3-1997

Automatic Synthesis of VLSI Layout for CMOS Continuous-Time Filters

Wei Han
Portland State University

THESIS APPROVAL

The abstract and thesis of Wei Han for the Master of Science in Electrical and Computer Engineering were presented on January 6, 1997, and accepted by the thesis committee and the department.

COMMITTEE APPROVALS:

Dr. W. Robert Daasch, Chair

Dr. Rolf Schaumann

Dr. Tom Schubert
Representative of the Office of graduate Studies

DEPARTMENT APPROVAL:

Rolf Schaumann, Chair
Department of Electrical Engineering

****

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by  on 19 March 1997

ABSTRACT

AN ABSTRACT OF THE THESIS of Wei Han for the Master of Science in Electrical and Computer Engineering presented on January 6, 1997.

Title: Automatic Synthesis of VLSI layout for CMOS Continuous–Time Filters.

Automatic synthesis of digital VLSI layout has been available for many years. It has become a necessary part of the design industry as the window of time from conception to production shrinks with ever increasing competition. However, automatic synthesis of analog VLSI layout remains rare.

With digital circuits, there is often room for signal drift. In a digital circuit, a signal can drift within a range before hitting the threshold which triggers a change in logic state. The effect of parasitic capacitances for the most part, hinders the timing margins of the signal, but not its functionality. The logic functionality is protected by the inherent noise immunity of digital circuits.

With analog circuits, however, there is little room for signal drift. Parasitic directly influence signal integrity and the functionality of the circuit. The underlying problem, that the automatic VLSI layout programs face, is how to minimize this influence.

This thesis describes a software tool that was written to show that the minimization of parasitic influence is possible in the case of automatic layout of continuous–time filters using transconductance–capacitor methods.

AUTOMATIC SYNTHESIS OF VLSI LAYOUT FOR CMOS CONTINUOUS-TIME FILTERS

by

Wei Han

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL AND COMPUTER ENGINEERING

Portland State University
1997

ACKNOWLEDGMENTS

I would like to acknowledged the previous work done on this project by Martine Wedlake, Dr. W. Robert Daasch and David Lyle Robinson. They were the real pioneers in this tool. I was fortunate to be the one to inherit the work in progress. I wish to express my deepest gratitude to my advisor, Dr. Daasch, for his patience and perseverance on my behalf, his timely suggestions, and his properate guidance. I would also like to thank Dr. Schaumann for his help me to gain knowledge beyond the research area.

My sincere appreciation goes to the members of the committee, Dr. Schaumann and Dr. Schubert for their helpful comments and understanding. I am very grateful to David Chiang, Girish Chandrasekaran for their special contribution in build the standard cell library. The tremendous works they had done have provided me a lot of convenience in my thesis work. I also thank my co–researcher Jonathon W. Walker, all the help and encouragement he provided are great appreciated.

Finally, my mother and my sister deserve my special thanks for their devotion, support, understanding throughout this whole project. And I also thank my good friends for their love, encouragement and understanding. I will forever be indebted to them for all that they have done.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION AND OBJECTIVE

The process of computer software completing one or more IC design tasks is called design automation and exists in many varieties. Design automation software came about because of the demand to be more aggressive and competitive in a tight market. Increasing demands on engineering resources provoked research into productivity tools. The sometimes mundane task of layout was a prime candidate for automation. Initially, layout was accomplished by creating a text file of numerical coordinates describing plot locations for the mask layers. The first step towards improving layout efficiency was the introduction of graphical oriented layout tools such as MAGIC [10]. Furthermore, designers began to reuse the layout of some common circuit elements. Terminology would be evolved to fit the practice. "Full Custom Design" meant that everything was designed by hand using tools such as MAGIC. A good example of the Full Custom Design is the Zilog Z–80 microprocessor. One characteristic of Full Custom Design is that the overall layout of the chip shows very little regularity [7].

In contrast to Full Custom Design, the Intel Pentium microprocessor, which came about two decades later, uses the "Semi–Custom" methodology, where a library of standard cells was developed and reused wherever possible.

Nowadays, with the advent of hardware description languages, standard cells can be automatically selected from a library and arranged by the computer following a digital designer's software instructions. However, currently the most mature design automation is only for digital circuits.

This thesis will discuss the area of analog IC design automation and describe a specialized analog layout tool, OTACC (Operational Transconductance Amplifier–Capacitance Compiler). This software tool is the focus of a research project in automating the layout of a CMOS continuous–time filter while minimizing the effects of parasitic capacitance and increasing the control of the shape (geometry) of the layout. Automating this part of the design cycle of these circuits simplifies the task of the designer and permits others to use the prior work of more experienced analog circuit designers.

OTACC initially was written by Martine Wedlake and Dr. W. Robert Daasch [4][5][6], and modified by David L. Robinson [7]. After receiving the initial prototype, this project added necessary features to both CFL and OTACC, so that the tool could be fully integrated into the filter design cycle. This involved CFL enhancement, changing the input and output interface of OTACC, expanding the functions so that OTACC has the capability to realize all necessary *LC* ladder components, expanding OTACC to accept both normalized and non–normalized *LC* ladder input, and adjusting the layout size to fit floor plan proportions. OTACC, beginning with its conceptual foundations, is described in the following chapters.

1.2 THESIS OUTLINE

This thesis is organized as follows:

Chapter 1 : General introduction.

Chapter 2 : An introduction of basic concept about continuous–time filters and design automation.

Chapter 3 : Description about the basic structure and some main functions of OTACC.

Chapter 4 : Software testing and result simulation. Several filters are realized with different OTAs by using OTACC, and their simulation results are discussed in detail.

Chapter 5 : Conclusions and future work.

CHAPTER 2

BACKGROUND AND CONCEPT

2.1 ANALOG IC FILTERS

The trend of integrated circuits is to reduce the dimensions of the transistors and incorporate in a single chip as many building blocks as possible. In the design of high–performance electronic circuits, the application of analog filters is unavoidable. In telecommunication applications, for instance, active filters ranging from a few kiloHertz (kHz) up to several hundred MegaHertz (MHz) are required. Some other communication systems, such as radio and video frequency receivers, demand high–frequency and high–selectivity analog filters as well. There are also other considerable interests in many applications such as read–write channels in magnetic discs and antialiasing filters in digital signal processing systems [3].

2.1.1 Types of Filters

Generally, filters are electrical networks that can provide frequency–weighted transmission and can be typically categorized according to their filtering function. With the magnitude (gain, attenuation) as the primary characteristic, filters can be classified as lowpass, high pass, bandpass, or bandreject networks [1].

The function of the lowpass (**LP**) filter is to pass the low frequencies from DC to some desired cutoff frequency and to attenuate the high frequencies. This kind filter is

specified by its cutoff frequency ω_c, stopband (**SB**) frequency ω_s, DC gain, stopband attenuation, and passband (**PB**) ripple. The passband of the filter is defined as the frequency range $0 \leq \omega \leq \omega_c$, the stopband as the frequency range $\omega \geq \omega_s$, and the transition band (**TB**) as the frequency range $\omega_c \leq \omega \leq \omega_s$. These specifications are shown in Figure 2.1.

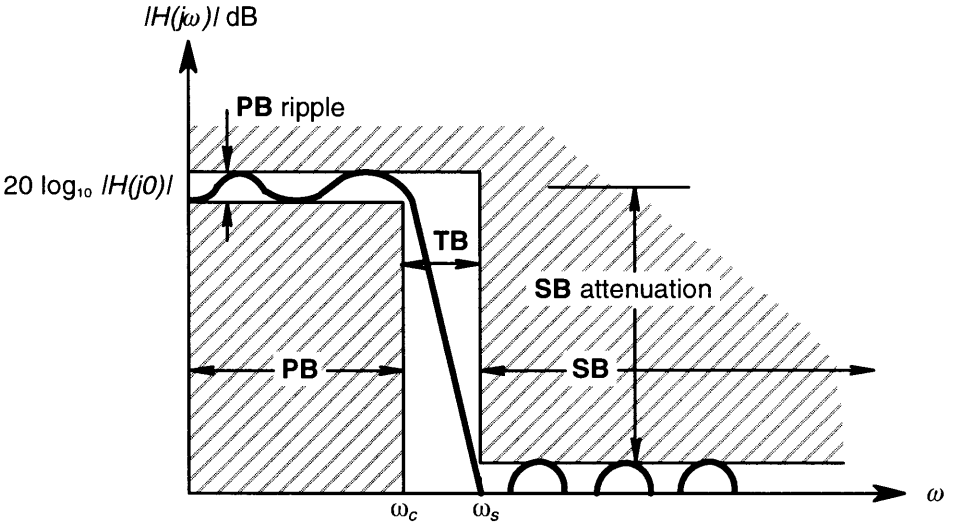


Figure 2.1 Lowpass filter characteristic [1].

Opposite to the lowpass filter, the function of the highpass (**HP**) filter is to pass the high frequencies above the desired cutoff frequency ω_c and to attenuate the low frequencies from DC to the specified stopband frequency ω_s. The highpass filter is specified in the same manner as the lowpass filter. In principle, the passband of the highpass filter extends to $\omega = \infty$. In practice, however, it is limited in active filters by the finite bandwidth of the active devices and by parasitic capacitances. As a result, the gain of the highpass filter will eventually roll off at high frequencies, as illustrated in Figure 2.2.

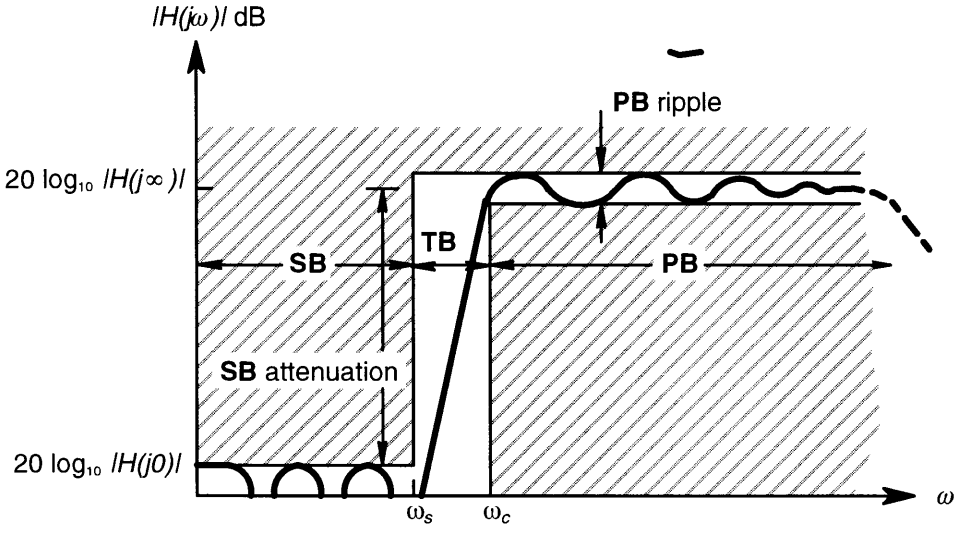


Figure 2.2 Highpass filter characteristic [1].

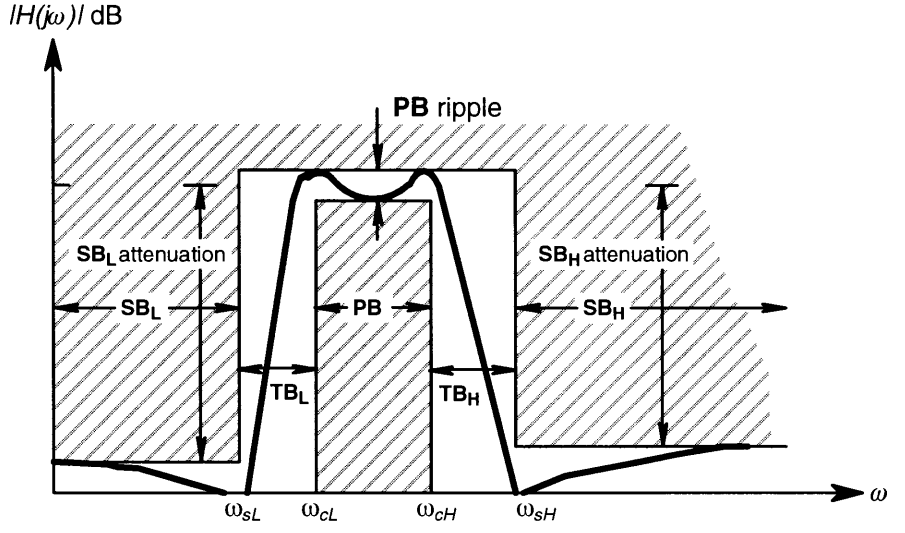


Figure 2.3 Bandpass filter characteristic [1].

The function of the bandpass (**BP**) filter is to pass a finite band of frequencies while attenuating both lower and higher frequencies. This filter has both a lower stopband, $\mathbf{SB}_L$, and an upper stopband, $\mathbf{SB}_H$. Generally, as illustrated in Figure 2.3, the bandpass filter

may not be symmetrical, the attenuation in the lower and upper **SB**s may be different, and the upper and lower transition bands $\mathbf{TB}_L$ and $\mathbf{TB}_H$ may not be the same.

The function of the bandreject (**BR**) filter is to attenuate a finite band of frequencies while passing both lower and higher frequencies. As a result, this kind of filter has both lower and higher frequency passbands, $\mathbf{PB}_L$ and $\mathbf{PB}_H$. The specification, which is similar to the bandpass filter, is shown in Figure 2.4. Just like the highpass filter, in reality the upper passband is limited due to the band–limited active devices and parasitics.

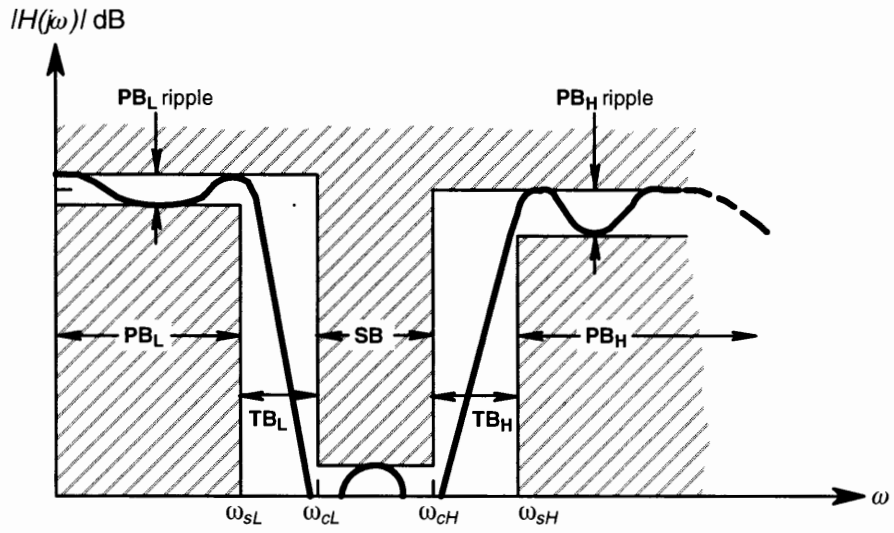


Figure 2.4 Band–rejection filter characteristic [1].

2.1.2 CMOS Continuous–Time Filters

Typically, most of the important characteristics of the continuous–time filter are sensitive to the process parameter tolerances. Nowadays, the available technologies for integrated circuits are Complementary–Metal–Oxide–Semiconductor (CMOS), Bipolar, BICMOS (BIpolar–CMOS), Gallium–Arsenide, and devices based on other III–V substrate. Each one of these technologies has advantages and drawbacks. Actually, CMOS

is the most popular technology due to its high circuit density, low cost and maturity. At this time, the commonly used techniques for the digital signal processing are implemented in CMOS technologies [3].

For the design of high–performance CMOS active filters, there are three main types of continuous–time filters, namely, RC active filters, MOSFET–C filters, and OTA based filters. The accuracy of the active RC filter is very limited. The MOSFET–C filters are basically active RC filters but with the resistors implemented by equivalent CMOS tunable resistors [26]. Due to the use of the resistors, buffered OPAMPs have to be employed. The limited frequency response of a two or more stage OPAMP reduces the application of the filters to low–frequency [3]. "Transconductance amplifiers appear to be the appropriate components because they are easy to implement in monolithic form, can be tuned electronically, and lead to intriguingly simple filter circuitry. Therefore, Operational Transconductance Amplifiers (OTAs) are likely to play an increasingly important role in analog filter design" [2]. The OTA is a functional circuit block that takes the input voltage and generates the output current with the relationship as:

$$I_{out} = g_m \cdot V_{in}$$

where V_{in} is the input voltage, I_{out} is the output current, and g_m is the transconductance value of OTA. It appears appropriate to take the OTA–based filter design strategy in our filter design automation tool.

A minimum OPAMP gain, around 30 dB, is necessary for well behaved OPAMP based filters. Otherwise, the Gain–BandWidth (GBW) effect of the OPAMP degrades the performance of the system. For a single–pole OPAMP, this implies a useful frequency range of the order of GBW/30. This fact makes it extremely difficult to design high–fre-

quency OPAMP–based filters. Because the OTA–based integrators are implemented in open loop, the functional frequency range of these circuits is limited only by the OTA gain–bandwidth product [3].

The main topic of this research project is automating the layout of CMOS continuous–time filters. For the OTA based filters, the Operational Transconductance Amplifier–Capacitor (OTA–C) filters have already been reported with frequency ranges from a few kHz up to hundreds MHz. The basic building blocks of OTA–C filters are the Operational Transconductance Amplifier (OTA) and integrated capacitors. A major advantage of the OTA–based continuous–time filters is their extremely large frequency range [15][16][17][18]. Actually, OTA–based filters with resonant frequencies in the range of several hundred MHz have been reported [19][20]. A comparison of the useful frequency range for both OPAMP and OTA–based circuits is shown in Figure 2.5.

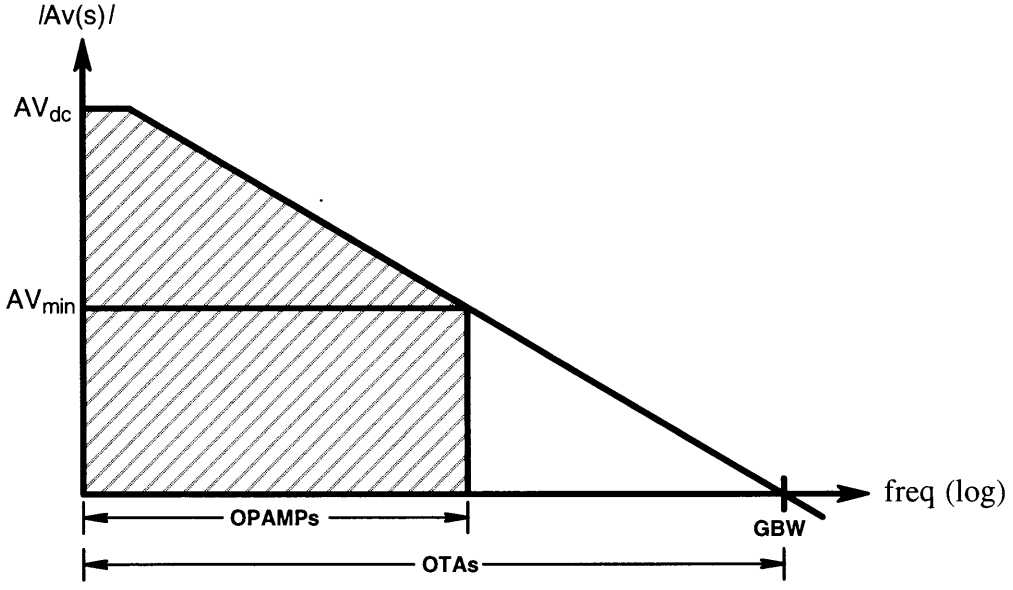


Figure 2.5 Frequency response for a typical single–pole OPAMP/OTA [3].

The **OTACC** (**O**perational **T**ransconductance **A**mplifier–**C**apacitance **C**ompiler), being the main topic of this thesis, focus on automating the design cycle of CMOS continuous–time filters by the simulation of *LC* ladder prototypes.

2.2 FILTER REALIZATION

2.2.1 LC Ladder

As an example, a fourth–order elliptic *LC* lowpass filter, with a 10MHz passband and nearly 50 dB of attenuation in the stopband, is shown in Figure 2.6. Its magnitude plot (simulated with ideal elements) is shown in Figure 2.7.

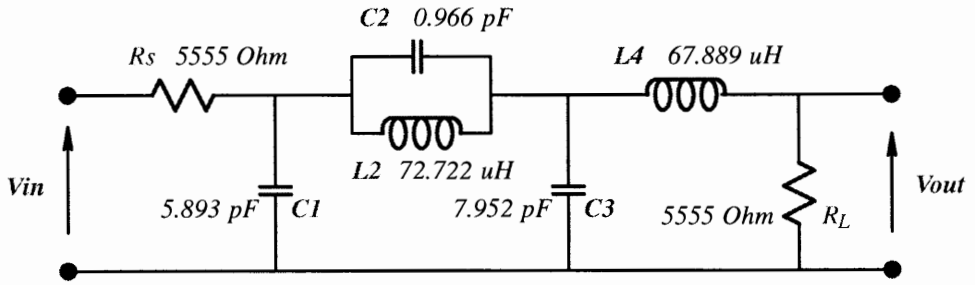


Figure 2.6 Fourth–Order Elliptic *LC* Lowpass Ladder.

The active *LC* ladder prototype is chosen for this project primarily in response to the filter design automation concerns [4]. For a desired transfer function, a filter is realized by the active simulation of an *LC* ladder topology derived from its input and output impedance. In addition, the *LC* ladder prototype was chosen because of its unique characteristic of low sensitivity to component variations and because a circuit capacitor is generally present at most of the circuit nodes in the original *LC* prototype. So the original value of the capacitance can be adjusted to include the value of the parasitic capacitance. That is:

$$C_{desired} = C_{circuit} + C_{parasitic}$$

If enough is known about the parasitics, we can combine the circuit capacitance with the parasitic capacitance associated with corresponding circuit node, and therefore, significantly limit the influence of the parasitic capacitance. The combination of the circuit capacitance with the parasitic capacitance will be referred to as *predistortion.*

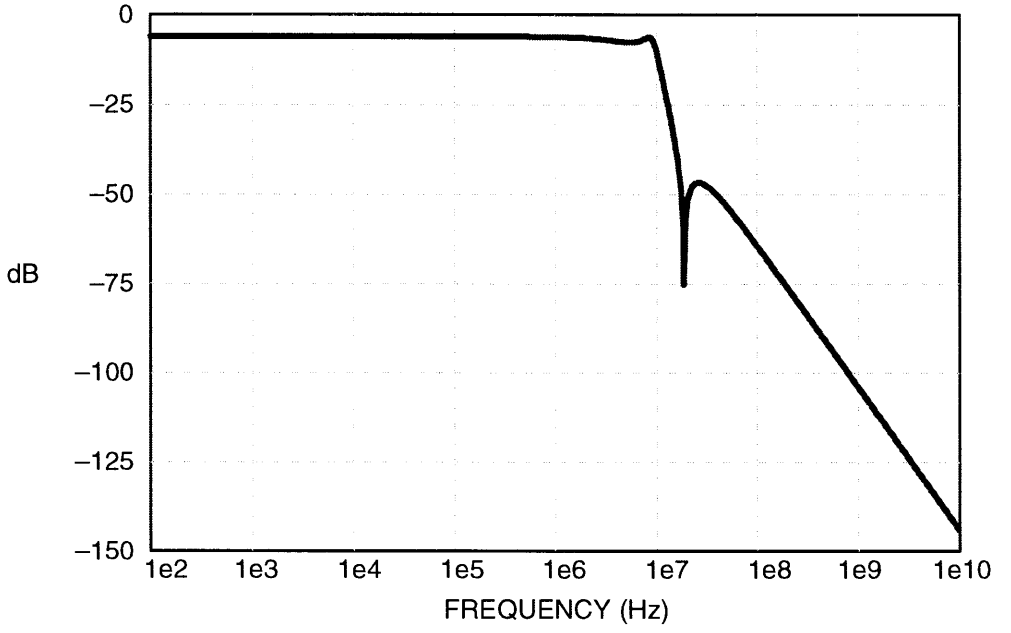


Figure 2.7 Magnitude plot of Fourth–Order Elliptic *LC* Lowpass Ladder shown in Figure 2.6.

2.2.2 Normalization

Like most electrical networks, the *LC* ladder is usually normalized with *Impedance–Level Normalization* and *Frequency Normalization.* Normalization causes no loss in generality and is performed only for simplicity in numerical computations, especially in hand calculations, because it can minimize the effects of round–off errors [1]. Frequency normalization is simply the frequency scaling by dividing the frequency variable by a conve-

niently chosen normalizing frequency Ω_0. Similarly, impedance–level normalization is performed by dividing all impedances in the circuit by a normalizing resistance R_0. The normalized element values are, therefore,

$$R_n = \frac{R}{R_0} \qquad L_n = L\frac{\Omega_0}{R_0} \qquad C_n = C\Omega_0 R_0$$

The schematic of the normalized LC ladder corresponding to Figure 2.6 is shown in Figure 2.8.

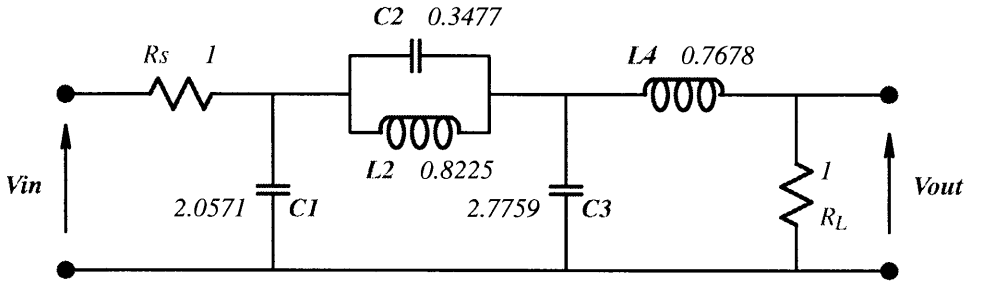


Figure 2.8 Normalized Fourth–Order Elliptic LC Lowpass Ladder.

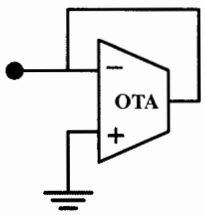
In our case, identical transconductance cells (with same g_m value), which converts the input voltage to output current , are used to realize the resistors and simulate inductors. So the actual denormalized physical parameters R, L, and C are,

$$R = \frac{1}{g_m} \qquad L = \frac{L_n}{g_m\Omega_0} \qquad C = C_n\frac{g_m}{\Omega_0}$$

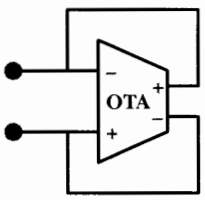
Where the Ω_0 is the cutoff frequency of the filter. The gm parameter, which is necessary to calculate the circuit R, L, and C in OTACC, can be obtained from the technology file which is associated with the OTA standard cells in the library. The format and contents of the technology file used by OTACC is shown in Appendix D.

2.2.3 Element Substitution

To achieve the integrated implementations of active filters, the R and L elements in the LC ladder must be replaced by active simulations with only OTAs and capacitors. These two elements are sufficient in general and lead to circuits that are easy to integrate in a typical IC process. The g_m simulation of the grounded resistor and grounded inductor are shown in Figure 2.9 and Figure 2.10.



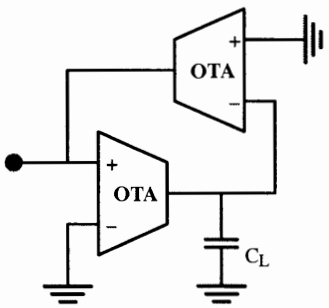
$$R = \frac{1}{g_m}$$



Single–Ended OTA

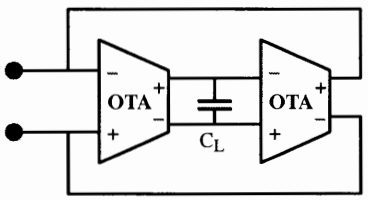
Differential OTA

Figure 2.9 Simulation for grounded resistor



$$L = \frac{C_L}{g_m{}^2}$$

For identical OTAs



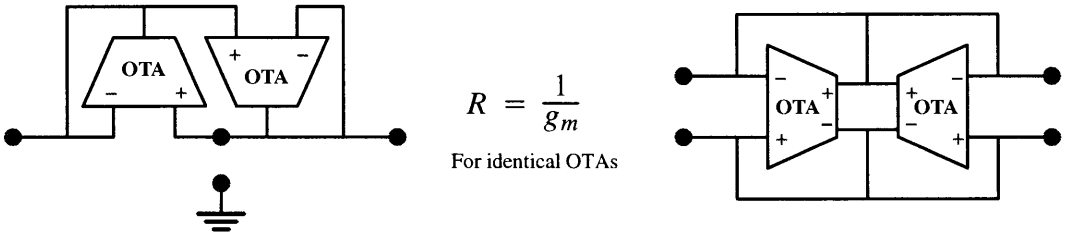
Single–Ended OTA

Differential OTA

Figure 2.10 Simulation for grounded inductor

When we need a floating resistor or a floating inductor instead, we must "lift the circuit off ground" and devise a method for the current to flow out of the second terminal.

As shown in Figure 2.11 and Figure 2.12, extra OTAs are needed for simulating the floating resistor and floating inductor.



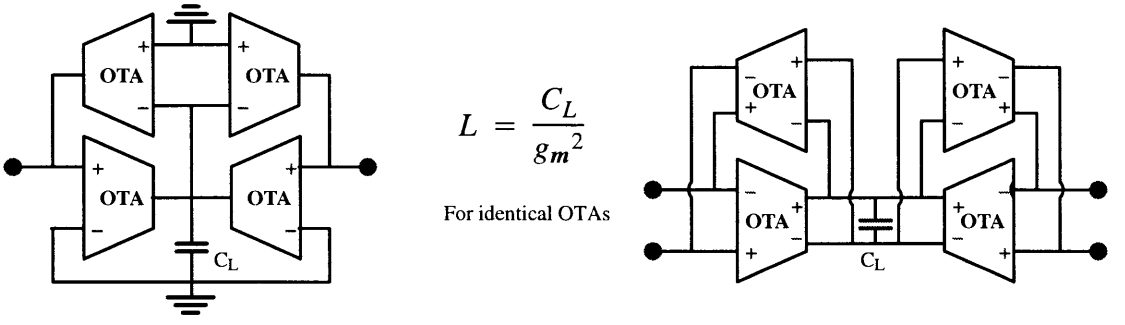
$$R = \frac{1}{g_m}$$

For identical OTAs

Single–Ended OTA

Differential OTA

Figure 2.11 Simulation for floating resistor



$$L = \frac{C_L}{g_m{}^2}$$

For identical OTAs

Single–Ended OTA

Differential OTA

Figure 2.12 Simulation for floating inductor

2.3 LAYOUT AUTOMATION

There are a several methods to automate the layout process of digital and analog circuits. However, the way that parasitics, specifically parasitic capacitance and resistance, are handled is vital to the success of any analog design automation method.

The first method is polygon generation. In this method, nearly all the layout is generated by the software. Each transistor, for example, may be independently sized and routed

to the next transistor for the entire design. This method has many variations that can be unforgiving in large analog subsystems. As the complexity of the system increases, the errors caused by parasitic capacitance and resistance begin to accumulate exponentially. However, there have been some successes with the automatic generation of small analog components [21][22][23] and digital systems [9][29].

The second method is module generation [22][24][25]. With this method, a library of basic building blocks containing modules such as logic devices, current mirrors, differential pairs, etc. is available. These modules are pre–assembled in such a way as to minimize internal and external layout problems. The methodology is general, but since the parasitic capacitances can still wreck havoc with analog configurations, considerable designer intervention is required for complex analog subsystems [24][25]. The work is ongoing with this method and may one day be the preferred methodology if the parasitic problem can be solved.

The third method is tiling. This takes the idea of modules a step further in that the modules (tiles) are more complex and are configured such that the signal routing may also be tiled. A library of pre–designed and characterized tiles are available as basic building blocks. Since routing is included in the tiles, the routing operation is static. Routing occurs simply by abutting two or more tiles together [6] as shown in Figure 2.13.

The disadvantage to this approach is library maintenance. If additional circuitry is added into the library, often a set of support tiles will need to be created also. For the digital logic application, there are no specialized routing tiles required. However for something more complex such as a tunable operational transconductance amplifier, the key element of the active filter in this thesis, additional support tiles are needed that match its layout. As an example shown in Figure 2.13, an additional route tile would be required

to support the OTA's layout to realize the multiplication by **–1** using differential OTAs. So, not only would an additional tile need to be created, but this tile might need to be linked to a specific OTA in order to assure proper wire location.

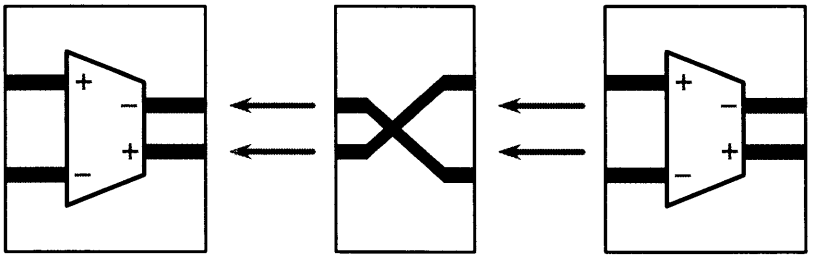


Figure 2.13 Example requiring a route tile [7].

Tiling is the basic approach that OTACC will take. Although less general than module generation, it uses a standard cell library and predistortion to provide better accuracy that would otherwise be unattainable. This is a reasonable tradeoff in seeking for better analog automation tools. To reduce the number of tiles that are associated with each OTA, a polygon generation method is used in this project to generate some route tiles from a set of basic route tiles. This is really a hybrid of module, polygon generation, and tiling approach. Only the route tiles that really depend on the geometry character of the OTA have to be created when a new OTA cell is added to the library. The details of how the tool works and how to develop the support library will be covered in a later chapter.

2.4 FILTER DESIGN AUTOMATION

Generally, there are four necessary procedures in the filter design cycle. They are filter approximation, layout, simulation, and verification. To automate the filter design cycle, user friendly interface is necessary to integrate the procedures to complete the filter design automation tool, as depicted in Figure 2.14.

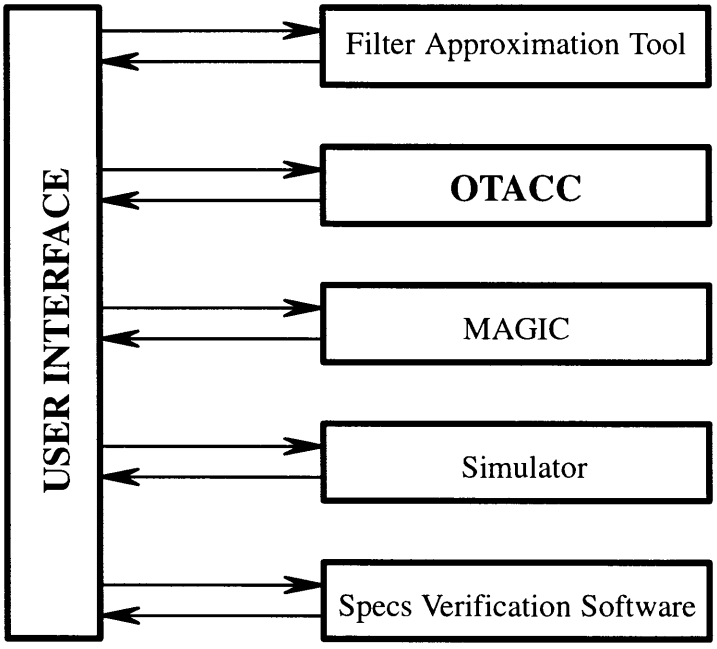


Figure 2.14 Diagram of automatic filter design cycle.

In Figure 2.14, the user interface is a window based, graphical user interface. It is a separate project related to this research subject [28]. The filter approximation could be done by some filter approximation tool, such as FiltorX which is developed and distributed by Department of Electrical Engineering, University of Toronto [12]. The layout of the filter will be generated in MAGIC [10] format, and the circuit netlist can be extracted for simulation. The simulator could be some general purpose circuit simulation programs, such as SPICE. To complete the picture of the filter design automation, the verification software is a planned separate project.

CHAPTER 3

OTACC

OTACC is an automatic layout tool written in C intended to realize the design automation for the class of OTA–C continuous–time filters. It uses a set of library functions known as CFL library that interface to the MAGIC layout . It takes MAGIC cells as input and generates the layout output in MAGIC format. OTACC, beginning with its conceptual foundations, is described in the following sections.

3.1 CFL

3.1.1 What is CFL ?

CFL is a C function library to be used for the construction of VLSI circuit layouts. It is organized algebraically in that there is a data type called SYMBOL, a set of primitive operands of this type, and a set of operators that generate new SYMBOLs by forming combinations of existing SYMBOLs.

All inputs for the calculations required by the CFL operators are retrieved from the descriptions of the borders of the SYMBOLs. The information in the border descriptions includes the bounding box and lists of coordinates of the points where each kind of material in the SYMBOL makes contact with the bounding box, such as the example shown in Figure 3.1. When a border description is available for a particular SYMBOL, CFL will

not access the rest of the geometry of SYMBOL [11], which provides considerable performance gains.

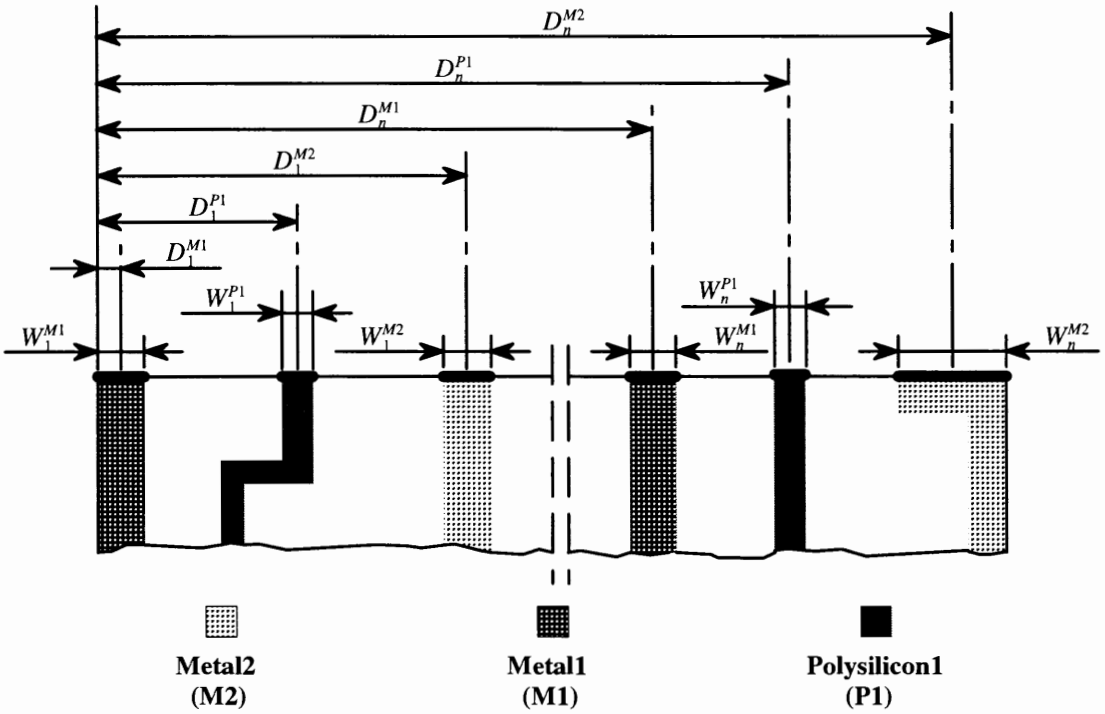


Figure 3.1 Different layers on the top border of the SYMBOL.

The system generates a border description from the geometry whenever the need arises and saves library symbols on disk. In this way, modules that have a large number of rectangles may be accessed from the library without the need of reading all the files associated with their sub–modules. This capability allows CFL to assemble large blocks of circuitry extremely fast. Additionally, CFL also provides automatic hierarchy compression when symbols are written to disk so that only those symbols that represent meaningful layout need be saved [11].

With a complete set of placement and routing primitives, this tiling language is simple and flexible. The database is stored in the same format as the popular public domain

VLSI CAD package, MAGIC [10], which has the capability to support large design layouts and includes an extractor that is used to create simulation netlist

3.1.2 CFL Enhancement

To insure the widest variety of chip assembly options, CFL includes approximately 70 operators for juxtaposing, transforming, and replicating hierarchies of one or more symbols. Unfortunately, some of the original routing functions of CFL were not flexible enough to be used by our application. The major problems in the original CFL were:

1. Most routing functions could only place the routing wires with constant wire width.
2. Mask layers on the border could not be routed with a contact's original layers. For example, the METAL2 contact on the border of a symbol could not be routed with METAL1 or METAL2

Because of the limited routing functions within CFL, CFL routing facilities had to be modified. The new version of CFL is compatible with the old CFL–based layout program. That is, any C program that used the original CFL, can still use the new version CFL library without modification and with no differences in the final layout. The new features of the enhanced CFL are:

1. All routing functions have the ability to run the routing with the wire width of its original width at the border.
2. The border descriptor, as argument to some of the routers, has the capability to contain not only a given mask layer on the border, but also the contact layers related to it.
3. New user functions permit the user to exclude or include certain mask layers on the border from the border descriptor.
4. A new user function permits the user to switch the active layer of a point on the border. This means that CFL has the ability to complete the point–to–point routing with the layer that is associated with the source point instead of the destination point.

Features 2, 3 and 4 can be used only when the new format technology file, which contains important process oriented information, is supplied. Use of the new operators, without the new format technology file, will cause the program to exit with a warning that the new technology file is required.

More details about the enhanced CFL functions and their applications are listed in Appendix A.

3.2 BASIC STRUCTURE OF OTACC

The communication structure of the OTACC module is based on a stack. All operations obtain their input from the stack, and then push the results back onto the stack. Each stack element is a structure of pointers to layout symbols and miscellaneous information for communication between functions. A stack is preferred because to pass information back and forth between the routines, a large number of global variables would be required. Also, using a stack machine allows each tile to be read from the disk only once, with the tiles remaining in the memory for the duration of the whole layout procedure. This improves the efficiency of OTACC since multiple reads of the disk for each tile are avoided. An idea of the number of global variables that would have been required if a stack machine were not used can be obtained by viewing Table 3.1. For more detail about the stack structure definition please see Appendix B.

OTACC Stack Structure	Type of Structure Elements				
	SYMBOL Structure	Integer Array	Enumerator	Integer	Floating
Number	21	2	3	8	10

Table 3.1 Number of elements in the OTACC stack structure.

3.3 INPUT AND OUTPUT INTERFACE OF OTACC

3.3.1 FiltorX Interface

The filter design cycle consists of taking a filter specification and generating the transfer function, possibly using a classical methodology. Next, the transfer function is approximated with real circuit elements and finally implemented. The University of Toronto has created a design tool called FiltorX to assist in the approximation and realization process [12]. OTACC's grammar set was designed to be compatible with the output of FiltorX so that the two tools could be combined into one, allowing the approximation, realization, and implementation phase of the filter design cycle to be assisted or automated by the software.

For a given transfer function, one possible output of FiltorX is in terms of the elements that make up an *LC* ladder filter. Reading the schematic showing in Figure 3.2 from left to right yields the FiltorX description shown in Table 3.2. Where the *filter_name* will be the file name used for final layout of the filter.

By matching the input grammar of OTACC to the output of FiltorX, we create a link that is necessary to automate the most computationally intensive parts of the filter design cycle.

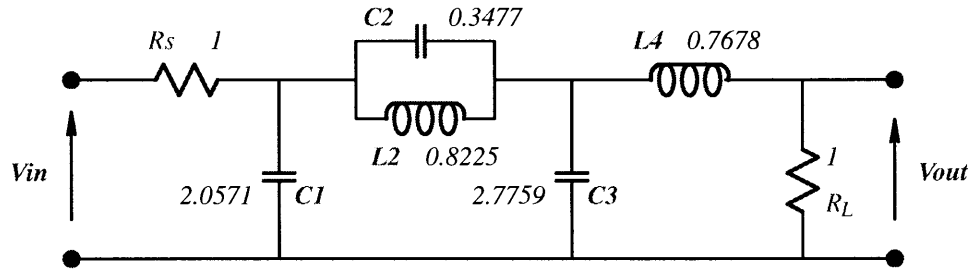


Figure 3.2 Example of Fourth Order Elliptic Lowpass Ladder.

ladder filter_name (
seriesR 1.0;
shuntC 2.0571;
seriesLC 0.8225, 0.3477;
shuntC 2.7759;
seriesL 0.7678;
shuntG 1.0;
)

Table 3.2 OTACC input from FiltorX

3.3.2 Command Line and Options

To increase the flexibility, several command line switches are supported by OTACC. The designer can control the geometry of the layout and easily update some of the required parameters to permit fine adjustments to be made during filter design. For example, most OTAs are designed with tuning nodes to adjust their transconductance value and the designer can predict the tuning effect on the transconductance by simply changing the transconductance value at the command line.

A list of command options for OTACC are outlined in Appendix C.

3.3.3 OTA–Technology File Scanning

OTACC caches all of the required technology parameters from the OTA–technology file that is associated with the OTA cells in the library. The tile file contains the important parameters of the OTA and important process parameters. The contents and format of the OTA–technology file are described in Appendix D.

Currently, the only required parameters for executing OTACC, are the name of the OTA cell, transconductance value and cut off frequency of OTA, and parasitic capacitance values. The other parameters have been reserved for future development, and they can be omitted for current applications. Comment lines are allowed in the OTA–technology file to make it readable.

3.3.4 Error Message Coding

To communicate with a top–level program such as a Graphic User Interface (GUI) or the designer, OTACC returns an encoded integer number for both normal exit and abnormal exit. For the non–fatal errors, a binary bitwise encoding method is used to encode the warning messages. This gives the designer or GUI a way to trace the program and a hint for debugging the problem when an error is detected during the OTACC execution.

The entire set of the error codes are defined in Appendix E.

3.4 SYMBOLIC LAYOUT AND TILE GENERATION

As described in Chapter 2, OTACC uses a hybrid of module, polygon generation and tiling methodology. Generally, there are three types of tiles required:

1. Transconductance tile.
2. Unit capacitor tile.
3. A set of route tiles.

The route tiles act as an interface between the transconductance and an array of capacitor tiles. For a double polysilicon, double metal CMOS process, an example of how the three types of tiles fit together is shown in Figure 3.3.

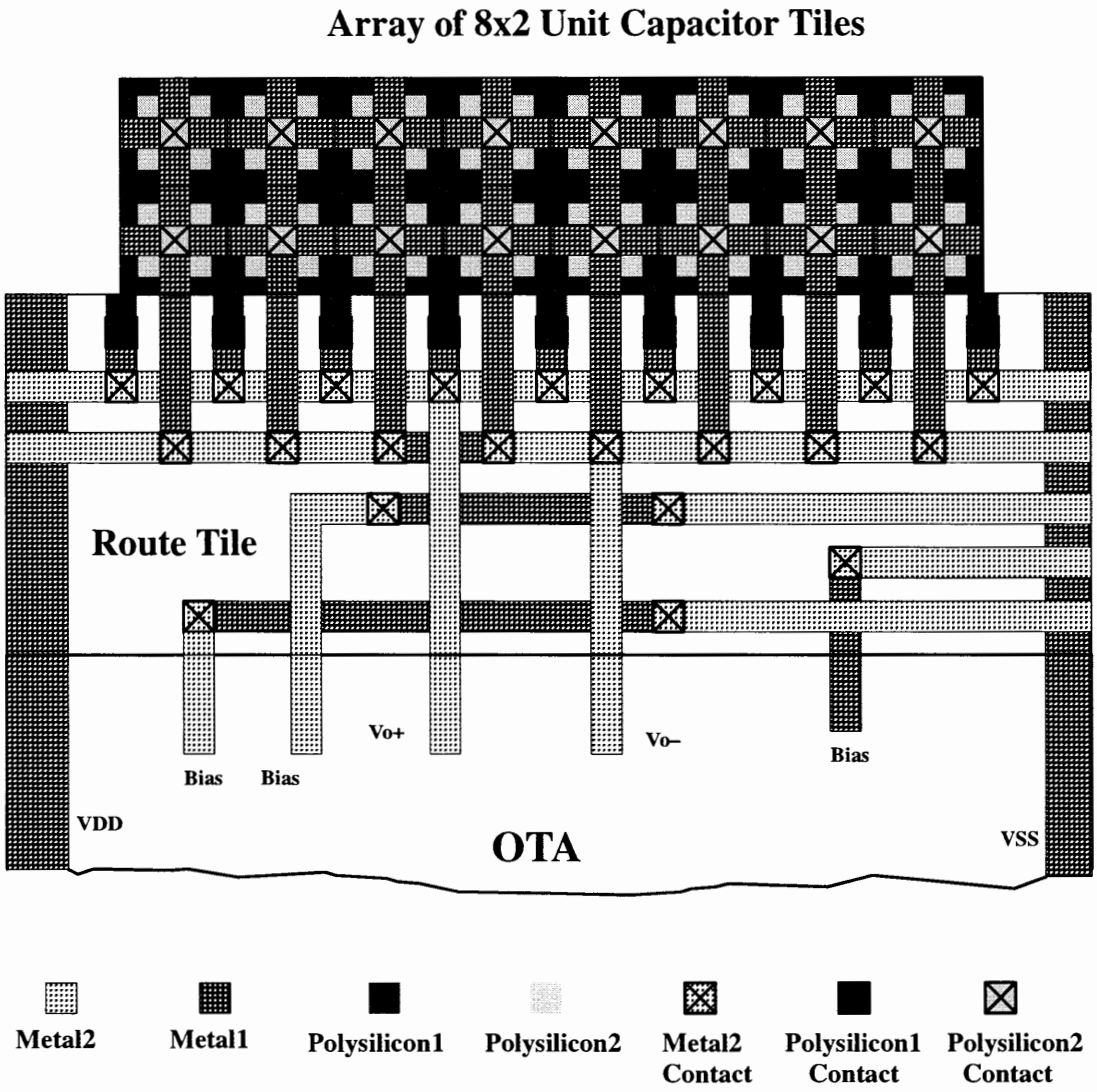


Figure 3.3 An example of the tiles fitting together.

By "routing" the cells with pre–designed tiles, the desired amount of node capacitance can be predistorted according to the amount of parasitic capacitance as described in the OTA–technology file. This method of layout generation greatly reduces the unpredictable parasitic capacitance problem that plagues automatic synthesis of VLSI analog circuitry. The disadvantage of routing with tiles is that the synthesis software can be very specialized, and the standard cell library maintenance becomes more difficult. For the

current version OTACC, fourteen route tiles are needed for *LC* ladder synthesis. i.e. beside the capacitor and OTA itself, fourteen route tiles need to be prepared for the filter synthesis.

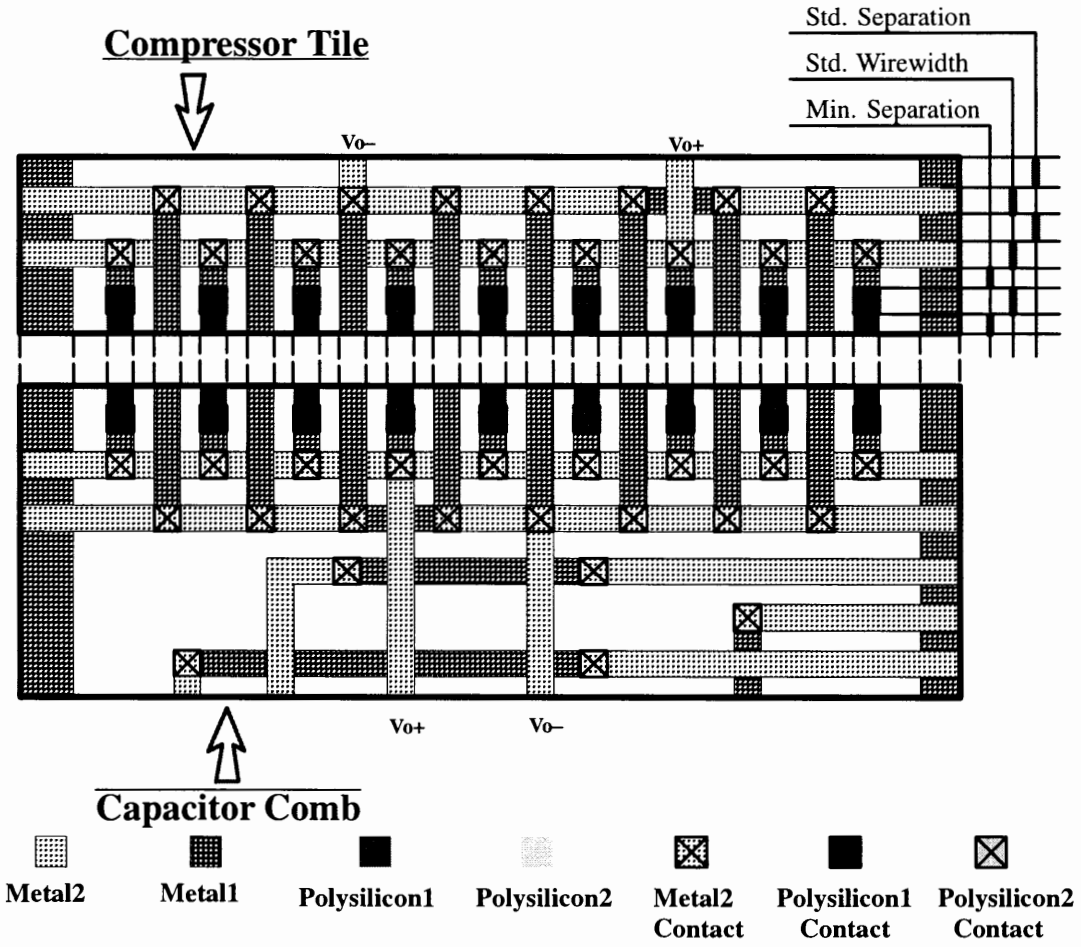


Figure 3.4 Example for route tile generation.

To limit the difficulty of the library maintenance, a second version OTACC includes functions to generate some of the route tiles based upon a fixed set of tiles. Software examines the geometry characteristic of the existing tiles and then generates the desired route tile to guarantee they are perfectly matched for later assembly. An example of route tile generation, in which part of the compressor tile is generated from the existing capaci-

tor comb, is shown in Figure 3.4. The compressor tile is important of the final assembly and will be fully described in Chapter 3.6.3.

The contents of the library for a specific OTA will include the MAGIC files of the OTA, unit capacitor tile, and 6 basic route tiles. All other necessary route tiles will be generated by the software automatically. The relationships between the different route tiles are shown in Table 3.3. Some basic route tiles need to be pre–designed, because they are dependent on the geometry of a specific OTA.

Name	Level	Derived From	Function in OTACC	Application	Status
combinP	Basic	Pre–designed		In+ Comb for capacitor array	Existing
combinN	Basic	Pre–designed		In– Comb for capacitor array	Existing
comboutP	Basic	Pre–designed		Out+ Comb for capacitor array	Existing
comboutN	Basic	Pre–designed		Out– Comb for capacitor array	Existing
dcrib	Basic	Pre–designed		DC bias route tile	Existing
ldrib	Basic	Pre–designed		Special DC bias route tile	Existing
dcCenter	Derivative	dcrib	mkCframe	Additional DC bias route tile	Existing
dcEdge	Derivative	unitC, comboutP	mkCframe	DC signal connection stub	Existing
zCenter	Derivative	UnitC	mkZCframe	"zero" capacitor array route tile	Existing
zEdge	Derivative	UnitC	mkZCframe	"zero" capacitor connection stub	Existing
compressor	Derivative	dcrib, comboutP	mkcompressor	Ac signal compressor tile	New
acisolator	Derivative	dcrib, comboutP	mkacisolator	AC signal isolator tile	New
floatcombm	Derivative	dcrib, comboutP	mkfloatcomb	Metal1 comb for float capacitors	New
floatcombp	Derivative	dcrib, comboutP	mkfloatcomb	Poly1 comb for float capacitors	New

Table 3.3 Route tiles for OTACC

To easily identify the route tiles and improve the maintenance of the library, all route tiles use the name of the OTA cell as the prefix of their name. For example, the MAGIC file name of the compressor tile for a transconductance cell named *OTAD* will be *OTAD.compressor.mag*.

3.5 OPERATORS

To build a filter, there are five main stack operators in OTACC to synthesis the *LC* ladder. They are: *do_fs()*, *do_zero()*, *do_add()*, *do_inv()* and *do_separation()*. There are two additional stack operators, *do_source()* and *do_load()*, for the input source and the load resistor block of the *LC* ladder. All stack operators are stand alone, and are coordinated by the state of the program stack. For example, the *do_add()* operator checks for and then pops two elements from the stack, properly assembles them to get a larger component, and then the result is pushed back to stack. The order of the calls is dependent on the input data stream. Each of these operators are discussed in following paragraphs.

3.5.1 Software Implementation and Circuit Synthesis

As shown in Table 3.4, each stack operator has a specific task to perform during *LC* ladder filter synthesis. Combining these operators with the filter descriptor, OTACC can realize an *LC* ladder filter of any order and transmission characteristics.

Stack Operators	Input	Stack Operation	Stack Status	Objective
do_inv()	Nothing	POP 1, PUSH 1	Stack Unchanged	Add an Inversion
do_fs(double C)	Capacitance Value	POP 1, PUSH 2	Stack Increase by 1	Capacitor Array
do_add()	Nothing	POP 2, PUSH 1	Stack Decrease by 1	Addition Operation
do_zero(double C)	Capacitance Value	POP 1, PUSH 2	Stack increase by 1	Transmission Zero Capacitor Array
do_separation()	Nothing	POP 1, PUSH 2	Stack increase by 1	AC Signal Isolator
do_source()	Nothing	POP 1, PUSH 2	Stack increase by 1	Source Transformation
do_load()	Nothing	POP 1, PUSH 2	Stack increase by 1	Grounded Load Resistor

Table 3.4 Stack operators in OTACC.

The stack element stores the tile symbols and miscellaneous information such as the parasitic capacitance values contributed by the OTA inputs and outputs, the route tiles and the capacitor tile. Other than modifying the stack, the stack operators have no output or return message when operating normally. Error messages are output when some error has occurred. For example, if a operator is used inappropriately or the stack is corrupted.

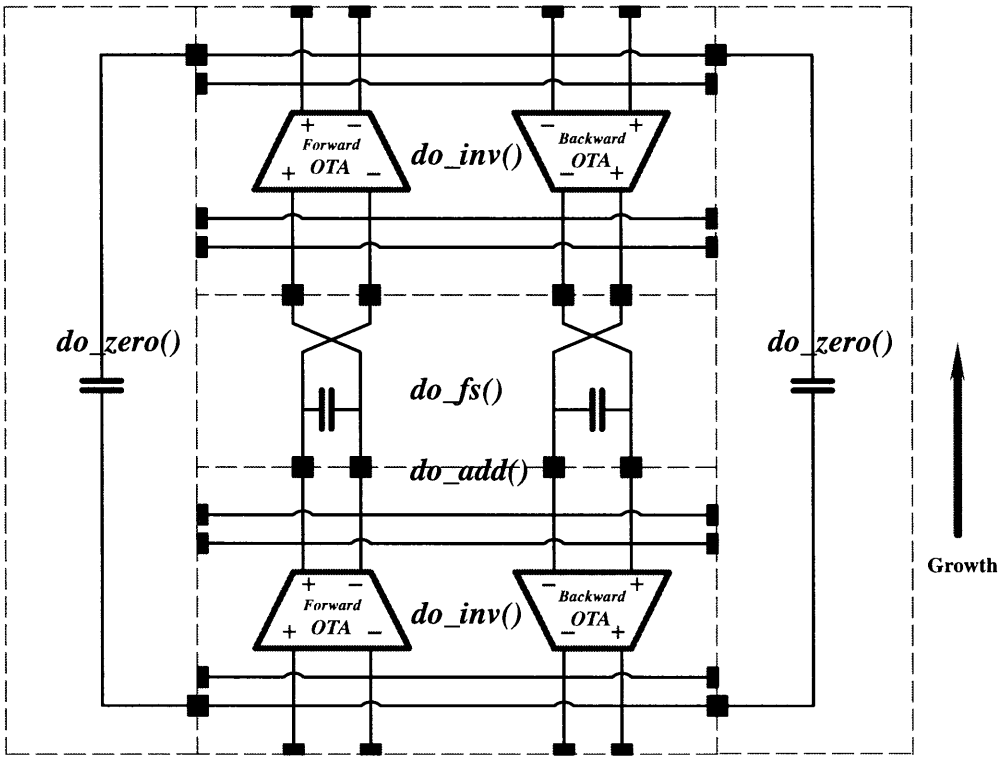


Figure 3.5 Example for the synthesis of series *LC* block.

An example for the synthesis of series *LC* block is shown in Figure 3.5. First the *do_inv()* operator puts a so–called forward OTA and backward OTA together, which is the core building block for the simulation of the inductance element as described in section 2.2.3. Next the *do_fs()* operator generates the capacitor array that is needed to simulate a specific value of inductance. Finally *do_add()* operator attaches the fragment onto the main structure of the layout. To complete a floating inductor, another OTA pair would be attached by the *do_inv()* operator. If a series capacitor array is needed, it would be gen-

erated by *do_zero()* operator and stored in the stack with all the necessary information attached to the capacitor array for the final assembly.

Other elements are constructed by a similar combination of these operators. Generally, by using these operators, any functional block for the *LC* ladder filer could be constructed.

In the *do_fs()* and *do_zero()* operators, CFL utilities are used to construct the layout from OTA tiles, unit capacitor tiles, and route tiles. The parasitic capacitance of inputs and outputs of the OTA and the route tiles are known before the capacitor array is generated. For a desired capacitance, C, the number of capacitor tiles to be tiled for the circuit capacitor array is computed using:

$$N = \frac{C - \left(C_{I/O} + C_{route}\right)}{\left(C_{unit} + C_{unitP}\right)} \tag{3–1}$$

where N is the number of capacitor tiles required to make the circuit capacitor array. $C_{I/O}$ is the sum of the OTA inputs and outputs parasitic capacitance. C_{route} is the parasitic capacitance of the route tiles. C_{unit} is the capacitance value of a single capacitor tile and C_{unitP} is the parasitic capacitance of a single capacitor tile.

A capacitor array is constructed as a two dimensional array of the capacitor tiles, as shown in Figure 3.6. The number of capacitor tiles per row in the array is determined by the width of OTA cells and any required DC routing space. After the rows of unit capacitors are created, if the last row is incomplete, the remaining spaces are filled by "*nullC*" capacitor tiles as a place holders (i.e. insignificant contribution to the total capacitor array). One polarity of the power supply passes through the "*DC edge*" on each side. Another polarity of the power supply and the DC bias routing pass through the "*DC center*". Therefore the AC routing and the DC routing can be completed in single steps.

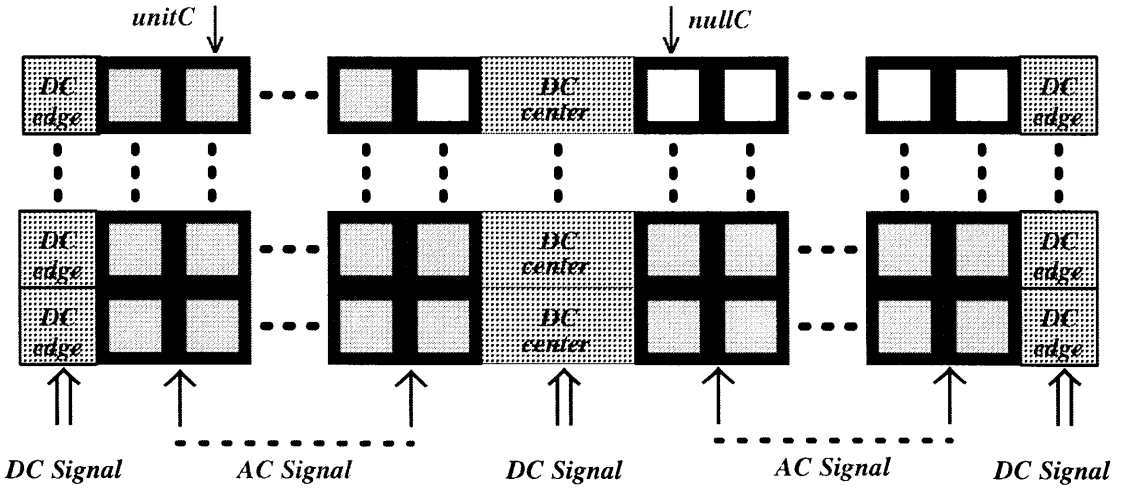


Figure 3.6 Capacitor array generation

If a series *LC* block is required in the filter, which is a key element for realizing a finite transmission zero in filter transfer function, the transmission zero capacitor array that is in parallel with the series *L* block is generated in the vertical direction. The number of the capacitor tiles per column in a transmission zero capacitor array is set by the height of OTA cells.

3.5.2 Data Communication

As noted earlier, the data communication in OTACC uses two stack operations: *push()* and *pop()*. The functions obtain their inputs from the stack element, and return their results to the stack.

Generally, the operators outlined will affect the stack, as shown in Table 3.4, in three ways.

1. Adding a new element on the top of the stack. The stack length is increased by 1.
2. Combine the first and second element's SYMBOLs. The stack length is decreased by 1.

3. Merge a new structure with the first element on the stack. The stack length is unchanged.

3.6 FLOOR PLANNING

In its simplest form, the layout generated by OTACC grows in one direction along the "backbone" of OTA cells and AC, DC interconnection. The width of the backbone is set primarily by the sum of the widths of the transconductance cells and the DC channel route. The maximum width of the final filter includes the width of any transmission zero capacitor array attached to the side of the backbone. Although these capacitor arrays may vary in size, they typically do not exceed 150 microns each, or a total of 300 microns in a 2 micron technology.

The basic building blocks of the filter layout consist of so–called forward transconductance, backward transconductance, and DC route tiles in the configuration shown in Figure 3.7 and Figure 3.8. The layout grows vertically because these meta–elements are stacked with capacitor arrays sandwiched in between.

3.6.1 Series *LC* Block

As shown in Figure 3.7, the inductor L is simulated by four OTA cells and the capacitor array C_L. The transmission zero capacitor arrays are attached on both sides of the backbone for a fully balanced design (there is only signal transmission zero capacitor array on one side of the backbone for a single–ended design). The separation between the transmission zero capacitor array and the backbone has been verified to limit the cross–talk between the capacitor array and the backbone [9].

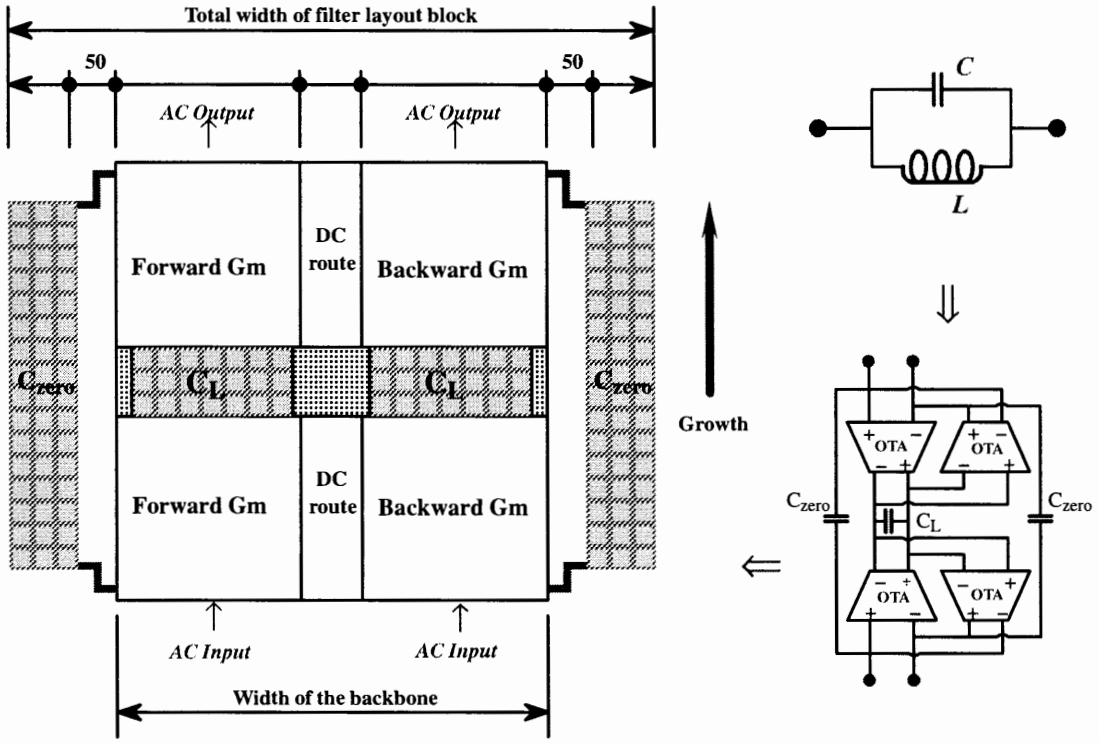


Figure 3.7 Basic building block of series *LC*.

For a series *L* branch, the C_{zero} capacitor array is not required. The rest of the structure is the same as the series *LC* branch.

3.6.2 Shunt *LC* Block

As shown in Figure 3.8, all the components of the shunt *LC* branch remain within the backbone. Because of that, each shunt *L* branch must have a grounded end for the AC signal. An AC signal isolator is applied to reduce the cross–talk between the upper and lower block. The structure of the AC signal isolator also includes some vertical DC connection wires to carry the power and bias supply along the backbone, and a horizontal DC signal (AC ground) wire to isolate the AC signal. The AC signal along the backbone is

maintained by the wire jumpers which are generated automatically by OTACC on the sides of the backbone.

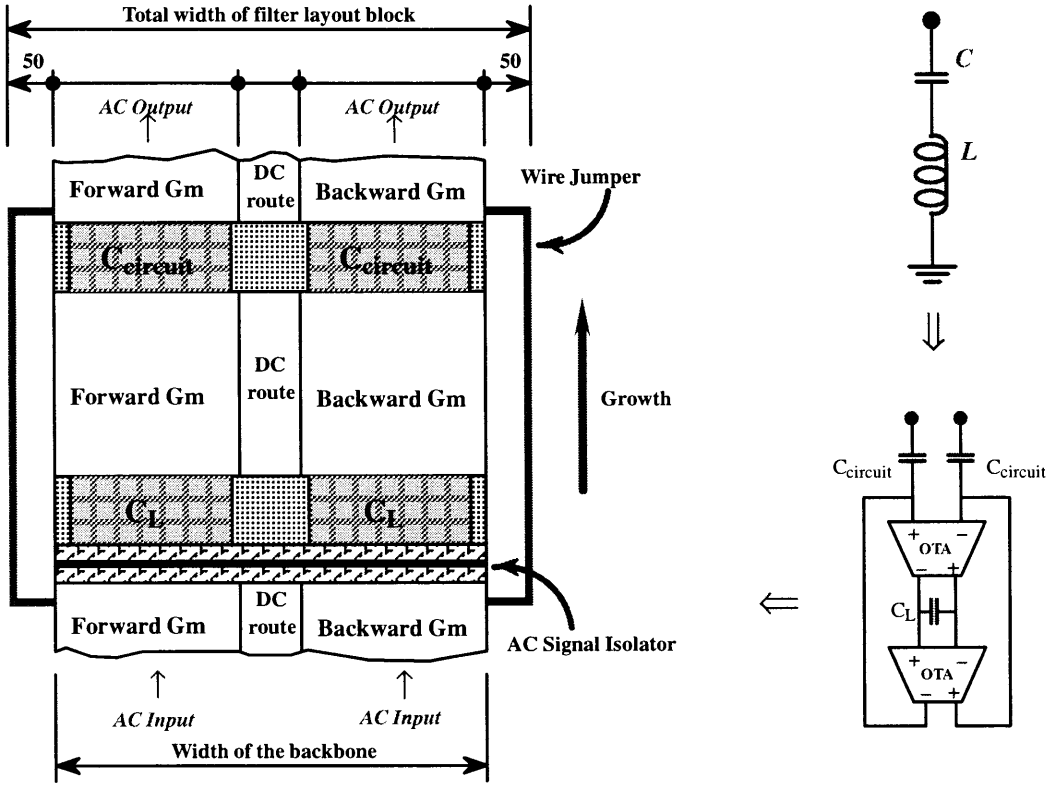


Figure 3.8 Basic building block of shunt *LC*.

For a shunt *L* branch, the series $C_{circuit}$ capacitor arrays is not required. The rest of the structure is the same as the shunt *LC* branch.

3.6.3 Folded Layout

Unfortunately, a tall, thin layout can be a large waste of silicon and difficult to fit into a larger design. It is generally desirable to have the layout to be roughly square, i.e., the width and height should grow at about the same rate. In order to utilize the space better, a height limit can be set by the designer to control the vertical growth of the layout to pro-

duce a zigzag layout. The single–line layout and zigzag layout are compared in Figure 3.9. As illustrated, each layout has approximately the same area.

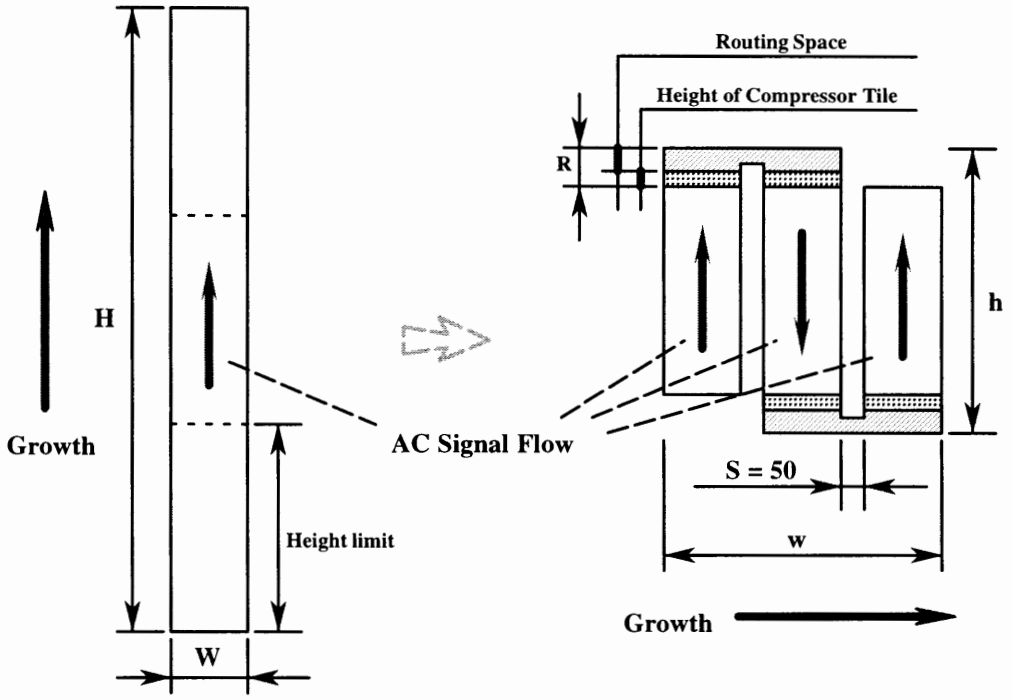


Figure 3.9 Converting a linear growth filter to a zigzag layout.

Note that layout of an active filter IC may include one or more transmission zeros or shunt *LC* branches which are not shown in Figure 3.9. The arrows in the figure indicate the direction of growth and the flow of the AC signal path.

As shown in Figure 3.9, the filter must complete a U–turn in the AC signal path when the height limit is reached. The height limit is set by the user through the command line. If the user does not specify the height limit, the software will attempt to identify the best square layout. From Figure 3.9, for an ideal square, we have

$$h = w$$

where h and w are the height and width of the final layout. Equivalently we have

$$\frac{H + 2R(n-1)}{n} = nW + (n-1)S \tag{3–2}$$

where H and W are the height and width of the primary layout as shown in Figure 3.9. R is the sum of the routing space and the height of the so–called compressor tile. n is the number of segments that the primary layout will be divided into. S is the separation between the segments. Rearranging Equation (3–2) yields the quadratic equation.

$$(W + S)n^2 - (2R + S)n - (H - 2R) = 0$$

Solving this quadratic equation for n, we get

$$n = \frac{(2R + S) + \sqrt{(2R + S)^2 + 4(W + S)(H - 2R)}}{2(W + S)} \tag{3–3}$$

To solve n from Equation (3–3), two OTACC layouts are necessary. OTACC will generate a single–line layout first to determine the primary height (H) and primary width (W) of the layout, then the number of segments for the final layout is calculated. The height limit for the final layout is easily determined from:

$$h = \frac{H}{n}$$

If users specify a CIS style layout, i.e. input and output terminals on the same side of the layout, n is selected to be the nearest even number. If users specify a TRANS style layout, i.e. input and output terminals on opposite side of the layout, n is selected to be the nearest odd number.

The compressor tile is necessary to connect the two adjacent segments. The compressor tile combines the parallel signals into a single wire which is much more suitable for turning the filter's signal path to the opposite direction. A break for the U–turn occurs only at the boundary of capacitor array which contains many fingers of identical AC sig-

nals. If each of these AC signals were routed separately, a large routing area would be needed and a large unpredictable parasitic capacitance would be introduced.

The final rules for creating the zigzag structure are following:

1. When a filter's layout reaches the height limit, the capacitor fingers are "capped" with a compressor tile. The local routing for transmission zero and shunt inductor in this segment are completed and the next segment begins with another compressor tile.
2. An U–turn in the layout may only be made at the junction of a capacitor array. But it can not be turned at the junction of the capacitor array used in the simulation of the inductors to avoid additional AC signal routing.
3. The height of each segment may differ somewhat from the absolute height limit. The difference is always less than the height of the OTA cell used in the layout.

3.6.4 Final Assembly

After the software has processed the filter specification input and built the filter likely with more than one segment, final assembly begins. The segments of the filter are popped off the stack in order, and assembled with enough separation to reduce the AC signal cross–talk, as shown in Figure 3.10.

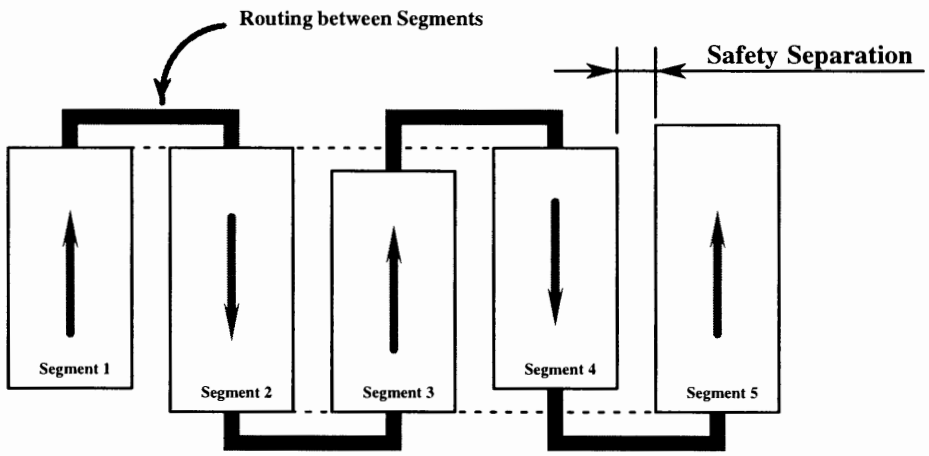


Figure 3.10 Filter segments routed together.

The height of each segment may vary slightly. To efficiently use the space, each segment is aligned with the current border of the previous segments instead of the border of the adjacent segment, as shown in Figure 3.10.

The AC signals between segments are routed with the layer metal 2 and the DC signals are routed with layer metal 1, as the example shown in Figure 3.11, To reduce the AC signal cross–talk as much as possible, all AC signals are separated by one or more DC signals.

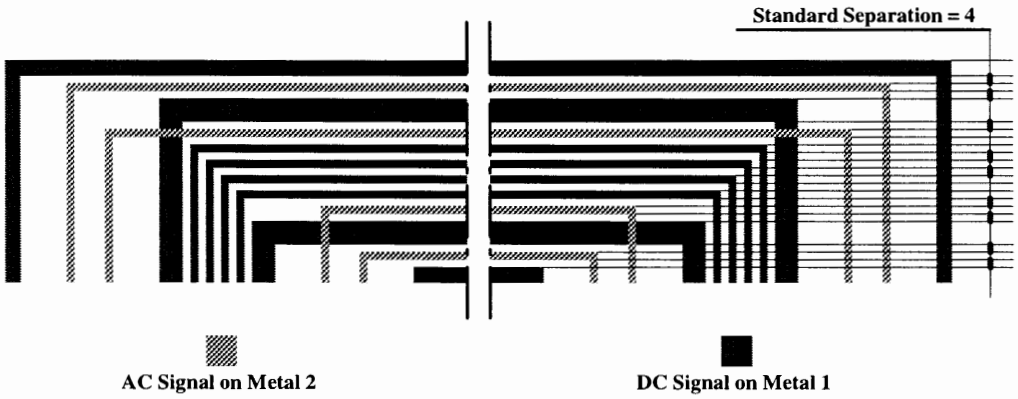


Figure 3.11 Routing wires between segments.

After the final assembly, the labels for the input and output nodes of the filter are added onto the layout, and the final layout is stored as a MAGIC format file with the name that is specified in the *LC* ladder description file (the *filter_name* in Table 3.2).

CHAPTER 4

TESTING AND SIMULATION RESULTS

Essentially, OTACC is the layout realization step in the filter design cycle. The previous step is the approximation step where the filter characteristics are converted into a set of element specifications that make up the filter. The step following the realization step is the verification step where the filter layout is simulated to verify that it meets the original specification. the OTACC input is the output of the approximation step, and the OTACC output is intended for verification/simulation.

To complete the verification using SPICE simulation, for example, the following procedures are required:

1. Generate the filter layout by running OTACC.
2. Extract the hierarchical circuit from the layout and generate the SPICE file.
3. Simulate the circuit by using SPICE.
4. Analyze the simulation results by comparing it with the original specification or the SPICE output of an ideal filter with same specification.

In order to fully automate the filter design and simulation process, a top–level design centering software may be utilized to control OTACC, Magic, Spice, and the comparison software. The user interface controls the data flow between the programs. This design centering software is the subject of another research project [27] [28].

More than twenty simulations were performed to verify the validity of the layout produced by OTACC. In most cases, the simulation results of the extracted filter layout meet the filter specifications. As examples, three different filters have been generated with dif-

ferent OTAs and simulated. The simulation results for each filter and OTA selection will be discussed in following sections.

OTAs		FILTERs		
No.	Developer	Fourth–Order Lowpass Filter	Eighth–Order Lowpass Filter	Third–Order Bandpass Filter
I	Pan Wu	Fully Verified	Fully Verified	Fully Verified
II	David Chiang	Fully Verified	Fully Verified	Fully Verified
III	Girish Chandrasekaran	Fully Verified	Failed	Failed

Table 4.1 Filters be realized with different OTAs

The layouts of all filters listed in Table 4.1 are realized by running OTACC, and are simulated using U.C. Berkeley SPICE Version 3f4. The realization and simulation are executed on SPARC 5 work station using SunOS. The layouts of different filters are printed as a series of figures in Appendix F.

As shown in Table 4.1, OTA III failed to meet the specification in realizing an eighth–order lowpass filter and a third–order bandpass filter. OTACC generated the layout of the filter in one minute of wall–clock time or less. So the user can validate the filter design with certain OTAs in as little as ten minutes, and quickly fit the design with a suitable OTA.

4.1 Fourth–Order Elliptic LC Lowpass Filter [1]

The schematic of a normalized fourth–order elliptic *LC* lowpass ladder is shown in Figure 4.1, and the schematic of the corresponding Transconductance–C realization using fully differential OTAs is shown in Figure 4.2.

By using OTACC, three different versions of this design are generated using different OTA cells that are available in the current library. All SPICE simulations are based upon

the circuit layout extraction. The magnitude responses of those filters are shown in Figure 4.3 and their phase and delay plots are shown in Figure 4.4 and Figure 4.5. The DC bias set points for the different OTAs are shown in Table 4.2. A comparison of the important characteristic of the filters is shown in Table 4.3. All parameters in this table are based on the simulation results and are not subject to adjustment or tuning.

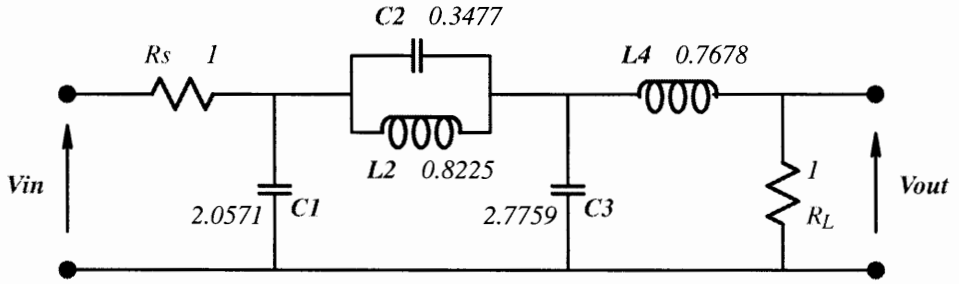


Figure 4.1 Fourth–Order Elliptic Lowpass Filter.

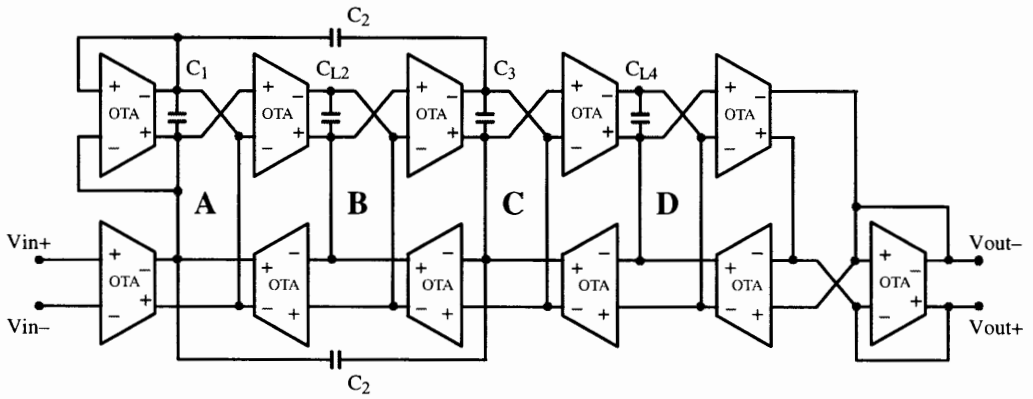


Figure 4.2 Schematic diagram of G_m–C Fourth–Order Lowpass Filter.

OTAs	Power	DC Input	g_m Control	r_o Control	Misc.	g_m Value
OTA I	±5 V	0 V	V_{is}=2.75 V	V_2=–2.68 V	V_c=0 V ; V_1=0 V	180 μS
OTA II	5 V	2.5 V	V_f=1.5 V	V_q=2.1 V	V_{bi}=1.13 V ; V_{i1}=2.5 V	150 μS
OTA III	5 V	2.5 V	V_{cf}=1.03 V	V_{cq}=2.65 V	V_{bias}=1.2 V	133 μS

Table 4.2 Bias set up for different OTAs in filter realization and simulation.

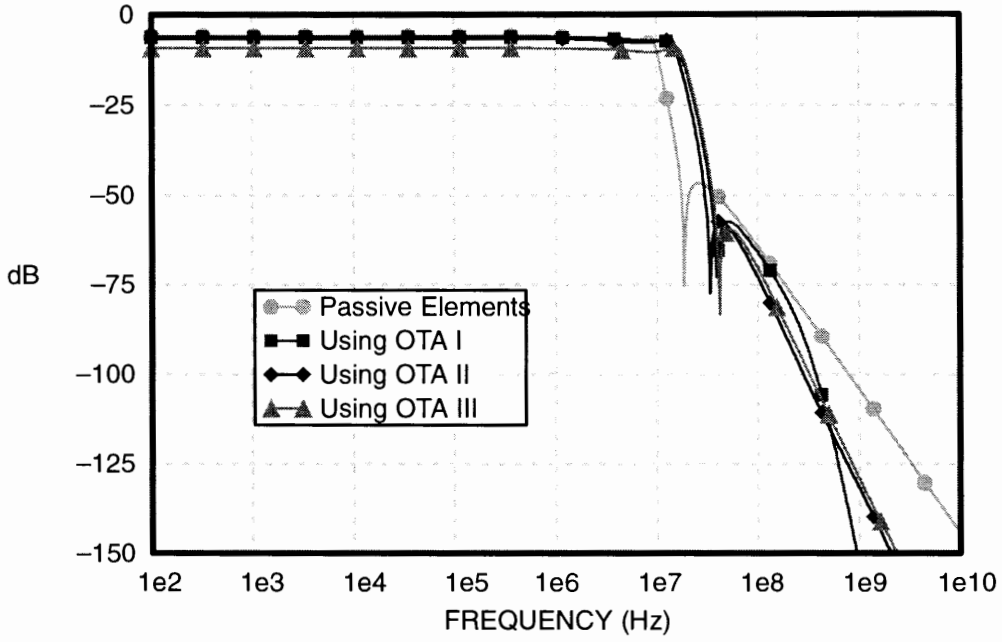


Figure 4.3 Magnitude response of Fourth–Order Lowpass Filter using different OTAs

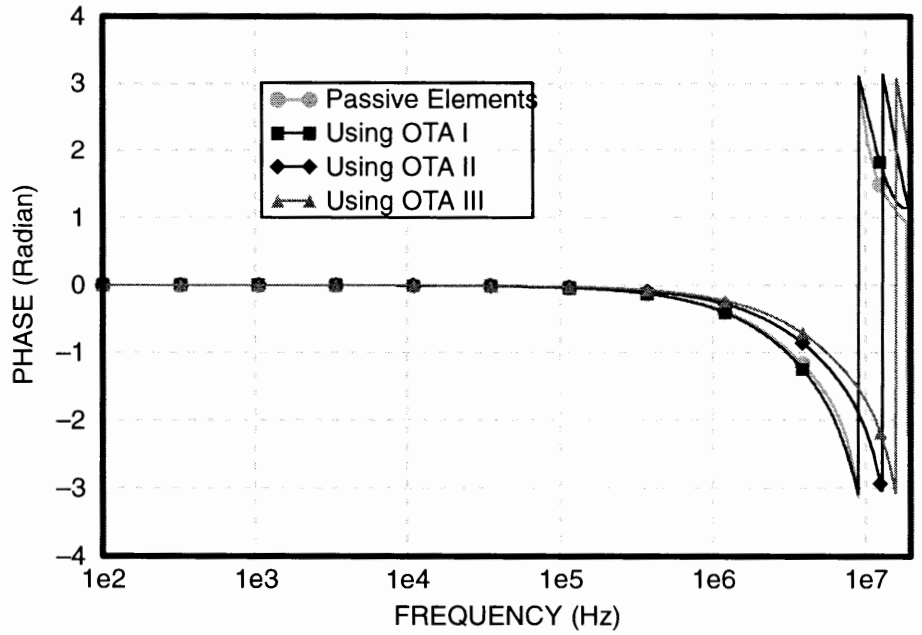


Figure 4.4 Phase plot of Fourth–Order Lowpass Filters using different OTAs

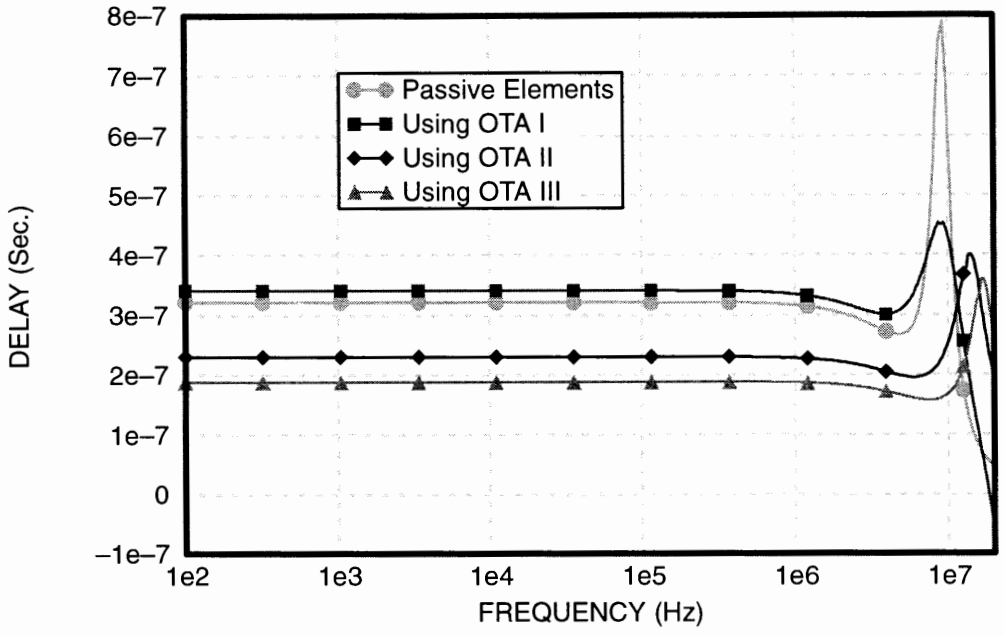


Figure 4.5 Delay of Fourth–Order Lowpass Filters using different OTAs

Fourth–Order LP Filters	Cutoff Frequency	Stopband Frequency	Stopband Attenuation	Passband Ripple	Geometry: WxH (um)	Linear Range % of OTA
Specification	10 MHz	18 MHz	40.5 dB	0 dB		
Using OTA I	17 MHz	39 MHz	50 dB	0.9 dB	1482x1274	100%
Using OTA II	16 MHz	34 MHz	51 dB	1 dB	1028x1126	100%
Using OTA III	17 MHz	41 MHz	48 dB	1 dB	1150x927	100%

Table 4.3 Comparison among the Fourth–Order Lowpass Filters using different OTAs.

Due to the parasitic capacitance of the miscellaneous routing wires that can not be completely accounted by predistortion, non–linear variation of the input, output capacitance of OTAs, and the non–ideal character of the OTAs, the filter cutoff frequency varies from the desired frequency. This problem can be improved by more precisely filling the parameters in OTA technology file by the OTA cell designers. Fortunately, all the OTA

cells in our library are designed with tuneable transconductance and this mechanism is proposed in the majority of designs in the literature. Therefore, the cutoff frequency of the filters can be adjusted to the desired frequency by tuning the g_m value of the OTAs. An example of the cutoff frequency tuning of the fourth–order elliptic lowpass filters is shown in Figure 4.6. Even though the notch is missing after the tuning of the transconductance value, the characteristics of the filter, such as stop band, pass band and transient band, are met by the tuned transconductance–C filter generated by OTACC.

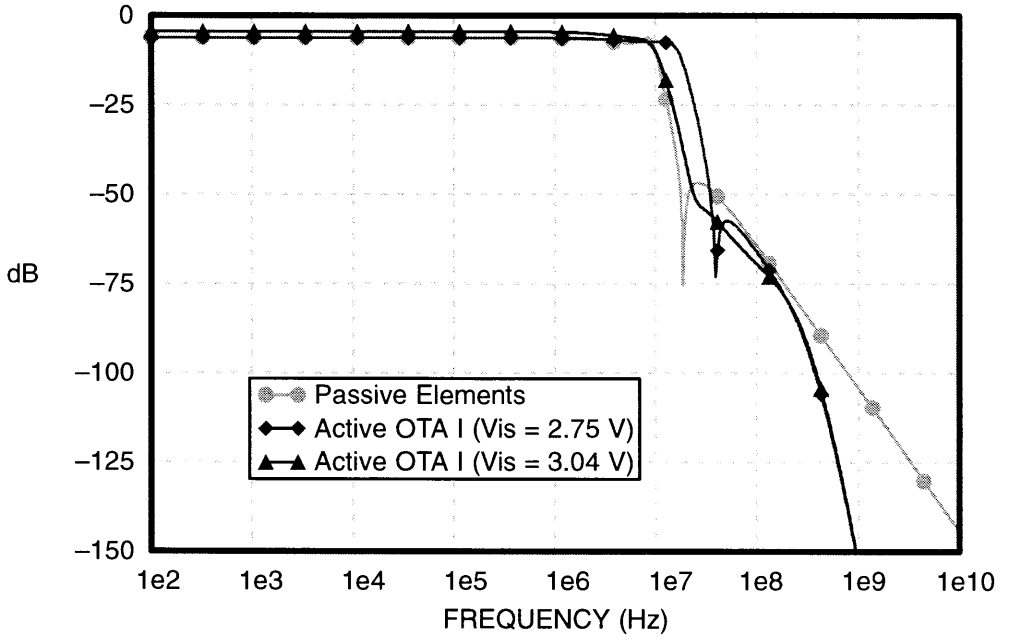


Figure 4.6 Zero frequency tuning of Fourth–Order Lowpass Filter using OTA I.

To further evaluate the performance of the filters and estimate the linear range of the overall filters, the voltage gains to each internal node of those filters at 200 mV AC input signal are plotted and shown in Figure 4.7, Figure 4.8, and Figure 4.9. The node names referred in those plots are corresponding to the names shown in Figure 4.2. From these plots, it can be seen that the voltage gain at each internal node is less than 1. Therefore the OTAs in the filters have their full dynamic range available.

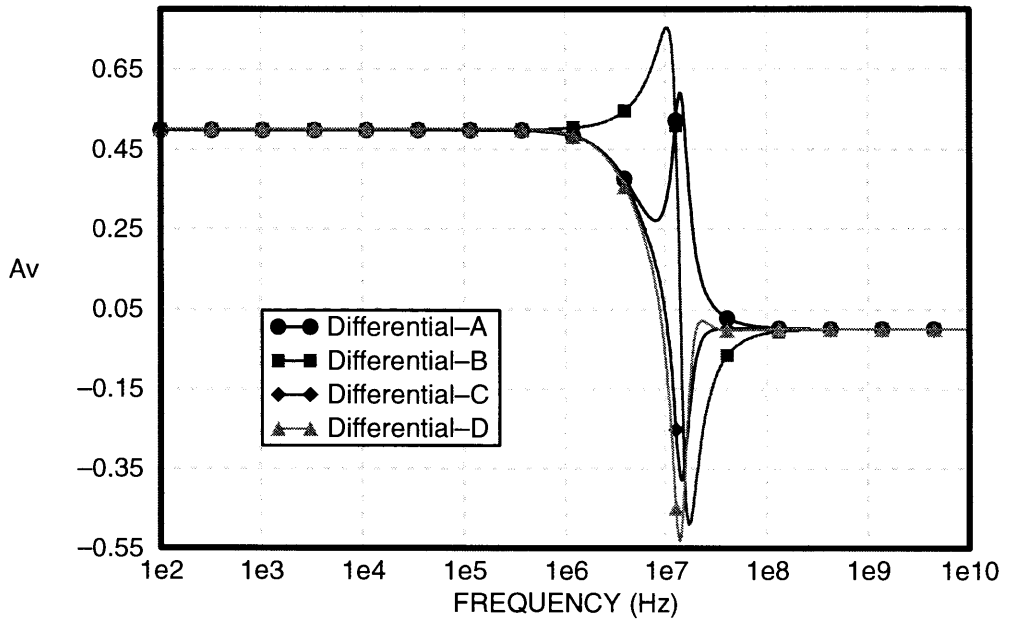


Figure 4.7 Voltage gain to the internal nodes of the Fourth–Order Lowpass Filter using OTA I.

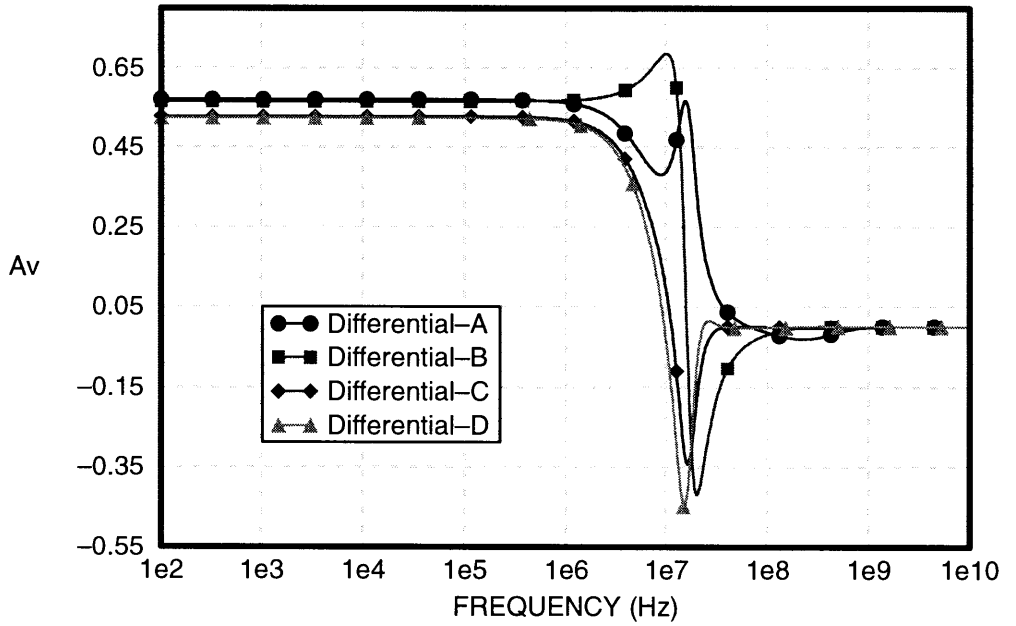


Figure 4.8 Voltage gain to the internal nodes of the Fourth–Order Lowpass Filter using OTA II.

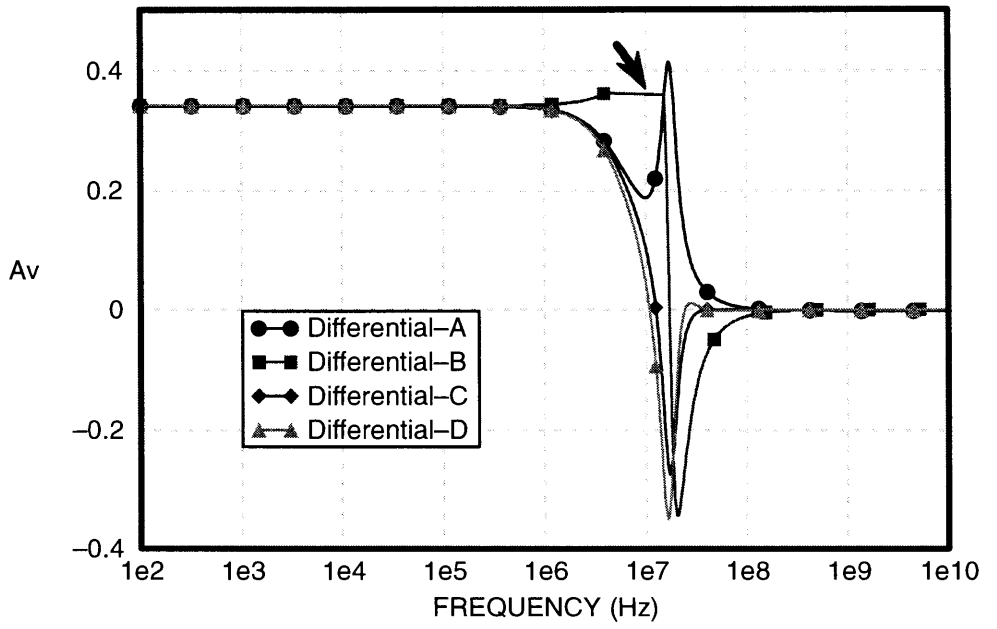


Figure 4.9 Voltage gain to the internal nodes of the Fourth–Order Lowpass Filter using OTA III.

By using OTACC, the layout of those filters were generated in one minute (wall–clock time) or less. The approximate wall–clock time for each step of the layout generation and simulation are shown in Table 4.4. If the user does not specify the height or width limit of the layout, OTACC will generate a layout that is roughly square. The geometric size of the layouts for the forth–order lowpass filter with different OTAs are shown in Table 4.5.

Fourth–Order Lowpass Filters	Processing Time (Sec)				
	OTACC	Extraction	Ext2spice	Editing	Spice
Using OTA I	5	10	2	1	60
Using OTA II	5	8	2	1	53
Using OTA III	5	7	2	1	35

Table 4.4 Approximate run time of AC analysis with 100 points per decade for the Fourth–Order Lowpass Filter on a SPARC 5 station

Fourth–Order Lowpass Filters	Size of Filters		Size of OTAs	
	Height (μm)	Width (μm)	Height (μm)	Width (μm)
Using OTA I	1274	1482	267	296
Using OTA II	1126	1028	206	180
Using OTA III	927	1150	154	213

Table 4.5 Geometric Size of the Layouts of the Forth–Order Lowpass Filters and the OTAs

From the test results for the Forth–Order Lowpass Filters shown above, it demonstrate that OTACC can realize the filter layout correctly and efficiently. From Figure 4.9, we can also find out that the filter using OTA III is in critical condition for normal operation. From Table 4.4 it can be concluded that the user can quickly optimize an OTA based filter design with OTACC.

4.2 Eighth–Order Inverse Chebyshev LC Lowpass Filter [1]

The schematic of the normalized eighth–order inverse Chebyshev *LC* lowpass ladder is shown in Figure 4.10. The schematic of the corresponding transconductance–C realization with fully differential OTAs is shown in Figure 4.11.

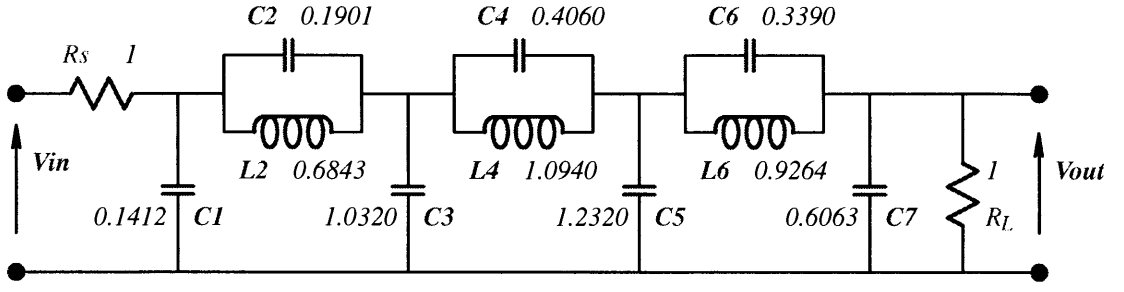


Figure 4.10 Eighth–Order Inverse Chebyshev Lowpass Filter.

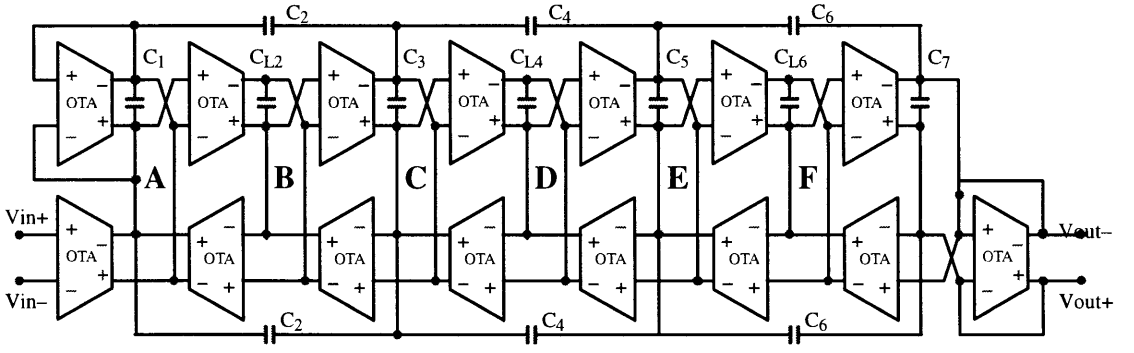


Figure 4.11 Schematic diagram of G_m–C Eighth–Order Inverse Chebyshev Lowpass Filter.

For the eighth–order inverse Chebyshev lowpass filter, OTA I and OTA II are chosen as examples. To examine the generality of OTACC, the same procedures as were used in the fourth–order lowpass filter, were used to test the eighth–order inverse Chebyshev lowpass filters. The bias set point for OTA I and OTA II are the same as shown in Table 4.2. The magnitude response for the two filters is shown in Figure 4.12. The phase and delay plots are shown in Figure 4.13 and Figure 4.14 respectively. The comparison of some important parameters of those filters is shown in Table 4.6. Because of the same reason in testing the Forth–Order Lowpass Filter, the filter cutoff frequency varies from the desired frequency. An example of the cutoff frequency tuning of the eighth–order lowpass filters is shown in Figure 4.15. The voltage gains to each internal node of those filters are plotted and shown in Figure 4.16, and Figure 4.17. The approximate wall–clock time for each step of the layout generation and simulation are shown in Table 4.7. The geometry size of the layouts for the eight–order lowpass filter with different OTAs are shown in Table 4.8.

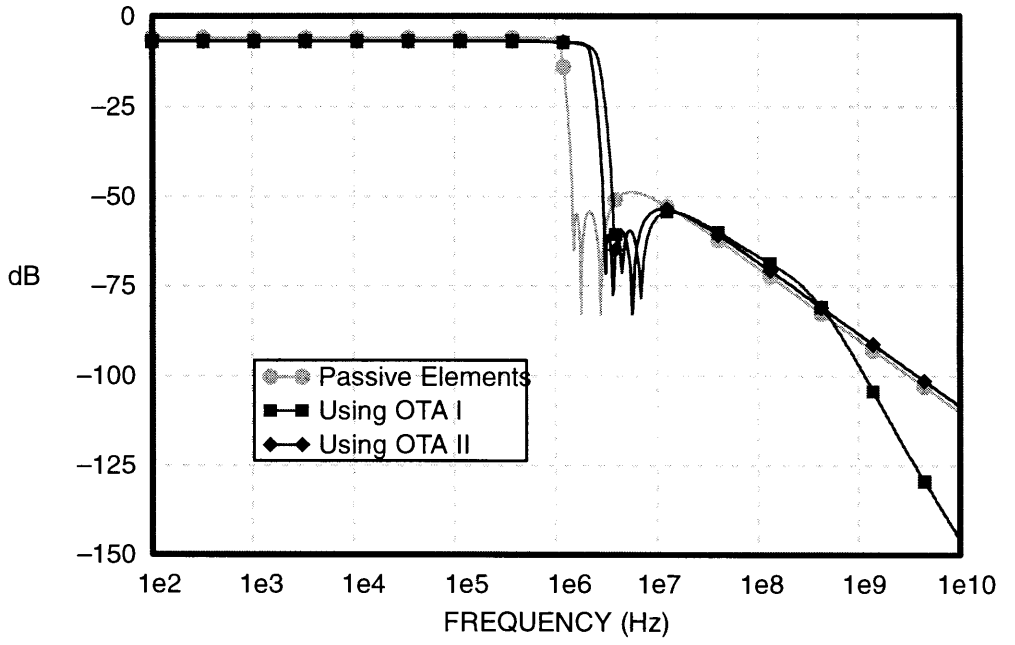


Figure 4.12 Magnitude response of Eighth–Order Lowpass Filter using different OTAs

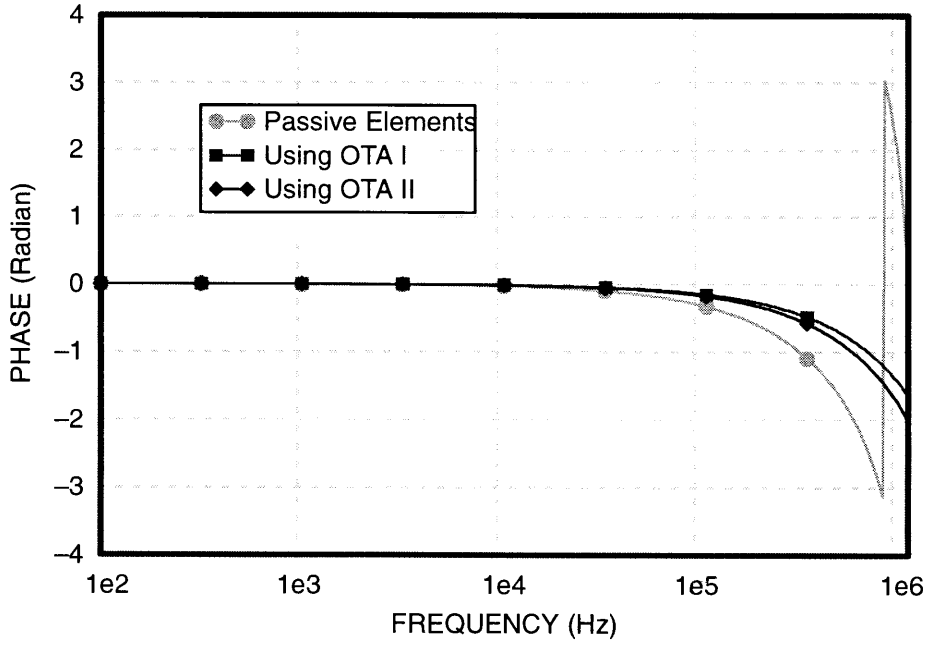


Figure 4.13 Phase plot of Eighth–Order Lowpass Filter using different OTAs

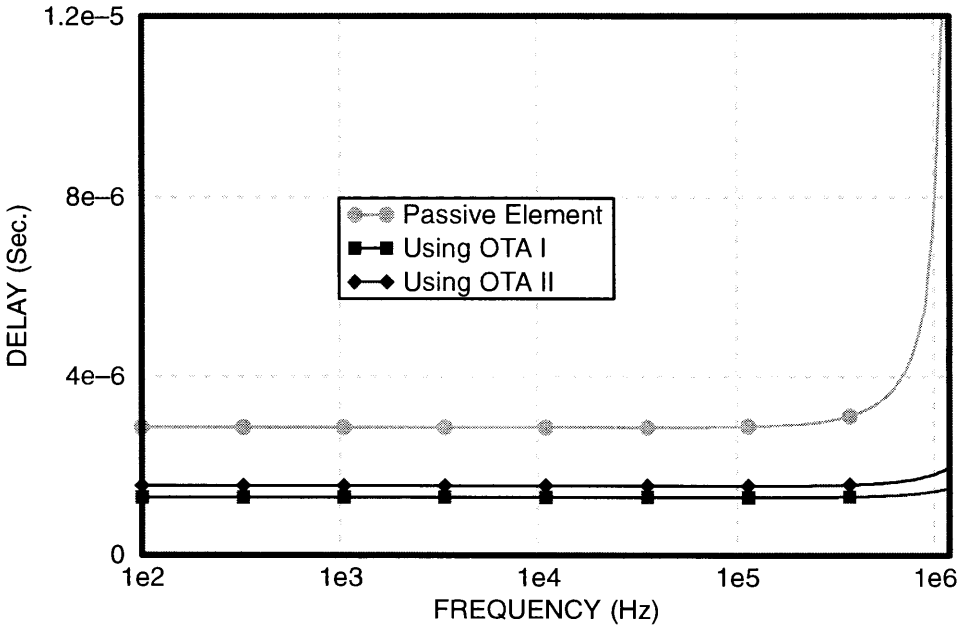


Figure 4.14 Delay of Eighth–Order Lowpass Filter using different OTAs

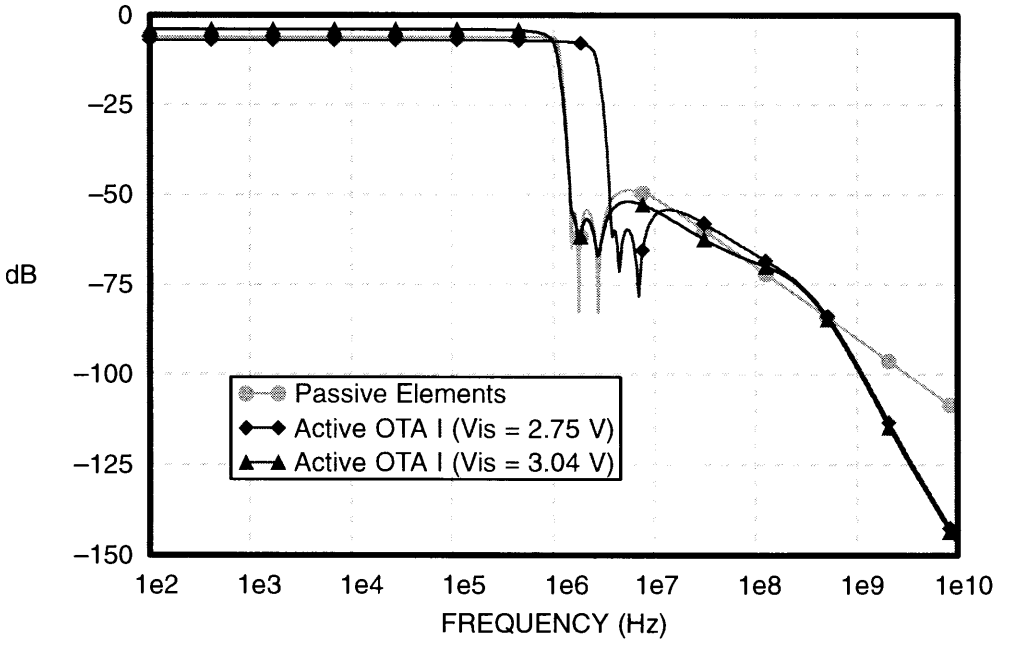


Figure 4.15 Zero frequency tuning of Eighth–Order Lowpass Filter using OTA I.

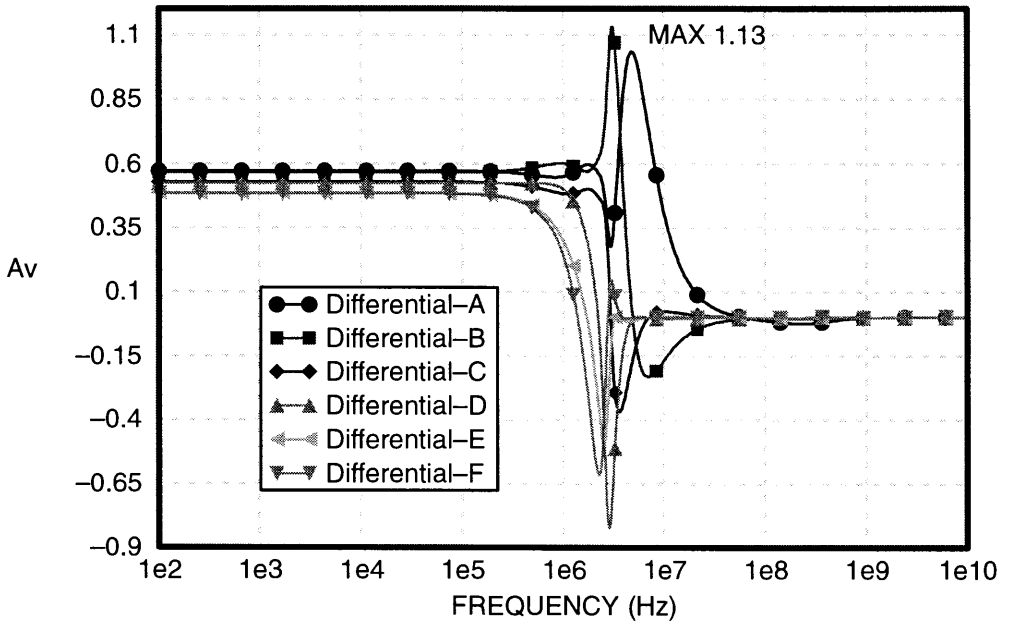


Figure 4.16 Voltage gain to the internal nodes of the Eighth–Order Lowpass Filter using OTA I.

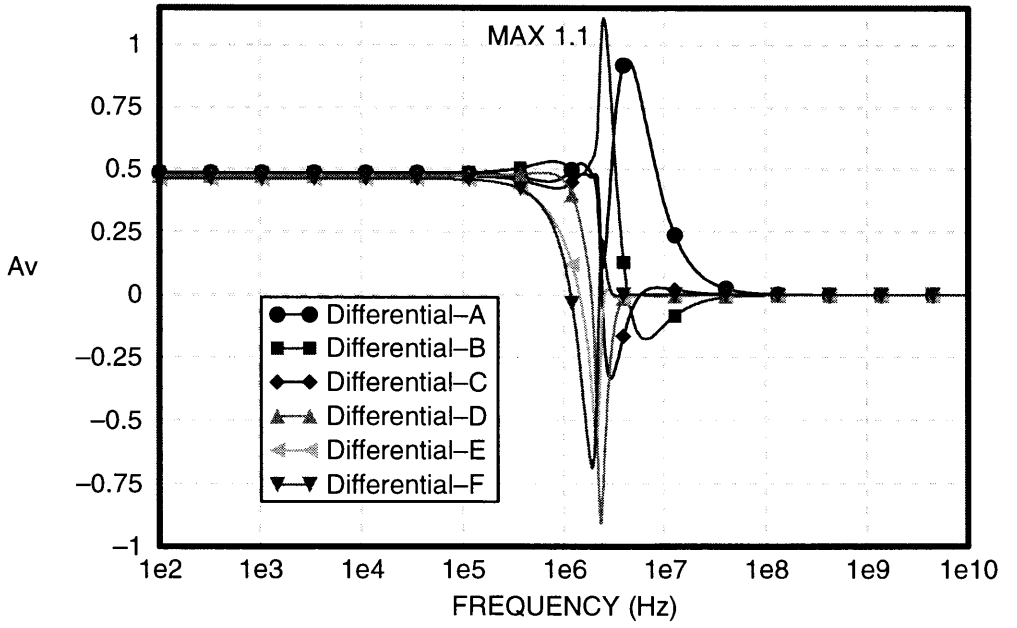


Figure 4.17 Voltage gain to the internal nodes of the Eighth–Order Lowpass Filter using OTA II.

Eighth–Order LP Filters	Cutoff Frequency	Stopband Frequency	Stopband Attenuation	Passband Ripple	Geometry: WxH (um)	Linearity: % of OTA
Specification	1 MHz	1.5 MHz	42 dB	0 dB		
Using OTA I	2.5 MHz	3.8 MHz	47 dB	0.2 dB	1186x2254	88%
Using OTA II	2.2 MHz	3.0 MHz	46 dB	0.2 dB	1432x2126	91%

Table 4.6 Comparison among the Eighth–Order Lowpass Filters using different OTAs.

Eighth–Order Lowpass Filters	Processing Time (Sec)				
	OTACC	Extraction	Ext2spice	Editing	Spice
Using OTA I	15	40	15	1	120
Using OTA II	15	38	12	1	104

Table 4.7 Approximate run time of AC analysis with 100 points per decade for the Eighth–Order Lowpass Filter on SPARC 5 station

Fourth–Order Lowpass Filters	Size of Filters		Size of OTAs	
	Height (μm)	Width (μm)	Height (μm)	Width (μm)
Using OTA I	2254	1886	267	296
Using OTA II	2126	1432	206	180

Table 4.8 Geometric size of the layouts of the Eighth–Order Lowpass Filters and the OTAs

The simulation results shown in the Figure 4.16 and Figure 4.17 reveal that the voltage gain to the internal node B (the node inside the first shunt inductor) exceeds 1. This means that to keep every OTA in the filter working in their linear range, the over all dynamic range of the filter must be reduced.

From the test results for the Eighth–Order Lowpass Filters shown above, it demonstrate that OTACC can realize the filter layout correctly and efficiently for relatively complex structure (there are 15 OTAs total in the Eighth–Order Lowpass Filters).

4.3 Third–Order Bandpass Filter [13]

The schematic of the normalized third–order bandpass ladder is shown in Figure 4.18. The schematic of the corresponding G_m–C realization with fully differential OTAs is shown in Figure 4.19.

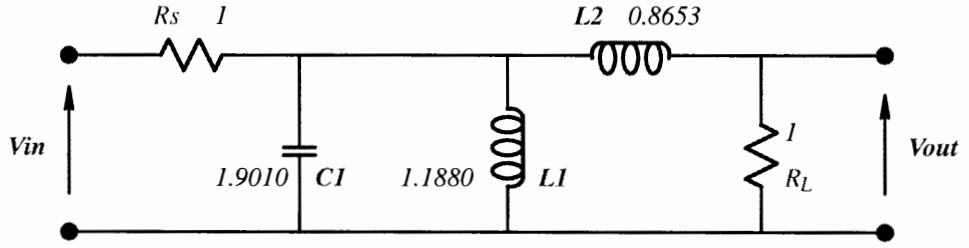


Figure 4.18 Third–Order Bandpass Filter.

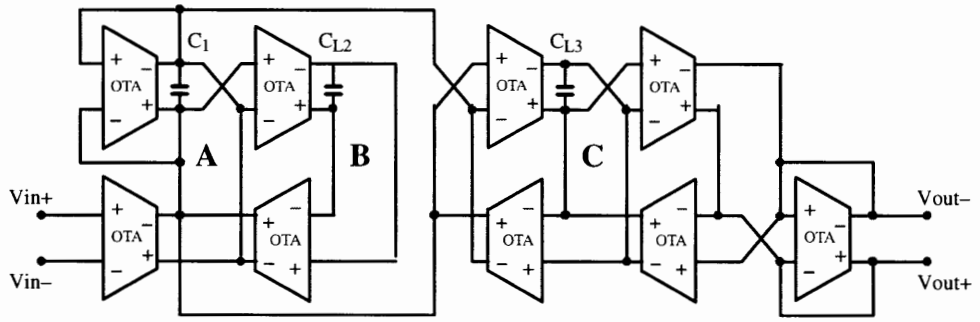


Figure 4.19 Schematic diagram of G_m–C Third–Order Bandpass Filter.

For the third–order bandpass filter, OTA I and OTA II are chosen as examples using the same procedures as the previous two cases . The bias settings for OTA I and OTA II are the same as shown in Table 4.2. The magnitude response of those filters is shown in Figure 4.20. A comparison of the important characteristics of the third–order bandpass filters is shown in Table 4.9. An example of the center frequency tuning of the third–order bandpass filters is shown in Figure 4.21. The voltage gains to every internal node of those filters are plotted and shown in Figure 4.22, and Figure 4.23. The approximate execution

times for each step of the layout generation and simulation are shown in Table 4.10. The geometry size of the layout for the third–order bandpass filters are shown in Table 4.11.

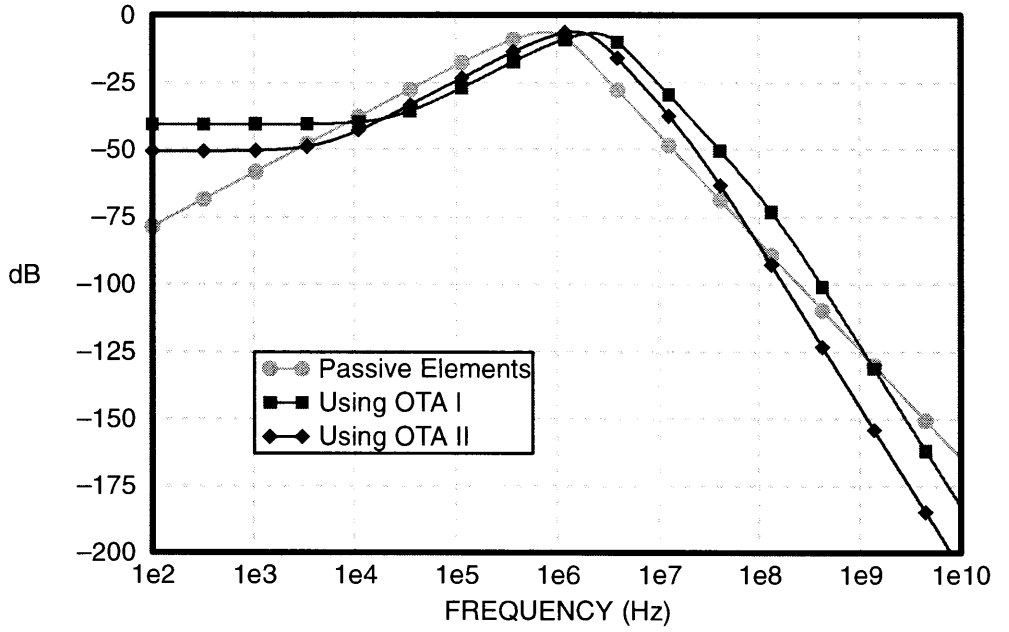


Figure 4.20 Magnitude response of Third–Order Bandpass Filter using different OTAs

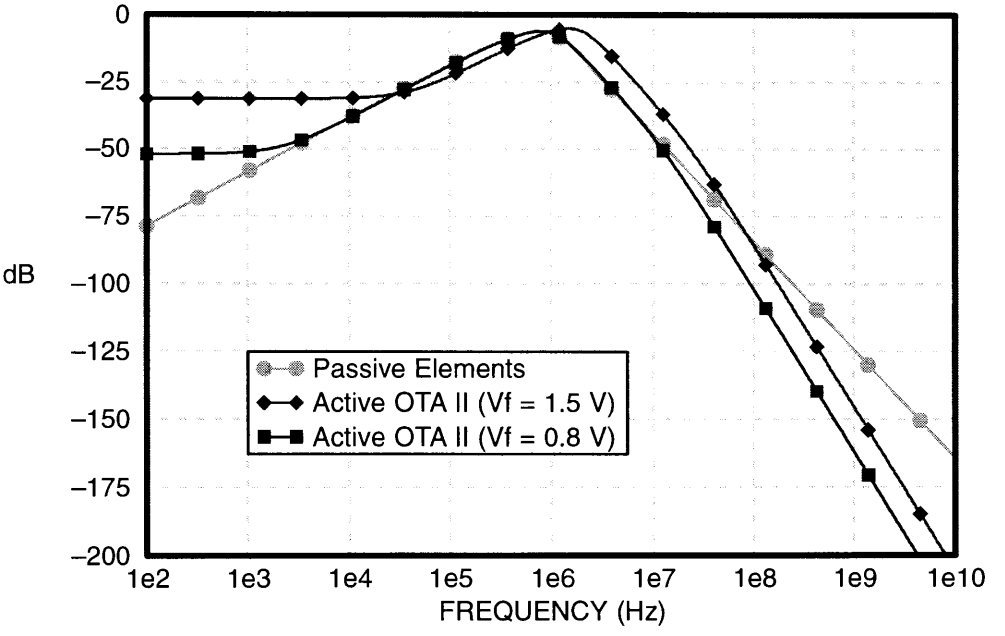


Figure 4.21 Center frequency tuning of Third–Order Bandpass Filter using OTA II.

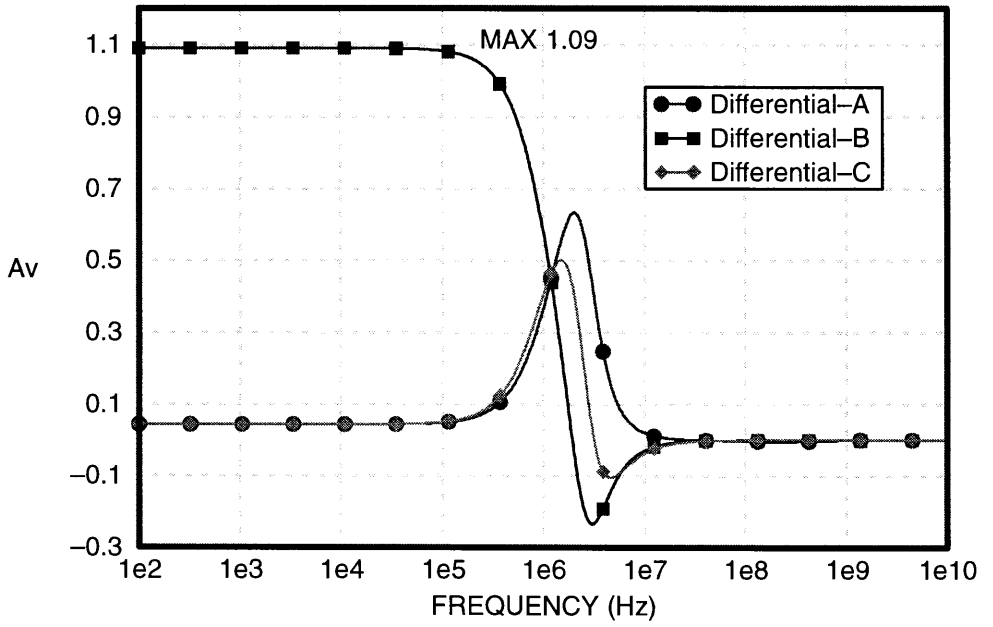


Figure 4.22 Voltage gain to the internal nodes of the Third–Order Bandpass Filter using OTA I.

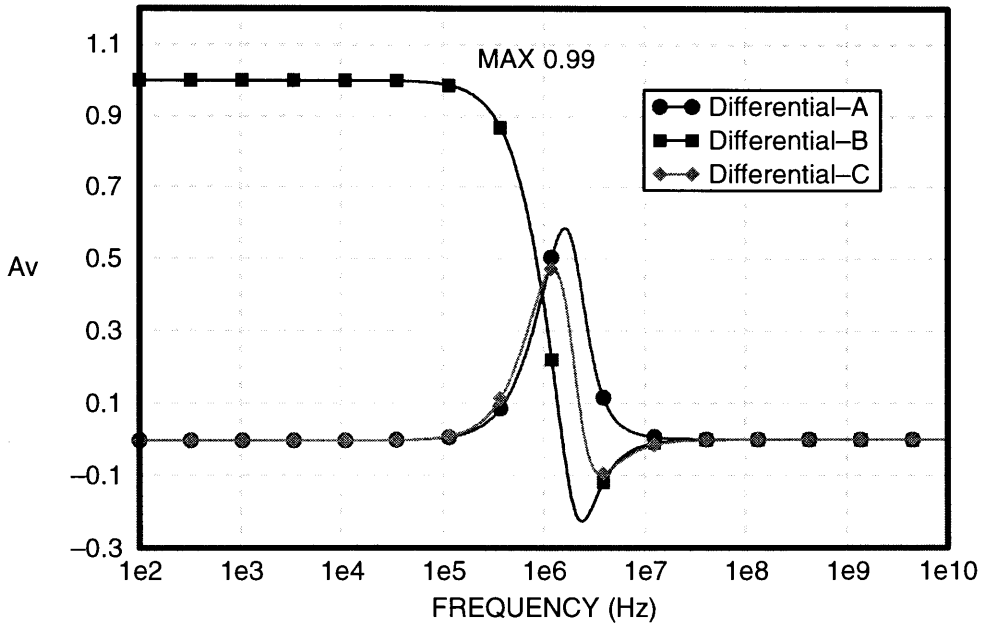


Figure 4.23 Voltage gain to the internal nodes of the Third–Order Bandpass Filter using OTA II.

Eighth–Order LP Filters	Center Frequency	3 dB (L) Frequency	3 dB (H) Frequency	Stopband Attenuation	Geometry: WxH (um)	Linearity: % of OTA
Specification	0.8 MHz	0.35 MHz	1.35 MHz			
Using OTA I	1.9 MHz	0.8 MHz	3.2 MHz	22 dB	1404x1312	92%
Using OTA II	1.5 MHz	0.8 MHz	2.5 MHz	46 dB	950x1286	100%

Table 4.9 Comparison among the Third–Order Bandpass Filters using different OTAs.

Third–Order Bandpass Filters	Processing Time (Sec)				
	OTACC	Extraction	Ext2spice	Editing	Spice
Using OTA I	7	25	7	1	22
Using OTA II	7	20	6	1	56

Table 4.10 Approximate run time of AC analysis with 100 points per decade for the Third–Order Bandpass Filter on SPARC 5 station

Fourth–Order Lowpass Filters	Size of Filters		Size of OTAs	
	Height (μm)	Width (μm)	Height (μm)	Width (μm)
Using OTA I	1312	1404	267	296
Using OTA II	1286	964	206	180

Table 4.11 Geometric size of the layouts of the Third–Order Bandpass Filters and the OTAs

From the simulation results shown above, we can see that the third–order bandpass filter realized with OTA I has relatively poor low stopband attenuation. This is mainly because the inductors simulated by OTAs are really lossy inductors, and the OTA I has relatively poor output impedance. The equivalent circuit for a practical grounded inductor simulated by OTAs is shown in Figure 4.24. Where the C_L is the circuit capacitance for simulating the inductor, g_m is the transconductance value of the OTA, r_o is the output resistance of the OTA cell. For higher OTA output resistance, a higher quality inductor can be simulated, and better stopband attenuation can be obtained for the third–order

bandpass filter. The simulation of the equivalent circuit shown in Figure 4.24 using ideal elements shows the same DC feed though as the practical inductor simulated with OTAs,

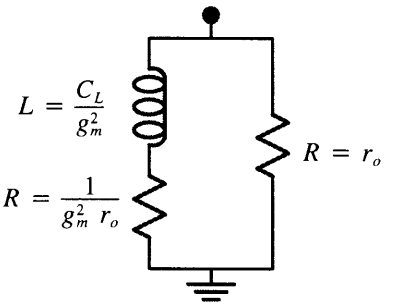


Figure 4.24 Equivalent circuit for a practical inductor simulated with OTAs.

Inside our library, the OTA II and OTA III are designed with a Negative Resistance Load (NRL) for compensating the parasitic output resistance of the OTA. By adjusting the output resistance of the OTA, improved low stopband attenuation of the third–order bandpass filter can be obtained. An example of the output resistance tuning effect is shown in Figure 4.25.

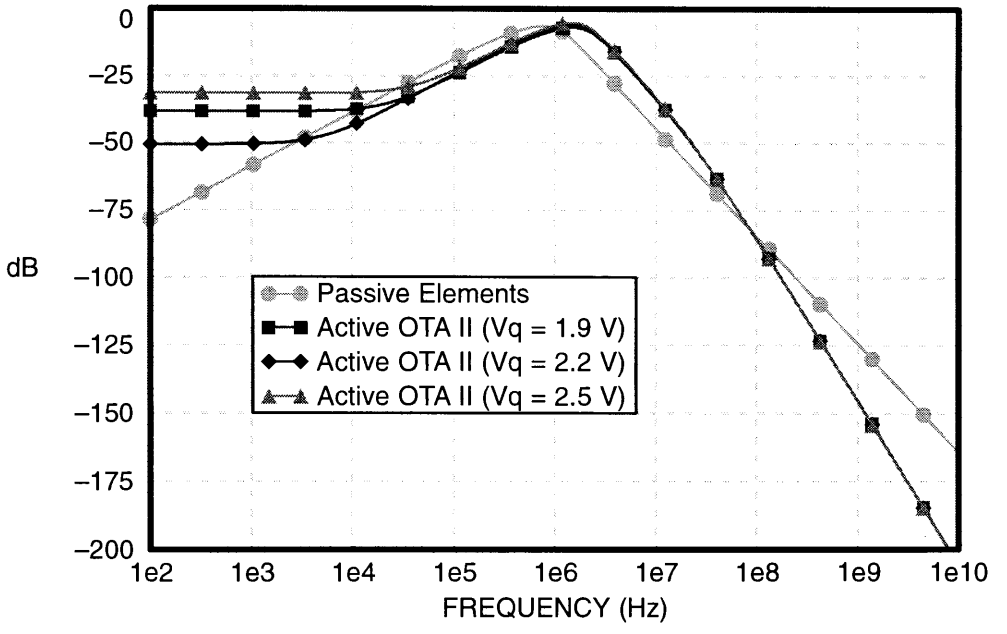


Figure 4.25 Magnitude response of Third–Order Bandpass Filter with output impedance adjustment using OTA II.

From the test results for the Third–Order Bandpass Filters shown above, it demonstrate that OTACC has the capability to realize various kind filter layouts correctly and efficiently.

4.4 Summary

From the testing and simulation results based on three different kind filters using different OTAs, it can be concluded that OTACC is an efficient, generic, and reliable layout synthesis tool for CMOS OTA–C continuous–time filters. By using OTACC, user can finish their filter layout and design optimization in minutes instead of hours or days, that will make the filter design work more efficiency and greatly save the engineering resource.

CHAPTER 5

CONCLUSIONS

Automatic synthesis of VLSI layout has traditionally been a tool only available to digital designers. This is because of the differences in sensitivity of the influence of parasitics between digital and analog circuits. Analog signals are typically measured in terms of millivolts. So slight variation in analog signals caused by the unavoidable parasitic capacitances results very large deviation in the circuit response. Digital circuits are able to address parasitic capacitance problems because they use larger noise margins, and also because they can be overdriven to compensate for any signal loss. This same approach to circuit design in analog circuits would be disastrous.

To overcome parasitic capacitances in analog circuits, a different approach is required. Instead of overdriving or ignoring the parasitic capacitance, they must all be carefully accounted for and if possible used as part of a desired circuit capacitor. The process of absorbing a parasitic capacitance into a circuit capacitor is called predistortion. Since predistortion can only occur on nodes where a circuit capacitor exists, then the scope of the solution is limited to specific configurations.

This research focused on *LC* ladder continuous–time filters using OTA–C methods. The advantage of the *LC* ladder configuration is that most nodes in the original circuit contain a capacitor and predistortion may be used.

From the testing and simulation results shown in Chapter 4, OTACC is a very efficient layout tool for the transconductance–capacitor continuous–time filter design automation and can reduce the time in the design cycle tremendously. Efficiently exploring several design alternatives, a designer can make the design optimization in minutes instead of hours or days.

5.1 LIMITATION

The parasitic capacitances of the OTA ultimately set the size of the smallest usable capacitor in the filters since any desired capacitor can be no less than the intrinsic parasitic capacitance. Correspondingly, this also sets the maximum obtained filter frequency for a given OTA. The following scaling equation shows the relationship between the filter's frequency and the size of the required filter capacitors:

$$\frac{g_m}{2\pi f_0} C_{normalized} = C_{circuit} > C_{parasitic}$$

It can be seen, as f_0 is increased, $C_{circuit}$ gets smaller. One way to prevent the desired capacitor from becoming too small is to increase the value of the transconductance g_m. If the transconductance value exceeds the maximum value of OTA's tuning range, another OTA with relatively large transconductance is expected. However, as the transconductance value of OTA is increased, the tendency is also to increase either the input, output or both parasitic capacitance. Therefore, we once again run into the problem where the parasitic capacitance becomes larger than the desired capacitor on the circuit node. This has the effect of limiting the realizable frequency of the filter.

Another limitation is the applicability to some OTA–C filters that have unloaded nodes. If there are unloaded nodes in the circuit configuration, i.e. there is no circuit capacitance resigned on those nodes, the predistortion may not occur. Because a parasitic

capacitance will always exist in real world, this will cause the transfer function to be changed radically. As shown in Figure 5.1, the highpass filter will really have a behavior more like a bandpass filter due to the parasitic capacitance that is always present at node A and node B.

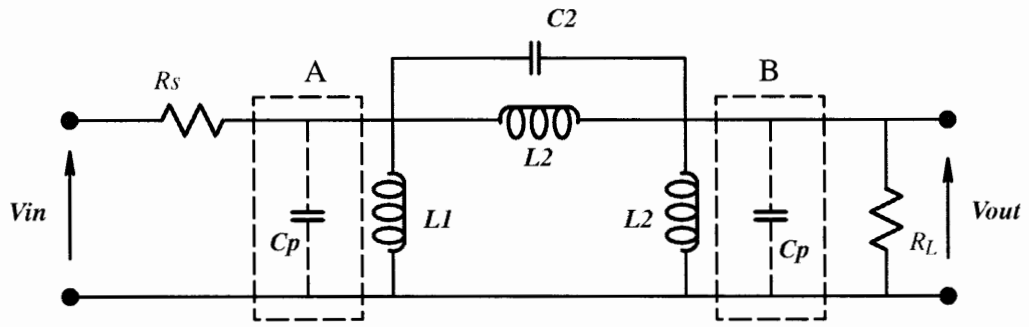


Figure 5.1 Example for the influence of parasitic capacitance on floating nodes in a highpass filter.

5.2 FUTURE WORK

It would be interesting to investigate the use of using the filters generated as standard cells themselves. The cascade of low–order (typically second–order) filters creates filters of higher order. One of the main advantages of cascade filters is that they are very easy to tune because each biquad is responsible for the realization of only one pole pair (and zero pair): the realizations of the individual critical frequencies of the filter are decoupled from each other [1]. Current OTACC is already capable of creating higher order filters. However, it does not have the capability to reuse its own designs as standard cells as would be required for the cascading of biquads. This would be an interesting area to expand the capabilities of OTACC.

REFERENCES

[1] Rolf Schaumann, Mohammed S. Ghausi and Kenneth R. Lader, "Design of Analog Filters – Passive, Active RC and Switched Capacitor", Prentice Hall, 1990.

[2] Rolf Schaumann, Mohammed S. Ghausi and Kenneth R. Lader, "Design of Analog Filters – Passive, Active RC and Switched Capacitor", Prentice Hall, 1990, Section 4.4, Page 235.

[3] Jose Silva–Martinez, Michiel Steyaert, Willy Sansen, "High–Performance CMOS Continuous–Time Filters", Kluwer Academic Publishers, 1993.

[4] W. R. Daasch, M. Wedlake, R. Schaumann and P. Wu, "Automation of the IC layout of continuous–time transconductance–capacitor filters", *Int. Journal of Circuit Theory and Applications*, 20, 267–282, 1992.

[5] W. R. Daasch, M. Wedlake and R. Schaumann, "Automatic Generation of CMOS Continuous–Time Elliptic Filters", *Electronics Letters*, 28, no. 24, pp. 2215–2216, 1992.

[6] W. R. Daasch and M. Wedlake, "Rapid Layout of a Continuous–Time Transconductance–C Filter", *IEEE Proc. Int. Symp. on Circuits and Systems*, pp. 2256–2259, 1992.

[7] David L. Robinson, "Automatic Synthesis of VLSI Layout for Analog Continuous–Time Filters", Portland State University Library, March 1995.

[8] M. A. Tan and R. Schaumann, "Generation of Transconductance–Grounded–Capacitor filters by Signal–Flow–Graph Methods for VLSI Implementation", *Electronics Letters*, Vol. 23, No. 20, September 1987.

[9] Neil H. E. Weste and Kamran Eshraghian, "Principles of CMOS VLSI Design", Second Edition, ADDISON–WESLEY Publishing Company, 1993.

[10] J. Ousterhout, "The Magic Layout System", *IEEE Design Test Computation*, 2, 19–30, 1985.

[11] Coordinate Free LAP Reference Manual, Version 1.2, University of Washington, Northwest LIS Release 3.2, Seattle, WA, September 1988.

[12] FiltorX Version 3.3, Department of Electrical Engineering, University of Toronto, 1993.

[13] Adel S. Sedra, and Peter O. Brackett, "Filter Theory and Design: Active and Passive", MATRIX Publishers Inc. 1978.

[14] Adel S. Sedra, and Kenneth C. Smith, "Microelectronic Circuits", Third Edition, Saunders College Publishing, 1991.

[15] R. L. Geiger and E. Sanchez–Sinencio, "Active Filter Design Using Operational Transconductance Amplifiers: A Tutorial", *IEEE Circuits and Devices Magazine*, Vol. 1, pp 20–32, March. 1990.

[16] H. Khorramabadi and P. R. Gray, "High Frequency CMOS Continuous–Time Filters", *IEEE Journal of Solid–State Circuits*, Vol. SC–19, pp. 939–948, December. 1984.

[17] C. S. Park and R. Schaumann, "Design of a 4–MHz Analog Integrated CMOS Transconductance–C Bandpass Filter", *IEEE Journal of Solid–State Circuits*, Vol. SC–23, pp. 987–996, August. 1988.

[18] V. Gopinathan and Y. P. Tsividis, "A 5V 7th–Order Elliptic Analog Filter for Digital Video Applications", *IEEE Int. Solid–State Circuits Conference*. San Francisco, California, pp. 208–209, February. 1990

[19] M. Snelgrove and A. Shoval, "A CMOS Biquad at VHF", IEEE Proc. CICC–91, San Diego, PP. 9.2.1–9.2.6, May 1991.

[20] B. Nauta, "CMOS VHF Transconductance–C Lowpass Filter", *IEEE Electronics Letters*, Vol. 26, pp. 421–422, March 1990.

[21] J. M. Cohn, D. J. Garrod, R. A. Rutenbar and L. R. Carley, "KOAN/ANAGRAM II: New Tools for Device–Level Analog Placement and Routing", *IEEE Journal of Solid–State Circuits*, Vol. SC–26(3), pp. 330–342, March 1991.

[22] H. Koh, C. Sequin and P. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Transaction on Computer–Aided Design*, Vol. CAD–9(2), pp. 113–125, February 1990.

[23] R. Harjani, R. A. Rutenbar and L. R. Carley, "OPASYN: A Framework for Analog Circuit Synthesis", *IEEE Transaction on Computer–Aided Design*, Vol. CAD–8(12), pp. 1247–1266, December 1989.

[24] B. R. Owen, R. Duncan, S. Jantzi, C. Ouslis, S. Rezania and K. Martin, "BALLISTIC: An Analog Layout Language", University of Toronto. WWW Sever: *http://www.eecg.toronto.edu*.

[25] V. M. Bexten, C. Moraga, R. Klinke, W. Brockherde and K. Hess, "ALSYN: Flexible Rule–Based Layout Synthesis for Analog IC's", *IEEE Journal of Solid–State Circuits*, Vol. 28, No. 3, pp. 261–267, March. 1983.

[26] M. Banu and Y. Tsividis, "Fully Integrated Active RC Filter in MOS Technology", *IEEE Journal of Solid–State Circuits*, Vol. SC–18, pp. 644–651, December 1983.

[27] M. A. Driscoll, W. R. Daasch, and C. Sembakutti, "Efficient Design Centering of Analog Integrated Circuits Using Binary Search", *Analog Integrated Circuits and signal Processing*, Vol. 6(2), pp. 157–169.

[28] Jonathon W. Walker, "User Interface for Continuous–Time Filter Design Automation Tool", Portland State University Library, December 1996, Expectantly

[29] M. Y. Hsueh and D. O. Pederson, "Computer–aided layout of LSI building blocks", *IEEE Proceedings of the 1979 International Symposium on Circuits and Systems*, Jul. 1979, Tokyo, Japan, pp. 474–477.

APPENDIX A

CFL ROUTING FUNCTIONS ENHANCEMENT

A.1 General Description

Generally, the new version CFL is compatible with the original CFL library. Any C file, that can work with the original CFL, can still work with the new version CFL library without any change and also give out the result without any difference.

The new features of the enhanced CFL are:

1. All routing functions have the ability to run the routing with the wire width of its original crossing width, if the wire width of the input argument equals zero.
2. The border descriptor, as arguments to some of the routers, has the capability to contain not only the crossings associated with the given layer, but also the crossings associated with the contact layers that related to the given layer according to the new version technology file. ✺
3. New user functions let the user exclude or include a certain crossing of certain layer or all crossings of certain layer in the resulting border descriptor. ✺
4. New user function to switch the active layer in the point descriptor, so the CFL has the ability to run the point–to–point routing with the layer that is associated with the original layer of the point. ✺

✺ This feature can only work when the new format technology file is supplied.

For some of the new features, a new format technology file is necessary. If the new format technology file is not supplied, the program will show the message,

No related fields assigned to contacts in the technology file *techfilename*

CONTINUE !

And the program will keep going without the new feature (with ✺ mark) showing up. The new operators can not work without the new format technology file supplied. The ap-

plication of these new operators, without the new format technology file supplied, will cause the program exit with a warning.

A.2 Application

All routing operators in the new version CFL, have the capability to run the routing with the wire width of the original width of the crossing, when the wire width arguments equal zero.

For routers:

```
pp(sym0, pt1, pt2, w)
pr(sym0, b1, b2, w)
ext(b, d, w)
elb(sym0, b1, b2, w1, w2, ct, rev)
```

with zero wire width input (w, w1, w2), the routing wire width will be the widths of the crossings in the descriptors (pt1, pt2, b, b1, b2). If any co–linear wire is necessary in the routing, the minimum width of each couple crossing is chosen for the co–linear wire width.

For routers:

```
plx(sym0, pt1, pt2, w, ct)
ply(sym0, pt1, pt2, w, ct)
tee(sym0, b1, b2, w, ct, rev)
```

with zero wire width input (w), the routing wire width will be the crossing widths of the first crossing descriptors (pt1, b1).

In the new version CFL, a new attribute setting command is added into the function cflsetc(a, v) . The user can set the attribute as:

cflsetc("ctinclude", "yes");

or

cflsetc("ctinclude", "no");

When the 'ctinclude' attribute is set to 'yes', the border descriptor will include the contact crossings that is related to the given layer, when a new format technology file is supplied. If the 'ctinclude' attribute is set to 'no', only the crossings of the given layer are included in the border descriptor, even though a new format technology file is supplied. The default status of the 'ctinclude' attribute is set to 'no'.

Example:

If we compose the program like following:

```
BORDER *b1;

cflsetc("ctinclude", "no");
b1(symbol0, "top", "metal2");
```

there will be only metal 2 crossings present in the b1. But if the program is as follows:

```
BORDER *b2;

cflsetc("ctinclude", "yes");
b2(symbol0, "top", "metal2");
```

the border descriptor b2 will include the crossings of metal 2 and the crossings of metal 2 contact on the top of symbol0.

New functions:

```
BORDER *b1;

bdexl(b1, layername, i);
bdinl(b1, layername, i);
```

The new operators **bdexl** and **bdinl** are provided for excluding and including specific crossing ordinals of the given layer from border descriptors. In general then, the border description facilities are capable of directing the routers to consider any subset of crossings along the side of a particular symbol

In the special instance that the ordinal argument **i** is zero, **bdexl** will exclude all crossings of a given layer from the resulting border, and **bdinl** will include all crossings of a given layer in the resulting border.

To obtain the number of crossings of certain layer currently in a border b1:

```
n = nbclayer(b1, layername, k);
```

When **k** is **TRUE** (**1**), the number of active crossings of the given layer is returned. If **k** is **FALSE** (**0**), the number of total crossings of the given layer is returned.

New function:

```
PT *pt;
```

ptlsw(pt, layername);

This operator is provided for switching the active layer of the point descriptor to the given layer. Then the point description facilities are capable of directing the routers, such as the point–to–point router, to perform the routing function with different, but reasonable layers, rather than only the original layer of the point descriptor.

If the '**layername**' argument equals '**reset**', then the active layer of this point descriptor will switch back to the original layer.

The given layer must be related to the original layer of the point descriptor according to the new format technology file. Otherwise, the program will exit with a warning.

A.3 List of New Functions

```
BORDER *bdexl(b1, layername, i)          /* Exclude i of layer from border descriptor */

BORDER *b1;
char       *layername;
int        i;
```

```
BORDER *bdinl(b1, layername, i)          /* Include i of layer from border descriptor */

BORDER *b1;
char       *layername;
int        i;
```

```
PT *ptlsw(pt, layername)                 /* Switch active layer in pointer descriptor */

PT *pt;
char *layername;
```

```
int nbclayer(b, layername, k)            /* Obtain the number of crossings of this layer
                                            in the border descriptor */

BORDER *b;
char       *layername;
int        k;
```

APPENDIX B

DEFINITION OF OTACC STACK ELEMENT STRUCTURE

```
typedef struct Stackelement *sptr
typedef struct Stackelement {
    filterkey type;              /* Filter type.                                  */
    SYMBOL *current;             /* Current filter                                */
    SYMBOL *cell;                /* Transconductance Tile                         */
    SYMBOL *Gf;                  /* "Forward" OTA with route tiles                */
    SYMBOL *Gf;                  /* "Backward" OTA with route tiles               */
    SYMBOL *unitC;               /* Unit capacitor tile                           */
    SYMBOL *nullC;               /* Non–capacitor tile                            */
    SYMBOL *acisolator;          /* AC signal isolator tile                       */
    SYMBOL *compressor;          /* AC signal compressor tile                     */
    SYMBOL *floatcombm;          /* Metal1 comb tile for float capacitor          */
    SYMBOL *floatcombp;          /* Poly comb tile for float capacitor            */
    SYMBOL *dcCenter;            /* Additional DC signal route tile               */
    SYMBOL *dcEdge;              /* DC signal connection stubs                    */
    SYMBOL *dcRib;               /* DC signal route tile                          */
    SYMBOL *ldRib;               /* Special DC route tile for load                */
    SYMBOL *combinP;             /* Positive input capacitor grid array route     */
    SYMBOL *combinN;             /* Negative input capacitor grid array route     */
    SYMBOL *comboutP;            /* Positive output capacitor grid array route    */
    SYMBOL *comboutN;            /* Negative output capacitor grid array route    */
    SYMBOL *zeroC;               /* Capacitor tile for placing "zeros" in filter  */
    SYMBOL *zCenter;             /* "zero" capacitor route tile                   */
    SYMBOL *zEdge;               /* "zero" capacitor connection stubs             */
    zeroinfo zero[10];           /* Array of "zero" capacitors                    */
    int sepinfo[10][2];          /* Array for separation information              */
    float GfCi;                  /* Input C parasitic of forward OTA tile         */
    float GfCo;                  /* Output C parasitic of forward OTA tile        */
    float GbCi;                  /* Input C parasitic of backward OTA tile        */
    float GbCo;                  /* Output C parasitic of backward OTA tile       */
    float unitcap;               /* Capacitance of a single unit cap tile         */
    float unitcapzero;           /* Capacitance of a single "zero" cap tile       */
```

```
    float unitparasitic;        /* Parasitic capacitance of unit capacitor    */
    float circuitcap;           /* Circuit capacitance in assembling          */
    float GmValue;              /* Gm value of the OTA                        */
    float FT;                   /* Cut off frequency of OTA                   */
    RUN run;                    /* Number of times program go through         */
    DIR direction;              /* Final layout direction in X or Y           */
    int Hlimit;                 /* Vertical size limit of filter layout       */
    int Wlimit;                 /* Horizontal size limit of filter layout     */
    int SizeLimit;              /* Size limit for running layout              */
    int NOcolumn;               /* Number of column in the final layout       */
    int RoutingSpace;           /* Size of the routing channel                */
    int columnNO;               /* Column No. which is currently working on   */
} stackelement;

typedef struct Zeroinfo *zptr;
typedef struct Zeroinfo {
    SYMBOL *array;              /* Transmission zero capacitor array          */
    int height;                 /* Current height of the backbone             */
    int crossing[2];            /* M2 crossing number for final assembly      */
    int Znumber;                /* The number of transmission of the filter   */
} zeroinfo;

typedef enum {
    UNKNOWN = 0x1,
    SINGLE = 0x2,
    BALANCED = 0x4,
    GROUNDED = 0x8,
    FLOATING = 0x16
} filterkeys;

typedef enum {
    FIRST = 0x0,
    SECOND = 0x1
} RUN;

typedef enum {
    DIR_X = 0x1,
    DIR_Y = 0x2
} DIR;
```

APPENDIX C

OTACC COMMAND OPTIONS

OTACC

[–io common]	Input directory and output path in common Default: mag/
[–i inpath]	Input directory instead of default one Default: mag/
[–o outdir]	Output directory instead of default one Default: mag/
[–c celltechfile]	Technology file of OTA instead of default one Default: cell.tech in OTACC_path or with OTA cell
[–f frequency]	ω_0 of the filter in Hertz Default: 1MHz
[–h height_limit]	Height limit of layout Default: No limitation
[–w width_limit]	Width limit of layout Default: No limitation
[–s style]	Layout style, TRANS or CIS Default: Arbitrary
[–d direction]	Layout growth in X or Y Default: Growth in Y
[–n normalization]	YES for normalized, NO for non–normalized ladder Default: Normalized
[–g Gmvalue]	Gmvalue update, instead of default one in techfile Default: The value from OTA cell technology file
[–v Trace_log]	YES for trace log printing, NO for no printing Default: Turn off the trace log printing
[infile]	Input ladder description file Must be specified

APPENDIX D

TECHNOLOGY FILES FOR DIFFERENT OTAS

The technology files for three OTA cells which currently exist in the cell library are shown as follows. These three OTA cells have different performance characters and are designed with different properties. For example, OTA II and OTA III are designed with Negative Resistance Load control capability (ctr_NRL nodes) to improve the output impedance of OTAs. And OTA I is designed with phase offset control capability (ctr_ph node) to adjust the phase shift of OTA.

D.1 OTA–Technology File For OTA I

```
CELL        OTAP                  # OTA cell designed by Pan Wu.
Technology  n3cn.prm              # Technology file name.
IOtype      Din Dout              # Differential Input and differential output.

Power       Vdd    5              # Power supply in Volts.
Power       Vss    –5             # Power supply in Volts.
Bias        V1     0              # Bias in Volts.
Bias        –V1    0              # Bias in Volts.
Bias        V2     –2.68          # Bias in Volts.
ctr_gm      IS     1.0   3.2      # g_m control, in Volts.
ctr_ph      Vc     0              # Phase control, in Volts

gm          135e–6   290e–6       # g_m value in Siemens.
ft          1.345e8               # Cut off frequency in Hertz.
CMRR        62.6 (10MHz)          # Common Mode Rejection Ratio in dB at 10 MHz.
```

```
PSRR      50.7 (10MHz)      # Power Supply Rejection Ratio in dB at 10 MHz.
THD       1.59 (1MHz)       # Total Harmonic Distortion in percentage at 1 MHz.

Cipos     0.0856            # Input parasitic capacitance (to Ground) in pF.
Cineg     0.078             # Input parasitic capacitance (to Ground) in pF.
Cid       0.0164            # Input parasitic capacitance (Differential) in pF.
Copos     0.470             # Output parasitic capacitance (to Ground) in pF.
Coneg     0.478             # Output parasitic capacitance (to Ground) in pF.
Cod       0.0096            # Output parasitic capacitance (Differential) in pF.
ripos     1e8               # Input resistance in Ohms.
rineg     1e8               # Input resistance in Ohms.
ropos     283.0e3           # Output resistance in Ohms.
roneg     283.0e3           # Output resistance in Ohms.

W         296               # Width of the transconductance cell in μm.
H         267               # Height of the transconductance cell in μm.

P1sub     1e–5              # Unit parasitic capacitance of Poly1 to GND in
                            #  pF/U^2.
p2p1cunit 0.000744          # Unit capacitance of Poly2 to Poly capacitor in
                            #  pF/U^2.
```

D.2 OTA–Technology File For OTA II

```
CELL      OTAD              # OTA cell designed by David Chiang.
Technology n52w.prm         # Technology file name.
IOtype    Din Dout          # Differential Input and differential output.

Power     Vdd   5           # Power supply in Volts.
Bias      bi    1.13        # Bias in Volts.
ctr_gm    Vf    0.7   1.9   # g_m control, in Volts.
ctr_NRL   Vq    1.5   2.5   # Negative Resistance Load control, in Volts

gm        128e–6  397e–6    # g_m value in Siemans.
ft        1e8               # Cut off frequency in Hertz.
CMRR      81.5 (10MHz)      # Common Mode Rejection Ratio in dB at 10 MHz.
```

```
PSRR      71.5 (10MHz)      # Power Supply Rejection Ratio in dB at 10 MHz.
THD       2.33 (1MHz)       # Total Harmonic Distortion in percentage at 1 MHz.

Cipos     0.080             # Input parasitic capacitance (to Ground) in pF.
Cineg     0.0856            # Input parasitic capacitance (to Ground) in pF.
Cid       0.0128            # Input parasitic capacitance (Differential) in pF.
Copos     0.333             # Output parasitic capacitance (to Ground) in pF.
Coneg     0.483             # Output parasitic capacitance (to Ground) in pF.
Cod       0.022             # Output parasitic capacitance (Differential) in pF.
ripos     1e8               # Input resistance in Ohms.
rineg     1e8               # Input resistance in Ohms.
ropos     1.1e6             # Output resistance in Ohms.
roneg     1.1e6             # Output resistance in Ohms.

W         180               # Width of the transconductance cell in μm.
H         203               # Height of the transconductance cell in μm.

P1sub     1e–5              # Unit parasitic capacitance of Poly1 to GND in
                            # pF/U^2.
p2p1cunit 0.000744          # Unit capacitance of Poly2 to Poly capacitor in
                            # pF/U^2.
```

D.3 OTA–Technology File For OTA III

```
CELL      OTAG              # OTA cell designed by Girish Chandrasekaran.
Technology n52w.prm         # Technology file name.
IOtype    Din Dout          # Differential Input and differential output.

Power     Vdd    5          # Power supply in Volts.
Bias      Vbias  1.2        # Bias in Volts.
ctr_gm    Vcf    0.8  1.3   # g_m control, in Volts.
ctr_NRL   Vcq    2.60 2.85  # Negative Resistance Load control, in Volts

gm        85e–6  280e–6     # g_m value in Siemans.
ft        1e8               # Cut off frequency in Hertz.
CMRR      206.9 (10MHz)     # Common Mode Rejection Ratio in dB at 10 MHz.
PSRR      94.4 (10MHz)      # Power Supply Rejection Ratio in dB at 10 MHz.
```

THD	4.34 (1MHz)	# Total Harmonic Distortion in percentage at 1 MHz.
Cipos	0.076	# Input parasitic capacitance (to Ground) in pF.
Cineg	0.089	# Input parasitic capacitance (to Ground) in pF.
Cid	0.012	# Input parasitic capacitance (Differential) in pF.
Copos	0.223	# Output parasitic capacitance (to Ground) in pF.
Coneg	0.220	# Output parasitic capacitance (to Ground) in pF.
Cod	0.013	# Output parasitic capacitance (Differential) in pF.
ripos	1e8	# Input resistance in Ohms.
rineg	1e8	# Input resistance in Ohms.
ropos	3.44e6	# Output resistance in Ohms.
roneg	3.44e6	# Output resistance in Ohms.
W	213	# Width of the transconductance cell in μm.
H	154	# Height of the transconductance cell in μm.
P1sub	1e–5	# Unit parasitic capacitance of Poly1 to GND in # pF/U^2.
p2p1cunit	0.000744	# Unit capacitance of Poly2 to Poly capacitor in # pF/U^2.

APPENDIX E

RETURN CODES OF OTACC

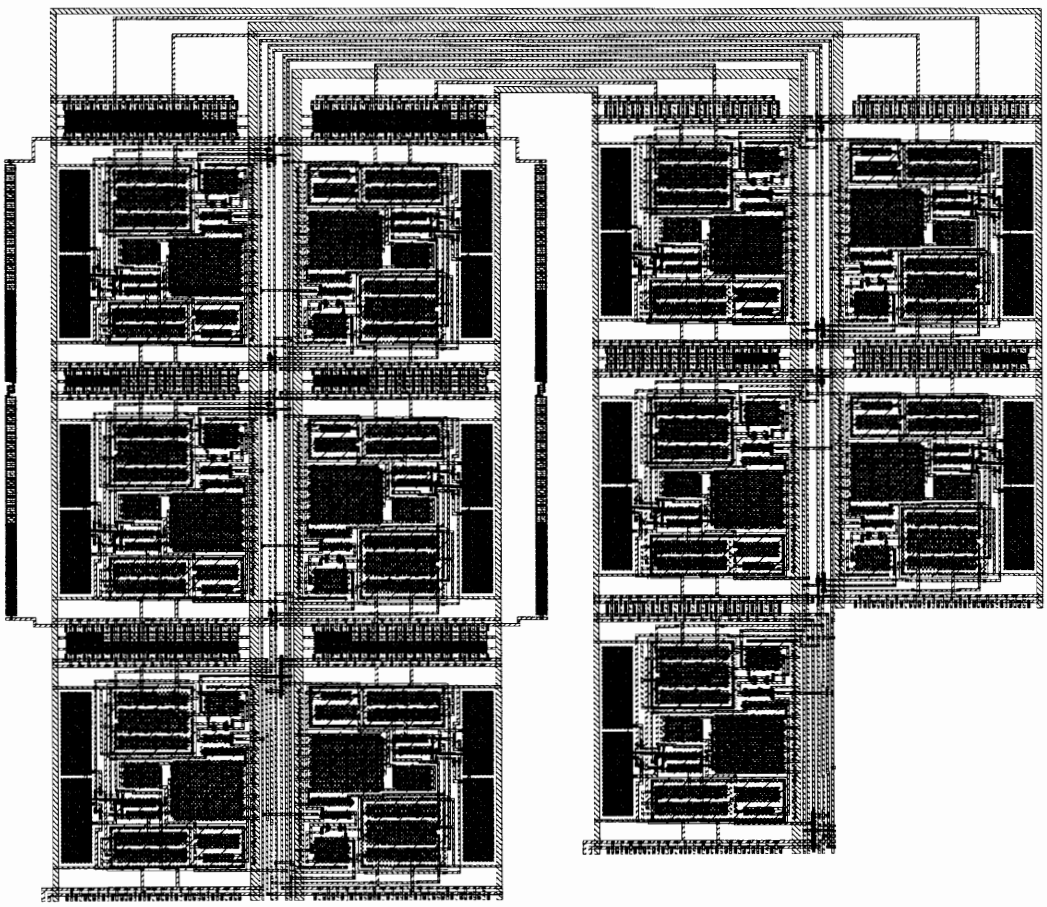
```
# Return Massage code

RTcode = 0x0;          # Normal exit

# Non–fatal Error Code
RTcode = 0x1;          # Width of the layout exceeds its limitation.
RTcode = 0x2;          # Height of the layout exceeds its limitation.
RTcode = 0x4;          # Warning for less Precision of capacitor tile,
                       # Number of unit capacitors less than 10.
RTcode = 0x8;          # Reserved.
RTcode = 0x16;         # Reserved.
RTcode = 0x32;         # Reserved.
RTcode = 0x64;         # Reserved.

# Fatal Error Code
RTcode = 130;          # Unknown option keyword in command line.
RTcode = 131;          # User specified an illegal height limit in command
                       # line.
RTcode = 132;          # User specified an illegal width limit in command
                       # line.
RTcode = 133;          # User specified an unknown layout style.
RTcode = 134;          # User specified an unknown layout direction.
RTcode = 135;          # User specified an unknown normalization option.
RTcode = 136;          # User specified an illegal working frequency ω0.
RTcode = 137;          # Technology file neither be specified nor found in
                       # default.
RTcode = 138;          # Can not open tech. file.
RTcode = 140;          # General format error in ladder description file.
RTcode = 141;          # General format error in tech. file.
RTcode = 150;          # Technical error in ladder description file.
```

```
RTcode = 160;       # Failed to allocate memory.
RTcode = 161;       # Stack overflow.
RTcode = 162;       # Stack blown, important element missing in stack.
RTcode = 170;       # Gm cell not suitable for this design, Number of
                    # unit capacitors less than 1.
RTcode = 171;       # Important parameter missing in technology file,
                    # gm value or capacitance value.
RTcode = 173;       # Unknown type of unit capacitor.
RTcode = 180;       # Number of signal wire does match the type of
                    # OTA.
RTcode = 200;       # Internal error, improper function calling.
```

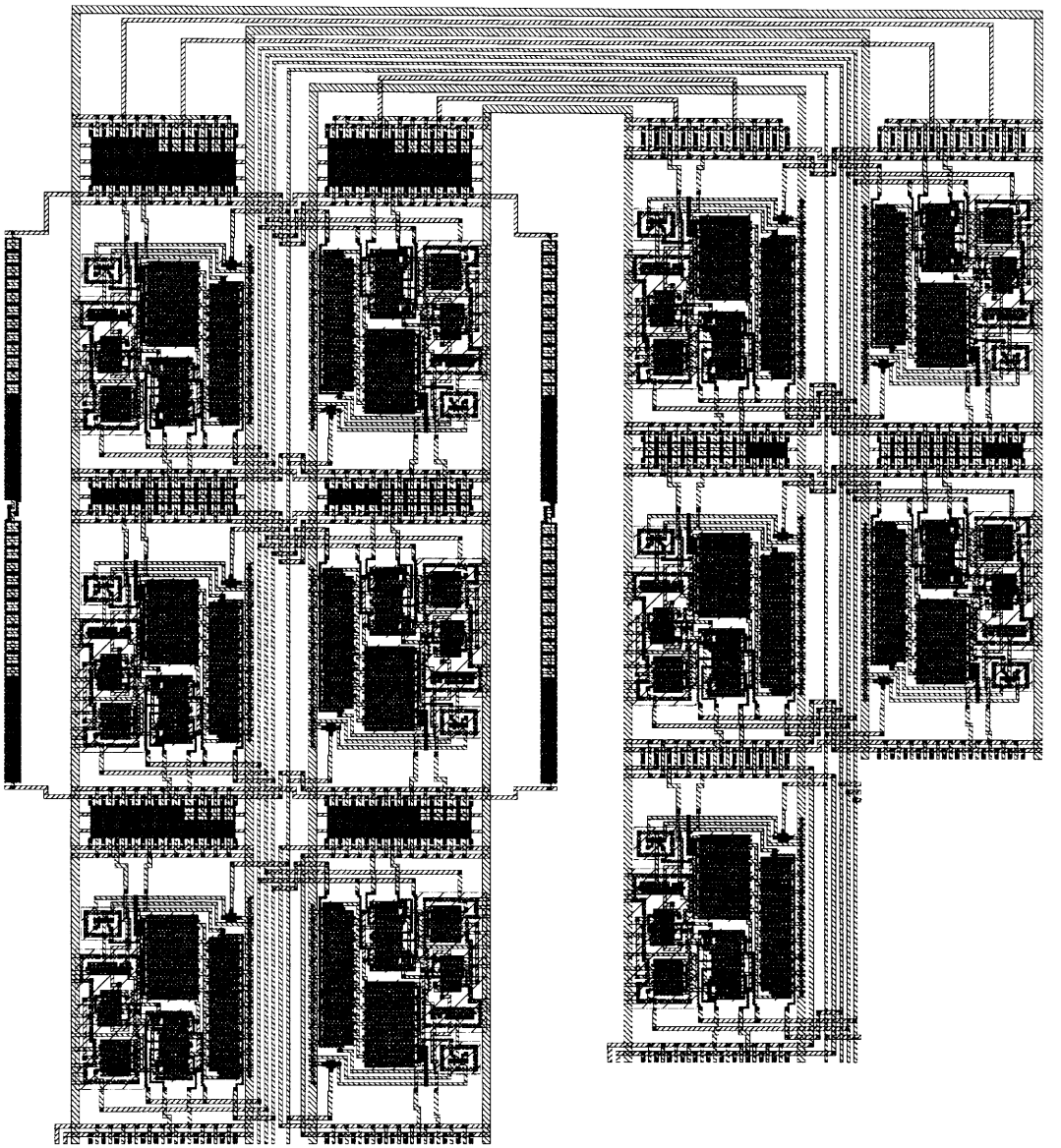
APPENDIX F

LAYOUT OF DIFFERENT FILTERS

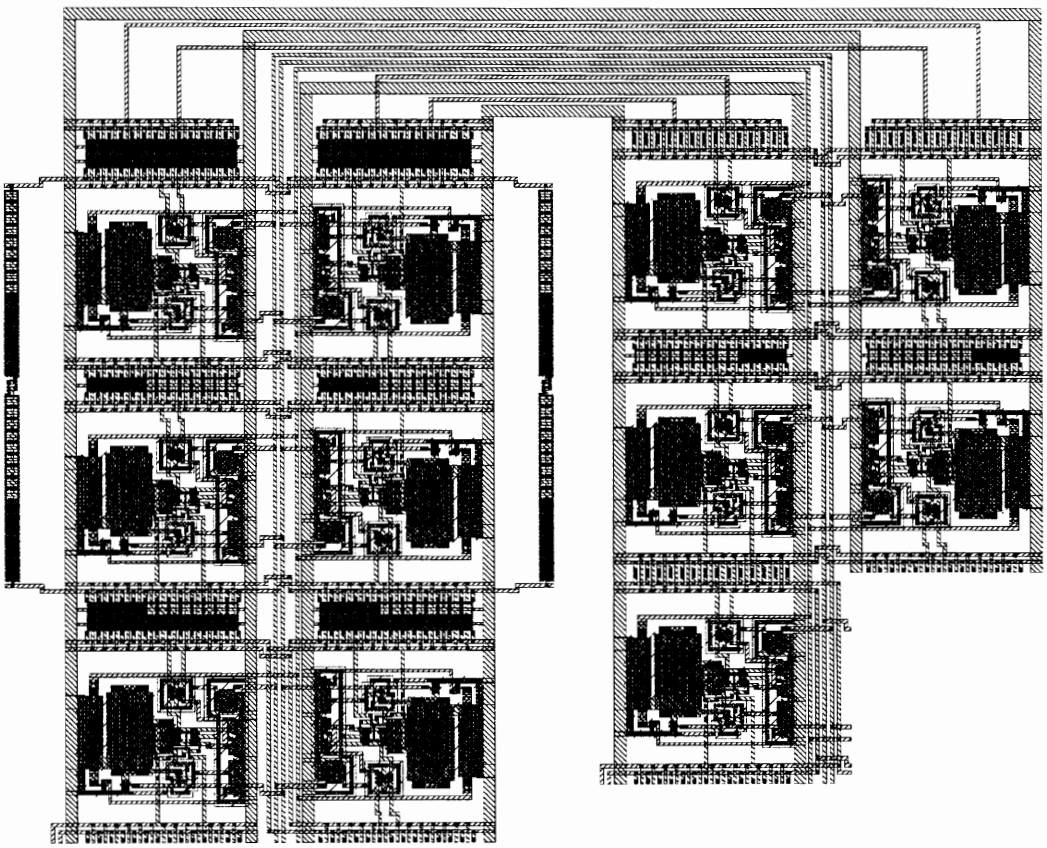
F.1 Layout of Fourth–Order Elliptic LC Lowpass Filter Using OTA I



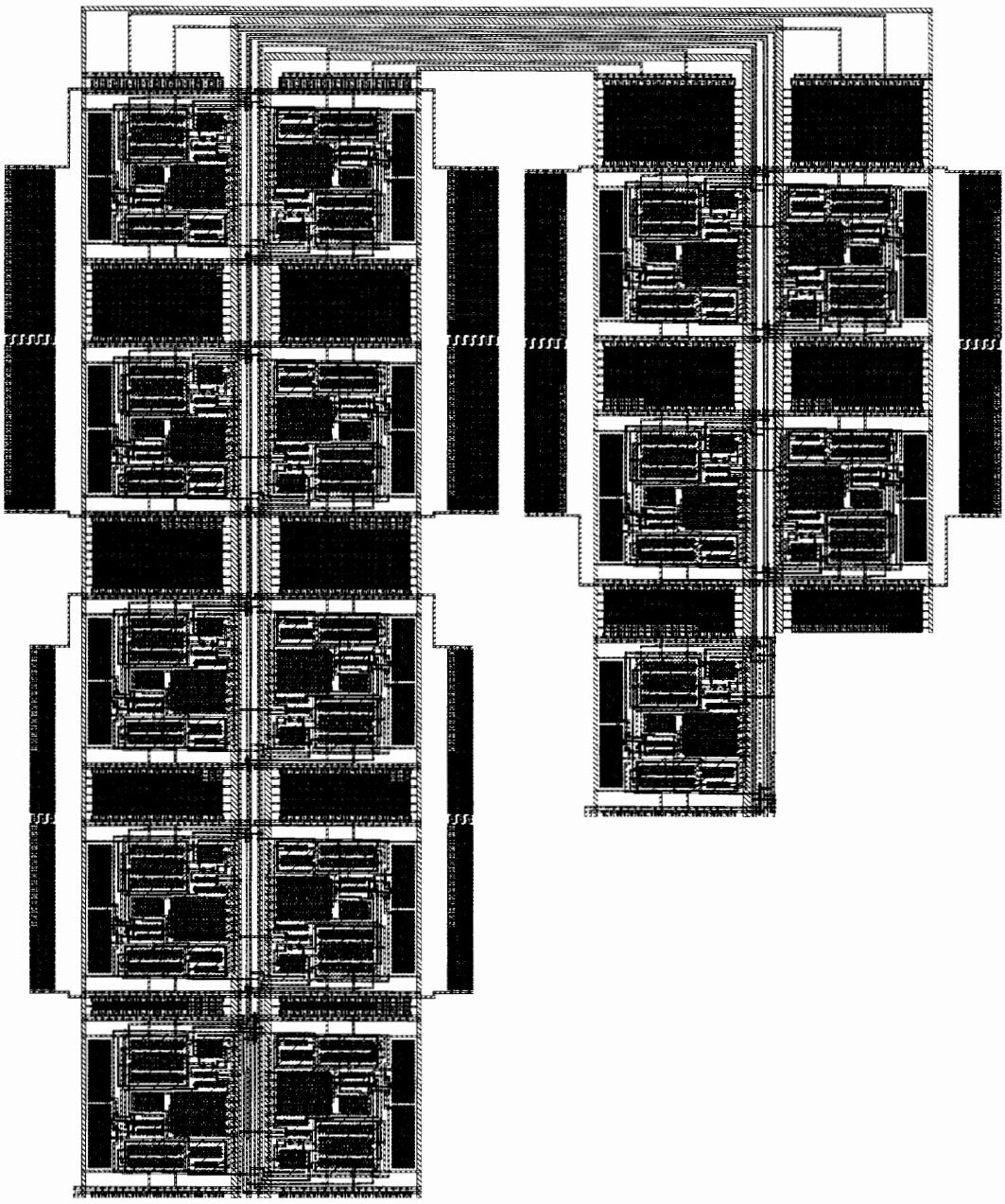
F.2 Layout of Fourth–Order Elliptic LC Lowpass Filter Using OTA II



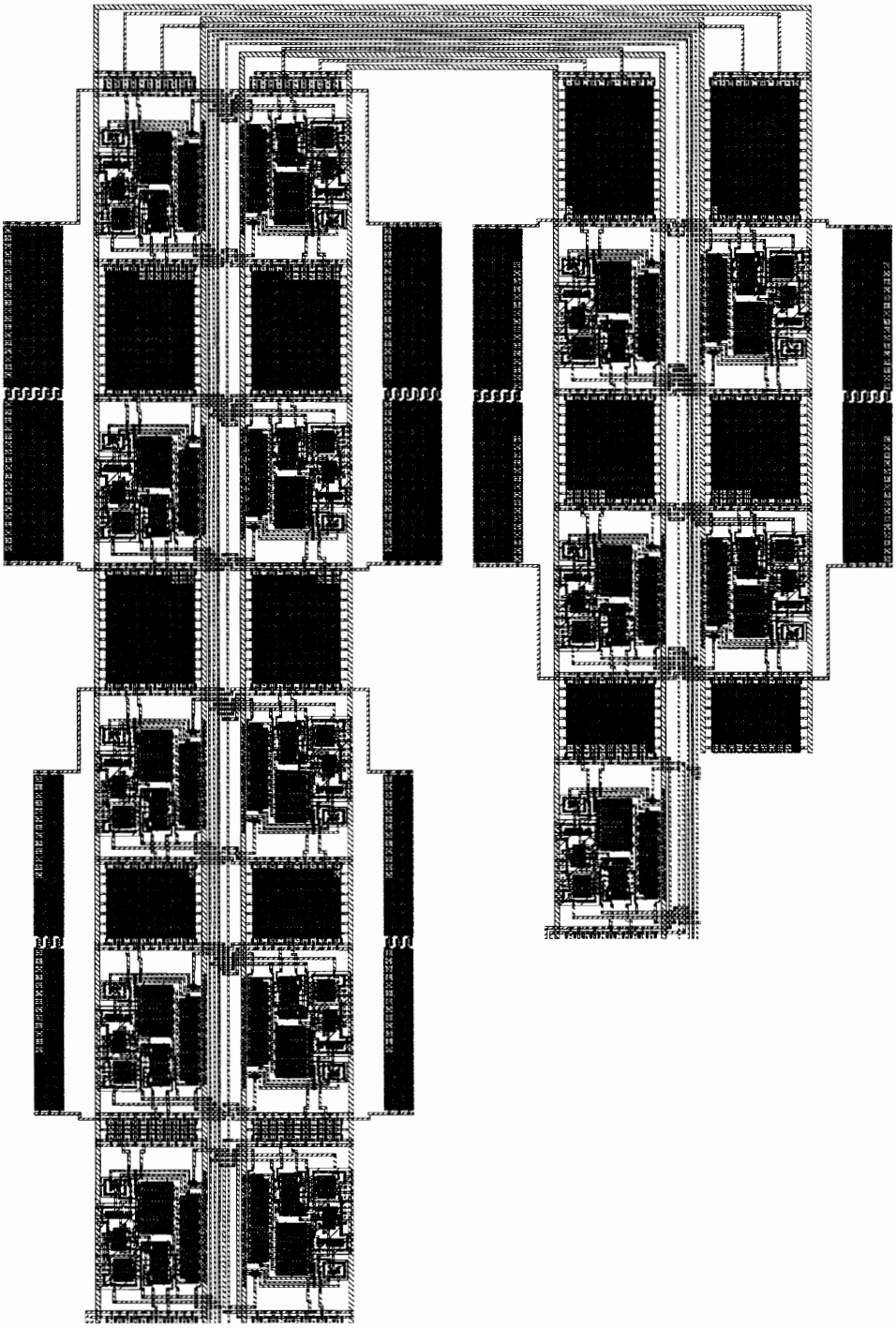
F.3 Layout of Fourth–Order Elliptic LC Lowpass Filter Using OTA III



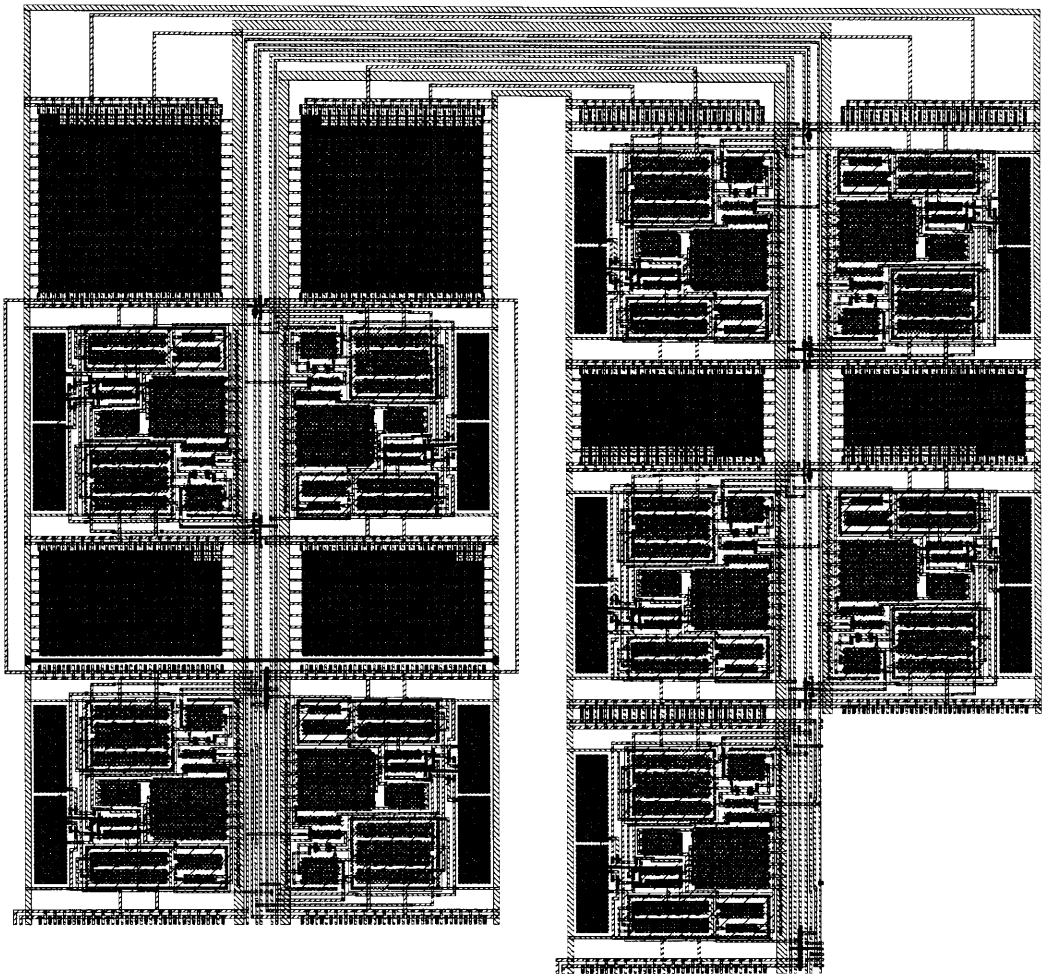
F.4 Layout of Eighth–Order Inverse Chebyshev Lowpass Filter Using OTA I



F.5 Layout of Eighth–Order Inverse Chebyshev Lowpass Filter Using OTA II



F.6 Layout of Third–Order Bandpass Filter Using OTA I



F.7 Layout of Third–Order Bandpass Filter Using OTA II