let us assume we have a module of 3 transistors: M1, M2, and M3. The placement and decomposition tree are shown in Figure 5.13. M1, M2, and M3 have gate widths and lengths (W_1, L_1) , (W_2, L_2) , and (W_3, L_3) respectively. The *objective* is to find the optimum shapes of the M1, M2, and M3 for a given aspect ratio or width or height of the final module while minimizing the area. Due to our layout style, a transistor will only have a single orientation since 90 degree rotation is not allowed and mirroring the device with respect to x or y axis will not yield a different dimension. However, as we mention earlier a



transistor can be sliced to yield more compact layout. Thus we can define a function that generates a transistor *actual width* and *height* for a given (W,L) and finger number. Let this function be:

$$f(W, L, i) \rightarrow (w_i, h_i) \quad (\text{EQ 5.1})$$

where i is the number of finger

W and L are the gate width and length of the transistor

w_i and h_i are the transistor actual width and height with finger = i

For a transistor the number of possible (w, h) 's for different finger numbers is limited by the lower bound of the minimum width allowable by a design rule of a particular given technology. If the minimum width allowed is w_{min} , the largest finger number would be:

$$finger_{MAX} = int\left(\frac{W}{w_{min}}\right) \quad (\text{EQ 5.2})$$

where W = the gate width of a transistor

For capacitor, the list of possible width and height is continuous since it is a function of aspect ratio. Thus to limit the number of possible (w,h) 's, a minimum and maximum aspect ratio of capacitor will be set for a given technology. Usually, the maximum and minimum aspect ratios are determined by the matching of capacitor ratio. Furthermore, to get a piece-wise linear function, the change of the width and height of the capacitor will be based on an increment of smallest dimension allowed or lambda unit.

For a module with n different width and height, the relation can be defined as:

$$R = \{ (x, y) \mid ((y \geq h_1, x \geq w_1) \cup (y \geq h_2, x \geq w_2) \cup \dots \cup (y \geq h_n, x \geq w_n)) \} \quad (\text{EQ 5.3})$$

For illustration, let us assume $(W_1, L_1) = (16, 2)$ and $w_{min} = 4$, the number of possible width and height will be 4. Referring to Figure 5.10, as the number of finger increases the actual transistor height will also increase on the other hand, the actual width decreases. Thus we can assume that $w_1 \geq w_2 \geq \dots \geq w_4$ and $h_1 \leq h_2 \leq \dots \leq h_4$. The shape constraint relation of this example is shown in Figure 5.14 with shaded area.

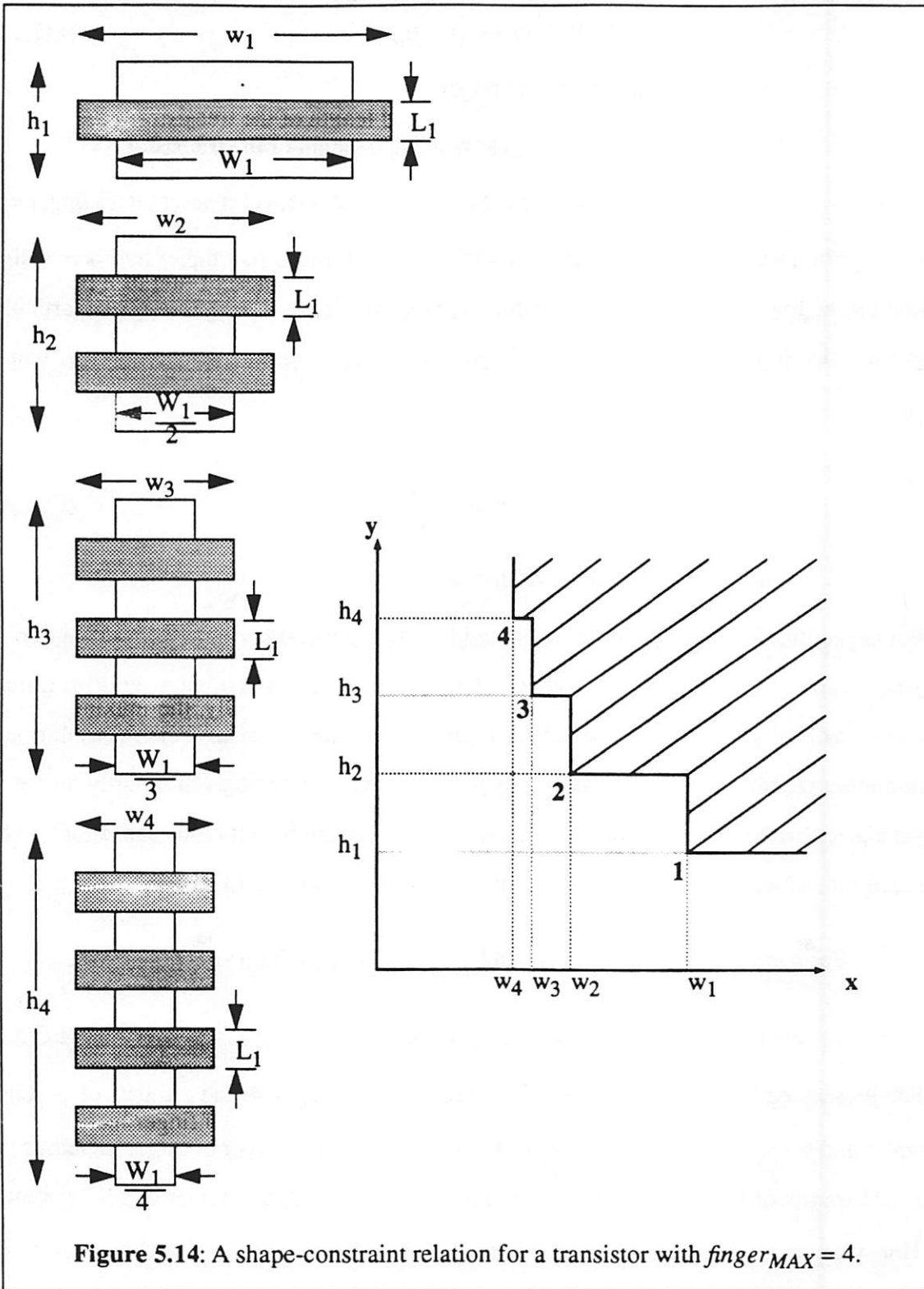


Figure 5.14: A shape-constraint relation for a transistor with $finger_{MAX} = 4$.

Referring back to Figure 5.13, M1 and M2 are placed on the top and bottom of each other respectively. We can then find the shape-constraint relation graph for the two modules combined. The shape-constraint relation of the resulting block is given by the following equation:

$$R_{M12} = \{ (x, y) \mid (x, y_1) \in R_{M1} \text{ and } (x, y_2) \in R_{M2} \text{ and } y = y_1 + y_2 \} \quad (\text{EQ 5.4})$$

where R_{M1} and R_{M2} = the shape constraint relations of M1 and M2

If we assume that M1 and M2 have $\text{finger}_{MAX} = 5$ and the shape-constraint relation graphs are shown in Figure 5.15(a) and (b) respectively, the combined shape-constraint relation can be obtained by graphically adding the two graphs in the y-axis direction. The shape-constraint relation of the combined module, M12 is shown in Figure 5.15(c).

The combined module, M12 and M3 are the left and right of each other. The equation describing the combined shape-constraint relation for two modules placed the left and right of each other is given by the following equation:

$$R_{M123} = \{ (x, y) \mid (x_1, y) \in R_{M12} \text{ and } (x_2, y) \in R_{M3} \text{ and } x = x_1 + x_2 \} \quad (\text{EQ 5.5})$$

where R_{M12} and R_{M3} = the shape-constraint relations of M12 and M3

By the same method, we can find the combined module of M12 and M3 by graphically adding the shape-constraint relation graphs of the two in the x-axis direction. If the shape-constraint relation of M3 is as shown in Figure 5.15(d), the resulting shape-constraint relation is shown in Figure 5.15(e). For a given width, the final height can be obtained from the graph and transistor structures within the module can then be gotten by traversing down the shaped-constraint relation graph in a reverse operation. If the module has final width = **W** and the final height = **H** as shown in Figure 5.15(f), M1, M2, and M3 have finger numbers of 2, 1, and 2 respectively.

Often for a given final shape-constraint relation graph, the given starting point falls

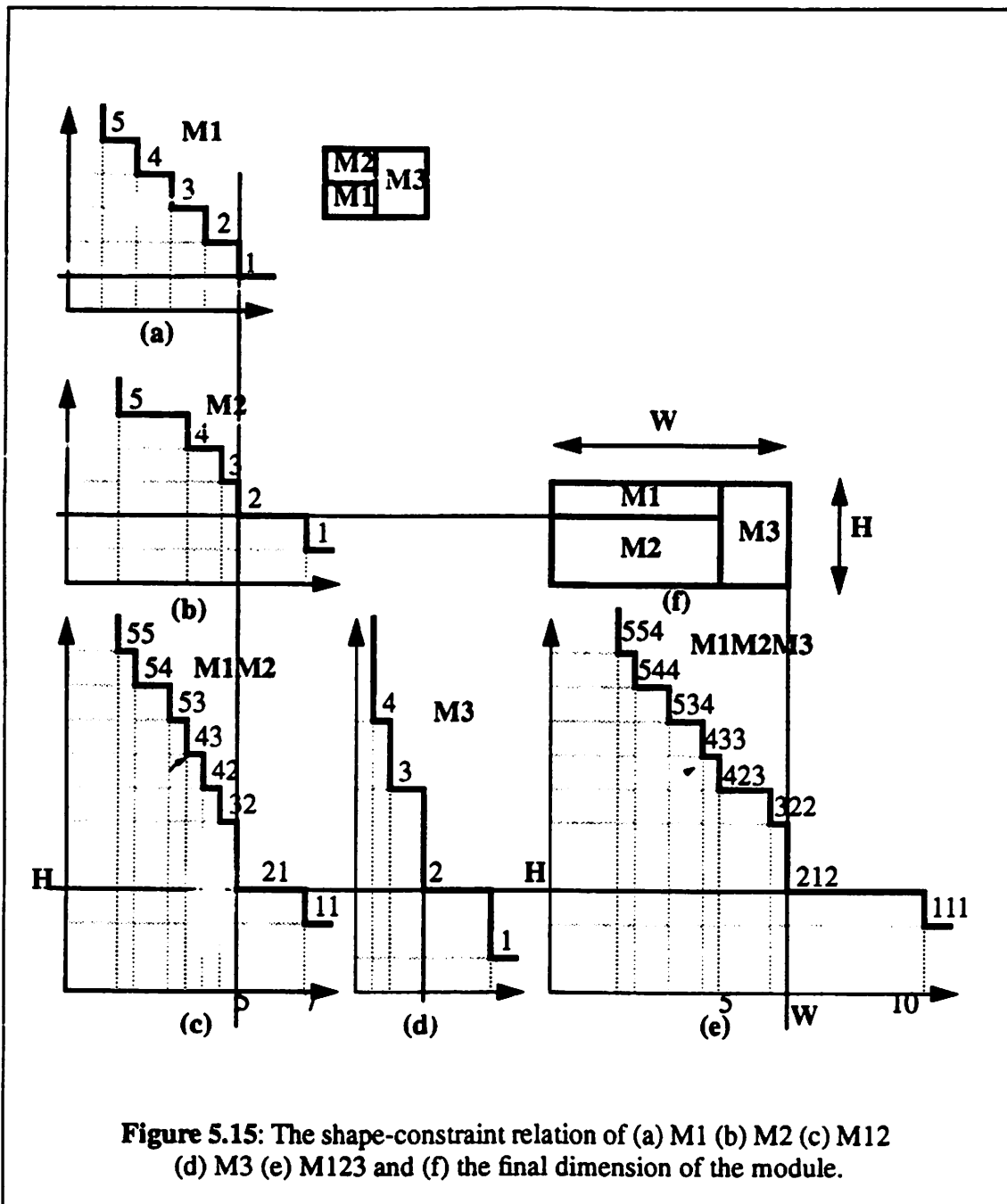


Figure 5.15: The shape-constraint relation of (a) M1 (b) M2 (c) M12 (d) M3 (e) M123 and (f) the final dimension of the module.

either within or outside the solution space. Thus a search needs to be carried out in order to find the solution point. Let us assume that we have a shape-constraint relation, R_i as shown in Figure 5.16(a) and a starting point (W_0, H_0) based on the given area (A) and aspect ratio

(R) of the requested final block.

Case 1: (W_0, H_0) is in the solution space as shown in Figure 5.16(b). There exists a point or points $p_i \in S$, where S is a set of feasible point on R ; bounded on the x-axis by W_0 and on the y-axis by H_0 .

$$p_i = (x_i, y_i) \quad (\text{EQ 5.6})$$

$$x_i \leq W_0 \quad (\text{EQ 5.7})$$

$$y_i \leq H_0 \quad (\text{EQ 5.8})$$

Since the solution points, p_2, p_3 , and p_4 lie inside the rectangular region bounded by the starting point, the solution points will yield an area which is smaller than the requested one. As a result, to choose an optimum solution point, we would like to find a point that has the closest aspect ratio to the requested aspect ratio. In other word, we would like to find a point, $p_{opt} \in S$ such that the aspect ratio error is minimized.

$$\text{Min} \left\| \frac{H_0}{W_0} - \frac{y_i}{x_i} \right\| \quad (\text{EQ 5.9})$$

Case 2: (W_0, H_0) is outside the solution space as shown in Figure 5.16(c).

There exists a solution point, $p_{opt} \in R$ where

$$x_{opt} \leq W_0 \text{ and } y_{opt} \geq H_0 \quad (\text{EQ 5.10})$$

$$\text{or } x_{opt} \geq W_0 \text{ and } y_{opt} \leq H_0 \quad (\text{EQ 5.11})$$

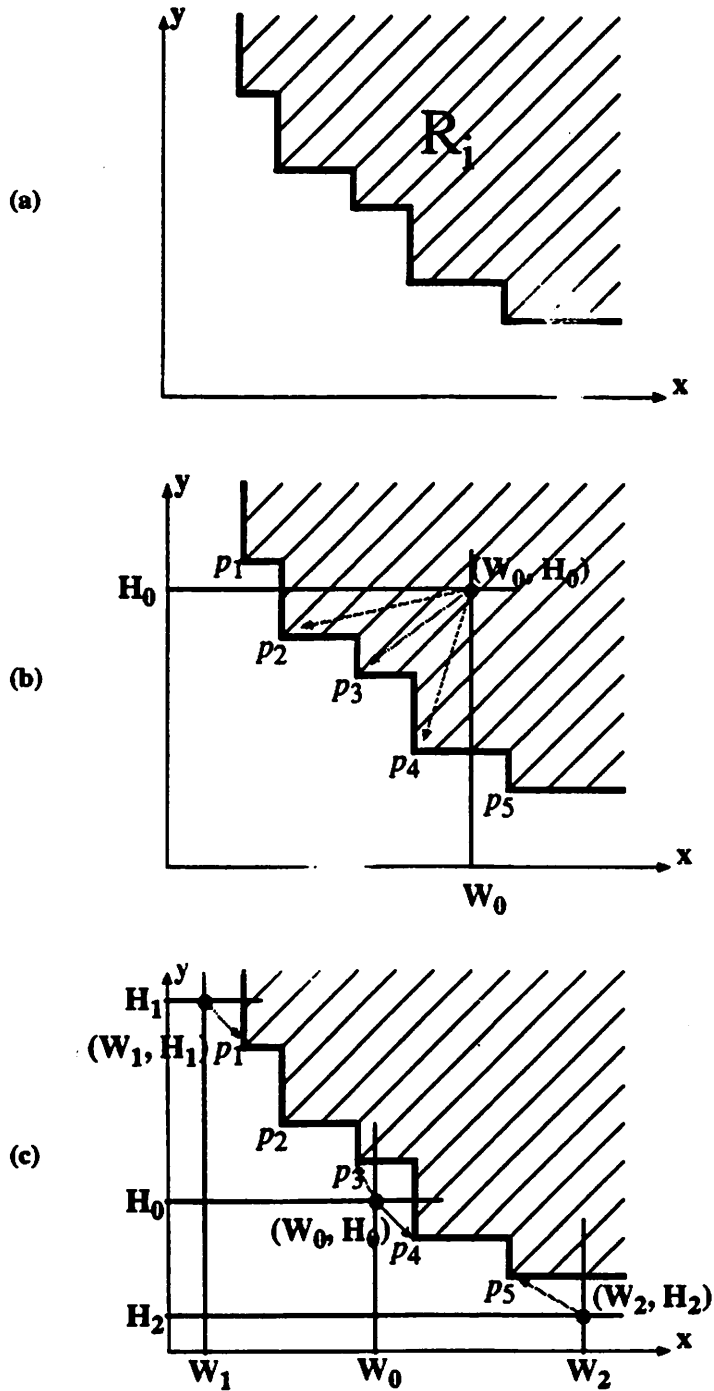


Figure 5.16: (a) A shape-constraint relation example (b) with starting point inside the solution space (c) with starting points outside the solution space.

The solution point can be determined by minimizing the resulting area *and* minimizing the aspect ratio error.

$$\text{Min } (x_i \cdot y_i) \text{ and } \text{Min } \left\| \frac{H_0}{W_0} - \frac{y_i}{x_i} \right\| \quad (\text{EQ 5.12})$$

Thus, the possible solution point(s) is always in the vicinity of the starting point. Referring to Figure 5.16(c), for starting point (W_1, H_1) , there will only be one possible solution point, p_1 and for starting point (W_0, H_0) , there are two possible solution points: p_3 and p_4 . Finally, for starting point (W_2, H_2) , p_5 is the only possible solution point.

In order to predict accurately what the total area is, the algorithm needs to be modified a little to take into account the area used for channel routing. Fortunately, this is an easy modification to do. The information about the channel area routing can be attached on to the nodes implicitly since the node representing horizontal or vertical cut can be viewed as a horizontal or vertical channel. The channel area can then be added directly to the shape-constraint relation graph. For vertical node, the addition is done to the x-axis and for horizontal node it is done to the y-axis. This channel area addition will show up on the graph as a constant shift in the x or y direction. In the next section, we will discuss how we do the channel area estimation.

5.3.3.3 Channel Area Estimation

There are many different ways to estimate the channel area needed to do the routing [KURD86], [ZIMM88]. These techniques are mainly geared toward reducing the number of horizontal tracks needed while doing the routing. Usually, they involve sophisticated database structures, graph theories, or algorithms. What we need for our channel area estimation is a *quick and simple way* of predicting the channel width needed to assure

that all nets within the channel will successfully be routed. The channel width information is then passed on to be used to modify the width and height of the final silicon area. As a result the total area estimation of the entire block will be much more accurate.

We chose to use a *simplified version* of zone representation of horizontal segments [YOSH82] to find the maximum density in a channel. The algorithm of this procedure will be given in the form of example. Let us consider a channel with two rows of terminals along its top and bottom edges as shown in Figure 5.17. Each terminal is represented by a netlist name which it belongs. For instance net-1 is a two-terminal net and net-2 is a three-terminal net. We assume that there are two layers for routing. One for horizontal tracks and one for vertical tracks.

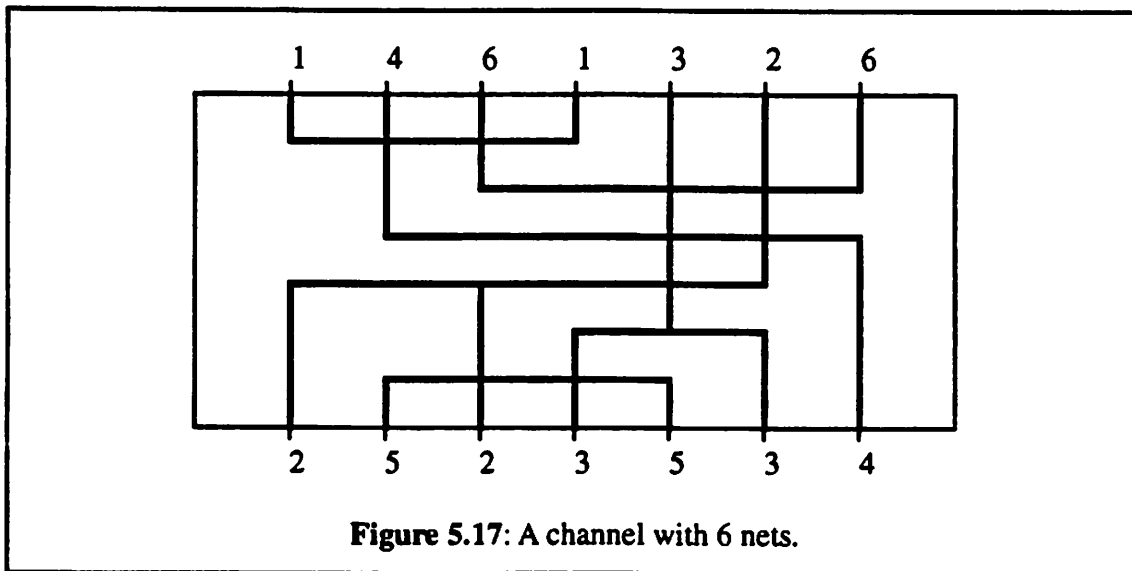


Figure 5.17: A channel with 6 nets.

Let $S(i)$ be the set of nets whose horizontal segments intersect column i . Starting from the most left column to the most right column, we can tabulate $S(i)$ as shown in TABLE 5-1. The number of nets in $S(i)$ is called local density of the i^{th} column. By finding the maximum number of nets among $S(i)$, we obtain the maximum channel density

needed to guarantee 100% completion. TABLE 5-1 shows that $S(4)$ has the densest nets which is 6. Thus the number of horizontal tracks needed to complete the routing is 6.

TABLE 5-1

Column	S(i)
1	1, 2
2	1, 2, 4, 5
3	1, 2, 4, 5, 6
4	1, 2, 3, 4, 5, 6
5	2, 3, 4, 5, 6
6	2, 3, 4, 6
7	4, 6

5.3.4 Data Structure

Due to the complexity of the input data that needs to be stored, a special data structure has been implemented. This data structure is designed so that input data can be stored and retrieved fairly quickly and efficiently. There are three types of input data: structured-netlist, floorplan information, and device structure information. All this information are stored within the data structure. Figure 5.18 shows a simplified version of the data structure used.

5.3.5 Routing

Channel routing is currently carried out by using a gridless channel router, GLITTER [CHEN86] developed here at Berkeley. Currently, a constraint-based channel router.

```

typedef struct blockCell {
    blockName;          /* name of the block */
    nodeName_list;     /* list of node names in ASCII */
    netlistType;       /* SUBCKT, SUBCKT_CALL, NMOS, PMOS, CAP, RES */
    cellType;          /* LEAFCELL, SUBBLOCK, TOP_BLOCK */
    lib;                /* STD_CELL, MOD_GEN, DON'T_CARE */
    union {            /* Information about a device such as: */
        transistor;    /* transistor type, width, length, number of slicing */
        capacitor;     /* width, height, ratio, value */
        resistor;      /* reserve for future use */
        cell;           /* width, height, number of nodes */
    } device;
    leafCell;          /* is the lowest level of a block where devices are defined */
    device_property;   /* information such as merged devices, rotation */
    net_property;     /* information such as matching nets */
    floorplan_tree;   /* floorplan information tree */
    struct blockCell *psubblock;
    struct blockCell *pNext;
} s_blockCell;

```

Figure 5.18: Simplified data structure for storing layout information.

ART [CHOU90] is being implemented into the layout synthesis as an alternative channel router. We are also studying the possibility of using a constraint-based maze router, RoAD [MALA90] to route the subcircuit.

The overall layout generation step can be summarized in the form of pseudo-C routine as shown in Figure 5.19. The layout generator is implemented according to the

OCT [HARR86] database. The complete flowchart of layout generator and netlist generator can be combined and is shown in Figure 5.20. Notice that the floorplan optimizer is

```
layoutGeneration()
{
    spice2db(); /* Store spice netlist into database */
    checkDb(); /* Make sure all subckt calls are defined */
    readFloorplan(); /* Store floorplan info into database */
    rearrangeNode(); /* for possible abutment and well/substrate contact */
    if (modified == true) then {
        optimizeFloorplan(); /* Generate new device structure */
    } /* end of if */
    generateDevice();
    recursively /* From bottom-up */
        placeCell();
        routeCell();
    } /* end of recursion */
} /* end of layoutGeneration() */
```

Figure 5.19: A pseudo-C routine for layout generation.

shared between the circuit synthesis and the layout synthesis. The idea is to give user a flexibility of modifying the generated ADC netlist before generating the layout. If the netlist is modified, the floorplan optimization needs to be executed again in order to generate a new set of device structure information. The outputs of this ADC module generator are complete ADC netlist, layout, and performance summary. In order to complete the loop, manual extraction of the layout needs to be done. The extracted netlist is then simu-

lated if desired before it is being fabricated. Evaluation results of the fabricated can then be obtained to improve the quality of the module generation.

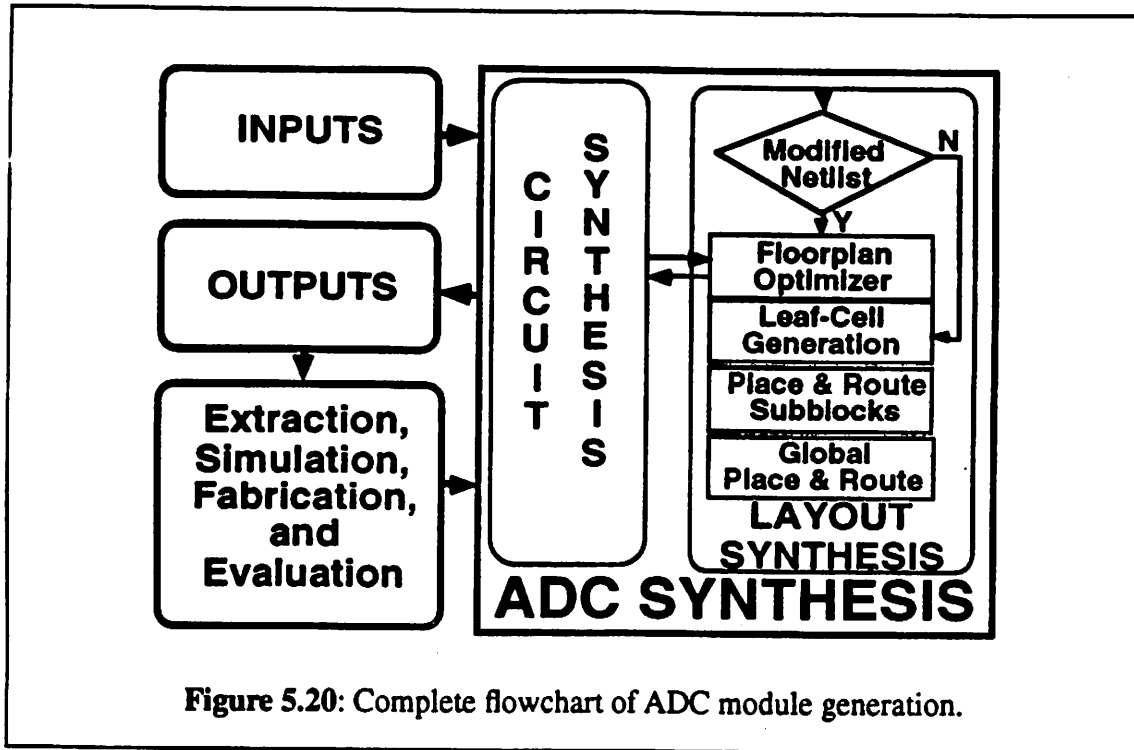


Figure 5.20: Complete flowchart of ADC module generation.

5.4 Summary

We have discussed an implementation of the layout generator that takes a spice netlist and generates a complete layout. Different implementation steps such circuit partitioning, floorplanning, floorplan optimization, and routing techniques are presented. Algorithms and pseudo-C routines of many of the techniques are also given. In the next chapter, examples from different input specifications will be presented. Evaluation results of a fabricated ADC prototype will then be summarized.

CHAPTER 6 Examples and Test Results

6.1 Introduction

In the following chapter, several ADC examples manually designed as well as automatically generated will be presented. One of the automatically generated examples has been fabricated in a 2- μm CMOS process through MOSIS. The test set-up as well as test results will also be shown.

6.2 Examples

6.2.1 Manually Designed Example

In order to understand the behavior and layout morphology of the new-improved algorithmic ADC proposed in Chapter 3, we chose to design and layout complete ADC block manually. While we were doing the layout, we have explored different ways of arranging subblocks and have chosen the one that gives the best floorplan and has the most efficient area. Many of the knowledge obtained are then implemented within the module

generator. We feel that this exercise is very important for one to understand what it takes to design and to implement an ADC module generator.

The manually designed ADC prototype was designed and fabricated in a 3- μm CMOS process through MOSIS. The specifications of this prototype is tabulated in TABLE 6-1 under manual design header. The layout is shown in Figure 6.1.

TABLE 6-1 : ADC specifications for MOSIS 3 μm CMOS process.

Specifications	MOSIS 3 μm¹		Units
	Manual Design	Synthesized	
Resolution	9	9	Bits
Conversion speed	100	109	kHz
Maximum DNL	0.5	0.40	LSB
Maximum INL	0.5	0.49	LSB
Gain error	-	0.729	-
Power dissipation	7.00	8.05	mWatts
Total area	7200	8826	mils ²
Aspect ratio (W/H)	1.0	1.19	-
Supply voltage	± 2.5	± 2.5	Volts
Maximum V_{REF}	1.0	1.2	Volts
Time	~ 10 months ²	~ 3 minutes ³	-

Note 1: Capacitor matching ratio for 1pF is assumed to be 0.1%
 Note 2: Time spent to learn and design manually from scratch
 Note 3: CPU time needed on DEC 5000

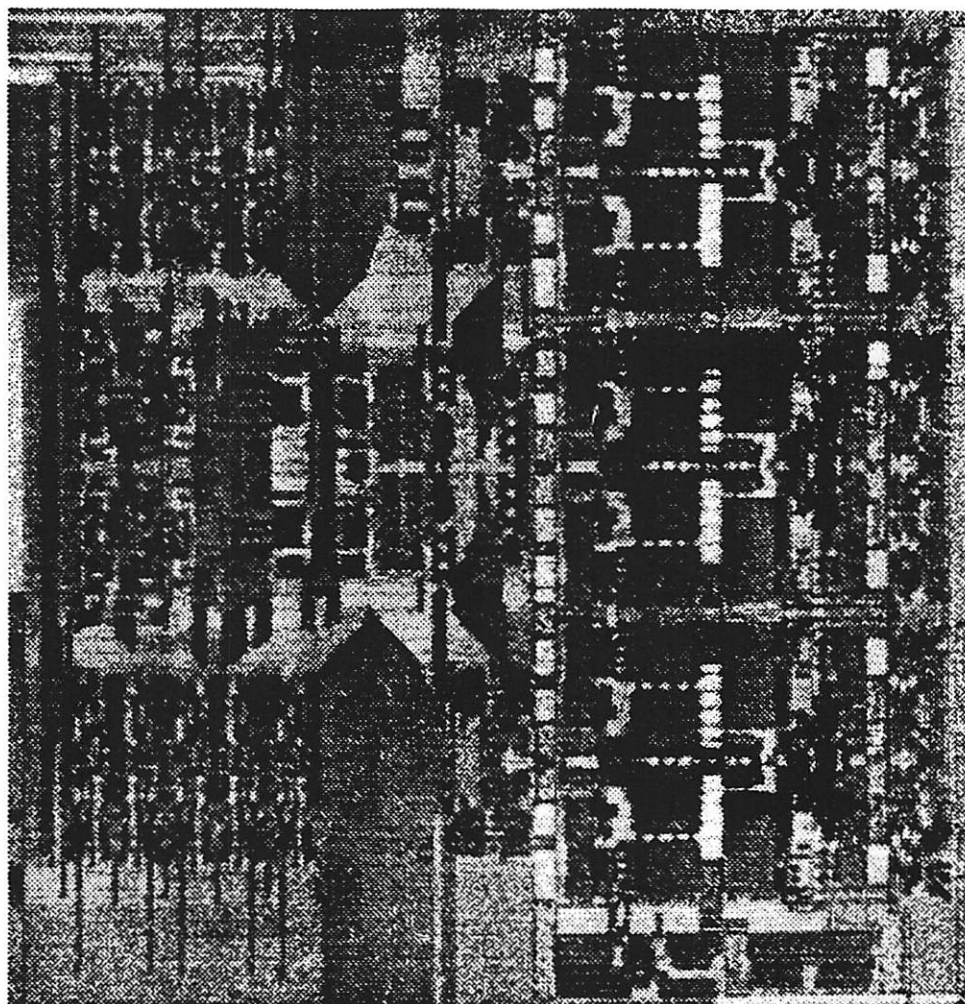
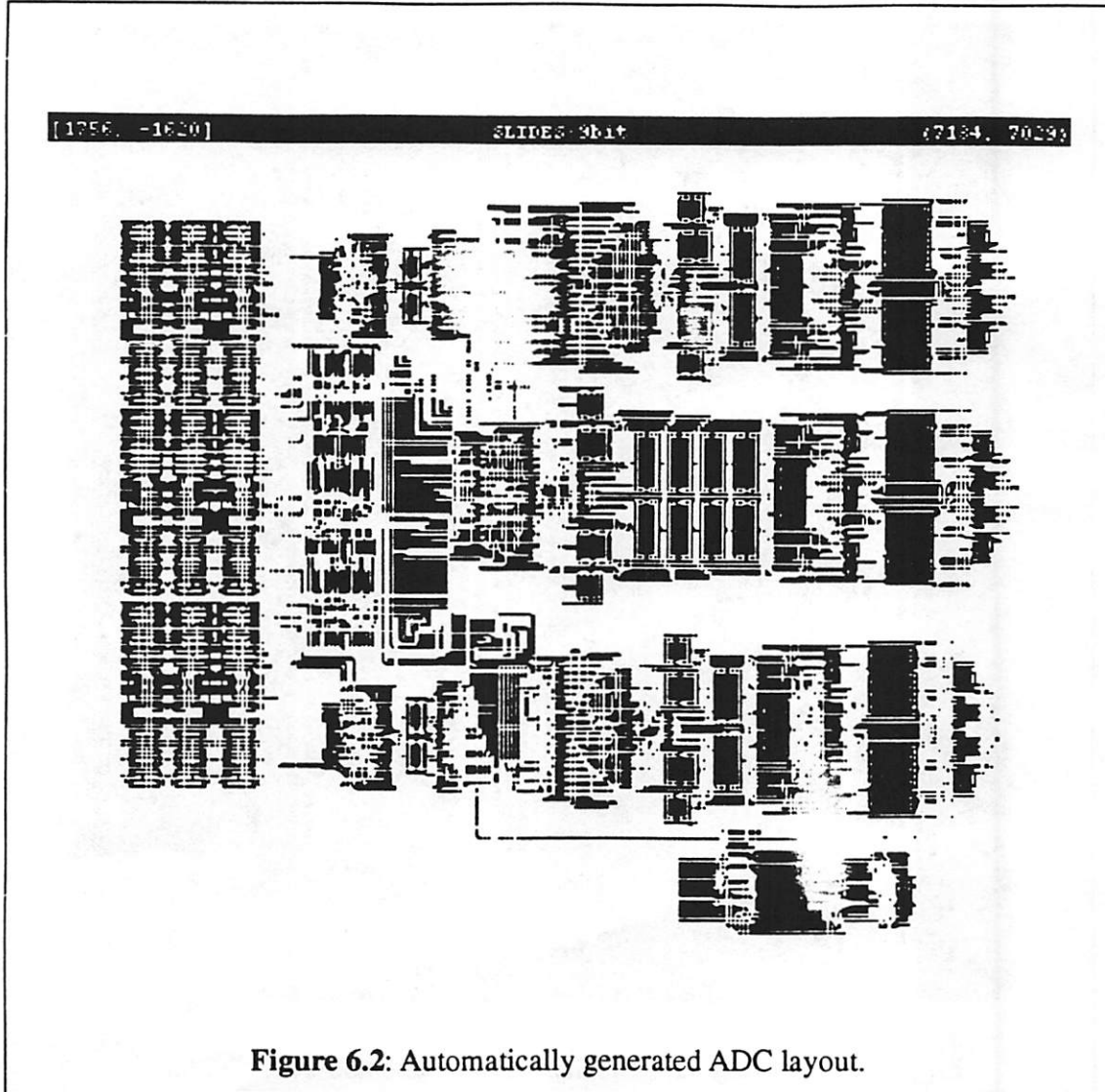


Figure 6.1: Layout of a manually designed and laid-out ADC block.

6.2.2 Automatically Generated Examples

We first generated an ADC from a set of specifications similar to the one that was used for the manually designed prototype. TABLE 6-1 tabulates the performance summary of the generated ADC block for a given set of specifications. The priorities were given to the first four rows of the specifications and they are *resolution*, *conversion speed*,

maximum differential non-linearity (DNL), and maximum integral non-linearity (INL).

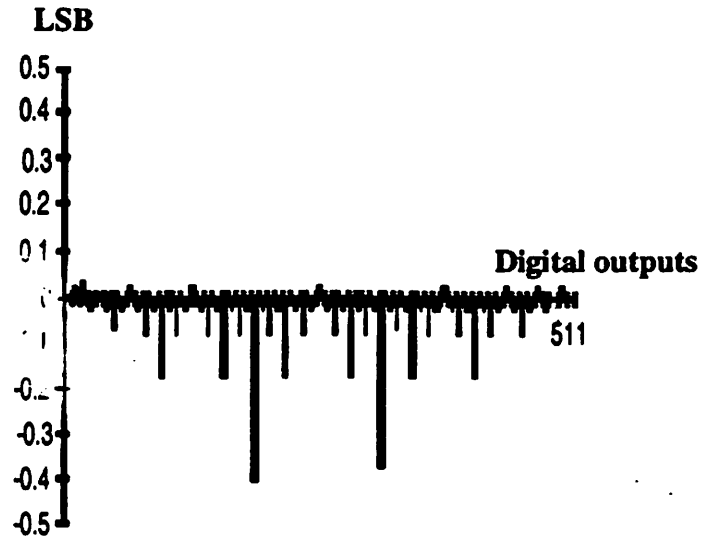


All the specifications requested are met by the module generator except the total area and power dissipation. What it means is the module generator needs more area and power than the given specifications in order to fulfill the higher priority specifications. The total area required is some how much larger than the specified one. The automatically gen-

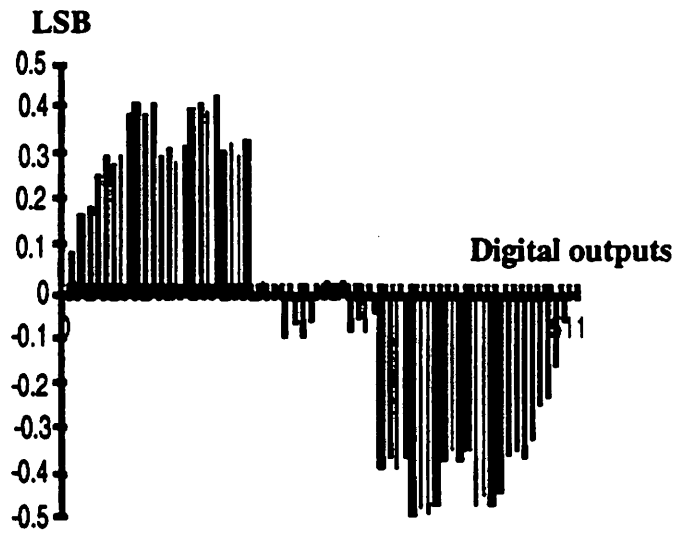
erated layout of the ADC is shown in Figure 6.2. This is expected because the automatically generated layout methodology is based on a hierarchical top-down and bottom-up layout procedure. As each subblock is built from the lower level blocks, there will be a bounding box called *frame* defining the boundaries of the rectangular region enclosing the block. Within the frame, there will be empty area that was unused while the subblock is generated. However, this empty space is invisible from the current level since only formal terminals of the lower blocks and the frame are visible during the placement and routing. As a result, once an empty space exists in a subblock, it will remain unused in the final layout.

A possible solution for this problem would be to do compaction to every block at each level of hierarchy as we traverse up to build the complete layout. We have tried available compaction program, SPARCS [BURN86] to compact each block at each level of hierarchy. Unfortunately, the result is very poor since the program is implemented for digital blocks. Symmetry property of devices and signals will be altered in the process of squeezing every empty spaces possible. On some blocks, the program halted due to over-constrained problem. Efforts to implement an analog compaction program is currently going on at Berkeley [FELT92]. Upon completion of this program, it could be integrated in the layout synthesis step of the module generator to get much denser layout.

Besides the performance summary, CADICS also provides *worst case* estimation of DNL and INL graphs generated by the behavioral simulator as shown in Figure 6.3(a) and (b). As a comparison, 3 different ADC specifications: 6, 8, and 10 bits were given to the ADC module generator and the resulting layouts are shown in Figure 6.4(a), (b), and (c) respectively. Notice that the sizes of the resulting layout scale appropriately. The digital error correction registers on the left hand side of each layout get bigger as the number of bits increases.

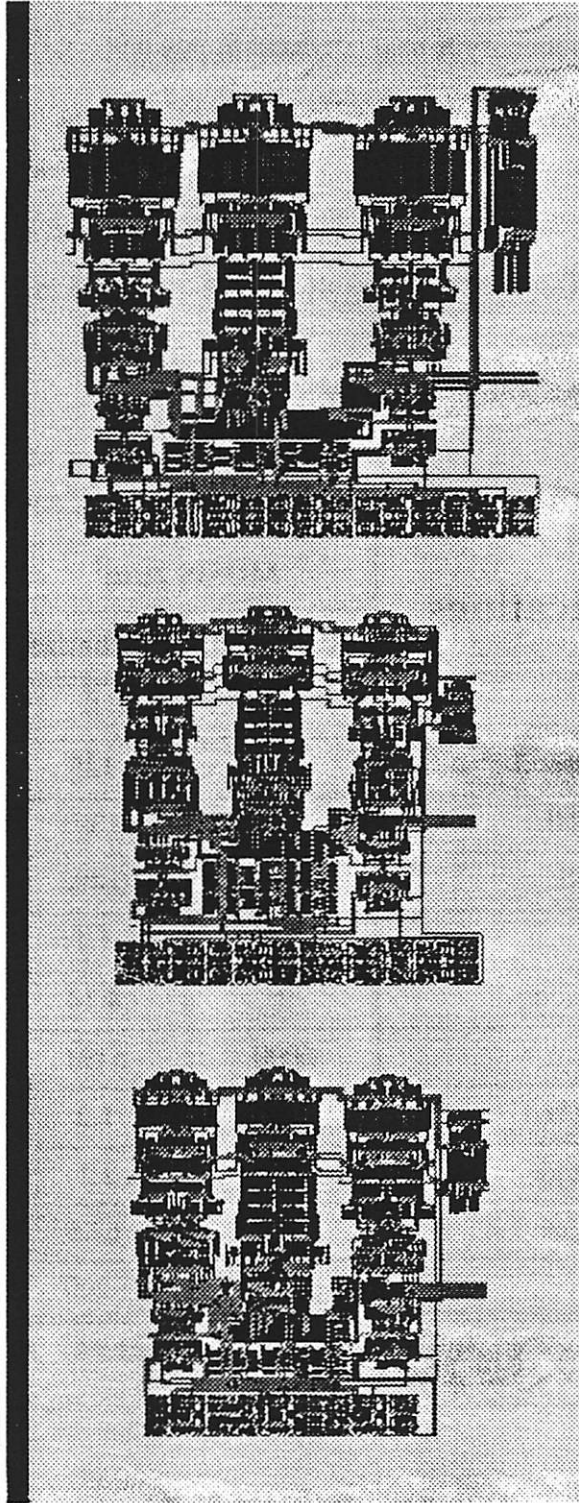


(a) Differential non-linearity



(b) Integral non-linearity

Figure 6.3: Simulated linearity graphs.



(c) 10-bit

(b) 8-bit

(a) 6-bit

Figure 6.4: ADC layouts of different specifications.

The ADC specifications for a 2- μm CMOS process is shown in TABLE 6-2. The resulting performance summary is also tabulated in this table. Semi automatic pad placement using MOSAICO [CASO89] was performed to the final layout in order to shield sensitive analog signals away from the digital signals. As a result, some blocks were adjusted to ease the final routing to the pads. This automatically generated ADC prototype has fabricated and evaluated. Test set-up and results will be presented in the following two sections.

TABLE 6-2 ADC specifications for MOSIS 2 μm CMOS process.

Specifications	MOSIS 2 μm^1		Units
	Requested	Synthesized	
Resolution	8	8	Bits
Conversion speed	100	125	kHz
Maximum DNL	0.5	0.395	LSB
Maximum INL	0.5	0.486	LSB
Gain error	2.0	1.7	-
Power dissipation	4	4.14	mWatts
Total area	6000	5191	mils ²
Aspect ratio (W/H)	1.0	1.2	-
Maximum V_{REF}	1.0	1.1	Volts
Supply voltages	± 2.5	± 2.5	Volts
Time	-	~ 200	Seconds

Note 1: Capacitor matching ratio for 1pF is assumed to be 0.1%
 Note 2: CPU time needed on DEC 5000

6.3 Test Set-up

The prototype is packaged in a 28-pin DIP. The bonding diagram is shown in Figure 6.5 and pin assignment is shown in TABLE 6-3. The test set-up shown in Figure 6.6 consists of clock generation circuitry, power supply and reference voltage generators, and a digital-to-analog converter (DAC) circuitry to convert digital signal back to its analog voltage. A wired-wrapped test board was designed to do the testing, unfortunately, the board was very noisy. Preliminary results using this board were shown in the next section. As an alternative, a printed-circuit board (PCB) was also designed to see if better performance can be obtained. The board was designed carefully so that analog ground and digital ground are isolated from each other. Details schematic of the test board will be not be shown here.

TABLE 6-3 Pin assignment.

Pin #	Pin Name	Descriptions
1	AVSS	Analog VSS supply voltage
2	VREFP	Positive voltage reference supply
3	VREFN_4	One fourth of negative reference supply
4	AGND	Analog ground
5	VREFP_4	One fourth of positive reference supply
6	VREFN	Negative reference voltage supply
7	VINN	Negative input voltage
8	VINP	Positive input voltage
9	AVDD	Analog VDD supply voltage
10	DVSS	Digital VSS supply voltage
11	$\overline{D7}$	Most significant bit active low
12-18	D6-D0	Bit 6 to 0 of the digital outputs

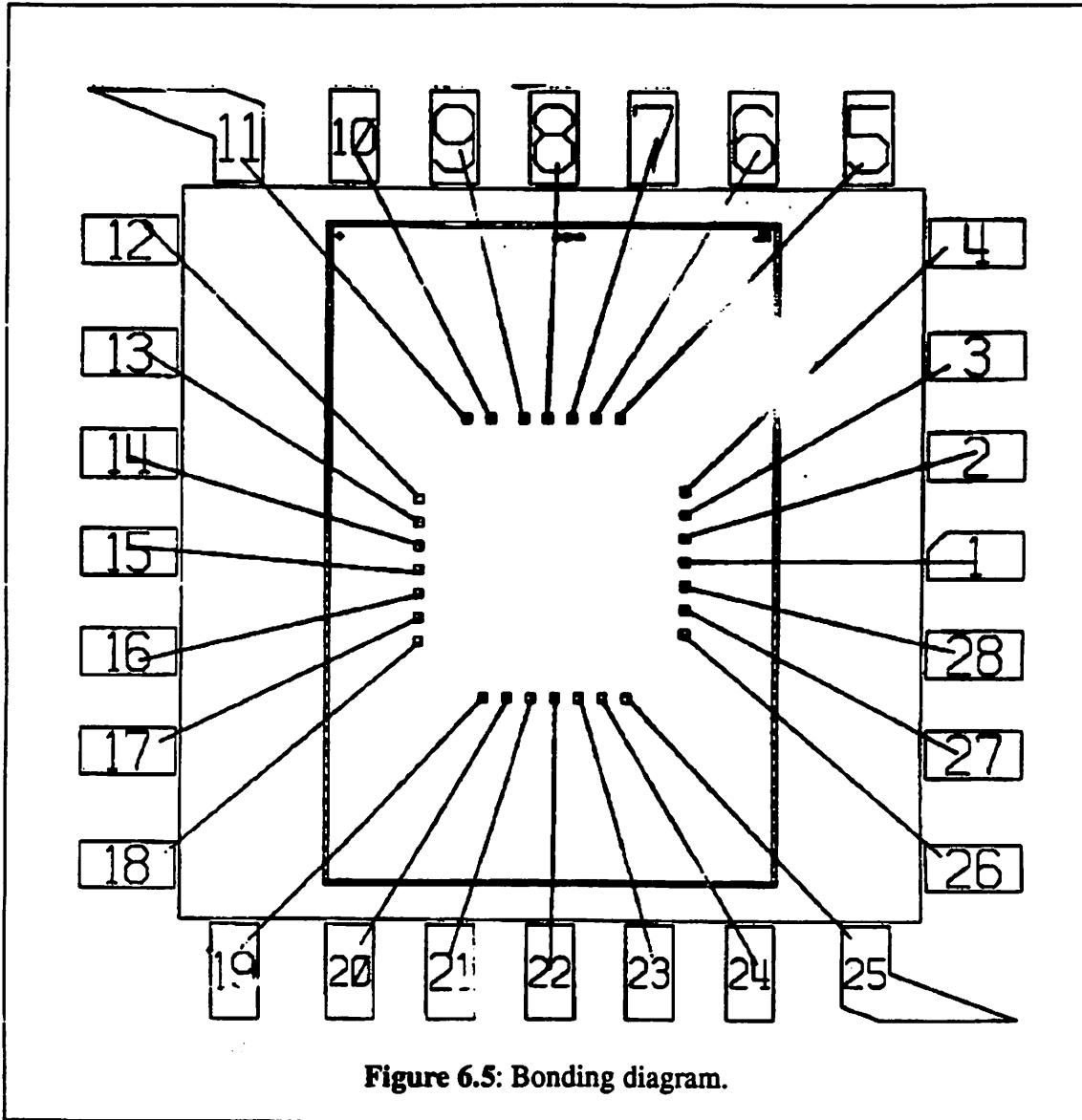


Figure 6.5: Bonding diagram.

TABLE 6-3 Pin assignment.

Pin #	Pin Name	Descriptions
19	CMP2	Comparator2 output
20	CMP1	Comparator1 output
21	PHI2CM	PHI2 common mode clock
22	PHI2	PHI2 clock
23	PHI1CM	PHI1 common mode clock
24	PHI1	PHI1 clock
25	PHISCM	PHIS common mode clock
26	PHIS	PHIS clock
27	DVDD	Digital VDD voltage supply
28	IBIAS	Bias current node

6.4 Test Results

Two different tests were performed using the wire-wrapped test board. The first test was to input a 2 kHz sinusoidal waveform and to sample the ADC at 100 kHz. The collected digital output data was reconstructed to get the sine-wave as shown Figure 6.8(a). A 1024-point discrete fourier transform (DFT) of the collected digital data was performed to find the dynamic range of the ADC. The result is shown in Figure 6.8(b). The result is not as good as we expected. The SNDR is around 38 dB which is slightly less than 7 bit resolution. We suspect that the noise radiation on the wire wrapped test board is so severely that it degrades the SNDR performance. Noise spike greater than 20 mV was observed. We expect to be able to obtain SNDR higher than 40 dB using the new test board.

A beat frequency test was carried out with input frequency of 92 kHz and sampling

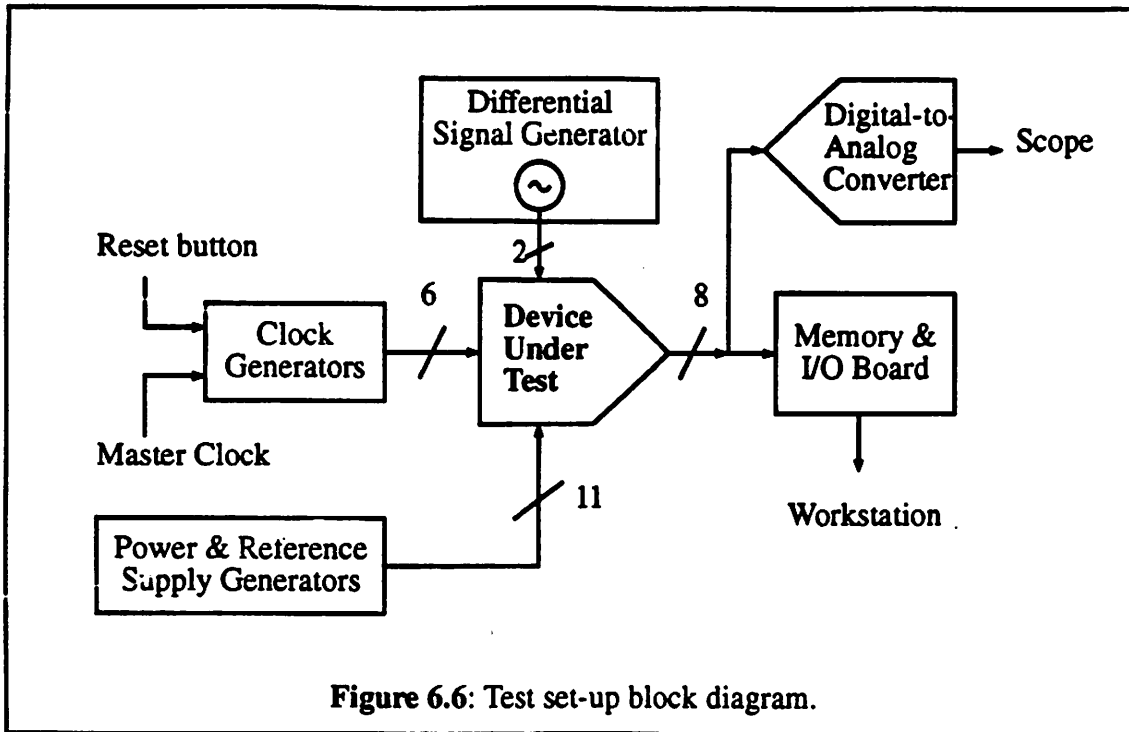
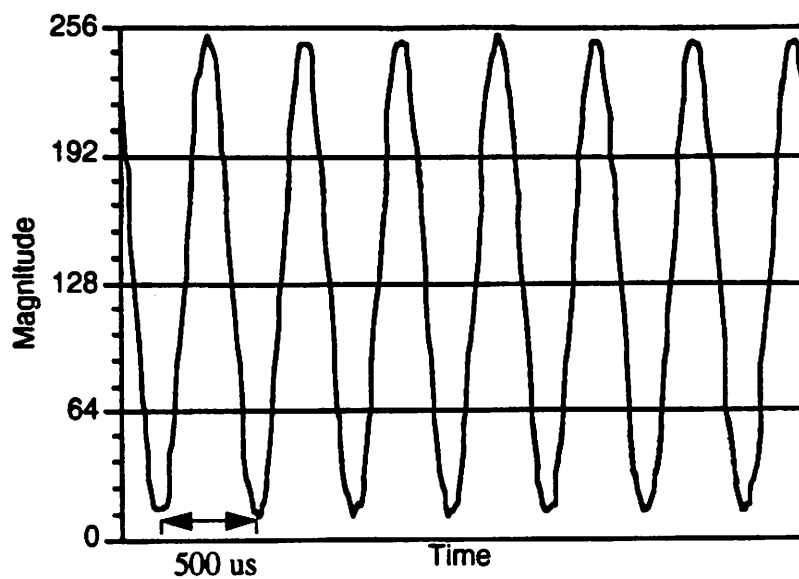


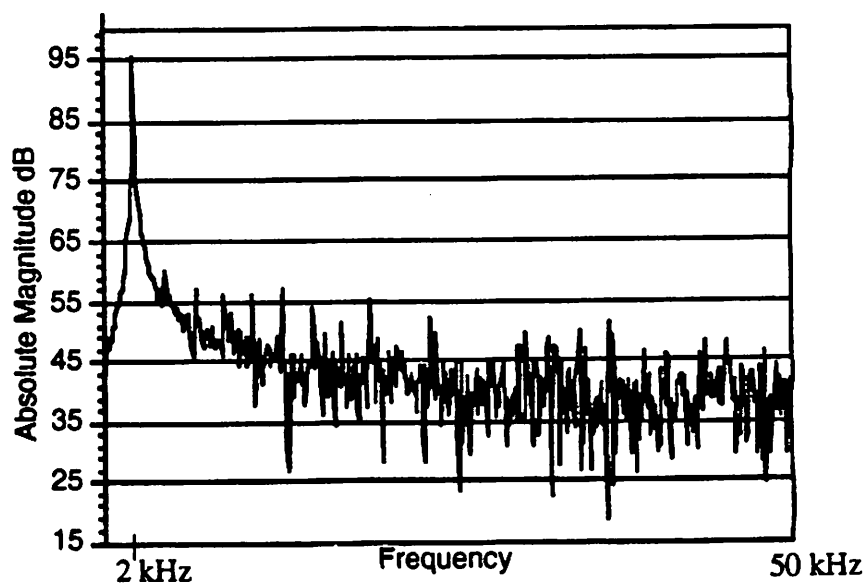
Figure 6.6: Test set-up block diagram.

frequency of 90 kHz. The reconstructed digitized input waveform and the DFT graph are shown in Figure 6.8(a) and (b) respectively. The noise floor is quite low but the harmonic distortions are still quite high. We tried to weaken the input signal amplitude to obtain better dynamic range but no significant improvement was observed.

We believe that the ADC prototype can have as high as 7.5 bit resolution. In the area of layout, we need to improve the layout of the ADC so that sensitive analog signal can be isolated much better than the digital signal. In the area of circuit generation, the op-amp gain requirements should probably be overestimated in order to guarantee the linearity and accuracy of the op-amp. Two crucial requirements would be to slightly over-estimate open-loop gain and to shorten the settling time of the op-amp.

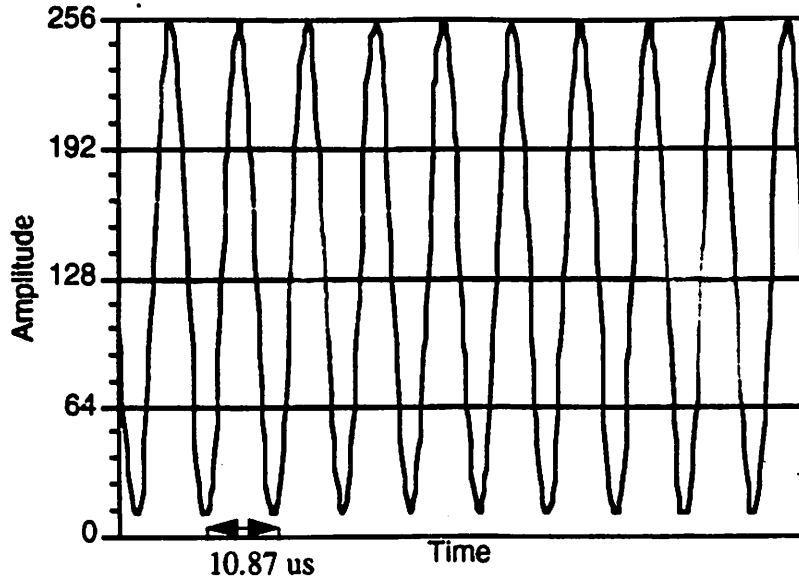


(a) Digitized input sine-wave amplitude vs. time.

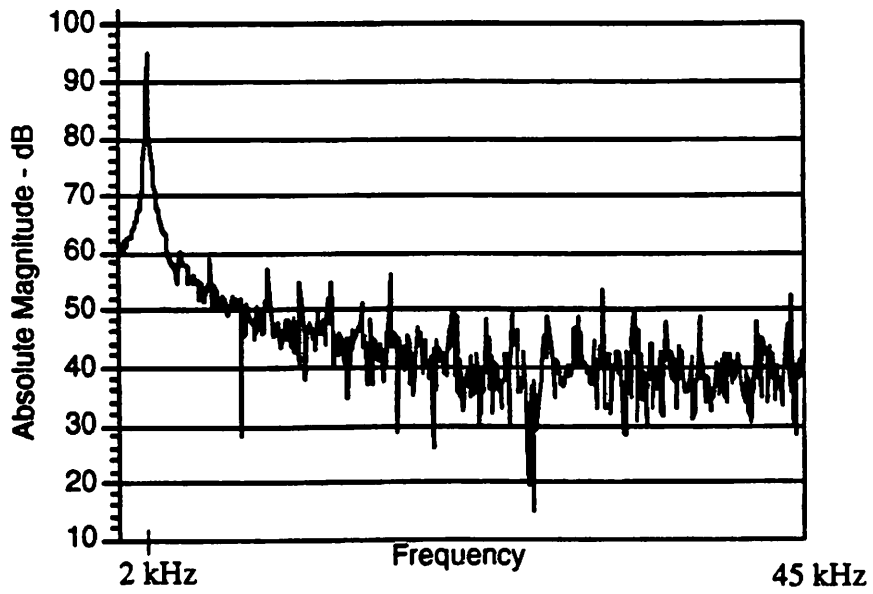


(b) 1024-point DFT output versus frequency

Figure 6.7: Measurement results with $f_s = 100 \text{ kHz}$ and $f_{in} = 2 \text{ kHz}$.



(a) Digitized input sine-wave amplitude vs. time.



(b) 1024-point DFT output versus frequency

Figure 6.8: Beat frequency measurements with $f_s = 90$ kHz and $f_{in} = 92$ kHz.

6.5 Summary

Examples of ADC blocks of different specifications have been presented. Layouts of manually designed as well as automatically generated ADC blocks were also shown. An automatically generated ADC prototype has been fabricated in a 2 μm CMOS process through MOSIS. The prototype has been tested and it functions as it is supposed to. The result of measurements are not as good as expected due to noise radiation in the wire-wrapped test board.

Examples and Test Results

CHAPTER 7 Conclusion

7.1 Summary of Research Results

This research has shown that the module generation of analog blocks in general and ADC in particular can be simplified by careful selection of architectural approach or implementation. An improved algorithmic ADC architecture that lends itself nicely into automatic generation has specifically been designed as the first building block of the ADC module generator, CADICS. The research results will be summarized into two different parts: *the improved algorithmic ADC architecture* and *ADC module generation*.

7.1.1 An Improved Algorithmic A/D Conversion Architecture

- The conventional algorithmic implementation does not fully utilize the conversion time. Idling periods exist among blocks during each conversion. Time-multiplexed two sample/hold blocks can remove the idling states and increase the overall throughput of the algorithmic ADC. The improvement in speed can be as high as a factor of 3 higher than the single sample/hold implementation.

Conclusion

- The performance of the algorithmic ADC is very sensitive to loop offset voltages. Offset voltage contribution from op-amps can be reduced by employing an auto zero amplifier that can be done at the beginning of every conversion. The offset voltage effect due to comparators is totally eliminated by using the proposed 2-comparator scheme and digital error correction. This comparison scheme allows us to use simple comparators for a wide range of sampling rates and resolution. The allowable comparator offset voltage is as large as $\pm \frac{V_{REF}}{4}$, where V_{REF} is the reference voltage of the ADC.
- Algorithmic ADC has a single parameter capability that can be easily adapted to satisfy requirements above 10 bits by adding a small module for self-calibration. It has been shown that the improved algorithmic ADC has very small numbers of variables that contributes to the overall linearity [LIU91]. Thus this type of ADC architecture is very suitable for automatic generation since the implementation of the netlist module generator will be made *easier* and *simpler*.
- The algorithmic ADC implementation is very area efficient compared to other ADC approaches. The trade-off between resolution and area can be made in straight forward way by scaling up/down the capacitor sizes to achieve the required $\frac{kT}{C}$ noise level and by adding the auto-calibration module if required.

7.1.2 ADC Module Generation

- Analog block generation is the bottleneck of the turn around time of mixed signal chips. While analog standard cell libraries will give short term solution, they do not offer flexibility to the change in technology feature size. It will be very desirable to have a general design methodology that can generate all possible analog

blocks automatically. Unfortunately, this is very difficult to do due to analog circuit design complexity. Thus module generation of commonly used analog functions such as ADC's is a good starting point to the realization of a general automatic mixed-signal design framework.

- A parameterized schematic of known architecture implementation is a better approach because it can generate highly-effective circuits based on known architecture for complex analog functions in a flexible environment by optimizing the device sizes with respect to various performance measures. CADICS is a performance-driven CMOS ADC module generator based on design rules and spice parameters.
- For ADC functions, the realm of performance maps into several classes of realizations spanning the conversion rate spectrum and two distinct regions of accuracy or linearity. Therefore, architectural study of different ADC blocks is a crucial part of the module generation implementation because careful selection of ADC blocks will limit the number of module generators needed in order to cover the widest spectrum of resolution and sampling rates.
- At least 5 different ADC implementations (serial, oversampled, algorithmic, pipelined, and flash) are needed to cover a wide spectrum of resolutions and sampling rates. As the first step, we developed an algorithmic ADC module generator.
- ADC module generation does not have a well-defined algorithm to do the synthesis making it very difficult to implement. In order to ease the development of the software, many analog circuit design techniques are incorporated into the module

Conclusion

generation. A new improved algorithmic ADC without high precision comparators was developed as the first ADC technique in CADICS.

- The module generator takes a set of ADC specifications, spice parameters, device matching information, design rules, and floorplan information. It generates the complete ADC netlist, layout, and the overall performance summary. It consists of three major steps: architectural selection, netlist generation, and layout generation.
- The netlist generation is implemented using a hierarchical optimization approach. The hierarchical optimization procedure is very suitable for block functions that can be partitioned into distinct subblocks. This procedure allows the flexibility to include other low-level block synthesis tools such as OPASYN to generate op-amps and comparators. Moreover, this type of implementation is much faster than doing a global optimization of the circuits all at once.
- Behavioral simulator is an important part of the module generation of complex blocks since some specifications can only be obtained through simulation runs. A discrete event simulation program of the new algorithmic ADC has been written and used within the netlist generator.
- Coupling between netlist and layout generation is an important aspect of analog block module generation since analog circuit performances depend on a fine detail of layout. Attempt to link the two generators is done by sharing an area optimizer.
- A top-down, bottom-up layout generation was implemented. The flooplanning is based on a slicing structure. A fixed floorplan is used throughout the layout gen-

eration but flexibility of changing the lower level block floorplan is allowed but not recommended.

- A structured-netlist using sub-circuit definition has been adapted. This style of netlisting accurately reflects the block partitioning of the ADC block. A data structure that stores the netlist efficiently has also been chosen.
- A fully differential layout style is used throughout except in the **digital blocks**.
- A modified Stockmeyer's algorithm is used to do the floorplan optimization. Channel area is also estimated in order to predict the actual silicon area of the final layout.
- The methodology introduced here can be extended to do a performance-driven hierarchical top-down design of mixed signal systems.

7.2 Future Work

The ADC module generator, CADICS needs to be improved by adding more ADC netlist module generators. More ADC topologies need to be added into the CADICS database in order to cover as wide range of resolution and sampling rates. More research to come up with high performance ADC implementation that is insensitive to small number of parameters is very important. This is very crucial because it will simplify the module generation implementation. More ADC's need to be fabricated and characterized in order to improve the quality of the existing module generator. Implementation of self-calibration module to extend the existing ADC topology to go beyond 10 bits is also very desirable.

The continuation of research in the area of generic analog tools such as router, placement, compaction, verification, and testing needs to be intensified in order to achieve

Conclusion

the maturity as those of digital circuits. In the future, a mixed-signal design framework will consist of many different module generators. This design framework effectively will function as a design manager that coordinates the generation of the requested mixed-signal blocks.

BIBLIOGRAPHY

-
- [ALLE86] P. E. Allen and P. R. Barton, "A silicon compiler for successive approximation A/D and D/A converters," in *Proc. Custom Integrated Circuits Conf.*, June 1986, pp. 552-555.
- [ALLS82] D. J. Allstot, "A Precision variable-Supply CMOS Comparator," *IEEE Journal of Solid-State Circuits*, SC-17(6): 1080-1087, December 1982.
- [BERK88] E. Berkan, M.d'Abreu, and W. Laughton, "Analog Compilation Based on Successive Decompositions," *Proceeding of 25th Design Automation Conference*, pp. 369-375, ACM/IEEE, 1988.
- [BOSE88] B. Boser and B. Wooley, "Quantization error spectrum of sigma-delta modulators," *1988 IEEE Int. Symp. Circuits Syst.*, pp. 2331-2334, June 1988.
- [BROD90] R.W. Brodersen et al, *LAGER Tool Set Users' Manual*. Berkeley, CA:1990.
- [BURN86] J.R. Burns and A.R. Newton, "SPARCS: a new constraint-based IC symbolic layout spacer," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1986, pp. 534-539.
- [CAND81] J. C. Candy and O. Benjamin, "The structure of quantization noise from sigma-delta modulation," *IEEE Trans. Communications*, vol. COM-29, pp.1316-1323, September 1981.
-

BIBLIOGRAPHY

- [CAND85] J. C. Candy, "A use of double integration in sigma delta modulation," *IEEE Trans. Communications*, Vol. COM-33, pp. 249-258, March 1985.
- [CASO89] A. Casotto, *MOSAICO Users' Manual*. Berkeley, CA: 1989.
- [CAST84] R. Costello, "Low-Voltage Low-Power MOS Switched-Capacitor Signal Processing Techniques," University of California at Berkeley, August 1984. *Ph. D. Thesis*.
- [CHAN92] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff, and P.R. Gray, "A top-down, constraint-driven design methodology for analog integrated circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1992, pp. 8.4.1-4.
- [CHAR92] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto, and A. Sangiovanni-Vincentelli, "A Constraint-driven placement methodology for analog integrated circuits," in *Proc. Custom Integrated Circuits Conf.*, May 1992, pp. 28.2.1-4.
- [CHEC78] W. Check, E. Cheng, G. Hill, M. Holler, and J. Miller, "Microcontroller includes A-D converter for lowest-cost analog interfacing," *Electronics*, May 1978.
- [CHEN86] H. Chen, *Glitter Users' Manual*. Berkeley, CA: 1986/
- [CHOU90] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-based channel routing for analog and mixed analog/digital circuits," in *Proc. International Conference on Computer-Aided Design*, Nov. 1990, pp. 198-201.
- [CHOU93] U. Choudhury and A. Sangiovanni-Vincentelli, "Automatic generation of parasitic constraints for performance-constrained physical design of analog circuits," *IEEE Trans. Computer-Aided Design*, vol. 12-2, pp. 208-224, February 1993.
- [COHN91] J. Cohn et. al, "KOAN/ANAGRAM II: New Tools for Device Level Analog Placement and Routing," *IEEE Journal of Solid-State Circuits*, SC-26(3):330-342, 1991.
- [DEGR85] M. Degrauwe, E. Vittoz, and I. Verbauwhede, "A Micropower CMOS Instrumentation Amplifier," *IEEE Journal of Solid-State Circuits*, SC-20(3): 805-807, June 1985.

BIBLIOGRAPHY

- [DEGR87] Marc G.R. Degrauwe *et al*, "IDAC: An Interactive Design Tool for Analog CMOS Circuits," *IEEE Journal of Solid-State Circuits*, SC-22(6):1106-1116, Dec. 1987.
- [DOER88] J. Doernberg, "High-Speed Analog-To-Digital Conversion Using 2-Step Flash Architecture," University of California at Berkeley, November 1988. *Ph. D. Thesis*.
- [EATO87] G.V. Eaton, D.G. Nairn, W.M. Snelgrove, and A.S. Sedra, "SICOMP: a silicon compiler for switched-capacitor filters." in *Proc. IEEE International Symposium on Circuits and Systems*, May 1987, pp. 321-324.
- [FELT92] E. Felt, E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli, "An efficient methodology for analog integrated circuits," submitted to the *European Design Automation Conference*, 1992.
- [GIEL92] G. Gielen and W. Sansen, "Open analog synthesis system based on declarative models," in *Proc. Advances in Analog Circuit Design*, April 1992, pp. 397-418.
- [GRAH53] R. S. Graham, "Relay computer for network analysis," *Bell Labs. Rec.*, vol. 31, pp. 152-157, Apr. 1953.
- [GAJS86] D. D. Gajski *et al*, "Silicon compilation (tutorial)," in *Proc. Custom Integrated Circuits Conf.*, June 1986, pp. 102-110.
- [GRAY84] P. R. Gray and R.G. Meyer, *Analysis and Design of Analog Integrated Circuits*. New York, NY: John Wiley & Sons, Inc., 1984
- [GRAY87] P. R. Gray, "Analog IC's in the submicron era: trends and perspectives," in *Proc. IEDM Dig. Tech. papers*, Dec. 1987, pp. 5-9.
- [HARJ89] R. Harjani, R.A. Rutenbar, and L.R. Carley, "OASYS: A Framework for Analog Circuit Synthesis," *IEEE Transaction on Computer-Aided Design*, 8(12): 1247-1266, Dec. 1989.
- [HARR86] D. Harrison, P. Moore, R.L. Spickelmier, and A.R. Newton, "Data management and graphics editing in the berkeley design environment," in *Proc. International Conference on Computer-Aided Design*, pp. 24-27, Nov 1986.
- [HAUS85] M. Hauser, P. Hurst, and R. Brodersen, "MOS ADC-filter combination that does not require precision analog components," *ISSCC Dig. Tech. Papers*, pp. 80-82, February 1985.

BIBLIOGRAPHY

- [HELM87] W. J. Helms and B. R. Byrnett, "Compiler generation of A to D converters," in *Proc. Custom Integrated Circuits Conf.*, June 1987, pp. 161-164.
- [HORN72] O.A. Horna, "A 150 Mbps A/D and D/A conversion system," *Comstat Technical Review*, 1972.
- [HOST84] B. J. Hosticka, W. Brockherde, U. Kleine and G. Zimmer, "Switched-capacitor FSK modulator and demodulator in CMOS technology," *IEEE Journal of Solid-State Circuits*, SC-19(3): 389-396, June 1984.
- [INOS62] H. Inose *et al*, "A telemetering system by code modulation- $\Sigma\Delta$ modulation," *IEEE Trans. Space Electronics and Telemetry*, vol. SET-8, pp. 204-209, Sep. 1962.
- [JUSU90] G. Jusuf, P.R. Gray, and A. Sangiovanni-Vincentelli, "CADICS - Cyclic analog-to-digital converter synthesis," in *Proc. of IEEE International Conference on Computer-Aided Design*, Nov. 1990, pp. 286-289.
- [JUSU90b] G. Jusuf and P.R. Gray, "A 1-bit/cycle algorithmic analog-to-digital converter without high-precision comparators," *Electronics Research Memorandum*, No. UCB/ERL M90/69, University of California at Berkeley, August 1990.
- [JUSU93] G. Jusuf, *CADICS Users' Manual*. Berkeley, CA: 1993.
- [KANE83] R. T. Kaneshiro, "Circuit and Technology Considerations For High-Frequency Switched-Capacitor Filters," University of California at Berkeley, July 1983. *Ph. D. Thesis*.
- [KOH89] H.Y. Koh, "Design Synthesis of Monolithic Operational Amplifier," University of California at Berkeley, May 1989. *Ph.D. Thesis*.
- [KOH90] H.Y. Koh, C.H. Sequin, and P.R. Gray, "OPASYN: A Compiler For CMOS Operational Amplifiers," *IEEE Transactions on Computer-Aided Design*, 9(2): 113-125, Feb. 1990.
- [KURD86] F.J. Kurdahi and A.C. Parker, "PLEST: a program for area estimation for VLSI integrated circuits," in *Proc. of Design Automation Conference*, May 1986, pp. 467-473.
- [LAKS86] K. R. Lakshmikumar, R. A. Hadaway and M. A. Copeland, "Characterization and modeling of mismatch in MOS transistors for precision analog design," SC-21(6): 1057-1066, December 1986.

BIBLIOGRAPHY

- [LEE84] H.S. Lee, D.A.Hodges, P.R.Gray, "Self-calibrating 15-bit A/D converter," *IEEE Journal of Solid-State Circuits*, SC-19(6): 813-819, December 1984.
- [LEWI87] S. H. Lewis and P. R. Gray, "A pipelined 5MHz 9b ADC," in *ISSCC Dig. Tech. Papers*, Feb. 1987, pp. 210-211.
- [LEWI87b] S. H. Lewis, "Video-Rate Analog-To-Digital Conversion Using Pipelined Architectures," University of California at Berkeley, November 1987. *Ph. D. Thesis*.
- [LI84] P. W. Li, "Ratio-independent Algorithmic Analog To Digital Converter Techniques," University of California at Berkeley, August 1984. *Ph. D. Thesis*.
- [LIN90] Y. M. Lin, "Performance Limitations on High-Resolution Video-Rate Analog-Digital Interface," University of California at Berkeley, June 1988. *Ph. D. Thesis*.
- [LIU91] E. Liu, A. Sangiovanni-Vincentelli, G. Gielen, and P.R. Gray, "A behavioral representation for nyquist rate A/D converters," in *Proc. IEEE International Conference on Computer-Aided Design*, Nov 1991.
- [LUEN84] D.G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley Publishing Company, 1984.
- [LUCA87] D. Lucas, "Analog silicon compiler for switched-capacitor filters," in *Proc. of IEEE International Conference on Computer-Aided Design*, Nov. 1987, pp. 506-513.
- [MALA90] E. Malavasi, U. Choudhury, and A. Sangiovanni-Vincentelli, "A routing methodology for analog and mixed-analog integrated circuits," in *Proc. of IEEE International Conference on Computer-Aided Design*, Nov. 1990, pp. 202-205.
- [MALA91] E. Malavasi, E. Charbon, G. Jusuf, R. Totaro, and A. Sangiovanni-Vincentelli, "Virtual symmetry axis for the layout of analog IC's," in *Proc. of IEEE 1991 International Conference on VLSI and CAD*, Seoul, South Korea, October 1991, pp. 195-198.
- [MATS87] Y. Matsuya *et al*, "A 16-bit oversampling a-to-d conversion technology using triple-integration noise shaping," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 921-929, Dec. 1987.

BIBLIOGRAPHY

- [MCCR75] J. L. McCreary and P. R. Gray, "All MOS charge redistribution analog-to-digital conversion techniques-Part 1," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 371-379, Dec. 1975.
- [MCCR81] J. L. McCreary, "Matching properties, and voltage and temperature dependence of MOS capacitors," *IEEE Journal of Solid-State Circuits*, SC-16(6): 608-616, December 1981.
- [NYE88] William Nye et. al, "DELIGHT.SPICE: An Optimization-Based System for The Design of Integrated Circuits," *IEEE Transaction on Computer-Aided Design*, vol. 7-4, pp. 501-519, Apr. 1988.
- [OHAR87] H. Ohara *et al*, "A CMOS programmable self-calibrating 13-bit 8-channel data acquisition peripheral," *IEEE J. Solid-State Circuits*, vol. SC- 22, pp. 362-369, Dec. 1987.
- [ONOD88] H. Onodera, T. Tateishi, K. Tamaru, "A cyclic A/D converter that does not require ratio-matched components," *IEEE Journal of Solid-State Circuits*, SC-23(1), February 1988.
- [OPTE91] F. Op't Eynde, G.M. Yin, and W. Sansen, "A CMOS fourth-order 14b 500k-sample/s sigma-delta ADC converter," in *ISSCC Dig. Tech. Papers*, Feb. 1991, pp. 62-63.
- [OTTE82] R. Otten, "Automatic floorplan design," in *Proc. of Design Automatian Conference*, June 1982, pp. 261-267.
- [PELG89] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, SC-24(6): 1033-1040, December 1989.
- [RABI78] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall Inc., 1978.
- [SANCS85] E. Sanchez-Sinocio and J. Ramirez-Angulo, "AROMA: an area optimized CAD program for cascade SC filter design," *IEEE Transaction on Computer-Aided Design*, vol. 4-3, pp. 296-303, July 1985.
- [SAUL82] P. H. Saul, "A high-speed comparator design technique," *IEEE Journal of Solid-State Circuits*, SC-17(6): 529-532, June 1982.
- [SEND82] D. Senderowicz, S. F. Dryer, J. H. Huggins, C. F. Rahim and C. A. Laber, "A family of differential NMOS analog circuits for a PCM

BIBLIOGRAPHY

- codec filter chip," *IEEE Journal of Solid-State Circuits*, SC-17(6): 1014-1023, December 1982.
- [SHIEH81] K. C. Shieh, "Noise Limitations In Switched-Capacitor Filters." University of California at Berkeley, December 1981. *Ph. D. Thesis*.
- [SHIH85] C. C. Shih, "Precision Analog To Digital Conversion Using Reference Recirculating Algorithmic Architectures," University of California at Berkeley, July 1985. *Ph. D. Thesis*.
- [SHYU 81] J. B. Shyu, G. C. Temes and K. Yao, "Random errors in MOS capacitors," *IEEE Journal of Solid-State Circuits*, SC-17(6): 1070-1076, December 1981.
- [SHYU84] J. B. Shyu, G. C. Temes and F. Krummenacher, "Random error effects in matched MOS capacitors and current sources," *IEEE Journal of Solid-State Circuits*, SC-19(6): 948-955, December 1984.
- [SOO85] D. C. Soo, "High-Frequency Voltage Amplification and Comparison in a One-Micron NMOS Technology," University of California at Berkeley, December 1985. *Ph. D. Thesis*.
- [STOC83] L. Stockmeyer, "Optimal orientations of cells in slicing floor-plan designs," *Information and Control*, vol 57-2/3, May 1983, pp. 91-101.
- [SZEN77] G. Szentirmai, "FILSYN - a general purpose filter synthesis program," in *Proceedings of IEEE*, 65:1443-1458, October 1977.
- [TEWK78] S. K. Tewksbury and R. W. Hallock, "Oversampled, linear predictive and noise shaping coders of order > 1 ," *IEEE Trans. Circuits and Systems*, vol. CAS-25, pp. 436-447, 1978.
- [TOWN80] M. Townsend *et al*, "An NMOS microprocessor for analog signal processing," *IEEE J. Solid State-State Circuits*, vol. SC-15, pp. 33-38, Feb. 1980.
- [TRON87] L. Trontelj *et al*, "Analog silicon compiler for switched capacitor circuits," in *Proc. of IEEE International Conference on Computer-Aided Design*, Nov. 1986, pp. 506-509.
- [WU88] J. T. Wu, "High-Speed Analog-To-Digital Conversion in CMOS VLSI," Stanford University, March 1988. *Ph. D. Thesis*.

BIBLIOGRAPHY

- [YAGU86] H. Yaguthiel *et al*, "Methodology for automated layout of switched capacitor filters," in *Proc. of IEEE International Conference on Computer-Aided Design*, Nov. 1986, pp. 444-447.
- [YOSH82] T. Yoshimura and E. S. Kuh, "Efficient Algorithms for Channel Routing," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-1, pp.25-35, Jan. 1982.
- [YUKA85] A. Yukawa, "A CMOS 8-bit high-speed A/D converter IC," *IEEE Journal of Solid-State Circuits*, SC-20(3): 775-779, June 1985.
- [ZIMM88] G. Zimmerman, "A new area and shape function estimation technique for VLSI layouts," in *Proc. of Design Automation Conference*, May 1988, pp. 60-65.

APPENDIX A: Building Blocks

A.1 Introduction

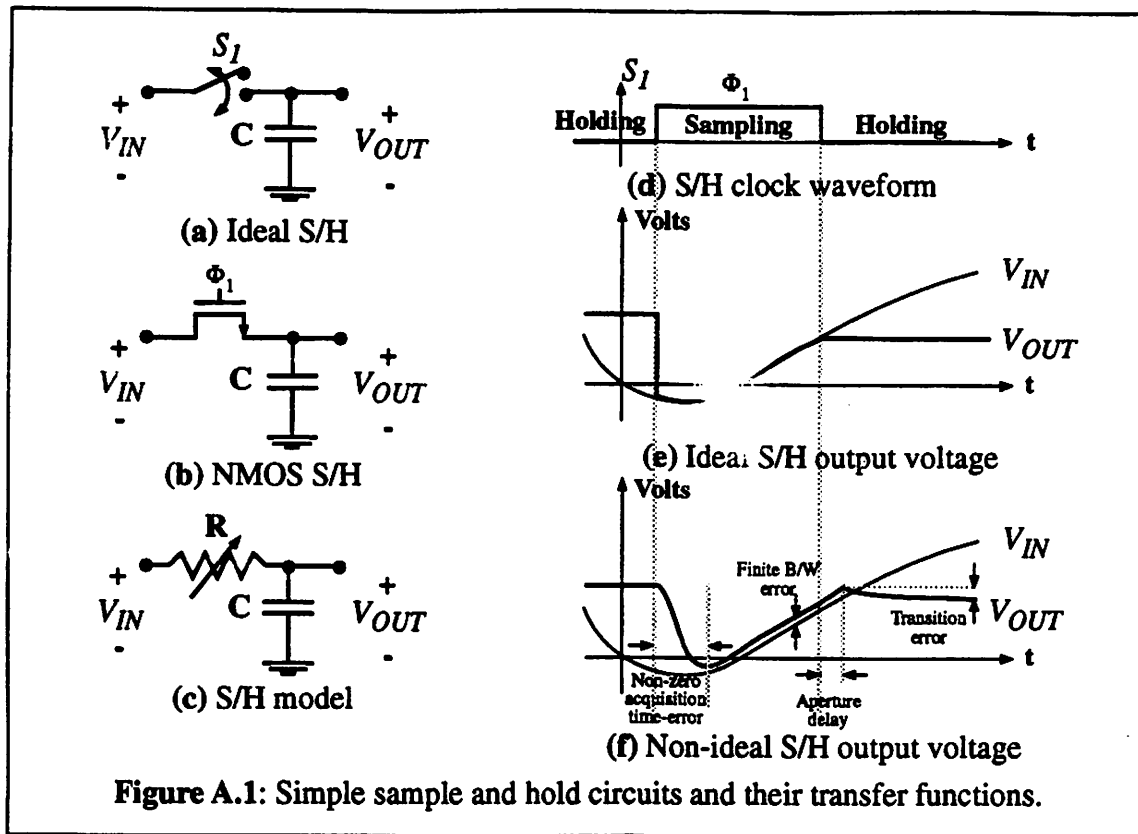
In this appendix, a detailed discussion of all the basic building blocks used in implementing the 1-bit/cycle algorithmic ADC will be presented. Discussion on Sample and Hold (S/H), Gain of 2 (2X), operational amplifier (op-amp), comparator (CMP), and digital circuitry will be discussed in succession. We will only derive in details equations that are not commonly used.

A.2 Sample and Hold Block

In this section the design of a capacitor mismatch insensitive sample and hold (S/H) circuit will be described. The design of the op-amp will be described in section A.4.

A.2.1 S/H Circuit

One of the simplest form of sample and hold circuit can implemented using a switch and a capacitor as shown in Figure A.1(a). During the sampling period, switch S_1



connects the input to the output nodes charging/discharging the capacitor C to the current V_{IN} . After a finite time T , V_{OUT} will track V_{IN} . When the switch is turned off, the input voltage has been sampled onto C and is now being held as shown in Figure A.1(e). One simple implementation of the switch is using NMOS transistor (Figure A.1(b)). Unfortunately, when Φ_1 is high the switch has some finite resistance (Figure A.1(c)) that will contribute to some errors such as non-zero acquisition time, aperture delay, transition error, and tracking error as illustrated in Figure A.1(f). A detailed discussion of these errors can be found in [LEWI87b]. Besides the errors mentioned above there are two other errors that affect the sampling mode, they are charge-injection voltage and thermal noise. These problems can be reduced by doing bottom-plate sampling, fully differential architecture, and CMOS switches.

One S/H implementation that was often used is shown in Figure A.2. When Φ_1 and Φ_{1a} are high and Φ_2 is low, the input voltage V_{IN} is sampled onto capacitor C_S while capacitor C_I is being discharged to ground. When Φ_1 switches from high to low, the instantaneous input voltage at this time is being stored in C_S and a delay later Φ_{1a} switches to low. The reason for doing this delay switching on the sampling mode will be discussed in details later in this section. When Φ_2 goes from low to high, the charges stored in C_S are transferred to C_I . This period is called the hold mode.

The overall closed-loop transfer function during the hold mode can be written as:

$$\frac{V_{OUT}}{V_{IN}} = - \left[\frac{C_S}{C_I + \frac{C_S + C_I + C_P}{A_v}} \right] \quad (\text{EQ A.1})$$

Where C_S = sampling capacitor

C_I = integrating capacitor

C_P = total parasitic capacitance on the summing node

A_v = open loop gain of the op-amp

Assuming the switches are ideal and the op-amp gain A_v is infinite, the transfer function will depend on the ratio of the two capacitors: $\frac{C_S}{C_I}$. For a unity gain sample and hold function, $C_S = C_I$. However, due to the imperfection in photolithography and process variation, these two capacitors will have some variations.

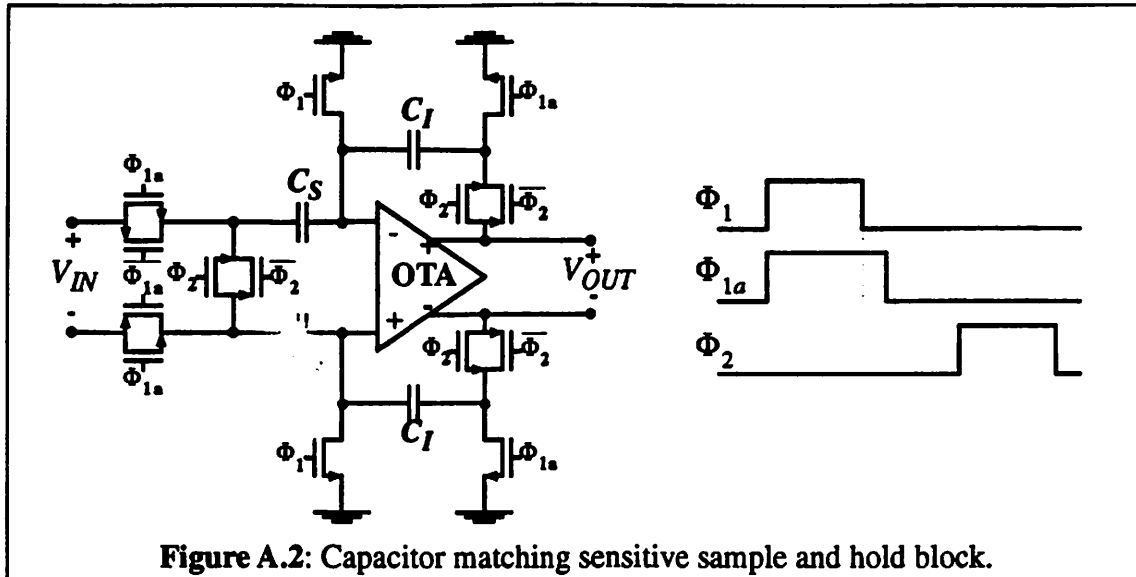


Figure A.2: Capacitor matching sensitive sample and hold block.

A better S/H circuit is the one that is independent of capacitor ratio and therefore, is insensitive to capacitor matching as shown in Figure A.3. The switching sequence, transfer function, error sources of the circuit, and calculation of switch and capacitor sizes will be discussed next.

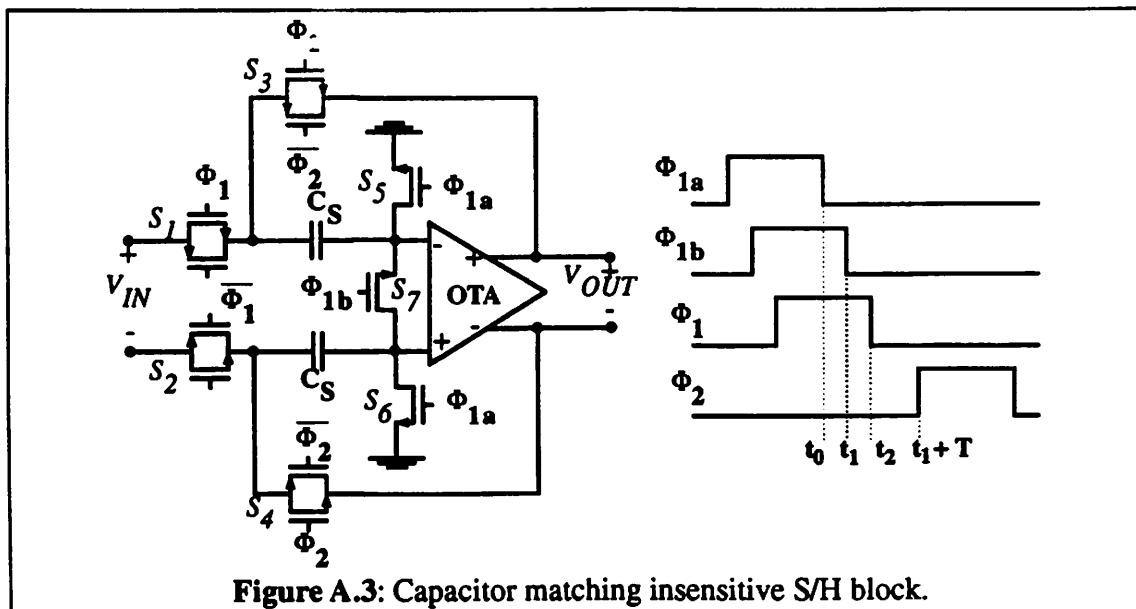


Figure A.3: Capacitor matching insensitive S/H block.

A.2.2 Switching Sequence

The S/H circuit shown in Figure A.3 has 4 different clocks: Φ_1 , Φ_{1a} , Φ_{1b} , and Φ_2 . Effectively there are only two clocks since Φ_1 is a delayed version of Φ_{1b} which in turn is a delayed version of Φ_{1a} . During the sampling mode, V_{IN} is sampled onto C_S by closing S_1 , S_2 , S_5 , S_6 , and S_7 switches (Figure A.4(a)). At the end of sampling period, Φ_{1a} will go low first and therefore, S_5 and S_6 get turned off first isolating the summing nodes of the op-amp (Figure A.4(b)). The charge injected by S_5 and S_6 will then reside in the summing nodes. Ideally S_5 and S_6 have exactly the same width and length, but due to limitations in photolithography and the etching process there will be a slight mismatch. As a result, the charge injected by S_5 and S_6 are not exactly equal. The difference will become a differential error voltage at the summing nodes.

To minimize the difference, S_7 is added, connecting the two summing nodes so that turning off S_7 after S_5 and S_6 equalizes the differential error voltage between the two nodes (Figure A.4(c)). Moreover, having S_7 will make S_5 and S_6 smaller in sizes because S_7 will now function as the sampling switch instead of S_5 and S_6 . Therefore, S_5 and S_6 can be made small to minimize the amount of charge injection. The total charge injection from S_5 , S_6 , and S_7 will contribute to a common offset voltage to the op-amp. As a result, proper switch size needs to be chosen to minimize this effect. To disconnect the sampling capacitor from the input voltage, Φ_1 goes from high to low turning off S_1 and S_2 . Turning off S_5 , S_6 , and S_7 prior to S_1 and S_2 results in no sampled charge injection due to turning off S_1 and S_2 .

When Φ_2 goes from low to high, S_3 and S_4 will be on, closing the loop by connecting nodes A and B to the outputs to do the hold mode (Figure A.4(d)). The size of S_3 and S_4 are determined by the allowable additional phase shift at the output nodes and the settling time of the op-amp.

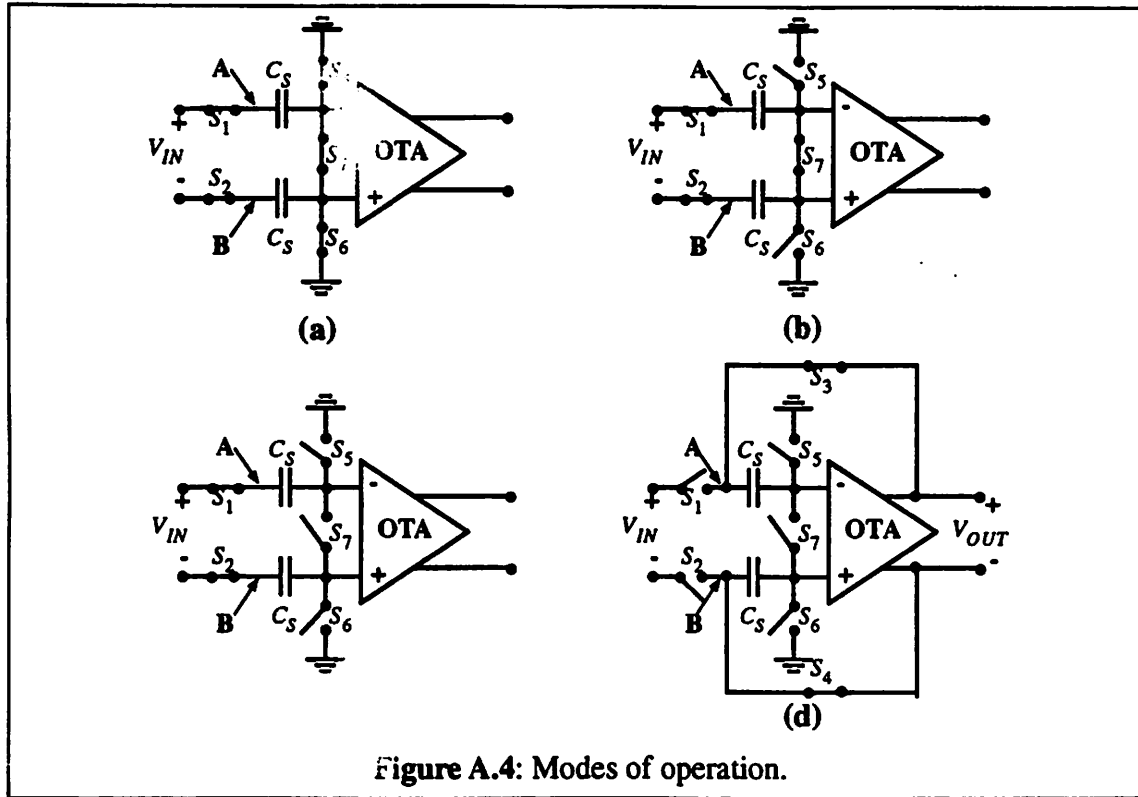


Figure A.4: Modes of operation.

A.2.3 Transfer Function

The S/H transfer function can be analyzed using charge conservation. For simplicity, a single ended version is used. When Φ_b goes from high to low at time t_1 (Figure A.3), the input voltage is being sampled onto the capacitor C_S . The total charge at node X during the sampling mode (Figure A.5(a)) can be written as:

$$Q_X(t_1) = C_S \cdot [0 - V_{IN}(t_1)] \quad (\text{EQ A.2})$$

where $V_{IN}(t_1)$ is the sampled voltage

When the S/H block goes into the hold mode at time $t_1 + T$ (Figure A.3) (the simplified hold mode configuration is shown in Figure A.5(b)), the total charge at node X is

$$Q_X(t_1 + T) = C_S \cdot [V_X(t_1 + T) - V_{OUT}(t_1 + T)] + C_P \cdot V_X(t_1 + T) \quad (\text{EQ A.3})$$

where $V_X(t)$ = voltage at node X at time t

$V_{OUT}(t)$ = output voltage at time t

C_P = total capacitance at summing node excluding C_S

Assume that the open loop gain of the operational amplifier is A_V ,

$$V_{OUT}(t) = -A_V \cdot V_X(t) \quad (\text{EQ A.4})$$

By conservation of charge, the total charge at node X during sample mode is equal to hold mode since node X has no dc path to the ground. Equating EQ A.2 and EQ A.3:

$$Q_X(t_1) = Q_X(t_1 + T) \quad (\text{EQ A.5})$$

$$-C_S \cdot V_{IN}(t_1) = C_S \cdot [V_X(t_1 + T) - V_{OUT}(t_1 + T)] + C_P \cdot V_X(t_1 + T) \quad (\text{EQ A.6})$$

Substituting V_X in term of V_{OUT} using EQ A.4 and solve for V_{OUT} , we get

$$V_{OUT}(t_1 + T) = \left[\frac{1}{1 + \frac{C_S + C_P}{A_V \cdot C_S}} \right] \cdot V_{IN}(t_1) \quad (\text{EQ A.7})$$

EQ A.7 shows that the output of this S/H block is independent of any capacitor ratio other than C_P . The above derivation can easily be extended to cover the fully differential case.

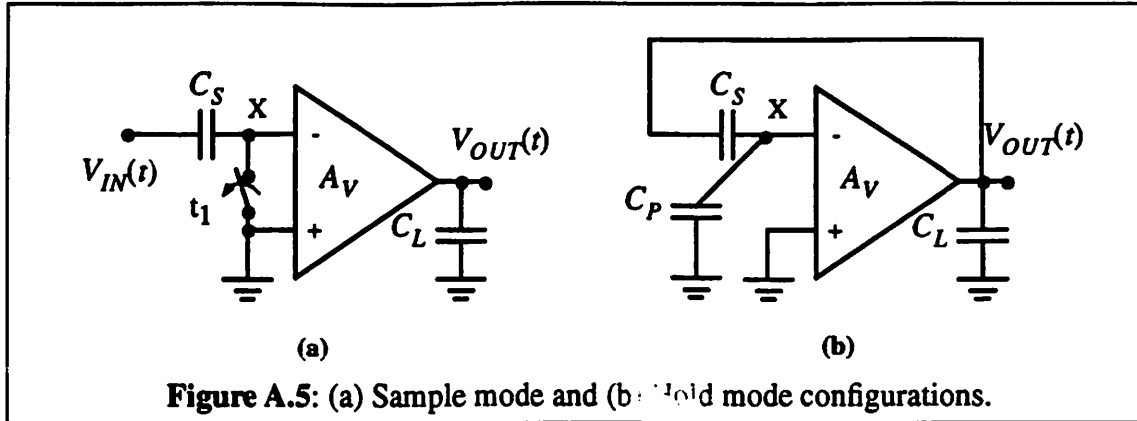


Figure A.5: (a) Sample mode and (b) hold mode configurations.

A.2.4 Noise Source

The noise sources of the S/H circuit (assuming an ideal op-amp) are the thermal noise generators due to non-zero channel resistance of the switches that get sampled onto the sampling capacitors. The only significant noise source of the S/H block shown in Figure A.3 comes from the middle switch S_7 . The reason is very obvious because S_7 is the switch that disconnects the sampling path, thus its thermal noise gets sampled onto the sampling capacitors [SHIEH81], [GRAY84].

Let us assume the S/H is sampling a zero input voltage. Right before S_7 is turned off, the noise model of this switch is an equivalent non-zero resistance R with thermal noise voltage generator $\bar{v}^2 = 4kTR\Delta f$ as shown in Figure A.5(a). When S_7 is turned off, the thermal noise is sampled onto both capacitors.

$$\bar{v}_A^2 = \frac{kT}{2C_T} \quad (\text{EQ A.8})$$

$$\bar{v}_B^2 = \frac{kT}{2C_T} \quad (\text{EQ A.9})$$

where $C_T = C_S + C_P$.

It can be shown that the input referred mean square noise contribution from one capacitor S/H block (Figure A.3) is smaller by a factor of 6 than the one from two-capacitor type (Figure A.2) assuming $C_S = C_I = C_P = C$.

$$\frac{V_{input}^2}{KT} = \frac{C}{C} \quad (\text{EQ A.15})$$

If $C_S = C$ and $C_P = C$, then $C_T = 2C$ and the input referred mean square noise is:

$$\frac{V_{input}^2}{2KT} = \frac{C_T}{2KT} \quad (\text{EQ A.14})$$

$$V_{input}^2 \approx \left[\frac{C_S}{C_T} \right]^2 \cdot V_{out}^2 \quad (\text{EQ A.13})$$

square of the gain.

The input referred mean square noise can be obtained by dividing EQ A.12 by the inverse

$$V_{out}^2 \approx \frac{C_T}{2KT} \cdot \left[\frac{C_S}{C_T} \right]^2 \quad (\text{EQ A.12})$$

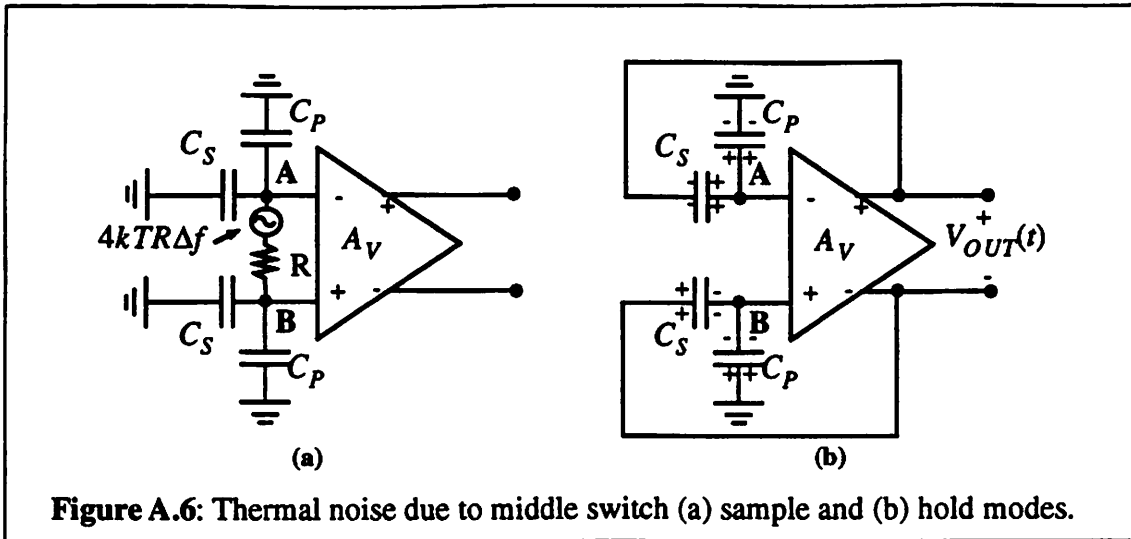
$$V_{out}^2 \approx \left[\frac{C_S}{C_T} \right]^2 \cdot \left[\frac{2C_T}{KT} + \frac{2C_T}{2KT} + \frac{2C_T}{2KT} \right] \quad (\text{EQ A.11})$$

Substituting V_A^2 and V_B^2 into EQ A.10:

$$V_{out}^2 \approx \left[\frac{C_S}{C_T} \right]^2 \cdot [V_A^2 + V_B^2 + 2(\sqrt{V_A^2} \cdot \sqrt{V_B^2})] \quad (\text{EQ A.10})$$

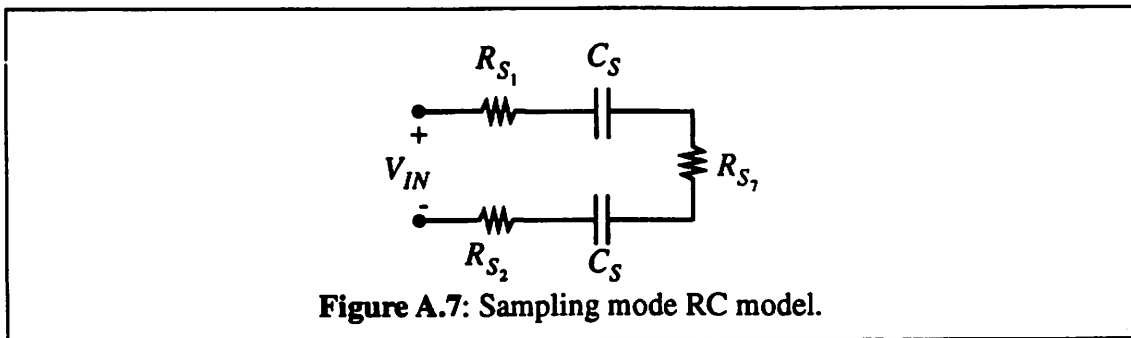
A.5(b)) at the output is:

Because V_A^2 and V_B^2 are correlated, the effective total noise (during the hold mode Figure



A.2.5 Switch Size Calculation

The following analysis will discuss how each switch size is determined. During the sampling mode, S_5 and S_6 (Figure A.3) will be turned off first. The switches will be in the configuration shown in Figure A.4(b). Due to the finite “on” resistance of the switches, we can simplify the network as RC network as shown in Figure A.5.



This RC network has to settle to within half LSB of input voltage when the middle switch S_7 is opened. This acquisition time, t_{aq} can be written as:

Sample and Hold Block

$$e^{-\frac{t_{aq}}{\tau}} \leq \frac{1}{2} \cdot \frac{2}{2^N} \quad (\text{EQ A.16})$$

$$t_{aq} \geq N \cdot \tau \cdot \ln(2) \quad (\text{EQ A.17})$$

$$\tau \leq \frac{t_{aq}}{N \cdot \ln(2)} \quad (\text{EQ A.18})$$

where τ = the time constant of the RC network

N = the number of bits

t_{aq} = the acquisition time

$$\tau = R_T \cdot C_T \quad (\text{EQ A.19})$$

$$R_T = R_{S_1} + R_{S_2} + R_{S_3} \quad (\text{EQ A.20})$$

$$C_T = 0.5 \cdot C_S \quad (\text{EQ A.21})$$

Substituting EQ A.19 and EQ A.21 into EQ A.18 and solve for R_T :

$$R_T \leq \frac{t_{aq}}{N \cdot C_T \cdot \ln(2)} \quad (\text{EQ A.22})$$

$$t_{aq} \leq \frac{1}{2} \cdot \frac{1}{f_s} \quad (\text{EQ A.23})$$

$$R_T \leq \frac{1}{N \cdot f_s \cdot C_S \cdot \ln(2)} \quad (\text{EQ A.24})$$

Another requirement that needs to be considered is the distortion due to this sampling network. This requirement is usually specified in term of maximum allowable phase shift, ϕ .

The phase shift requirement is very important in video applications. Typically it is in the order of 4° or less. Based on this requirement, another equation for R_T can be derived.

Detailed derivation of the phase shift equation can be found in [LEWI87b].

$$\phi \leq -\text{atan} \left(\frac{f_{in}}{f_{-3dB}} \right) \quad (\text{EQ A.25})$$

$$f_{-3dB} \geq \frac{f_{in}}{\tan \phi} \quad (\text{EQ A.26})$$

where ϕ = phase shift

f_{in} = input frequency

f_{-3dB} = -3 dB frequency of the RC network, $(2\pi \cdot R_T \cdot C_T)^{-1}$

According the Nyquist theory, the maximum allowable input frequency for the A/D converter that we are discussing should be less than or equal to conversion frequency, $f_{conversion}$.

$$f_{in} \leq \frac{1}{2} \cdot f_{conversion} \quad (\text{EQ A.27})$$

The conversion frequency can be related to the sampling frequency, f_s to be

$$f_{conversion} = \frac{f_s}{\text{int} \left(\frac{N+1}{2} \right)} \quad (\text{EQ A.28})$$

EQ A.28 is only valid for 1 bit/cyclic Algorithmic ADC discussed in this thesis. Substituting EQ A.27 and EQ A.28 into EQ A.26 and assuming $f_{in} = \lambda \cdot f_{conversion}$ (for $\lambda \leq 0.5$) and solve for R_T :

$$R_T \leq \frac{\text{int} \left(\frac{N+1}{2} \right) \cdot \tan \phi}{\pi \cdot \lambda \cdot f_s \cdot C_S} \quad (\text{EQ A.29})$$

Assuming $\phi = 4^0$, we can then compare EQ A.24 and EQ A.29 for all N . If

$\lambda = 0.5$, for $N \leq 8$ it will be phase shift limited and for $N > 8$ it will be time acquisition limited. Since the A/D converter is intended for mid-frequency application, the phase shift requirement is much more relax. Moreover, most of the time the input frequency is much slower than the Nyquist rate i.e. $\lambda < 0.5$. For $\phi = 4^\circ$ and $\lambda < 0.3$, for all N, the switch size will be time acquisition limited. For the remaining of the discussion of this section will be assuming that this is in fact true. Therefore, EQ A.24 will be used when calculating the switch sizes.

Let us assume that $R_{S_7} = \alpha \cdot R_{S_1} = \alpha \cdot R_{S_2} = R$ where $\alpha > 1$. Since S_7 is an NMOS switch, the on resistance of switch R_{S_7} can be approximated by

$$\frac{1}{R_{S_7}} = \mu_n \cdot C_{ox} \cdot \frac{W_7}{L_7} \cdot (V_H - V_{TN}) \quad (\text{EQ A.30})$$

where μ_n = mobility electron in the n -channel

C_{ox} = gate oxide capacitor per unit area

$\frac{W_7}{L_7}$ = aspect ratio of the gate

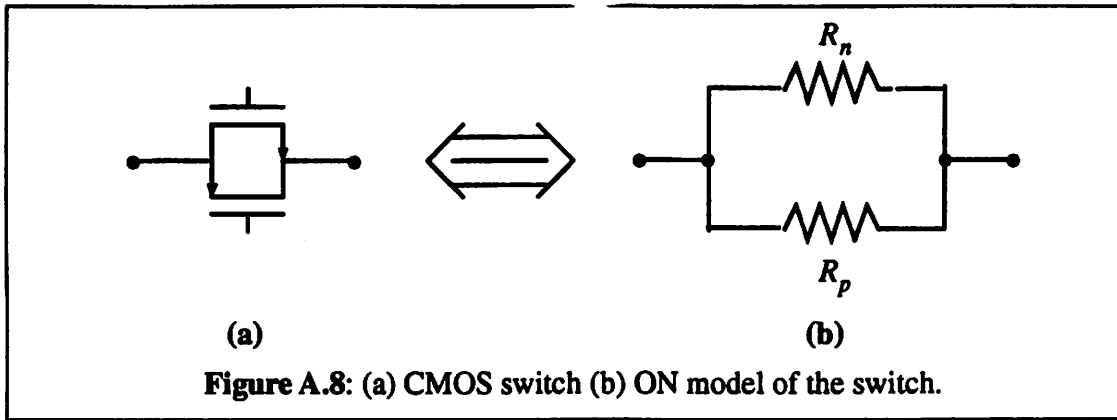
V_H = gate to source voltage applied to turn the switch on

V_{TN} = threshold voltage

Substituting EQ A.30 into EQ A.24 and solve for the aspect ratio, we obtain the following equation:

$$\frac{W_7}{L_7} \geq \frac{N \cdot \ln(2) \cdot f_s \cdot C_S \cdot \left(\frac{2}{\alpha} + 1\right)}{\mu_n C_{ox} \cdot (V_H - V_{TN})} \quad (\text{EQ A.31})$$

For S_1 and S_2 , we can solve for their equivalent resistor of CMOS switch with the same width and length. When a CMOS switch is on, we can model it as two resistors, R_n and R_p in parallel as shown in Figure A.8.



$$\frac{1}{R} = \frac{1}{R_n} + \frac{1}{R_p} \quad (\text{EQ A.32})$$

$$\frac{1}{R_n} = \mu_n \cdot C_{ox} \cdot \frac{W_n}{L_n} \cdot (V_{GS_n} - V_{TN}) \quad (\text{EQ A.33})$$

$$\frac{1}{R_p} = \mu_p \cdot C_{ox} \cdot \frac{W_p}{L_p} \cdot (|V_{GS_p}| - |V_{TP}|) \quad (\text{EQ A.34})$$

where μ_n = electron mobility in the n -channel

μ_p = hole mobility in the p -channel

C_{ox} = gate oxide capacitor per unit area

Sample and Hold Block

W_n = the gate width of the NMOS

L_n = the gate length of the NMOS

W_p = the gate width of the PMOS

L_p = the gate length of the PMOS

V_{GS} = the gate to source voltage to turn the switch on

V_{TN} = n -channel threshold voltage

V_{TP} = p -channel threshold voltage

Let us assume:

$$V_{TN} = |V_{TP}| = V_T \quad (\text{EQ A.35})$$

$$\mu_n = b \cdot \mu_p \quad (\text{EQ A.36})$$

$$V_{GS_n} = |V_{GS_p}| = V_H \quad (\text{EQ A.37})$$

$$\frac{W_n}{L_n} = \frac{W_p}{L_p} = \frac{W_1}{L_1} \quad (\text{EQ A.38})$$

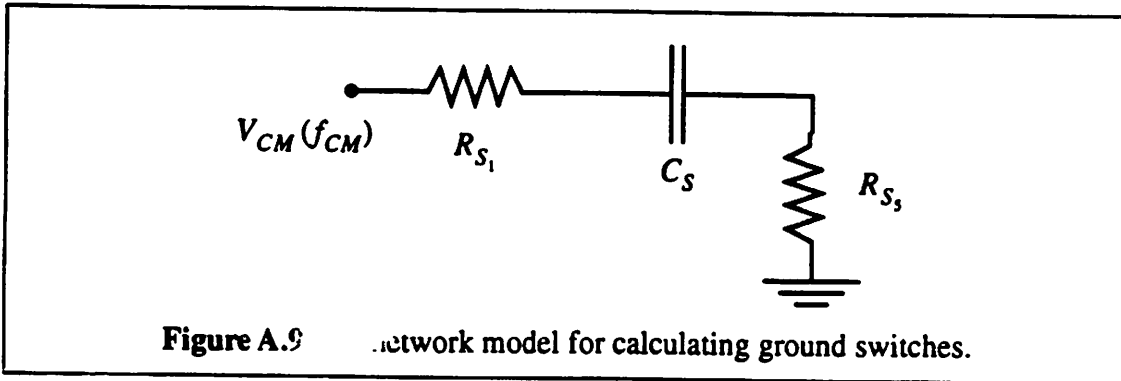
Substituting EQ A.33 - EQ A.38 into EQ A.32, we obtain:

$$\frac{1}{R_{S_1}} = (1 + b) \cdot \mu_n \cdot C_{ox} \cdot \frac{W_1}{L_1} \cdot (V_H - V_T) \quad (\text{EQ A.39})$$

Since $R_{S_2} = \alpha \cdot R_{S_1}$, we can solve for the aspect ratio of S_1 by substituting EQ A.31 into EQ A.39.

$$\frac{W_1}{L_1} \geq \frac{N \cdot \ln(2) \cdot f_s \cdot C_S \cdot (2 + \alpha)}{(1 + b) \cdot \mu_n \cdot C_{ox} \cdot (V_H - V_T)} \quad (\text{EQ A.40})$$

In order to calculate the ground switches S_5 and S_6 , we can simplify the analysis by ignoring the middle switch. When these switches are on, they again form an RC network as shown in Figure A.9. The requirement for this RC network is different from the previous one discussed. The acquisition time for this network is a function of the common mode frequency, f_{CM} of the input signal. Usually f_{CM} is at a much lower frequency than f_{in} . Since the amplitude of the common mode input voltage swing is usually much smaller than the differential input voltage, we can choose the RC network to have a bandwidth as large as f_{CM} .



$$(R_{S_1} + R_{S_2}) \cdot C_S \leq \frac{1}{2\pi \cdot k \cdot f_{in}} \quad (\text{EQ A.41})$$

where k = the multiplication factor, $k < 1$

$$R_{S_2} \leq \frac{1}{2\pi \cdot C_S \cdot k \cdot f_{in}} - R_{S_1} \quad (\text{EQ A.42})$$

Substituting EQ A.27, EQ A.28, EQ A.30, and EQ A.40 into EQ A.42 and solve for the

aspect ratio of NMOS switch S_5 , we obtain

$$\frac{W_5}{L_5} \geq \frac{f_s \cdot C_S}{\mu_n \cdot C_{ox} \cdot (V_H - V_T) \cdot \left[\frac{\text{int}\left(\frac{N+1}{2}\right)}{2\pi \cdot \lambda \cdot k} - \frac{1}{(2+\alpha) \cdot N \cdot \ln(2)} \right]} \quad (\text{EQ A.43})$$

By Assuming $\alpha = 2$, $\lambda = \frac{1}{2}$, $N \geq 6$, and worst case $k = 1$, we can ignore

$\frac{1}{(2+\alpha) \cdot N \cdot \ln(2)}$ contribution. EQ A.43 can be approximated to be

$$\frac{W_5}{L_5} \geq \frac{\pi \cdot f_s \cdot C_S}{\mu_n \cdot C_{ox} \cdot (V_H - V_T) \cdot \text{int}\left(\frac{N+1}{2}\right)} \quad (\text{EQ A.44})$$

Finally, during the hold mode, S_3 and S_4 (Figure A.4(d)) will be on. These switches can be modeled as a feedback resistor shown in Figure A.10. The sizes are determined by the settling time of the operational amplifier (op-amp). These switches will also contribute an additional phase shift to the overall loop of the block. Usually the contribution of these switches to the additional phase shift is negligible. Thus, the switches size will be a function of the op-amp settling time. Let τ be the settling time of the op-amp.

$$t_{aq} = N \cdot R_{on} \cdot C_L \cdot \ln(2) \quad (\text{EQ A.45})$$

where t_{aq} = acquisition time

N = the ADC resolution

R_{on} = the on resistance of the switch

C_L = total load capacitor at the output node

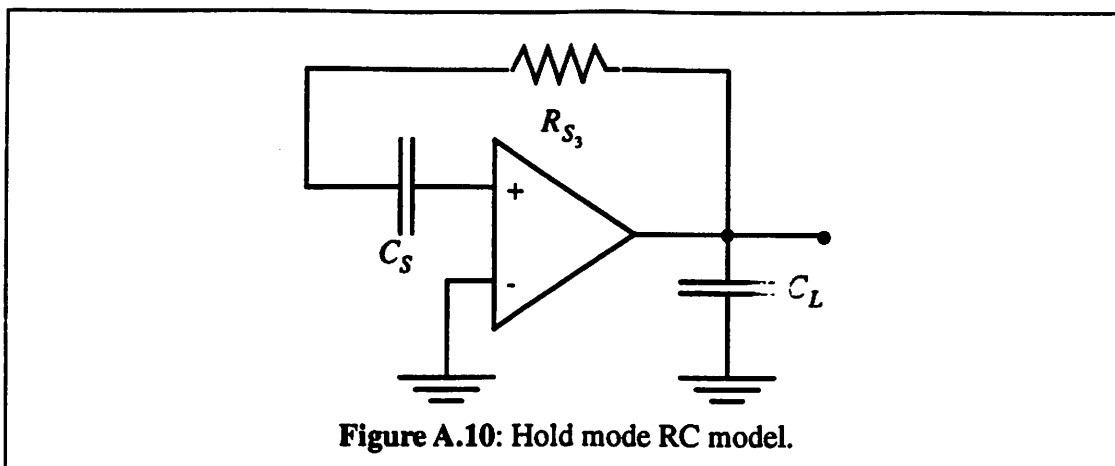


Figure A.10: Hold mode RC model.

For the holding network to settle as quickly as possible and contribute negligible phase shift, the switch size has to be chosen so that $t_{aq} \ll \tau$. Let $t_{aq} = \beta \cdot \tau$ where $\beta \ll 1$. Since S_3 and S_4 are CMOS switches, EQ A.39 can be used. Substituting t_{aq} and R_{on} into EQ A.45, and solve of the aspect ratio of the switch, we obtain

$$\frac{W_3}{L_3} \geq \frac{N \cdot C_L \cdot \ln(2)}{\beta \cdot \tau \cdot (1+b) \cdot \mu_n \cdot C_{ox} \cdot (V_H - V_T)} \quad (\text{EQ A.46})$$

All equations obtained indicate that the larger the switch size the better the performance of the S/H block. Unfortunately, there are factors that give the upper limit as how large the switch sizes can be. The first is the charge injection voltage resulting from turning the switches on and off. For this particular S/H circuitry, the charge injection contribution will be from S_5 , S_6 , and S_7 will be the critical ones. One needs to determine how much charge injection voltage can be injected to the summing before it affects the performance of the op-amp. Usually, it is dictated by the input common mode range of the op-amp. The second factor is the parasitic capacitance contribution from a large switch will be larger. Usually, for mid-frequency applications, the switch size will be fairly small and therefore, the parasitic capacitance contribution is negligible. Thus, the combination of the

two factors mentioned above will determine the upper limit of the switch.

Notice that all the equations derived above are for calculating the switch sizes of the input S/H block (S/H-1). For the second S/H block (S/H-2), all equations are valid except the requirements are much more relax. For example, the resolution is reduced by one, the input signal is pretty much dc, and the common mode input frequency is non-existence. Therefore, all the switch sizes in S/H-2 will be much smaller than S/H-1.

A.2.6 Amplifier

Ideally the S/H-1, S/H-2, and 2X blocks have different op-amp performance since the requirements of these blocks are different. But to simplify the design procedure, the three blocks use the same op-amp. Among the three blocks, the 2X block is the critical one in that its op-amp gain requirement has to be better than the S/H blocks due to its large feedback factor. Thus the discussion of amplifier design will be described in section A.3.2

A.3 Gain of Two Block

In the following section, the gain of two block circuitry and switching scheme will be presented. Transfer function and the noise source of this block will also be discussed.

A.3.1 2X Circuit

The gain of two circuit is the same as shown in Figure A.2 except the capacitor ratio $\frac{C_S}{C_I}$ is equal to two. In order to fully utilize this 2X block, the reference subtraction circuit is also integrated inside by adding reference capacitors C_{REF} and CMOS switches which select V_{REF} , GND , or $-V_{REF}$. The complete 2X block is shown in Figure A.11.

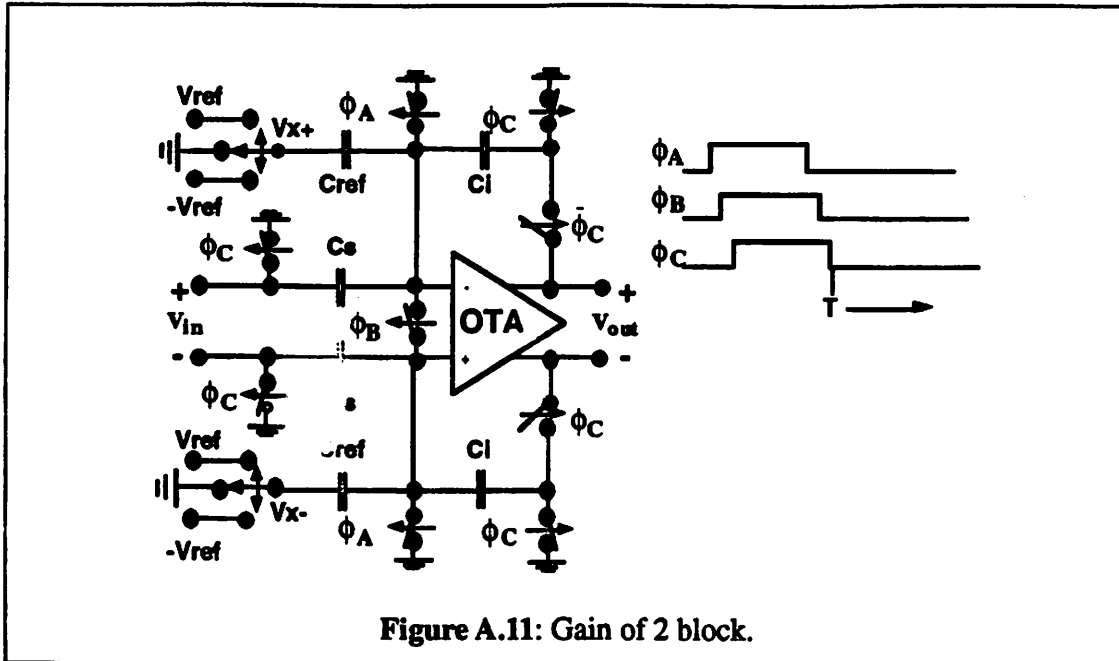
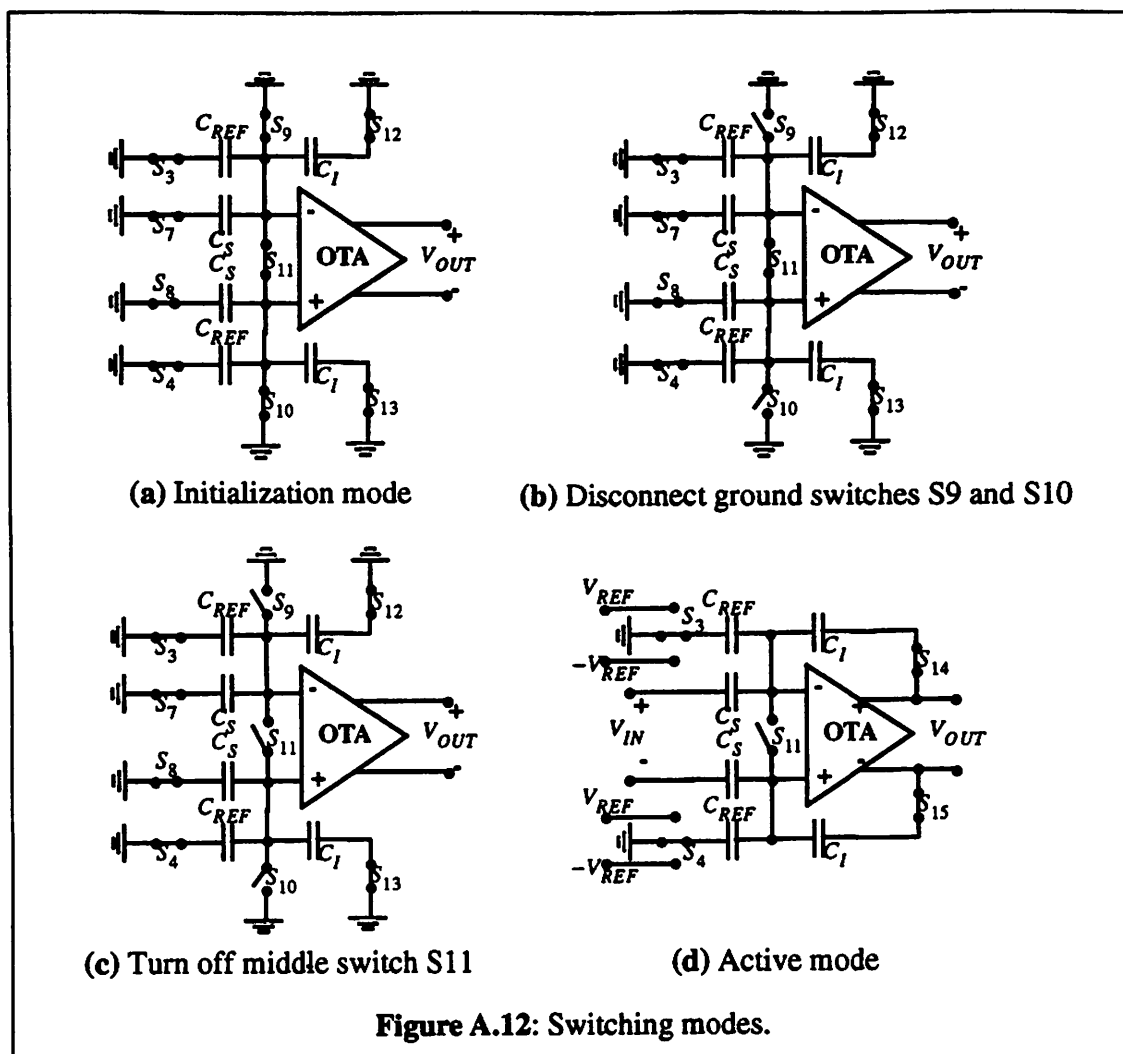


Figure A.11: Gain of 2 block.

A.3.2 Switching Scheme

At the beginning of each bit conversion, all the capacitors in the block will be initialized to zero by closing switches $S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}$ (Figure A.12(a)). During this period, the op-amp also undergoes an initialization period where the common mode voltage for the outputs are initialized, and the offset voltage and noise of the op-amp are stored. The details of the op-amp initialization will be discussed later. The initialization period is carried out during the sampling of the input signal. Just like the S/H block, the ground switches S_9 and S_{10} are turned off first (Figure A.12(b)), followed by S_{11} (Figure A.12(c)). The initialization period is completed when S_7, S_8, S_{12} and S_{13} are turned off and S_{14} and S_{15} are turned on, connecting the integrating capacitors to the outputs (Figure A.12(d)). At this time, the 2X block is ready to perform multiplication for the rest of the conversion cycles. The reference subtraction switches S_1, S_2, S_3, S_4, S_5 , and S_6 will be controlled by the comparator outputs of the previous cycle. Only one switch pair (i.e. $S_1 -$

S_2 or S_3 - S_4 or S_5 - S_6) will be on at any given time.



A.3.3 Transfer Function

The transfer function of the 2X block can also be analyzed by using conservation of charge on the capacitors. Once again the analysis is simplified by considering a single ended case. During the initialization mode (Figure A.13(a)), the charge at node X is zero since all the capacitors are discharged.

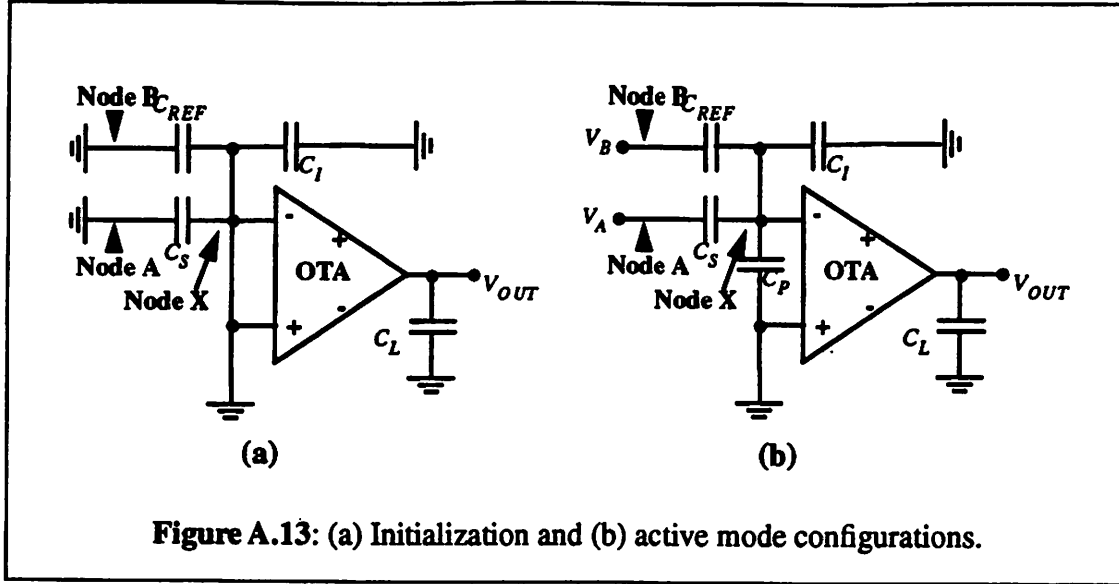


Figure A.13: (a) Initialization and (b) active mode configurations.

$$Q_T = 0 \quad (\text{EQ A.47})$$

Let us assume nodes A and B are connected to V_A and V_B respectively when the 2X block is in the active mode (Figure A.13(b)). Therefore, the total charge at node X at this time is

$$Q_{T+1} = C_S \cdot [V_X(T+1) - V_A(T+1)] + C_{REF} \cdot [V_X(T+1) - V_B(T+1)] \\ + C_I \cdot [V_X(T+1) - V_{OUT}(T+1)] + C_P \cdot V_X(T+1) \quad (\text{EQ A.48})$$

where $V_i(t)$ = voltage at node i at time t

$$V_{OUT}(t) = -[A_V \cdot V_X(t)] \quad (\text{EQ A.49})$$

Equating the charge at node X before and after the initialization:

$$0 = C_S \cdot [V_X(T+1) - V_A(T+1)] + C_{REF} \cdot [V_X(T+1) - V_B(T+1)] \\ + C_I \cdot [V_X(T+1) - V_{OUT}(T+1)] + C_P \cdot V_X(T+1) \quad (\text{EQ A.50})$$

Substituting EQ A.49 into EQ A.50 and solve V_{OUT} in terms of V_A and V_B , we get

$$V_{OUT}(T+1) = - \left(\left[\frac{C_S}{C_I + \frac{C_{Total}}{A_V}} \right] \cdot V_A(T+1) \right) - \left(\left[\frac{C_{REF}}{C_I + \frac{C_{Total}}{A_V}} \right] \cdot V_B(T+1) \right) \quad (EQ A.51)$$

where $C_{Total} = C_S + C_I + C_{REF} + C_P$

EQ A.51 clearly shows that the 2X block functions as a **pure multiplier**. The accuracy of the multiplier will depend again not only on the matching between C_S and C_I , and C_{REF} and C_I but also on the gain of the op-amp A_V . At high resolution, the accuracy of the multiplier will become critical in determining the overall linearity of the conversion. If the capacitor matching is not good enough, trim arrays for C_S and C_{REF} can be added. A method of calibration of two capacitors using trim arrays will be discussed at the end of this section.

A.3.4 Noise Source

Just like the S/H block, the thermal noise for this block comes from the middle switch S_{II} . Carrying out the same derivation, the total input referred mean square noise contribution for this block is

$$\bar{v}_{input}^2 = 2 \cdot \frac{kT}{C_T} \cdot \left[\frac{C_T}{C_S} \right]^2 \quad (EQ A.52)$$

where $C_T = C_S + C_{REF} + C_I + C_P$

Building Blocks

If $C_{REF} = C_I = C_P = C$ and $C_S = 2C$, then $C_T = 5C$ and the total input referred mean square noise is:

$$\overline{v^2}_{input} = \frac{5}{2} \cdot \frac{kT}{C} \quad (\text{EQ A.53})$$

Comparing the mean square input referred noise between the S/H and 2X blocks for the same value of unit capacitor, the 2X block is bigger by a factor of 2.5. Therefore, determination of the size of the smallest unit capacitor based on $\frac{kT}{C}$ noise will be dominated by noise contributed by the 2X block.

A.4 Amplifier

Ideally the S/H-1, S/H-2, and 2X blocks have different op-amp performances since the requirements of these blocks are different. But to simplify the design procedure, the three blocks use the same op-amp. Among the three blocks, the 2X block is the critical one in that its op-amp gain requirement has to be better than the S/H blocks due to its large feedback factor. Thus the following discussion of amplifier design will be intended for 2X block.

A.4.1 Amplifier Specifications

The specifications of the 2X block amplifier depend directly on the resolution and conversion speed of the A/D converter. Referring back to Chapter 3, the accuracy of the gain block has to be within $\epsilon = \frac{1}{2^{N-1}}$. For an 8-bit A/D converter, this means $\epsilon = 0.78\%$. The effective gain from input to output of the 2X block (see Figure A.11) can be written as

$$Gain = \frac{V_{OUT}}{V_{IN}} = - \left[\frac{C_S}{\frac{C_I + C_S + C_P + C_{REF}}{A_V} + C_I} \right] \geq -(2 - \epsilon) \quad (\text{EQ A.54})$$

If $C_I = C_{REF} = C$ and $C_S = C_P = 2C$, then EQ A.54 can be simplified to be

$$Gain = \frac{-2}{1 + \frac{6}{A_V}} \geq -(2 - \epsilon) \quad (\text{EQ A.55})$$

$$A_v \geq \frac{6}{\frac{1}{(1 - \frac{\epsilon}{2})} - 1} \quad (\text{EQ A.56})$$

$$\text{If } \frac{1}{1-x} \approx 1+x \text{ for } x \ll 1, \text{ then} \quad (\text{EQ A.57})$$

$$A_v \geq 6(2^N) = 1536 \quad (\text{EQ A.58})$$

From the above calculation, the op-amp has to have open loop gain greater than 1536. At higher resolutions, capacitor mismatch also degrades gain accuracy. One way to solve capacitor matching problem is to add a capacitor trim array (this will be discussed later in this appendix).

To achieve an 8-bit accuracy at 125kHz, clocks Φ_1 and Φ_2 shown in Figure 3.19 have to have a period of 2 μ s. Assuming the clocks have a 47% duty cycle, the total settling time for 2 op-amps (i.e. S/H and 2X blocks) is approximately 940 nanoseconds. A detailed clock division is shown in Figure A.14. The non-overlap time between Φ_1 and Φ_2 is approximately 60 nanoseconds, during which the comparators make their decisions and then latch the current bit into the register.

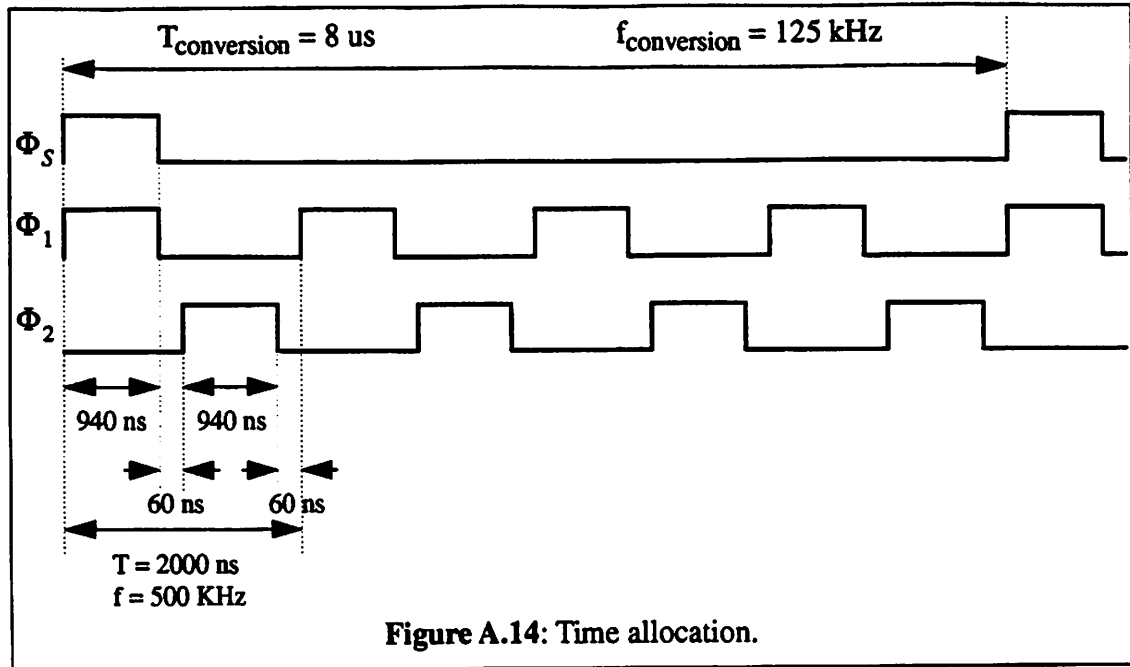


Figure A.14: Time allocation.

Referring to Figure A.15, Let t_a be the time required to acquire an input signal to N-bit accuracy for an op-amp in gain block configuration. t_a can be approximated [LEWI87b] to be

$$t_a \geq 0.7 \cdot N \cdot \tau_{FB} \quad (\text{EQ A.59})$$

where τ_{FB} = output time constant with feedback

Let t_{a1} and t_{a2} be the settling times for S/H and 2X blocks respectively.

For the 2X block:

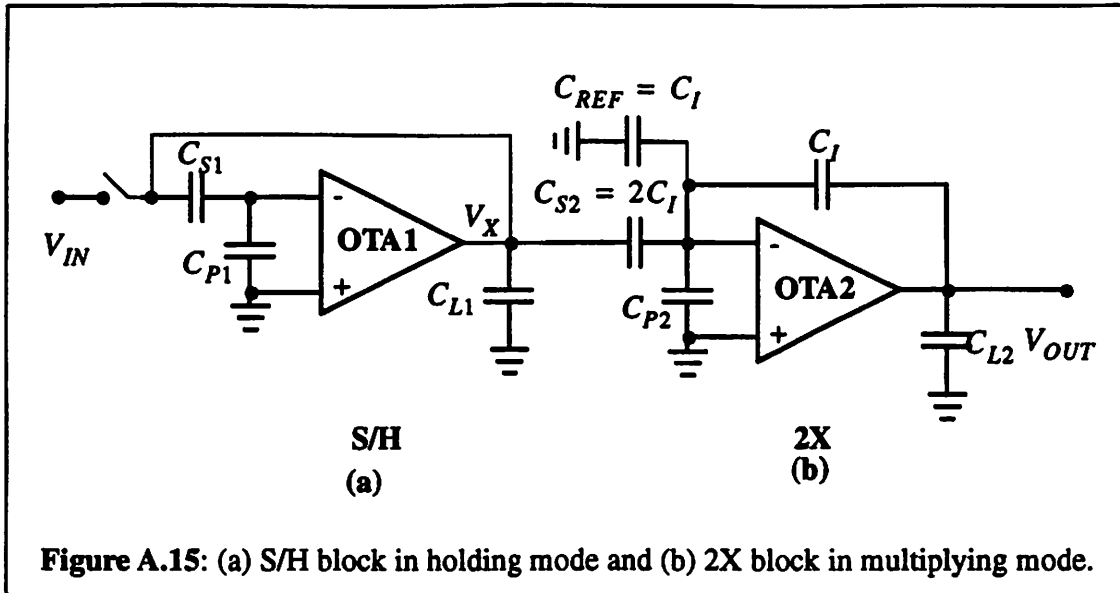


Figure A.15: (a) S/H block in holding mode and (b) 2X block in multiplying mode.

$$\tau_{FB2} = \frac{\tau_2}{1 + A_V \cdot \left[\frac{C_I}{C_S + C_I + C_{REF} + C_P} \right]} \quad (\text{EQ A.60})$$

where τ_2 = the output time constant of the 2X block

A_V = open-loop gain

For the S/H block:

$$\tau_{FB1} = \frac{\tau_1}{1 + A_V \cdot \left[\frac{C_I}{C_I + C_P} \right]} \quad (\text{EQ A.61})$$

$$\tau_i = R_{OUT} \cdot (C_{Li} + C_I) \quad (\text{EQ A.62})$$

where N = number of bits

R_{OUT} = the open-loop output resistance of the op-amp

C_{Li} = total loading capacitor at the output of the op-amp

i = the S/H or 2X block

Therefore, 940 ns is the total time $t_{a1} + t_{a2}$ for both blocks to settle within N bits of accuracy. The assumption here is not exactly accurate since while the S/H block is in the hold mode, the 2X block is in the multiplication mode. The output of the S/H block is connected to the input of 2X block. Therefore, as the output of the S/H block (the input of the 2X block) is slewing and settling, the 2X block is also slewing and settling. Effectively, the total slewing and settling time for the combined two blocks are less than 2 op-amp settling times. For illustration purpose, the 2 op-amp settling time is used. If $C_S = C_P = 2C_I = 2C_{REF}$ and $\tau_1 = \tau_2$, τ_{FB1} and τ_{FB2} can be approximated by:

$$\tau_{FB1} = \frac{\tau}{1 + \frac{A_V}{2}} \approx \frac{\tau}{\frac{A_V}{2}} \quad (\text{EQ A.63})$$

$$\tau_{FB2} = \frac{\tau}{1 + \frac{A_V}{6}} \approx \frac{\tau}{\frac{A_V}{6}} \quad (\text{EQ A.64})$$

$$\tau_{FB2} \approx 3 \cdot \tau_{FB1} \quad (\text{EQ A.65})$$

$$t_{Total} \geq t_{a1} + t_{a2} \approx 0.7 \cdot N \cdot (\tau_{FB1} + \tau_{FB2}) = 2.8 \cdot N \cdot \tau_{FB1} \quad (\text{EQ A.66})$$

For $N = 8$ and $t_{Total} \geq 940\text{ns}$, $\tau_{FB1} \leq 42\text{ns}$, corresponding to a closed-loop -3 dB bandwidth of about 3.8 MHz.

From EQ A.65, the time for the 2X block to settle is around $\frac{3}{4}$ of 940 ns which is around 705 ns. Assuming half of the 705 ns is for slewing and the other half is for settling, for $C_L + C_I = 5$ pF and 4 Volts peak-to-peak swing, the slew rate has to be least $11.36 \frac{\text{Volts}}{\mu\text{s}}$ and output charging current $71 \mu\text{A}$. The necessary input transconductance of the op-amp G_m would then be approximately $G_m = 0.4 \text{ mA/volt}$.

In the remaining of this section, the chosen op-amp topology and the techniques to reduce the offset and noise of the amplifier will be presented.

A.4.2 Basic Op-amp Topology

Before analyzing the op-amp behavior, all the symbols used in the equations are defined below.

A_0 = dc open loop gain

R_{OU} = equivalent output resistance

z_1 = the zero of the transfer function

p_1 = the dominant pole of the op-amp

p_2 = the first non dominant pole

p_3 = the second non dominant pole

C_{LX} = total capacitance at node X

g_{mi} = transconductance of transistor i

g_{oi} = output conductance of transistor i

C_{gdi} = gate to drain capacitor of transistor i

C_{gsi} = gate to source capacitor of transistor i

C_{dbi} = drain to bulk capacitor of transistor i

C_{sbi} = source to bulk capacitor of transistor i

V_{VDSATi} = saturation voltage of transistor i

V_{GS} = gate to source voltage

V_{ti} = threshold voltage of transistor i

$$\Delta V_{t(i-j)} = V_{ti} - V_{tj}$$

$\left(\frac{W}{L}\right)_i$ = the aspect ratio of transistor i

$$\left(\frac{W}{L}\right)_{(i-j)} = \frac{\left(\frac{W}{L}\right)_i + \left(\frac{W}{L}\right)_j}{2}$$

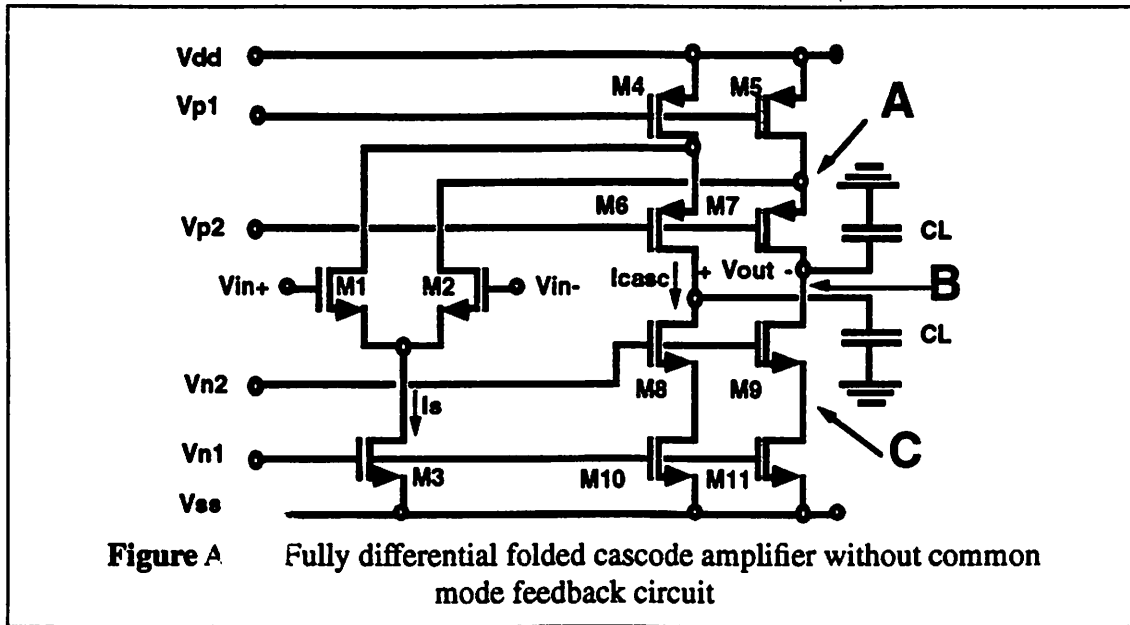
$$\Delta \left(\frac{W}{L}\right)_{(i-j)} = \left(\frac{W}{L}\right)_i - \left(\frac{W}{L}\right)_j$$

\bar{v}_i^2 = equivalent input referred noise for transistor i

K_f = flicker noise coefficient, $3 \times 10^{-12} \text{V}^2/\text{pF}$

R_X = equivalent resistance at node X

A fully differential folded-cascode CMOS amplifier shown in Figure A.16 [GRAY84] is chosen because it can achieve high gain and is simple in topology. The transfer function of the folded cascode is



$$\frac{V_{OUT}}{V_{IN}}(s) = A_d(s) = A_0 \frac{(1 + \frac{s}{z_1})}{(1 + \frac{s}{p_3})(1 + \frac{s}{p_2})(1 + \frac{s}{p_1})} \quad (\text{EQ A.67})$$

where $A_0 = -g_{m1}R_{OUT}$

$$R_{OUT} = \frac{1}{G_1 + G_2 + G_3}$$

Amplifier

$$G_1 = \frac{(g_{o1} + g_{o4}) g_{o6}}{g_{m6} + g_{mb6} + g_{o6}}$$

$$G_2 = \frac{g_{o8} g_{o10}}{g_{o10} + g_{o8} + g_{m8} + g_{mb8}}$$

$$G_3 = \frac{(g_{o1} + g_{o4}) g_{o6} g_{o10}}{(g_{m6} + g_{mb6} + g_{o6}) (g_{o10} + g_{o8} + g_{m8} + g_{mb8})}$$

$$z_1 \approx \frac{g_{o10} + g_{o8} + g_{m8} + g_{mb8}}{C_{LC}}$$

$$p_1 \approx \frac{1}{R_{OUT} C_{LB}}$$

$$p_2 \approx \frac{g_{o1} + g_{o4} + g_{o6} + g_{m6} + g_{mb6}}{C_{LA}}$$

$$p_3 \approx \frac{g_{o10} + g_{o8} + g_{m8} + g_{mb8}}{C_{LC}}$$

$$C_{LA} = C_{gd1} + C_{db1} + C_{gd4} + C_{gs6} + C_{sb6}$$

$$C_{LB} = C_{gd6} + C_{db6} + C_{gd8} + C_{db8} + C_L$$

$$C_{LC} = C_{gs8} + C_{sb8} + C_{gd10} + C_{db10}$$

Design procedures for this type of op-amp is summarized in Table A-1.

Table A-1: Op-amp performance summary.

Specification	Equation
Unity gain frequency	$\omega_0 = \frac{g_{m1}}{C_{LB}}$
Phase margin	$PM = \frac{\pi}{2} - \text{atan}\left(\frac{\omega_0}{p_2}\right)$
Slew rate	$SR = \frac{I_S}{2C_{LB}}$
Cascode bias current	$I_{casc} = \text{MAX}\left(I_S, \frac{1}{2}\omega_0 V_{DSAT} \tan(PM)\right)$
Output swings	$V_{OUT}^+ = V_{DD} - V_{DSAT4} - V_{DSAT6} $ $V_{OUT}^- = V_{SS} + V_{DSAT10} + V_{DSAT8} $
Total power	$PWR = (2I_{casc} + I_S)(V_{DD} - V_{SS})$

The input referred offset voltage is:

$$V_{OS} = \Delta V_{t(1-2)} + \frac{(V_{GS} - V_t)_{(1-2)}}{2} \left(\frac{\Delta\left(\frac{W}{L}\right)_{(1-2)}}{\left(\frac{W}{L}\right)_{(1-2)}} + \frac{\Delta\left(\frac{W}{L}\right)_{(4-5)}}{\left(\frac{W}{L}\right)_{(4-5)}} + \frac{\Delta\left(\frac{W}{L}\right)_{(10-11)}}{\left(\frac{W}{L}\right)_{(10-11)}} \right) \quad (\text{EQ A.68})$$

The equivalent input referred noise is:

$$\bar{v}_{eq}^2 = 2 \left[\bar{v}_1^2 + \bar{v}_4^2 \left(\frac{g_{m4}}{g_{m1}}\right)^2 + \bar{v}_{10}^2 \left(\frac{g_{m10}}{g_{m1}}\right)^2 + \bar{v}_6^2 \left(\frac{1}{g_{m1}R_A}\right)^2 + \bar{v}_8^2 \left(\frac{1}{g_{m1}R_C}\right)^2 \right] \quad (\text{EQ A.69})$$

$$\bar{v}_i^2 = \left[4kT \frac{2}{3} \frac{1}{g_{mi}} + \frac{k_f}{W_i L_i C_{oxf}} \right] \Delta f \quad (\text{EQ A.70})$$

A.4.3 Common Mode Feedback Circuit

The common mode feedback (CMFB) circuit is needed in order to control the common mode voltage of the output node [SEND82]. There are two types of CMFB circuits: continuous [KANE83] and dynamic [LEWI87b], [HOST84], [CAST84]. The difference between the two types is that the dynamic approach needs clocks, whereas, the continuous one does not. However, the continuous CMFB circuit is not very desirable because it limits the output swing of the op-amp.

A simple dynamic CMFB circuit can be obtained by using only two capacitors and a CMOS switch as shown in Figure A.17. The operation of the CMFB circuit is as follows: during the initialization cycle, the voltages across capacitors are initialized to their respective voltages. When the op-amp is in the active mode, S1 will be *opened* and the two capacitors C_{CM1} and C_{CM2} will sense the output common mode voltage. If the common mode output voltage is higher(lower) than the initial voltage, V_A will also go up/down. As a result, more/less current will be drawn by M10 and M11 to pull down/up the output common mode voltage. If there is a mismatch between C_{CM1} and C_{CM2} , noise from the supply line V_{SS} will be injected or coupled to the output nodes through $C_{gs10} - C_{CM1}$ and $C_{gs11} - C_{CM2}$ and it will appear as a differential output voltage. At high resolution, the amount of noise injected through the supply line may affect the linearity of the conversion.

Figure A.18 illustrates a more complicated CMFB circuit in that it consumes more area and power; however, power supply noise is not a problem. The basic operation is the same as before. During the initialization, S1 and S2 are closed to initialize the voltages at nodes A1 and A2. Then, S1 and S2 are opened and S3 is closed. Thus, C_{CM1} and C_{CM2}

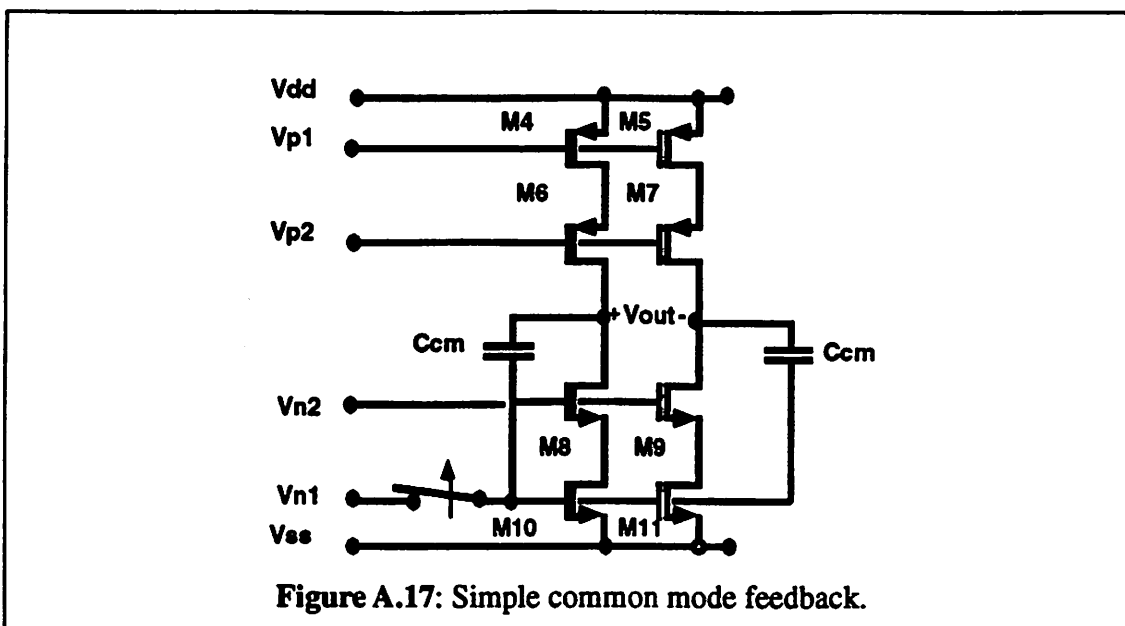
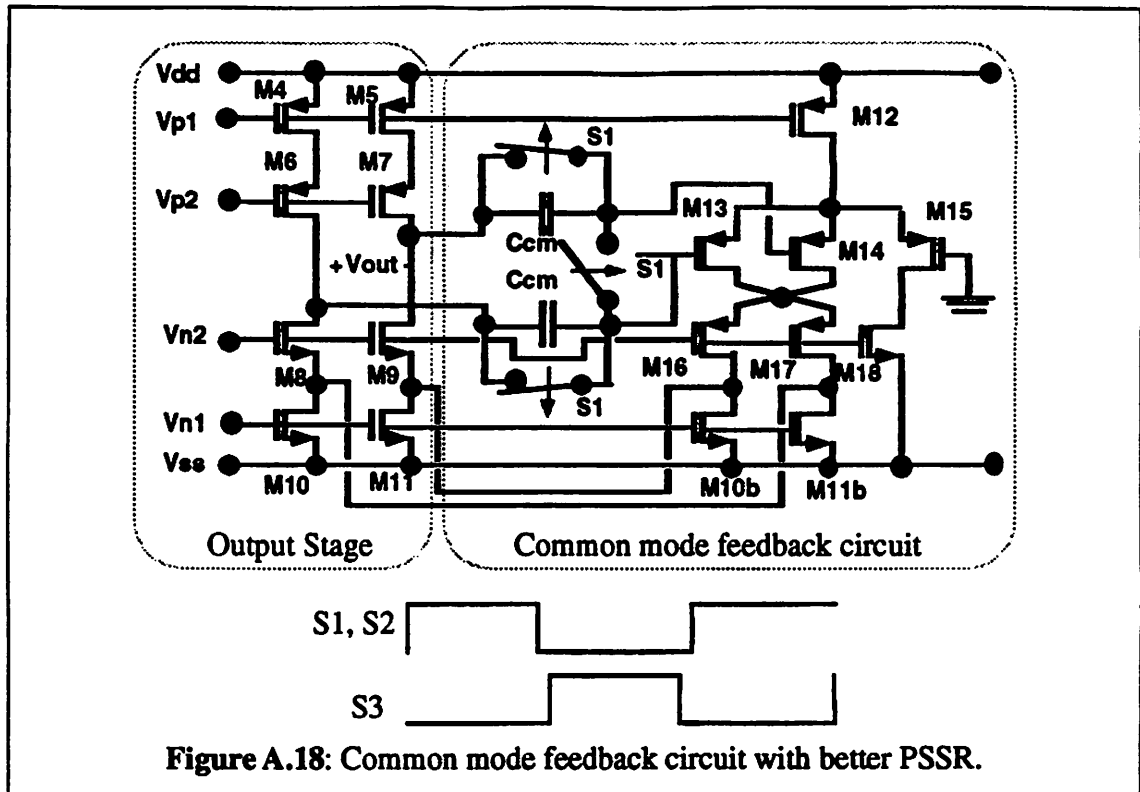


Figure A.17: Simple common mode feedback.

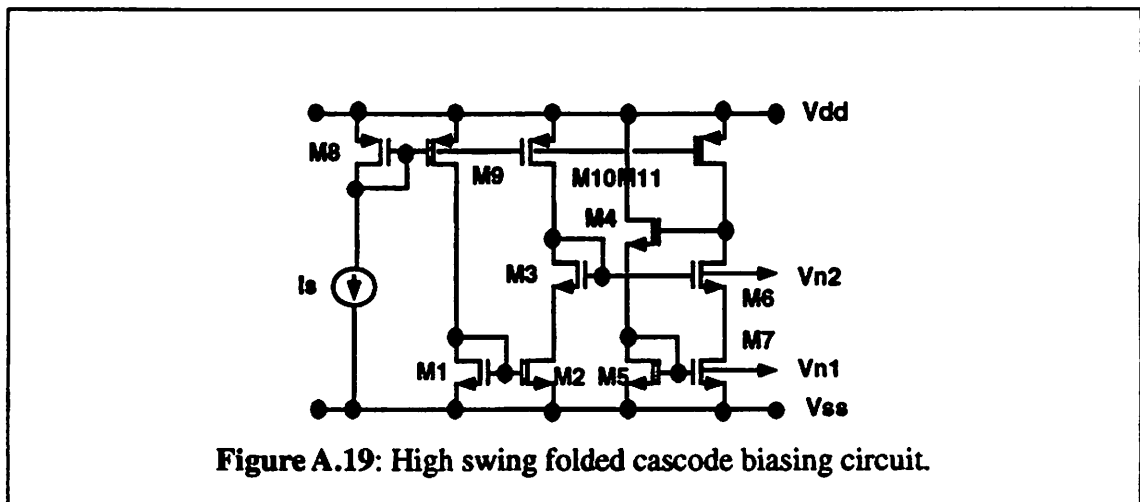
function as output common mode voltage sensors. In order to cover a broad spectrum of resolution of A/D converters (i.e. from 6 up to 12 bits), this CMFB circuit is implemented in this 8-bit algorithmic ADC prototype even though the simple CMFB circuit is sufficient.

A.4.4 Biasing Circuit

A high swing folded cascode biasing circuit generating the bias voltages V_{N1} and V_{N2} for NMOS transistors is shown in Figure A.19. The basic circuit is similar to the one reported previously [OHAR87] with some modifications. Transistor M2 is designed to be in the linear region by making its aspect ratio larger than that of M1. The purpose is to make the V_{DS} of M2 to be a low impedance voltage source and set the V_G of transistor M3. The resulting V_{G3} can be used to bias the cascode NMOS transistors of the core op-amp. In order to guarantee that the two bias lines V_{N1} and V_{N2} track, M4-M7 are added. Transistor M4 forms a feedback loop from V_{D6} so that M6 can operate in deep saturation region.

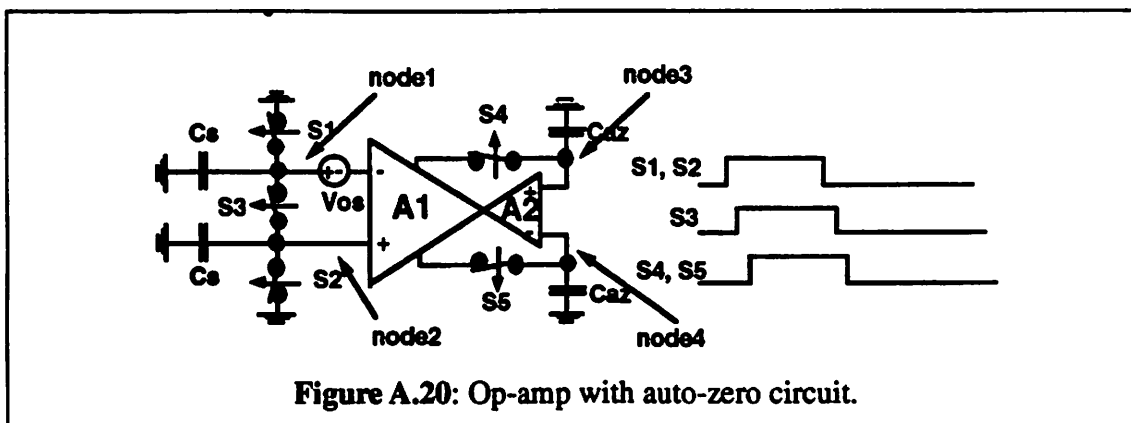


Unfortunately, this biasing circuit requires more power and area. The complementary bias circuit is used to generate biasing voltages V_{P1} and V_{P2} .



A.4.5 Auto-Zero Circuit

At high resolution, offset voltages and charge injection due to op-amp and CMOS switches will affect the overall linearity of algorithmic A/D converter. In order to reduce these effects, an auto-zero circuit can be added into the op-amp [OHAR87], [DEGR85] as shown in Figure A.20. This auto-zero circuit can be treated as an auxiliary op-amp (A_2) with its inputs connected to the outputs of the main op-amp (A_1) to form a feedback loop. The storing of these offset voltages can be carried out during the initialization of the block, thus no extra clock periods are needed to include the auto-zero circuit.



Let V_{os1} be the offset voltage of the op-amp. When ϕ_a , ϕ_b , and ϕ_c go high, all switches are closed, and V_{os1} gets amplified by A_1 , fed back by A_2 , and eventually an equilibrium is established. The voltage across node 3 and 4 is stored on capacitor C_{az} and is equal to

$$V_{3-4} = \frac{-A_1 V_{os1}}{1 + A_2} \quad (\text{EQ A.71})$$

When ϕ_a and ϕ_b go low, S1, S2 and S3 will get turned off and inject charges into node 1

and 2. The differential error voltages ΔV_1 between these two nodes will also get amplified by A_1 and stabilized since the feedback mechanism is still active. Therefore, the voltage across node 3 and 4 will be

$$V_{3-4} = \frac{-A_1 (V_{os1} + \Delta V_1)}{1 + A_2} \quad (\text{EQ A.72})$$

When ϕ_c goes low, S4 and S5 will inject charge into node 3 and 4. This injected charge results in a voltage across nodes 3 and 4. The total voltage difference across nodes 3 and 4 due to these 3 error sources is given by

$$V_{3-4} = \frac{-A_1 (V_{os1} + \Delta V_1)}{1 + A_2} + \Delta V_2 \quad (\text{EQ A.73})$$

Across the output nodes of the op-amp, the voltage will be equal to

$$V_{out} = -A_1 \left[\left(\frac{V_{os1} + \Delta V_1}{1 + A_2} \right) + \Delta V_2 \frac{A_2}{A_1} \right] \quad (\text{EQ A.74})$$

This voltage can be referred back to the input by dividing EQ A.74 by $-A_1$, resulting in an input referred offset of

$$V_{in} = \frac{V_{os1} + \Delta V_1}{1 + A_2} + \Delta V_2 \frac{A_2}{A_1} \quad (\text{EQ A.75})$$

Therefore, the offset voltage due to op-amp and charge injection at the input nodes are attenuated by $(1 + A_2)$, and the charge injection due to switch S4 and S5 is reduced by a factor of $\frac{A_2}{A_1}$. The optimum value for A_2 turns out to be $\frac{1}{10}$ of A_1 or in other words, g_{m2}

is equal to $\frac{g_{m1}}{10}$ where g_{m1} and g_{m2} are the input transconductances of the main and auxiliary op-amps respectively. For an 8-bit resolution $A_1 \geq 1536$ and $A_2 \approx 154$. The offset voltage due to the op-amp and switch charge injection at the input nodes is 0.65% of the actual value, whereas, those from S4 and S5 is 10% of the initial value. Thus the charge injection due to S4 and S5 will dominate the offset error. Careful design is needed to ensure that ΔV_2 is small enough.

A.4.6 Noise Reduction

The beauty of the auto-zero circuit described above is that it reduces the noise in the 2X block by a factor of $\frac{g_{m1}}{g_{m2}}$. Let us assume the equivalent mean square noise at the input of the op-amp be

$$\bar{v}_{eq}^2 \approx 4kT \frac{b}{g_{m1}} \Delta f \quad (\text{EQ A.76})$$

where b = a constant factor

This noise voltage will then get amplified and appear across v_{3-4} .

$$\bar{v}_{3-4}^2 \approx \left[\frac{g_{m1}}{g_{m2}} \right]^2 \bar{v}_{eq}^2 = \left[\frac{g_{m1}}{g_{m2}} \right]^2 4kT \frac{b}{g_{m1}} \Delta f \quad (\text{EQ A.77})$$

Let us also assume that the -3dB frequency of the auxiliary circuit is equal to

$$\omega_{-3dB} = \frac{g_{m2}}{C_{az}} \quad (\text{EQ A.78})$$

$$f_{-3dB} = \frac{1}{2\pi} \frac{g_{m2}}{C_{az}} \quad (\text{EQ A.79})$$

The equivalent noise bandwidth can then be approximated [GRAY84]

$$f_N = \frac{\pi}{2} f_{-3dB} = \frac{1}{4} \frac{g_{m2}}{C_{az}} \quad (\text{EQ A.80})$$

Substituting f_N for Δf in EQ A.77, the mean square noise across node 3 and 4

$$\bar{v}_{3-4}^2 = b \left[\frac{g_{m1}}{g_{m2}} \right] \left[\frac{kT}{C_{az}} \right] \quad (\text{EQ A.81})$$

When switches S4 and S5 are turned off, this noise is sampled onto C_{az} . Referring this

noise back to the op-amp input by dividing EQ A.81 by $\left[\frac{g_{m2}}{g_{m1}} \right]^2$, we get

$$\bar{v}_{eq, input}^2 = b \left[\frac{g_{m2}}{g_{m1}} \right] \left[\frac{kT}{C_{az}} \right] \quad (\text{EQ A.82})$$

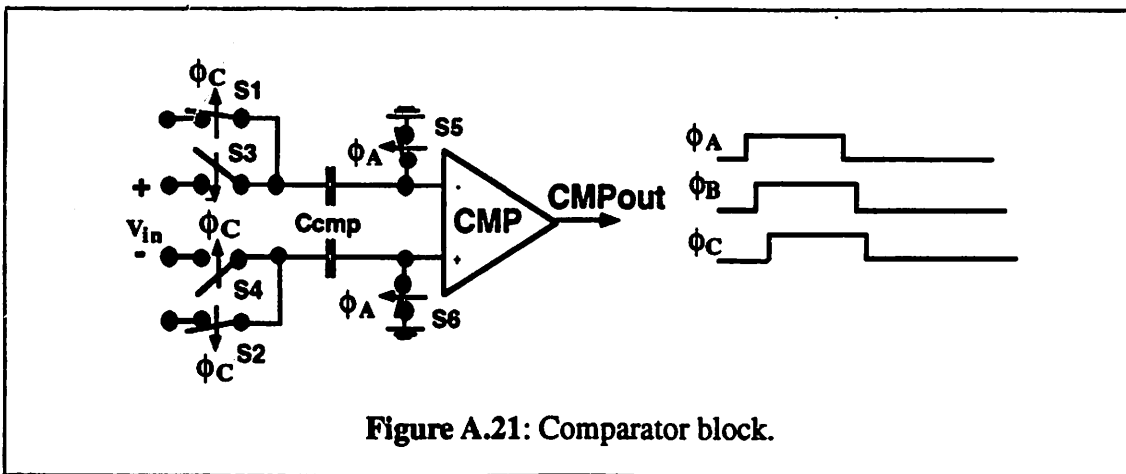
Therefore, the $\frac{kT}{C}$ noise on C_{az} gets attenuated by the same factor as the ratio between

$$\frac{g_{m2}}{g_{m1}}$$

A.5 Comparator Blocks

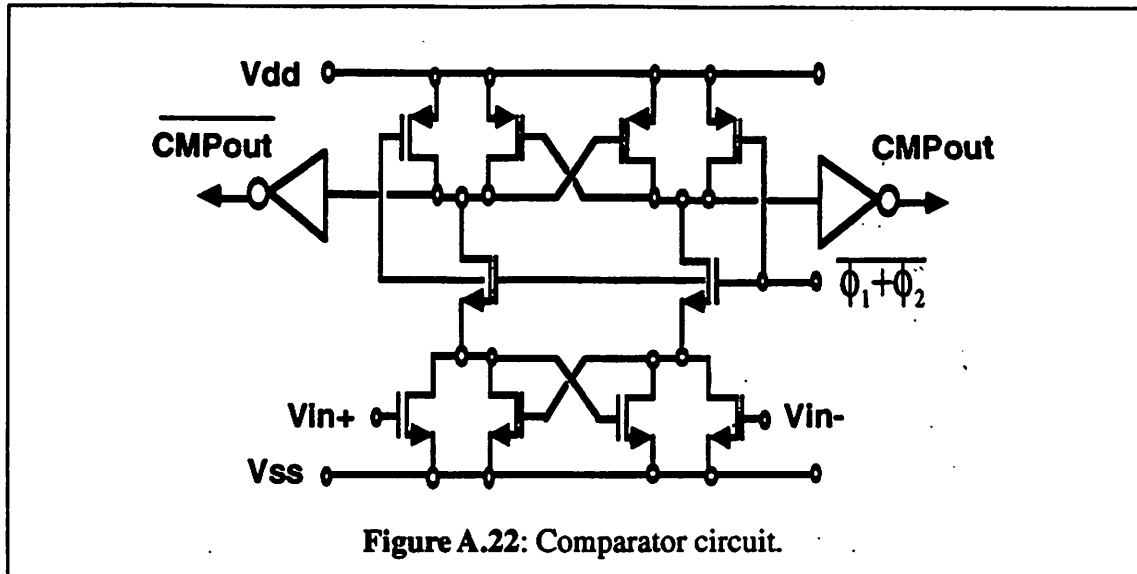
Figure A.21 shows a comparator block. At the beginning of the conversion i.e. when the input signal is sampled onto the S/H-1 block, the comparators are being initial-

ized. The threshold voltages ($\pm \frac{V_{REF}}{4}$) are sampled onto capacitors C_1 and C_2 by closing switches S1, S2, S5, and S6. At the end of the initialization, S5 and S6 are opened first, followed by the opening of S1, S2 and the closing of S3 and S4. Now the comparator is ready to do the comparison. At the beginning of the next input sample, the comparators will again be initialized.



The comparator design is very simple since the offset voltage of the comparator is not an issue here. As long as the comparators make decisions in less than the allowable time slot which is 60 ns, any type of comparator will do the job. Figure A.22 shows the schematic of a comparator circuit where the comparator is realized by a simple regenerative latch [DOER88], [WU88]. The operation of the comparator is as follows:

1. When ϕ_{COMP} is low, the PMOS transistors M8 and M9 are disconnected from the NMOS transistors (M1-M4) by M6 and M5. M7 and M10 are on, and thus they set the output voltage to high. During this time, the inputs of the comparator are changing and settling which in turn affects the values of effective resistors of M1 and M4. M7, and M10.



2. When ϕ_{COMP} goes high, the comparator outputs will go to logic 1 or 0 depending on the resistances and capacitances on the drains of M1-M10. To avoid unbalanced parasitic capacitance at the output nodes due to routing, two inverters are added.

A.6 Digital Circuits

All the necessary digital circuitry and combination logic are integrated into the final ADC netlist. Referring back to section 3.5.3.1, the decision bits from comparators are used to decide the current bit, the correction of previous bits if necessary, and for controlling the reference subtraction circuit switches. Table A-2 shows how the comparator outputs are decoded. The complete digital error correction block is shown in Figure A.23. The details of the reference clock logic schematic is omitted since it is trivial.

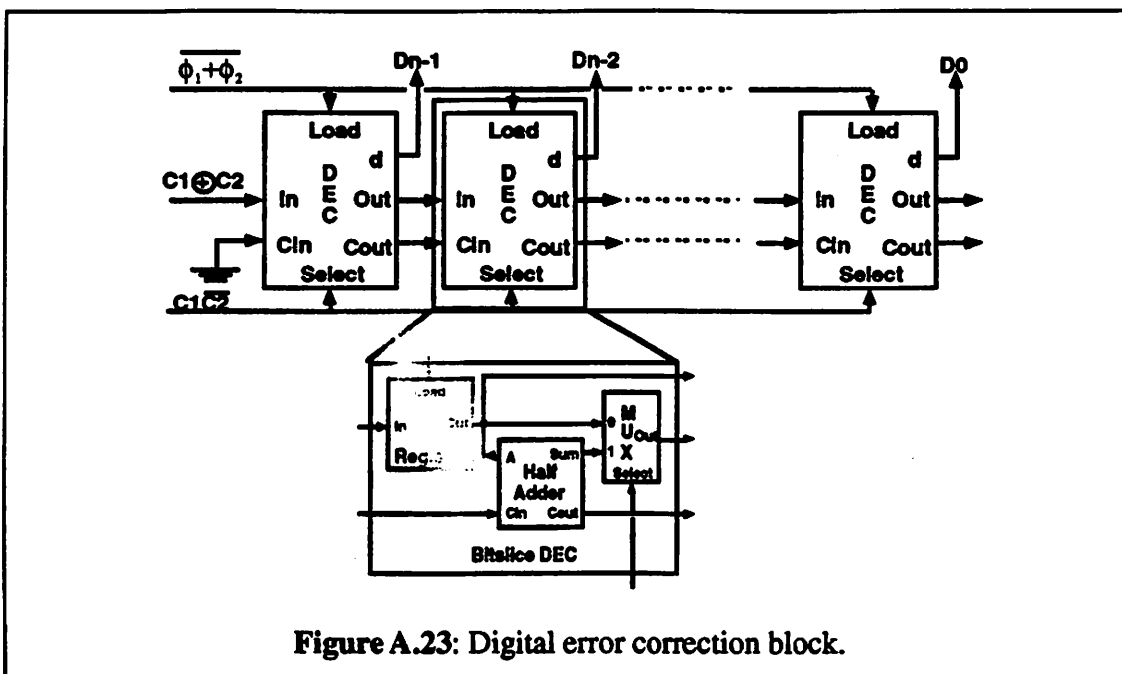


Figure A.23: Digital error correction block.

Table A-2: Decoder summary.

	C1	C2	Reference Clock	Current Bit	SELECT
Reset	0	0	CONNECT to GND	0	PASS
$V_{IN} \geq \frac{V_{REF}}{4}$	0	1	SUBTRACT V_{REF}	1	PASS
$V_{IN} \leq -\frac{V_{REF}}{4}$	1	0	ADD V_{REF}	1	CORRECT
$-\frac{V_{REF}}{4} < V_{IN} < \frac{V_{REF}}{4}$	1	1	CONNECT to GND	0	PASS

A.7 Trim Array

The capacitor trim array module has not been implemented in the ADC module generator, CADICS. In this section, a calibration step for the gain of two block will be presented. The calibration sequence for this 1-bit/cycle algorithmic A/D converter is different from the regular ones [OHAR87], [LEE84] because both C_S and C_{REF} need to be trimmed with respect to C_I . As a reminder, $C_S = 2C_I = 2C_{REF}$. The calibration of these capacitors will be carried out in two different procedures.

1. Calibrating C_{REF} to be equal to C_I . Refer back to Figure A.11 (for simplicity a single ended case is discussed here), initially capacitor C_{REF} is initialized to V_{REF} by connecting the top plate of C_{REF} to V_{REF} and C_I and C_S are discharged as shown in Figure A.24(a). The total charges, Q_T at node X at this time will be

$$Q_T = C_{REF}(0 - V_{REF}) \quad (\text{EQ A.83})$$

The charges stored in C_{REF} is then transfer to C_I by connecting the top plate of C_{REF} to ground. At the same time, the summing node of the op-amp node X is left floating and the top plate of C_I is connected to V_{REF} as shown in Figure A.24(b). At this time, the total charges at node X will be

$$Q_T = C_I(V_X - V_{REF}) \quad (\text{EQ A.84})$$

By conservation of charge, we can equate the two equations.

$$-C_{REF}V_{REF} = C_IV_X - C_IV_{REF} \quad (\text{EQ A.85})$$

$$V_X = 1 - \frac{C_{REF}}{C_I} V_{REF} \quad (\text{EQ A.86})$$

If $C_{REF} \neq C_I$, V_X will be non-zero. If $C_{REF} > C_I$, $V_X < 0$. As a result, the op-amp arranged as a high-gain comparator will give a logic "0". C_{REF} can then be trimmed down by adjusting the capacitor trim switches, eventually the output of the open-loop op-amp will switch from logic "0" to logic "1". At this time, the value of C_{REF} should be very close to C_I . The ΔC increments in the trim array can be designed so that it is good enough for the resolution specified. If $C_{REF} < C_I$, $V_X > 0$. The opposite operation should be performed.

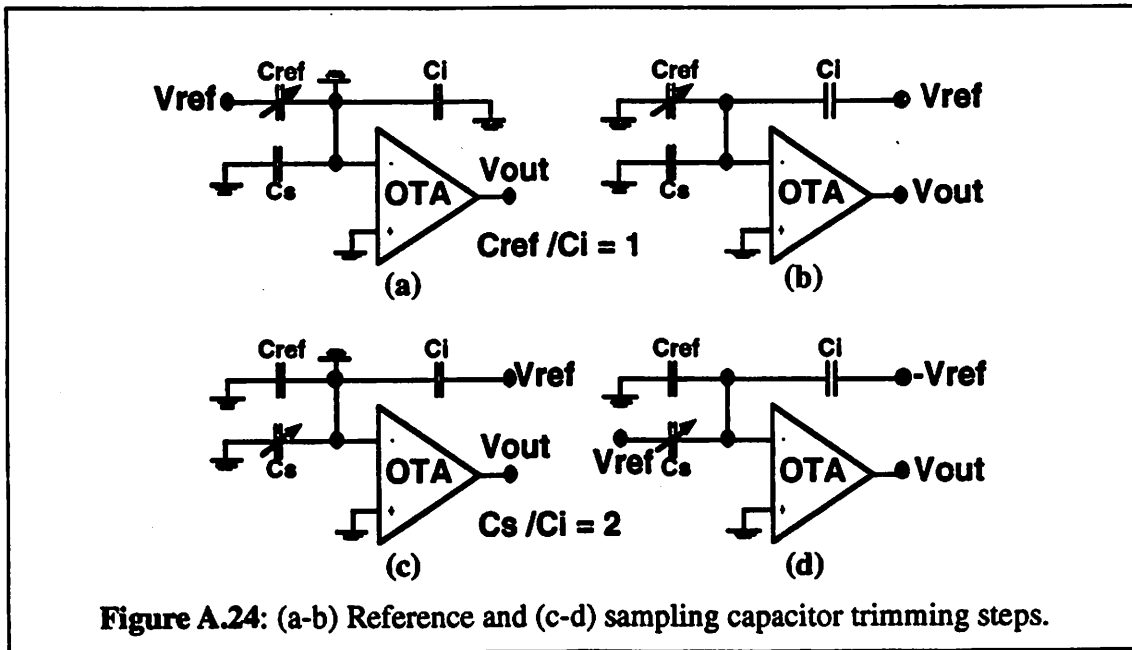


Figure A.24: (a-b) Reference and (c-d) sampling capacitor trimming steps.

2. Calibrating C_S to be $2C_I$. The same method described above can be applied to adjust C_S . First, C_{REF} and C_S are discharged and C_I is set to V_{REF} as shown in Figure A.24(c). The total charges, Q_T at node X will be

$$Q_T = C_I(0 - V_{REF}) \quad (\text{EQ A.87})$$

Then the summing node is open to float the summing node X, and the top plates of C_S and C_I are connected to V_{REF} and $-V_{REF}$ respectively as shown in Figure A.24(d). At this time, the total charges, Q_T at node X will be

$$Q_T = C_I(V_X + V_{REF}) + C_S(V_X - V_{REF}) \quad (\text{EQ A.88})$$

By conservation of charge, we can again equate the two equations.

$$-C_I V_{REF} = C_I V_X + C_I V_{REF} + C_S V_X - C_S V_{REF} \quad (\text{EQ A.89})$$

$$-V_{REF} = V_X + V_{REF} + \frac{C_S}{C_I} V_X - \frac{C_S}{C_I} V_{REF} \quad (\text{EQ A.90})$$

$$V_X \left(1 + \frac{C_S}{C_I}\right) = V_{REF} \left(\frac{C_S}{C_I} - 2\right) \quad (\text{EQ A.91})$$

$$V_X = \frac{V_{REF} \left(\frac{C_S}{C_I} - 2\right)}{1 + \frac{C_S}{C_I}} \quad (\text{EQ A.92})$$

If $C_S \neq 2C_I$, V_X will be non-zero. The same trimming technique described earlier can again be applied here to adjust C_S value.

Thus, the improved algorithmic ADC requires two trimming arrays in order to achieve higher resolution ADC. The current process, the capacitor mismatch is good up to 9 bit resolution. An example of a 3-bit trim array [LIN90] is shown in Figure A.25. As a rule of thumb, the number of bits needed for the trim array is roughly equal to $N-8$ where N is the number of bits and $N \geq 9$.

