ABSTRACT

Title of Document:	HIGH-PERFORMANCE FPAA DESIGN FOR HIERARCHICAL IMPLEMENTATION OF ANALOG AND MIXED-SIGNAL SYSTEMS Hu Huang, Doctor of Philosophy, 2007
Directed By:	Professor Martin Peckerar Professor Joseph Bernstein Department of Electrical & Computer Engineering

The design complexity of today's IC has increased dramatically due to the high integration allowed by advanced CMOS VLSI process. A key to manage the increased design complexity while meeting the shortening time-to-market is design automation. In digital world, the field-programmable gate arrays (FPGAs) have evolved to play a very important role by providing ASIC-compatible design methodologies that include design-for-testability, design optimization and rapid prototyping. On the analog side, the drive towards shorter design cycles has demanded the development of high performance analog circuits that are configurable and suitable for CAD methodologies.

Field-programmable analog arrays (FPAAs) are intended to achieve the benefits for analog system design as FPGAs have in the digital field. Despite of the obvious advantages of hierarchical analog design, namely short time-to-market and low non-

recurring engineering (NRE) costs, this approach has some apparent disadvantages. The redundant devices and routing resources for programmability requires extra chip area, while switch and interconnect parasitics cause considerable performance degradation. To deliver a high-performance FPAA, effective methodologies must be developed to minimize those adversary effects.

In this dissertation, three important aspects in the FPAA design are studied to achieve that goal: the programming technology, the configurable analog block (CAB) design and the routing architecture design. Enabled by the Laser Makelink^{TM} technology, which provides nearly ideal programmable switches, channel segmentation algorithms are developed to improve channel routability and reduce interconnect parasitics. Segmented routing are studied and performance metrics accounting for interconnect parasitics are proposed for performance-driven analog routing. For large scale arrays, buffer insertions are considered to further reduce interconnection delay and cross-coupling noise. A high-performance, highly flexible CAB is developed to realized both continuous-mode and switched-capacitor circuits. In the end, the implementation of an 8-bit, 50MSPS pipelined A/D converter using the proposed FPAA is presented as an example of the hierarchical analog design approach, with its key performance specifications discussed.

HIGH-PERFORMANCE FPAA DESIGN FOR HIERARCHICAL IMPLEMENTATION OF ANALOG AND MIXED-SIGNAL SYSTEMS

By

Hu Huang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:

Professor Martin Peckerar, Chair
Professor Joseph Bernstein
Professor Neil Goldsman
Professor Pamela Abshire
Professor Ali Mosleh

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Microelectronics technology has allowed a continuously increasing integration complexity. With today's advanced CMOS VLSI process, more and more complete systems that previously require one or multiple printed circuit boards (PCB) are being fabricated on a single chip. Examples of such systems-on-a-chip (SoC) in recent years are the new generations of telecommunication systems that include analog, digital and even radio-frequency (RF) sections on one chip [1.1], WiMAX wireless broadband platforms [1.2] and completely integrated DVD systems [1.3].

While most functions in such integrated systems are implemented by digital circuitry, there are some typical functions that will always remain analog [1.4]. Since all natural signals are analog – at a macroscopic level, mixed-signal circuits like sample-and-hold (S/H), analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) are required to interface the real world to the digital world. Moreover, analog signals usually need to be filtered and amplified to allow digitization with sufficient signal-to-noise ratio (SNR), or drive the outside load. Typical analog circuits used here are buffers, low-noise amplifiers (LNA), variable-gain amplifiers (VGA), filters, oscillators and mixers. In addition, all above circuits

need precise, stable voltage, current and timing references for their operation, which are generated by analog circuits too. A representative floor plan of a SoC is shown in Figure 1.1.

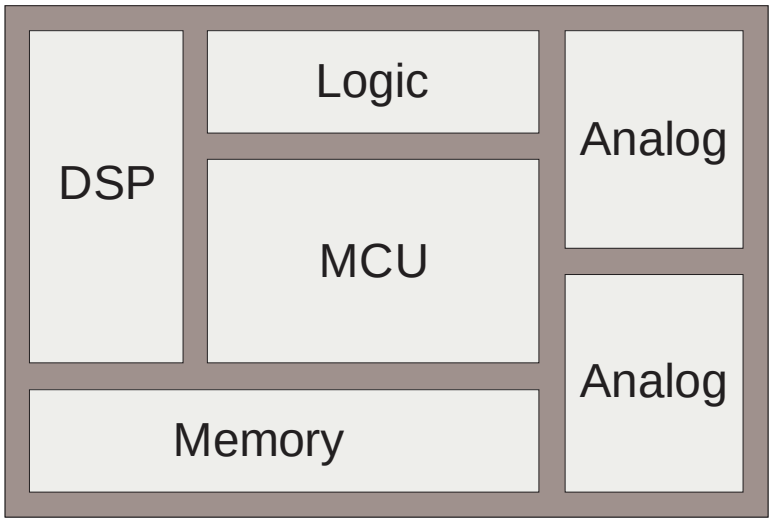


Figure 1.1 Representative floor plan of a System-on-a-Chip

Due to higher integration, more complex system architectures and signal processing algorithms, the design complexity of today's IC has increased dramatically. At the same time, many application-specific integrated circuit (ASIC) and application-specific stand part (ASSP) for consumer electronics, telecom and computer markets are characterized by shortening product lifecycles and tightening time-to-market requirements. If the initial market window was missed, the product can be totally out of competition.

A key to manage the increased design complexity while meeting the shortening time-to-market requirement is design automation with computer-aided design (CAD) tools. In the digital domain, today's CAD tools are fairly well developed and

commercially successful. The system can be described using a hardware description language such as VHDL or Verilog, either at the behavioral or structural level. Various synthesis tools can then translate the HDL specifications into a gate-level netlist, and physical design tools (place & rout) map the netlist into a mask-level layout based on a cell library specific for the selected technology. In recent years, the field-programmable gate arrays (FPGAs) have evolved to play a very important role in digital design by providing ASIC-compatible design methodologies that include design-for-testability, design optimization and rapid prototyping allowing the engineers to have direct and immediate access to all resources in the system while avoiding the encapsulation of standard parts and the high cost of ASICs [1.5]. The time-to-market pressures and low financial risk has made FPGAs and complex programmable logic devices (CPLDs) an increasingly popular vehicle for prototyping and, in many cases, actual production.

The story on the analog side is quite different. Due to the wide variety of components required for analog systems, the continuous nature and variable levels of analog signals, the analog design in general is perceived as less systematic and more knowledge-intensive than digital design. Unlike the digital systems, which can naturally be represented in terms of Boolean constructs, the larger variety of analog circuit topologies and the number of conflicting requirements make a unified description of analog functions very difficult. In addition, the analog circuits are more sensitive to non-idealities and all kinds of high-order effects and parasitic disturbances. Therefore, although analog circuits typically occupy only a small

fraction of the total mixed-signal IC, their design is often the bottleneck in both time and test cost.

Despite of those adversities, analog CAD and design automation over the past two decades has been a field of profound academic and industrial research activity, resulting in a slow but steady progress [1.6]. The simulation area has been well developed since the advent of SPICE. Analog circuit synthesis has recently shown promising results at the research level [1.7], and the development of analog and digital hardware description languages like VHDL-AMS and Verilog-A/MS is intended to provide a unifying trend needed in designing mixed signal ICs and SoCs of the future. However, the need of efficient analog system design methodologies beyond individual tools has also been clearly identified [1.8]. Particularly, the drive towards shorter design cycles for analog integrated circuits has demanded the development of high performance analog circuits that are configurable and suitable for high-level CAD methodologies, just like FPGAs in the digital world.

1.1 Hierarchical Analog Design

Hierarchical, or structured design, was already used by hardware designers in the late 1970s, when the increasing complexity of full-custom design for products such as microprocessors created serous bottlenecks. Like the traditional “divide and conquer” engineering methods adopted by software engineers in building complex software products, ideas such as modularization, information hiding and stepwise refinement were applied to VLSI design to allow more structured descriptions of the work, particularly when it contains iterated or conditional features. Digital systems

designed with FPGAs are typical examples of hierarchical design, where the circuits are divided into sub-circuits that can be realized with the building cells from the library provided by the vendor, either automatically or by the custom users. Then the place and route tools are used to map those sub-circuits to the basic logic elements (BLEs) and set the switches to make necessary connections.

Hierarchical analog design shares the same ideal. Figure 1.2 shows a hierarchical decomposition of a conceptual analog front-end processing unit.

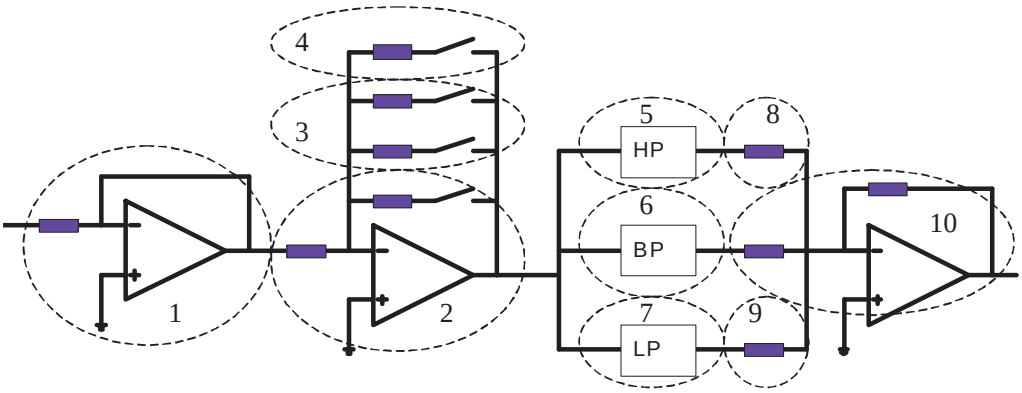
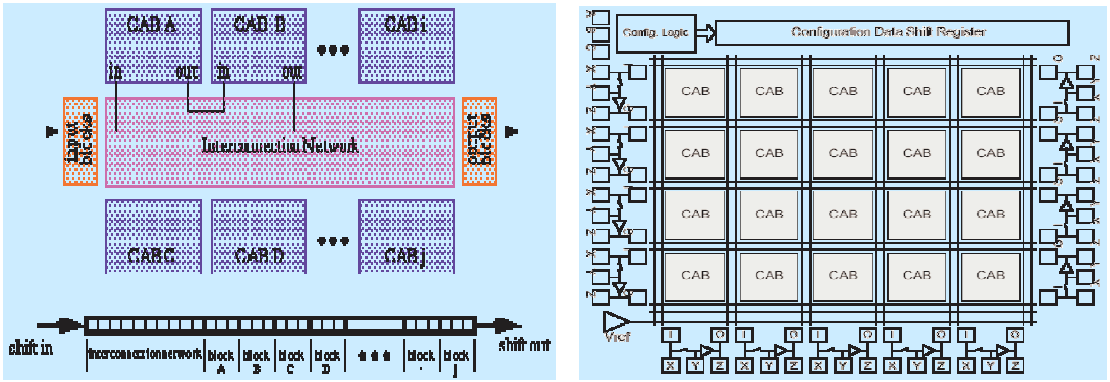


Figure 1.2 Hierarchical partitions of analog circuits

For the same reasons that lagged the development of analog design CAD tools, hierarchical analog design hasn't made as much progress as in the digital world. The advent of Field-programmable analog arrays (FPAAs) is intended to change this situation. Composed of configurable analog blocks (CABs) than can realized high-level analog functions like amplifiers, filters and references, and programmable interconnects to link them up and implement more complex systems, FPAAs are proposed as a straightforward vehicle for hierarchical analog system design. Steady

progress made at academic institutes ([1.9], shown in Figure 1.3a, [1.10]-[1.11]), and commercial products introduced recently ([1.12]-[1.13], [1.14], shown in Figure 1.3b) indicate renewed interest and further accomplishment in achieving this goal. However, the functionalities they can implement are still relatively limited and the signal bandwidth they can process is quite small (maximum 2MHz). A general purpose FPAA with good supporting CAD tool suitable for high frequency applications has not yet appeared.



(a) (b)

Figure 1.3 Examples of existing FPAA diagrams

1.2 Motivation of This Study

The advantages of hierarchical approach in analog design are obvious. The post-fabrication configurability and usage of high-level CAD methodologies make short time-to-market possible. By using pre-qualified software and hardware components, the non-recurring engineering (NRE) costs are greatly reduced. It is also a perfect way to build prototype systems that allow quick verification. In the meantime, this

approach has some disadvantages apparent, namely, the extra chip area for redundant devices and routing resources required by programmability, along with the performance degradation due to switch and interconnect parasitics. To deliver a high-performance FPAA, effective methodologies must be developed to minimize those adversary effects. In this dissertation, three important aspects in the FPAA design are investigated to achieve that goal: the programming technology, the configurable analog block (CAB) design and the routing architecture design, which are discussed in greater details below.

1.2.1 Programming Technology

Among the currently available programming technologies, SRAM programming is the most popular one [1.15], which uses memory cells to control pass transistors, multiplexers or tri-state buffers. However, the large space required by the memory cells and substantially high resistance make it not a good choice for FPAA. Anti-fuse programmed FPGA uses metal-metal plane capacitors with a very thin layer of amorphous silicon as insulation layer, producing smaller size and lower resistance (about 100 - 600Ω) links [1.16]. However, it is not compatible with standard CMOS process. It also needs programming voltage higher than standard, and has intolerable leakage current. The EPROM/EEPROM are re-programmable without requiring external storage, but they normally consume a large chip area and require multiple voltage sources, which might not otherwise be required [1.17].

For analog applications, ideally we need a programmed switch to behave just like a short metal wire. MakeLink^{TM}, a laser-induced metal-to-metal antifuse technology, is the only viable candidate for this purpose [1.18]. Experimental results show that for

the link structure of $4 \times 4\ \mu m^2$ holes and 3μm line, the average link resistance was found to be approximately 1.8Ω [1.19]. The principle of this technology is schematically illustrated in Figure 1.4. The top two levels of metals in a standard CMOS process are used as the upper frame and the lower line. When the lower metal line is impinged by laser beam, its temperature increases rapidly due to the absorption of the laser energy, but the temperature of the dielectric between those two metals will not changed much because of its low thermal conductivity and light absorbency. The stresses concentrating at the upper corner of the lower metal cause the initiation of a crack toward upper metal due to the thermal conductivity mismatch between the metal and the dielectrics, and molten metal simultaneously fills in the crack to form the link sheet. Figure 1.5 shows a FIB cross sectional image of a laser-induced vertical link between the lower and upper metal.

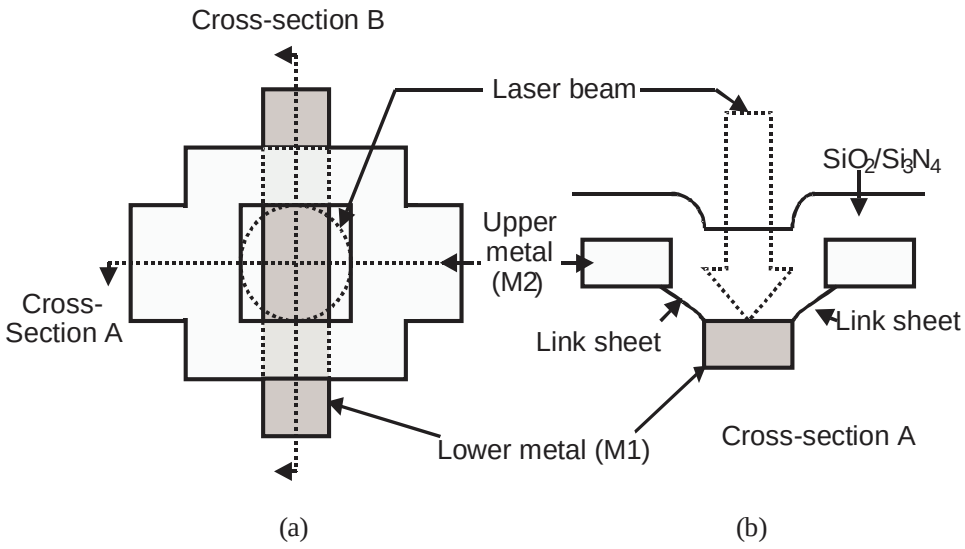


Figure 1.4 Schematics of MakeLink™ (a) top view (b) cross section A-A'

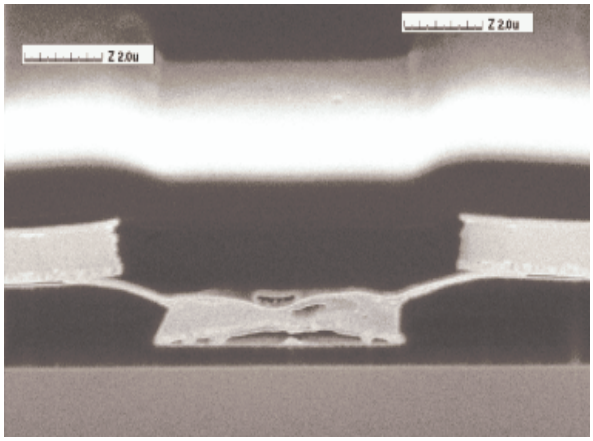


Figure 1.5 Cross-section of a vertical link.

1.2.2 Configurable Analog Blocks

As the core building elements, the CABs play a vital role in the realization of high performance FPAAs. The importance of CAB design is manifested in two aspects: the circuit design and the configuration topology.

Analog circuits usually cannot be simply described by their input/output transfer functions. A large variety of topologies exist for a similar analog function, each to achieve certain advantages on some performance specifications like gain, bandwidth, input range/output swing or noise. Therefore, it is very difficult to choose one circuit topology that is suitable for all applications. However, there are a wide set of high-level analog functions, like sample-and-hold, comparator, active-RC filters, that can be realized using the same basic elements, namely, OPAMP (or OTA), resistors and capacitors. In addition, some commonly used analog circuits bear great remembrance to each other, like a fully differential input pair and a Gilbert cell, which make a general configurable block for a large variety of applications possible.

On the other hand, different analog function requires different circuit topologies and element values. In addition, analog function can usually be implemented in both continuous-time mode and switched-capacitor circuits. While most existing FPAAs can only be used in either mode [1.20], CABs that are capable of implementing both of them are usually desirable because customers will have more choices and then higher chance to meet the design requirements. This demands an internal configuration topology flexible enough to accommodate all target applications, while highly efficient to not increase the chip area dramatically or introduce unacceptable interconnect parasitics.

1.2.3 Routing Architecture

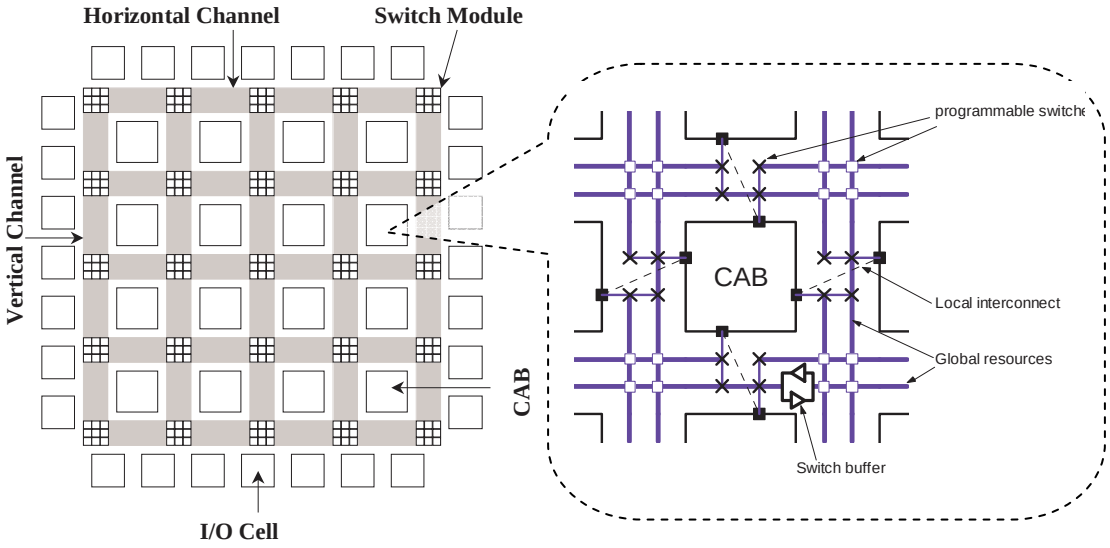


Figure 1.6 Array-based FPAA routing architecture.

Routing architecture is as important as the circuit design. Usually, the global routing architecture distributes the routing resources in the chip, and defines parameters like array aspect ratio, center/edge capacity ratio and directional-biasing

ratio. The detailed routing architecture specifies the connectivity of each wire segment and input/output pin. An conceptual array-based FPAA architecture is depicted in Figure 1.6.

Unlike the custom analog system design, the routing resources in an FPAA are prefabricated, and programming is realized by setting switches to make connections. A good routing architecture is essential for high performance FPAAs because most system performance degradation is due to routing rather than circuit, and most of the area of an FPAA is devoted to routing [1.21]. Moreover, since interconnect does not scale as well as transistors with process shrinks, the fraction of area and performance degradation due to routing in FPAAs is increasing with each technology scaling down. For high-frequency applications, the routing architecture is even important. It would be impossible for high-bandwidth FPAAs to realize their full potential if the routing delays and resource utilization were not handled well [1.22].

Due to the great difference in characteristics between analog and digital design, existing routing architecture for FPGAs cannot be taken over for FPAAs without major modifications. For example, they only consider topological parameters that deal with routability issues, which are sufficient for FPGAs since digital circuit are intrinsically robust to interconnect parasitics. However, for analog systems, the electrical issues, such as RC delay, cross coupling, voltage dropping and matching must be considered at the time of routing since they will greatly affect the overall performance. For those reasons, routing architectures specifically designed for analog systems are necessary.

1.3 Contribution and Organization of This Dissertation

In this dissertation, we proposed a design methodology for high performance FPAAs, targeting to facilitate hierarchical analog design approach while minimizing the performance penalty caused by configuring and routing. Begin with channel segmentation schemes aiming to improving routability and reduce interconnect parasitics, we developed performance-driven segmented routing algorithms and combined channel segmentation and buffer insertion algorithms to minimize interconnect delay and cross-coupling. The obtained results are applied in the design of a highly flexible CAB, and symmetrical routing channels for the analog array. Effectiveness of our design is demonstrated through the implementation of an 8-bit pipelined A/D converter running at 100 Mega-sample-per-second (MSPS).

The organization of this dissertation is as follows: Chapter 2 presents the channel segmentation algorithms, both parametric and no-paramedic. Chapter 3 presents the theory and implementation of a performance-driven segmented router for FPAAs. Electrical performance enhancement in terms of RC delay and cross-coupling deduction are investigated by buffer insertion in Chapter 4 and Chapter 5, respectively. The CAB circuits and its internal routing architecture design were described in Chapter 6. Finally, the implementation of the 8-bit pipelined A/D converter was presented in Chapter 7, with its key performance specifications discussed.

Chapter 2

Channel Segmentation

One important feature of a routing architecture is its channel segmentation scheme, which defines the lengths and locations of wire segments available in the routing channel. Just as there is a choice of partitioning the circuits, there is also a choice in partitioning the wiring scheme. A true hierarchical design would require a routing channel with wire segments of variable lengths and staggering locations. As shown in Figure 2.1, the channel segmentation schemes can be categorized into four different models.

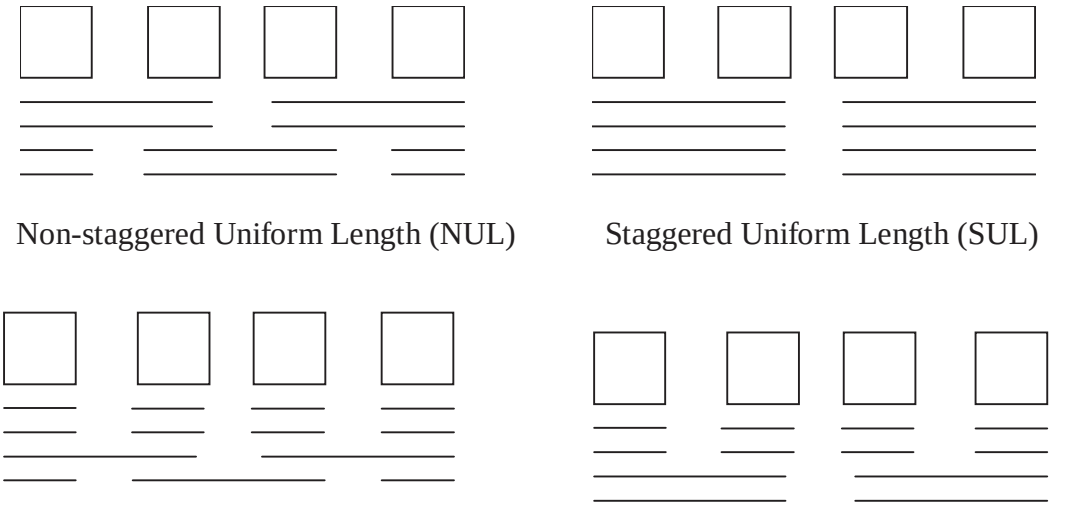


Figure 2.1 Different channel segmentation schemes

Channel segmentation has been studied for FPGA for years [2.1], yielding a lot of important results. However, it had never been considered for FPAAs because of the unacceptable performance degradation caused by the high resistance and capacitance associated with links offered by popular programming technologies. As shown in the previous works [2.2], the number of segments/switches, instead of wire length, used by a net is the most critical factor in determining the routing delay. The impact of switches on the electrical behavior of the circuit may even change its function. Therefore, almost all existing FPAAs employ crossbar routing channels that consist of tracks running through the whole chip to avoid excessive switches. However, MakeLink^{TM} technology, produces reliable, high quality, metal-to-metal links with extremely low-impedance and therefore makes channel segmentation practically attractive for FPAAs, especially for high bandwidth applications.

2.1 Introduction

Intuitively, there should be a strong correlation between the routing segmentation and the actual net distribution. Routing tracks composed of long segments usually have better performance because fewer switches in the signal path, but they also result in lower routability and higher wire wastage. On the other hand, tracks composed of short segments provide more flexibility and reduce the waste of wire, but performance is usually sacrificed [2.3]. Similarly, the locations of segments with respect to a net span are also very important in determining whether the net can be routed optimally. It is therefore the main object in constructing a routing architecture to match the channel segmentation scheme to the actual net distribution as closely as possible. By choosing segments of appropriate lengths and positions, it has been

clearly demonstrated that a well-segmented channel can greatly help the router to achieve effectively high routability and resource utilization comparable to a freely customized routing channel [2.4].

The rest of this chapter is organized as follows. Section 2.2 overviews the existing channel segmentation algorithms and defines our problem. Section 2.3 and Section 2.4 present the parametric and the nonparametric channel segmentation algorithms, respectively. Experiment results are presented in Section 2.5 and conclusions are given in Section 2.6.

2.2 Prior work and our problem

A number of channel segmentation design for FPGAs have been examined in many literatures. Zhu and Wong presented an algorithm for the channel segmentation design problem based also on a stochastic analysis [2.5]. K. Roy and M. Mehendale developed an algorithm which generates channel segmentation with fixed length tracks to approximate a given segment length distribution [2.6]. Pedram et al. presented an analytical model for the design and analysis of effective segmented channel architectures [2.7]. And later W. K. Mak extended the problem to 2-D symmetrical FPGAS [2.8]. Recently, Jai-Ming Ling et al. presented a unified segmentation and routing design for array-based FPGAs [2.9].

Here we adopted the staggered, non-uniform length (SNL) segmentation model [2.10] used by most researchers. The following notations are used in defining the problem:

L: Length of a channel

T: Total number of tracks in the channel

M: Maximum number of segments for routing a net

$h(x,l)$: Probability of a net with length l originating at x

Our design problem is formulated as follows: *Given L, T and* $h(x,l)_c$*, design a channel segmentation scheme that maximize success rate for M-segment routing while minimizing the average interconnection parasitics.*

2.3 Parametric channel segmentation

When the theoretical or empirical net distribution of the target application is used in the channel design, a situation assumed by most existing channel segmentation algorithms, we call it parametric channel segmentation.

2.3.1 Segment length selection

Like most of previous work, a channel is partitioned into several regions. The tracks in the rth region are divided into segments of length Λ_r, also called type r segments designated to route nets whose lengths fall in the range $(M\Lambda_{r-1}, M\Lambda_r]$ (assuming $\Lambda_{r-1}<\Lambda_r$). The segments are arranged in a staggered fashion to allow the maximum flexibility of routing nets starting at different locations.

We adopted the One-segment length selection algorithm in [2.5] and extended it to M-segment routing channels. For an arbitrary net length distribution $f(l)=\Sigma_x\, h(x,l)$, the Λ_r's are determined as follows: We set $\Lambda_J = L$, and choose Λ_r, $r = J$-1, J-2, … one by one as the largest value that satisfies

$$\sum_{l=M\Lambda_r+1}^{M\Lambda_{r+1}} f(l)\cdot l \Big/ \sum_{l=M\Lambda_r+1}^{M\Lambda_{r+1}} f(l) \geq \frac{M\Lambda_{r+1}}{\xi} \tag{2.1}$$

The parameter ξ is a constant greater than one and can be tuned to achieve the best results. An example of this segment length selection result is shown in Figure 2.2.

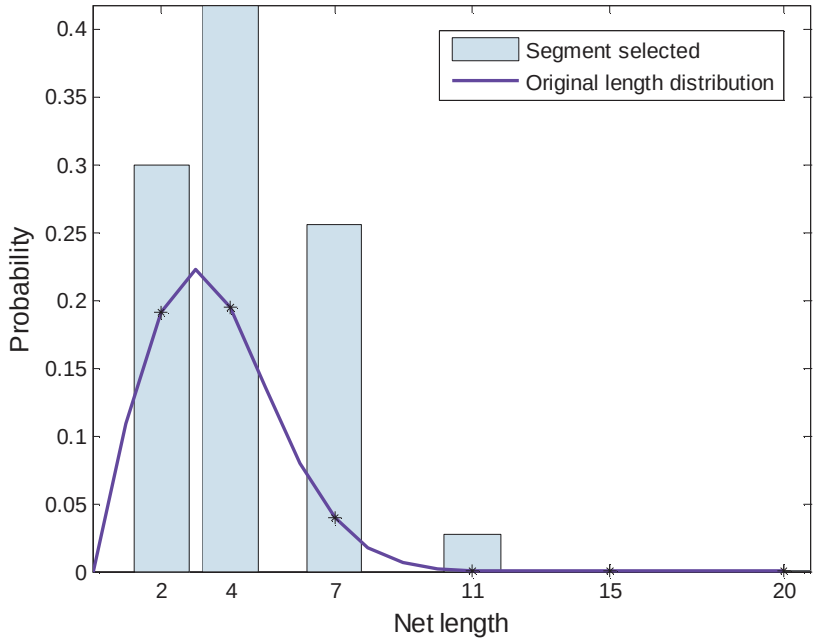


Figure 2.2 Segment length selection from net length distribution

A few shortcomings of this length selection algorithm were identified in [2.5] and a reconstruction procedure was suggested. However, during that procedure the short segments need be combined with their neighbors and result in very irregular segment lengths, which is not good for layout generation. A regular segmentation model where each track is divided into segments of equal length was adopted in [2.8], however, the

non-staggering nature of the channel will considerably limit the routability. Here we retain the segment lengths generated by this algorithm, but improve the track assignment by taking the staggering factors into account.

2.3.2 Track assignment

The number of tracks in each region should be proportional to the expected usage of that type of segments. Since the segments in one track are actually placed one by one, their originations (left ends) can only appear at multiples of the segment length. Those nets originate from other points (called off-grid nets) may fail to be routed by any track in their designated region, and require an extra track in regions containing longer segments. To calculate the expected usage of tracks in region *r*, we consider two cases similar to those described in [2.7] and introduce the staggering factors δ_1 and δ_2 to describe how much the off-grid situations are taken into consideration.

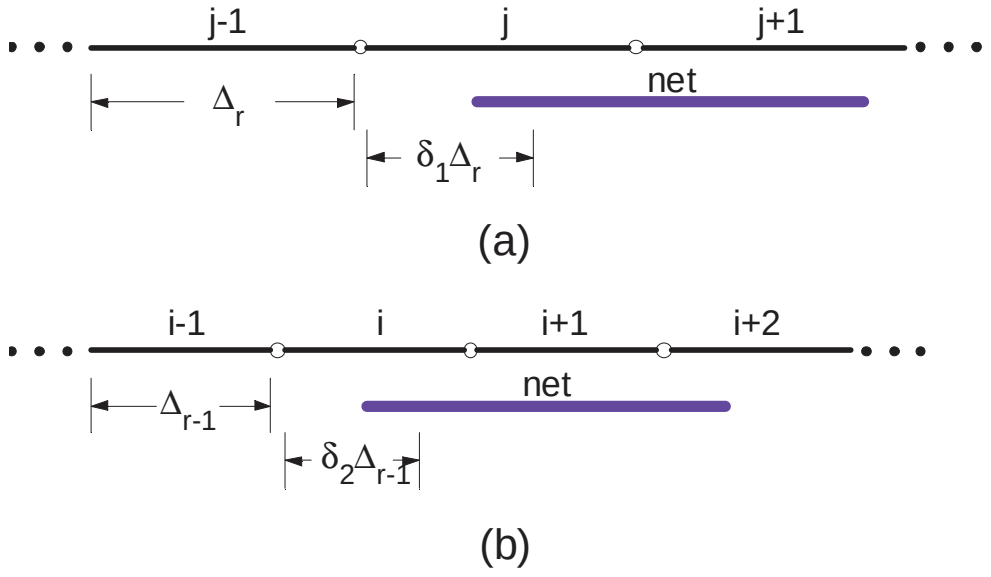


Figure 2.3. Illustration of the staggering factors.

The first case is those nets have length in the range ($M\Lambda_{r-1}$, $M\Lambda_r$] and can be routed using tracks in region r. For a net with origination x in the range of [$j\Lambda_r$, $(j+\delta_1)\Lambda_r$] ($0 \le \delta_1 \le 1$), its length should be no more than $(j+M)\,\Lambda_r - x$, as illustrated in Figure 2.3 (a) (M = 2). The expect number of tracks in region r for such nets is given by

$$n_j^r = \sum_{x=j\Lambda_r}^{(j+\delta_1)\Lambda_r} \sum_{l=M\Lambda_{r-1}+1}^{(j+M)\Lambda_r - x} h(x,l) \tag{2.2}$$

The second case is those nets have length in the range ($M\Lambda_{r-2}$, $M\Lambda_{r-1}$], but cannot be routed using tracks in region r-1. For a net with origination x in the range of [$i\Lambda_{r-1}$, $(i+\delta_2)\Lambda_{r-1}$] ($0 \le \delta_2 \le 1$), its length should be more than $(i+1)\,M\Lambda_{r-1} - x$, as illustrated in Figure 2.3(b) (M = 2). The expect number of tracks in region r for such nets is given by

$$m_i^r = \sum_{x=i\Lambda_{r-1}}^{(i+\delta_2)\Lambda_{r-1}} \sum_{l=Max\{M\Lambda_{r-2},(i+M)\Lambda_{r-1}-x\}+1}^{M\Lambda_{r-1}} h(x,l) \tag{2.3}$$

The total expected number of tracks for type r region is then given by the sum of the maximum expected usage of both cases

$$p^r = MAX_{j=0}^{L/\Lambda_r - 1} n_j^r + MAX_{i=0}^{L/\Lambda_{r-1} - 1} m_i^r \tag{2.4}$$

Note that when $\delta_1 = \delta_2 = 0$, $n_j^r = \sum_{l=M\Lambda_{r-1}+1}^{M\Lambda_r} h(j\Lambda_r, l)$ and $m_i^r = 0$, so p^r is simply the probability of a net length falling in the range ($M\Lambda_{r-1}$, $M\Lambda_r$].

Once p^r has been calculated for all r, the number of track allocated to region r is allocated proportionally to p^r. if there exist more than one track in a region, the tracks are displaced with an offset evenly chosen in $[0, \Lambda_r)$. An example of channel segmented by this method is shown in Figure 2.4.

Figure 2.4 An example of parametric channel segmentation.

2.4 Nonparametric channel segmentation

To determine Λ_j and T_j, the procedure above need a net distribution or a large set of benchmark circuits, which however, as we mentioned before, is not always available to FPAA designers. To meet this special situation, we propose a novel channel segmentation scheme without the net distribution information.

Given the channel length, L, we construct Λ_j as follows: $\Lambda_0 = L$, $\Lambda_1 = \lfloor \lambda \Lambda_0 \rfloor + 1$, where λ is the section coefficient, and $\lfloor x \rfloor$ stands for the maximum integer that is less than x. The other segment lengths are chosen by the formula

$$\Lambda_j = \Lambda_{j-2} - \Lambda_{j-1}, \quad j = 2, \ldots, J, \tag{2.5}$$

stopping at $\Lambda_J = 0$ or 1. Therefore, we have

$$L = \Lambda_0 = \Lambda_1 + \Lambda_2 = 2\Lambda_2 + \Lambda_3 = 3\Lambda_3 + 2\Lambda_4$$

$$= F_{j+1}\,\Lambda_j + F_j\,\Lambda_{j+1} \tag{2.6}$$

where F_j is the Fibonacci sequence that $F_0 = 0$, $F_1 = 1$ and $F_j = F_{j-1} + F_{j-2}$, $j = 2, 3, \ldots$

The *j*th region will then be constructed as follows:

The first group contains T_j tracks, each consisting of F_{j+1} segments of length Λ_j, and F_j segments of length Λ_{j+1}, from left to right; A new group is generated by a circular right shift of the previous group, i.e., put its right-most segment to its left end. This procedure stops when the new group is identical to the first group.

The procedure may result in uneven regions. T_j's are used to adjust the region width in favor of long or short segments, while ensuring the total number of tracks constructed not exceeding the predefined maximum number of tracks.

A nice property of this construction procedure is it won't produce unexpected segment length, that is, tracks are always made of segments of pre-defined lengths. This makes it easier to realize by commercial processes. An example when $L = 16$ is

shown in Figure 2.5 ($\Lambda_0 = 16$, $\Lambda_1 = 10$, $\Lambda_2 = 6$, $\Lambda_3 = 4$, $\Lambda_4 = 2$, $\Lambda_5 = 2$, and $\Lambda_6 = 0$. For simplicity, $T_j = 1$ for all j)

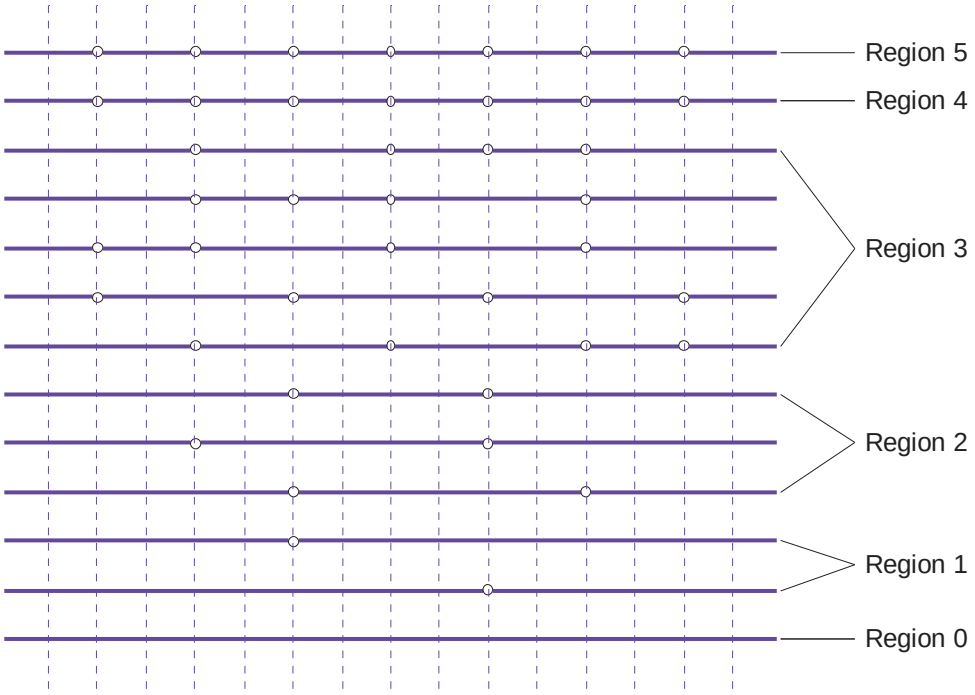


Figure 2.5 An example of nonparametric channel segmentation

2.5 Experimental Results

In our experiments, the proposed parametric channel segmentation algorithm was applied to six different net length distributions based on Geometric, Normal and Poisson distributions, as listed in Table 2.1. It is assumed that the net left-end points follow a uniform distribution, which is very close to reality as confirmed by empirical studies [2.10]. We set the channel length $L = 40$, total number of tracks $T = 20$, and compute the rate of successful routing completion for randomly generated routing instances according to those distributions.

Table 2.1 Net distributions used in the experiments.

	Ge1	Ge2	No1	No2	Po1	Po2
$f(l)$	0.6^{l}	$0.6^{l/4}$	$e^{-l^2/20}$	$e^{-l^2/120}$	$2^{l}/l!$	$6^{l}/l!$

First we investigated the effects of staggering factors δ_1, δ_2 on routability. We chose three different value of δ_1 (0, 0.4 and 1), and let δ_2 vary from 0 to 1. For each value of δ_1, δ_2, a segmented channel is constructed using the algorithm described in Section 2.3. Five hundred routing instances, each containing 30 nets, were generated randomly for each distribution. These were routed in the channels using the segmented routing algorithm that will be described in Chapter 3.

The one-segment routing success rates for instances with distribution Ge1 are shown in Figure 2.6. It is seen that the factor δ_1 doesn't have much effect on the routing results, causing a variation on the success rate of less than 11%. However, the choice on δ_2 does make a huge difference. When δ_2 is small, the success rate increases rapidly with δ_2 till it reaches a value around 0.5. After that, the success rate becomes rather flat and even decreases for larger δ_2. By choosing the δ_2, the success rate can be increase from about 10% to more than 95%, an order of magnitude. It can be explained that when δ_2 increases, more tracks are allocated to longer segments, which are more useful than short segments for 1-segment routing. However, as more tracks are assigned to long segments, the total number of segments decreases, which cancels the benefits brought by longer segments and finally makes the success rate drop.

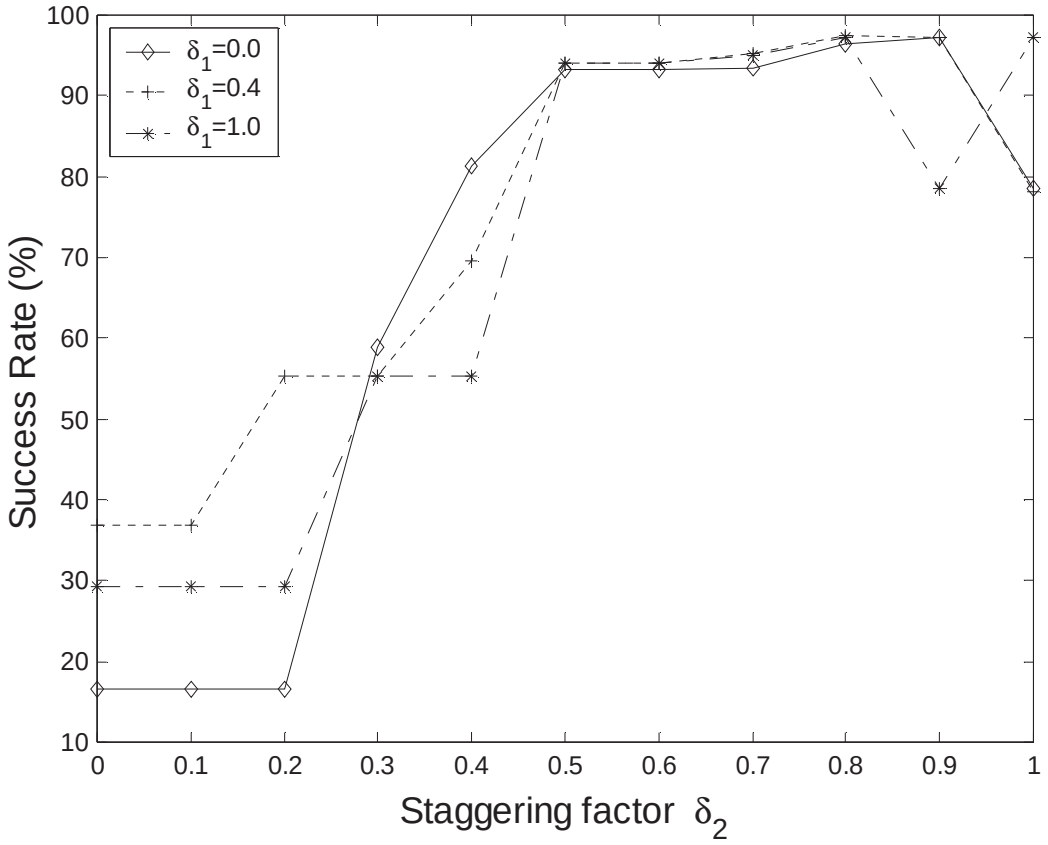


Figure 2.6 The effects of δ_1, δ_2 on 1-segment routability for net distribution Ge1.

Experiments on other five net distributions revealed similar results. The one-segment routing success rates for all six distributions are shown in Figure 2.7. Although the significances of the effect of δ_2 on the channel routability are different for different net distributions, they exhibit almost the same trend. Interestingly, the highest success rates are reached unanimously when δ_2 is around 0.5, indicating an optimum value rather independent on the actual net length distribution.

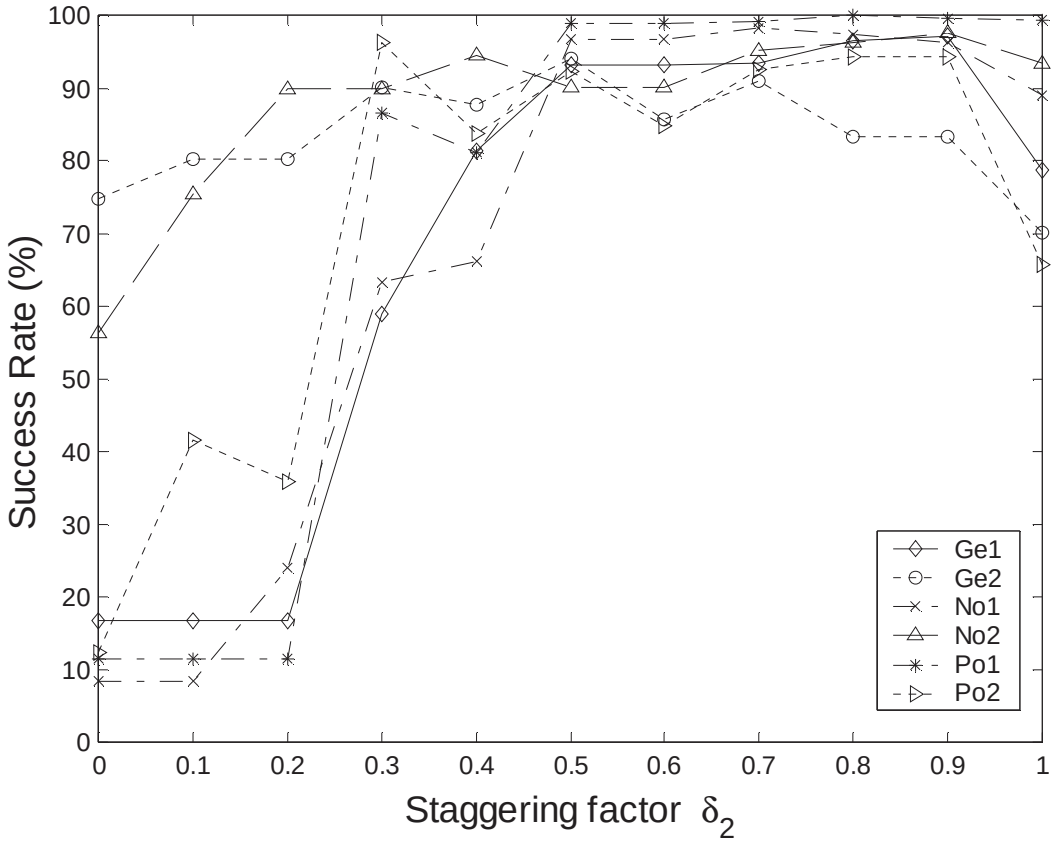


Figure 2.7 The effects of δ_2 on routability for one-segment routing.

The two-segment routing success rates for all six distributions are shown in Figure 2.8. Since two-segment routing have more choices in choosing wire segments, their success rates also increase with δ_2, but less significantly compared to one-segment routing. For the same reason, there seems no optimum δ_2 for all net distributions, while some value between 0.5 and 0.6 is still a good choice for most of them.

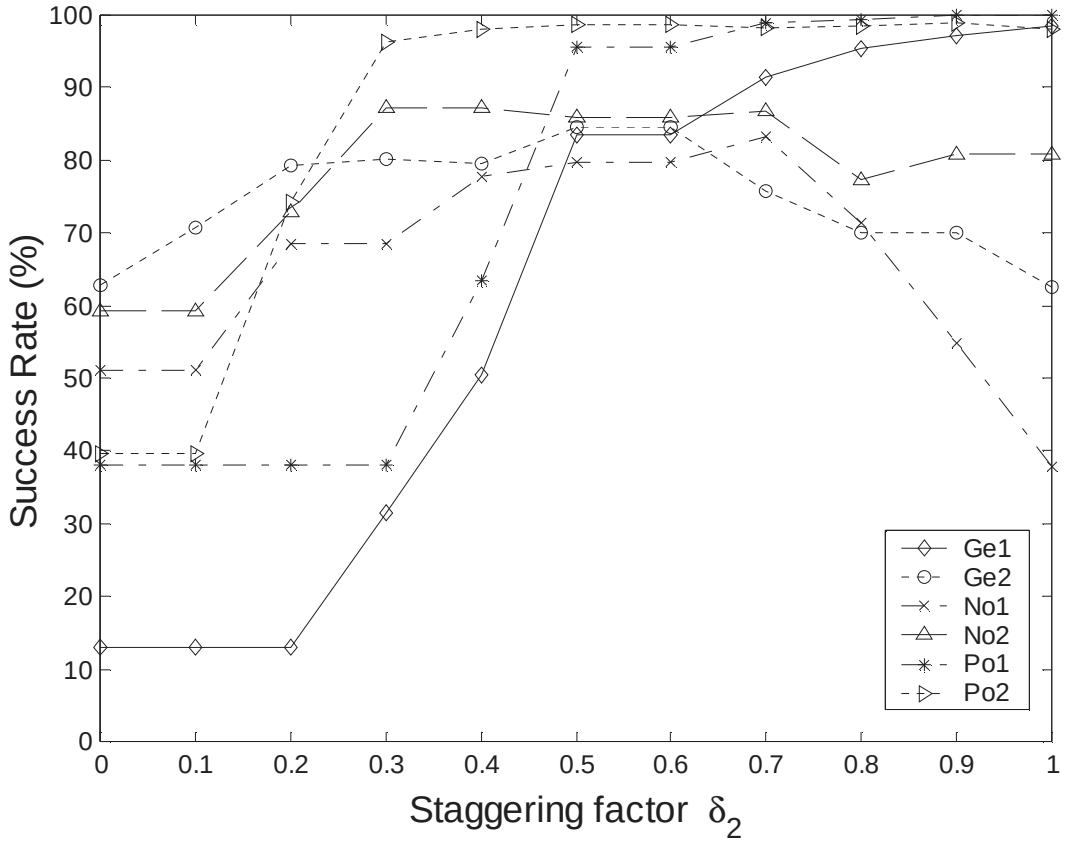


Figure 2.8 The effects of δ_2 on routability for two-segment routing

2.6 Conclusions

In this Chapter, we addressed problem of segmentation scheme in SNL routing channel design. Multiple-segment length selection algorithm was derived from existing one-segment algorithm, and a new track assignment algorithm was proposed to calculate more accurately the actual demand on each type of segments. Experimental results shown that, by choosing the proper staggering factors that take the needs of off-grid nets into account, the channel routability can be increased by an order of magnitude without increasing the channel area or number of tracks.

Chapter 3

Segmented Routing

Aiming at the same goal, high routability and low performance degradation, the efforts in FPAA designs are divided into two distinct but closely related approaches: the routing architecture and the placement/routing tools. A well-designed architecture could not attain its best performance without a corresponding CAD tools that can take full advantage of it. On the other hand, a sophisticated and efficient CAD tool could not improve the routing results if it has to deal with an architecture that does not support its advanced features, just like a CPU and its operating system.

The development of routing architecture and routing algorithm bears some analogy to the design of an automatic transportation system. The routing architecture design is to construct the optimum road system, like planning beforehand where should be a long freeway and where only a local one-way shortcut is needed, according to the average traffic pattern in that system. It also need to decide how these roads are connected to one another, their capacities and speed limits so routing algorithm can properly estimate the congestion and delay. The main purpose of the routing algorithm is helping the drivers to find the quickest path to their destinations with the knowledge of the road conditions like the distance and speed limit of each

road, the location of joints, as well as the current traffic conditions. A typical design flow for FPAA implementations is shown in Figure 3.1.

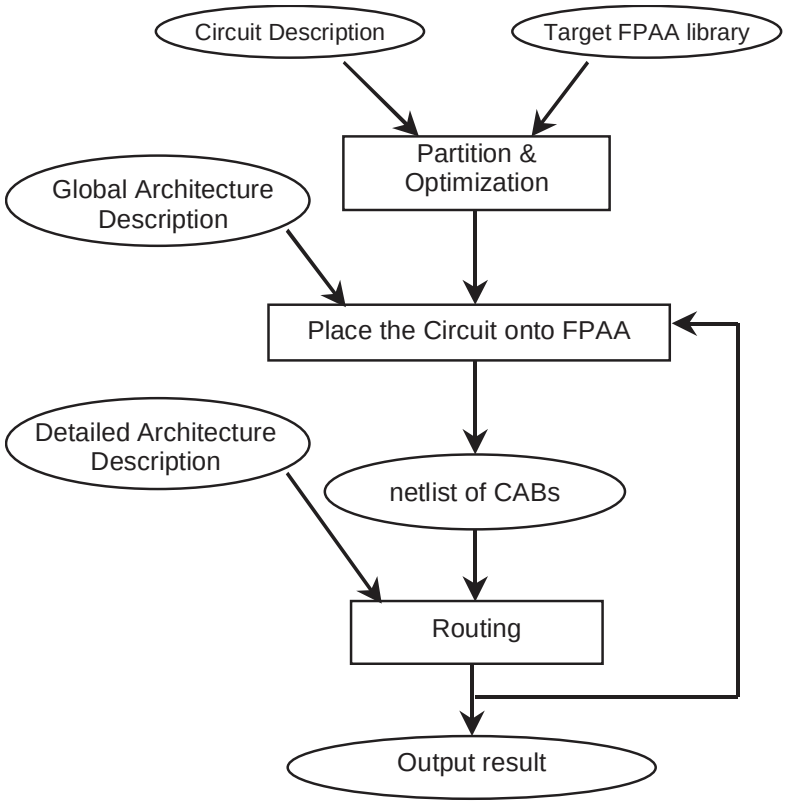


Figure 3.1 A typical FPAA design Flow.

3.1 Introduction

Once a circuit is partitioned into sub-blocks and placed, automatically or manually, across several CABs in the FPAA, those nets broken in the partition process need to be restored, which is the main task of the routing algorithm. By assigning each net to a set of wire interconnects in the routing channel and setting the proper switches, the router reinstates the connections and therefore the original circuit.

Unlike digital circuits, which are characterized by intrinsic robustness to parasitic effects, analog systems usually suffer from interconnection parasitic. Since the programmable switches introduce significant series resistance and capacitance, existing FPAAs use single-segment routing channel that consists of crossbar tracks running through the whole width of the chip to avoid excessive switches. A fast and effective routing algorithm for this type of FPAAs was presented in [3.2]. However, research indicates that a good channel-segmentation scheme will not only improve the channel routability, but also enhance the system performance by reducing interconnect parasitics [3.2].

Segmented routing algorithms have been studied extensively in the past two decades. It has been shown that the segmented channel routing problem is in general a NP-complete problem [3.4]. A routing structure with fixed orthogonal wire segments is described in [3.5], and in [3.6] a global router for symmetrical-array-based FPGAs was presented. Research results have shown that an efficient segmented routing algorithm can achieve nearly the same results of free channel routing [3.7]. Most of the algorithms are routability-driven or timing-driven, which are sufficient for digital systems. However, the factors that cause performance degradation to analog systems are not limited to delays. Variations in cross-coupling capacitances and stray resistances, for example, can dramatically degrade circuit performance or even cause system instability. Therefore, the existing algorithms cannot be directly applied to the FPAA routing without some major modifications. In this chapter, we focus on the routing problem of FPAAs with segmented routing channels. The rest is organized as follows. Section 3.2 describes the preliminaries of segmented routing.

Section 3.3 presents our performance-driven analog segmented routing algorithm. Experiments results are given in Section 3.4 and conclusions are given in Section 3.5.

3.2 Preliminaries

In this subsection, a summary of automated routing algorithms in current use is given. One of the most important algorithms is Lee's Maze router [3.8], of which most of existing automated routing algorithms use some variations. Lee's Maze router is best illustrated by Figure 3.2. Its task is to find a shortest path from source node s to target node t. First, grids defining where one wire can cross are marked by its relative distance to the source. The search begins at the source, finding all the grids at incremental distances until reaching the destination. This algorithm addresses the problem in a manner consistent with wave propagation. With this procedure it is guaranteed that the shortest path will be found.

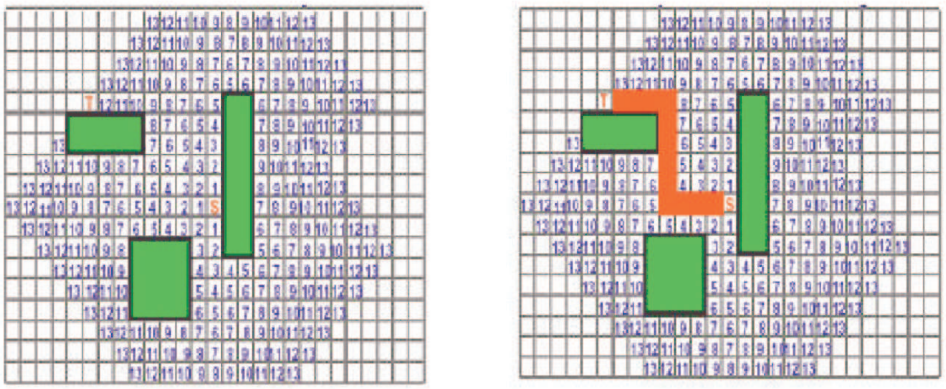


Figure 3.2 Lee's Maze router

A Maze router essentially consists of running Dijkstra's algorithm [3.9], which solves the single-source, shortest-path problem on a weighted, directed graph G =

(V,E), for the case in which all edge weights are non-negative values, as presented below:

```
Dijkstra(G,w,s)
1. for each u∈V[G] {
2.          dist(s,u) = ∞;
3.          pre(u) = NULL;}
4. dist(s,s) = 0;
5. Done = Φ;
6. Q = G;
7. while Q != Φ {
8.          find u ∈ Q with min. dist(s,u);
9.          Q = Q – {u};
10. for each v adjacent to u
11. if dist(s,v) > dist(s,u) + dist(u,v) {
12.          dist(s,v)=dist(s,u)+dist(u,v);
13.          pre[v] = u;}
14. Done = Done∪ {u};
15.}
```

Figure 3.3 Dijkstra algorithm

The searching strategy is very similar to the one used in Prim's algorithm [3.9], where a light edge is added at each step. The shortest path of a new vertex is calculated, with respect to the existing, partially finished tree (net). This algorithm applies a greedy strategy. The key to efficiently implementing Prim's algorithm is to make it easy to select a new edge to be added to the tree. During execution of the algorithm, all vertices that are not in the partial tree (net) are stored in a priority queue.

```
Prim (G,w,r)
```

```
1. for each u∈V[G] {
2.          do key [u] <- ∞;
3.           p [u] <- NIL
4. key [u] <- 0
5. Q <- V[G];
6. Q = G;
7. while Q != Φ {
8.   do u <- Extract Min (Q)
9.   for each v ∈ Adj [u]
10.         do if v ∈ Q and w (u, v) < key [u]
11.         then p [v] <- u
12.         key [v] <- w (u, v)
```

Figure 3.4 Prim algorithm

A pure routability-driven router may produce circuits with poor performance, while pure performance-driven routing may result in an unroutable circuit. A efficient way to balance these trade-offs is to incorporate costs into the routing, like the pathfinder negotiated routing algorithm [3.10], which negotiates routing repeatedly rips-up and re-routers every net in the circuit until all the congestions are eliminated. During the first routing iteration, every net is routed for minimum cost, even if this leads to congestion. The cost of overuse is increased after each iteration. The router can determine how to arrange the routing resource, based on the cost of each vertex. Consequently, if overuse exists at the end of a routing iteration, more iterations are performed to resolve this congestion. The detailed algorithm is shown below:

```
RT(neti): a linked list used to store the set of vertices in the current routing of net i
While (overused resources exist && max iteration not exceeded) {
For (each net, i) {
        If RT is not empty then Rip-up existing RT(neti) and update p(n) ;
        Initialize RT to the source terminal;
```

```
For(each sink net i) {
If PQ is not empty then free PQ and re-initialize PQ;
Initialize PQ to RT;
Mark all the vertices as un-reached by wave expansion;
Initialize PriorityQueue to RT(neti) and set pathcost equal to the base cost of
each vertex in RT;
If this sink j is not foundd in RT(neti) {
        do {
                Dequeue PQ;
                For (all fanout vertices n of node m){
                If (this fan-out is not a PIN or PAD and un-reached during
                previous wave expansion)
                add it to PQ & update pathcost(n) = pathcost(m) + cost(n);
                else if (this fanout is a sink)
                        add it to a sink list;
                else continue wave expansion;
                }
        } while (no sink has been found); /* Wave expansion ends here */
}
if ( more than one sinks are found during this wave expansion) {
        add those sinks and their parents to RT;
        update p(n) only if vertex n is not contained in RT;
}
for (all vertices in path from RT(i) to sink,j){ /* Backtrace from the linked list
        of sinks */
        Update p(n) only if vertex n is not contained in RT;
        Add n to RT(i);
} /* Backtracing ends here */
Update h(n) for all n;
} /*End of one iteration*/
```

Figure 3.5 The improved pathfinder negotiated routing algorithm

In addition to dealing with wire segments of various lengths, segmented routing differs from serial routing algorithms like Maze and Pathfinder Negotiated, which assume that the FPAA contains wire segments of only one length and find the routing

path by wave propagation, in two aspects. First, all the wire segments required to finishing routing a net are determined at the same time. Secondly, all nets in the routing instance are routing simultaneously. Basically, the segment routing is a weighted bipartite matching problem. , also known as the assignment problem. Since assigning a net to each of wire segments available in the channel comes with a cost, the routing algorithm tries to finish the assignment with the minimum total cost. Figure 3.6 shows the concept of minimum-cost-bipartite-matching.

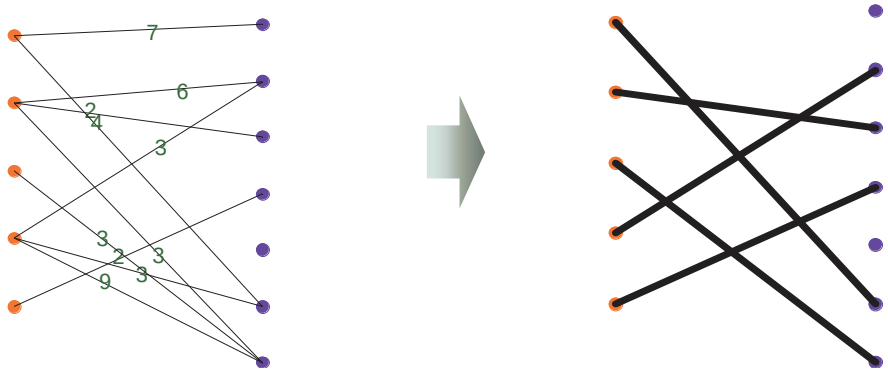
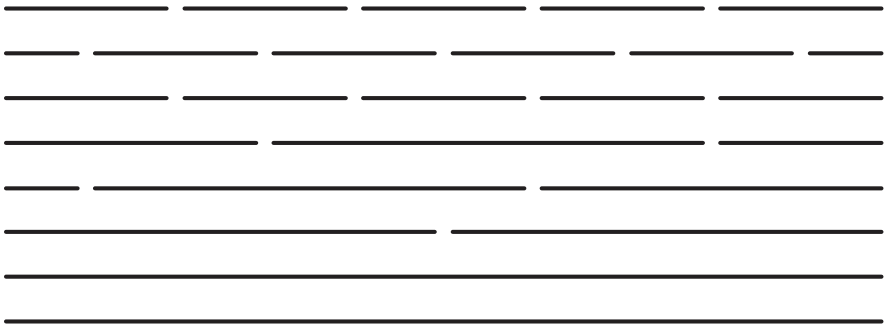
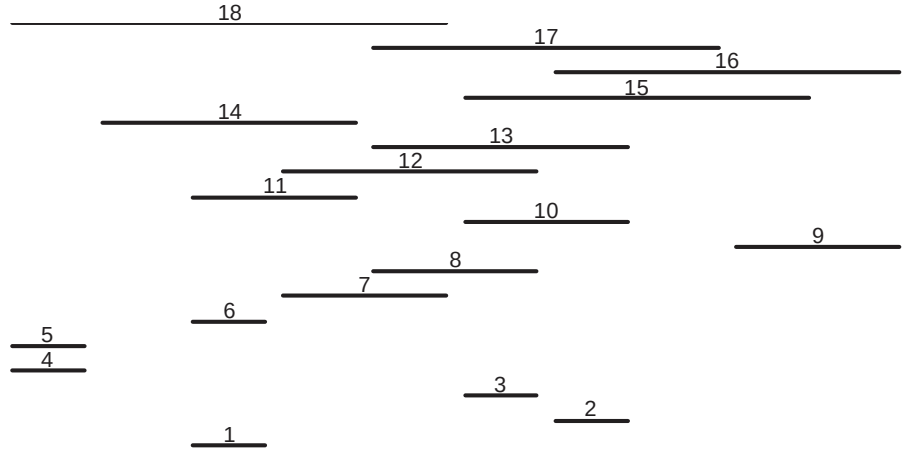


Figure 3.6 Minimum-cost-bipartite-matching

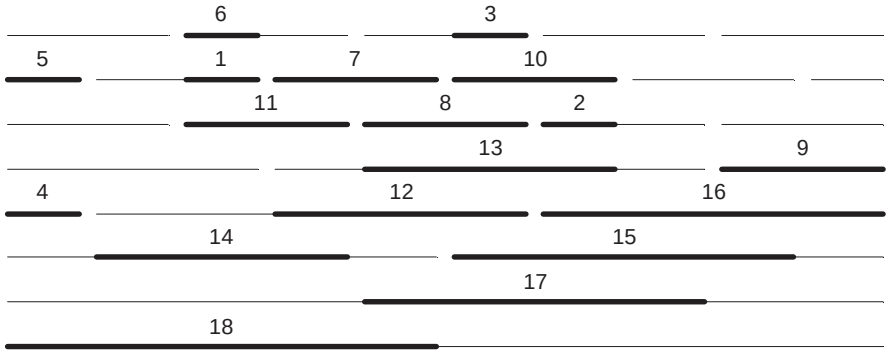
The weighted bipartite matching problem, can be solved by the primal-dual method — called the *Hungarian method*, which solves a complete bipartite graph with $2 \cdot |V|$ nodes in $O(|V^3|)$ arithmetic operations [3.11]. An example of channel segmentation and matching-based routing is shown in Figure 3.7.



(a)



(b)



(c)

Figure 3.7 (a) A segmented channel with eight tracks (b) eighteen nets to be assigned (c) minimum-cost routing results

3.3 Analog Segmented Routing

Analog routing aims at not only making the necessary connection, but also minimizing the performance degradation caused by interconnect parasitics. In this called performance-driven routing, the right choice of assigning cost function is essential to achieve the routing results we want. In digital layouts, the dependence of

electric performances on the details of physical implementation is limited to logic functions and delay requirements. However, more performance specifications are imposed on analog circuits. For example, the specifications for an operational amplifier include power, area, gain-bandwidth, DC gain, phase margin, gain margin, output range, common-mode input range, settling time, PSRR, CMRR, noise, input and output impedance, slew rate, offset, harmonic distortion, and so forth, which makes the definition of cost function much more complex in analog routing.

3.3.1 Parasitic Modeling

An accurate parasitic model is essential in define the cost function. Ideally highly accurate performance estimation can be obtained with a circuit simulator like SPICE, but the CPU time required to run SPICE on thousands of nets in a typical VLSI circuit is prohibitive. Though a distributed transmission lines model is more accurate for modern interconnection wires, most of existing literatures use the lumped RLC models [3.12], which provide a good trade-off between accuracy and efficiency.

There are two types of parasitic resistance, serial resistance of the programmable switches and metal wire resistance, which can be calculated by using the follow equation:

$$R_w = R_{\square} L/W + R_s M, \tag{3.1}$$

where the first term stands for the wire resistance, $R_{\square}$ is the sheet resistance, in $\Omega/\square$, L is the wire length and W is the wire width, while the second term stands for the

switch resistance, R_s is the on-resistance of a single switch, and M is the number of switches used in the path.

The typical interconnect capacitance at each node in a circuit is calculated using the model shown in Figure 3.8. It consists of two conduction layers over the substrate, considered as a reference plane (ground plane). There are three capacitance components at any node [3.13]:

- Overlap capacitance between two wires in different layers (C_{21a} and C_{23a}), which are proportional to the overlap area $W \cdot L$
- Fringing capacitance between two wires in different layers (C_{21fr} and C_{23fr}), which are proportional to the wire length L.
- Lateral capacitance between two wires in the same plane (C_{22lat}), which are proportional to the wire length L and inversely proportional to the wire spacing D.

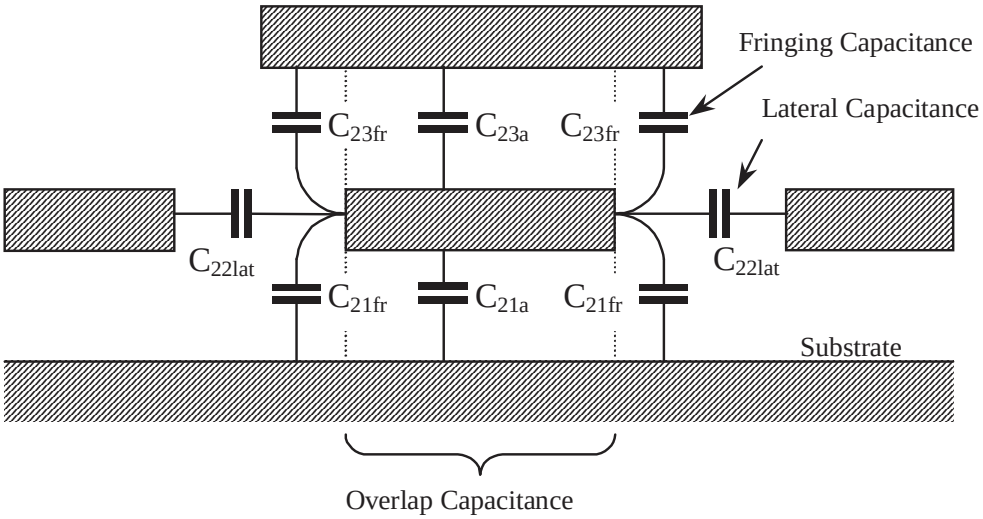


Figure 3.8 Interconnect capacitance model

Interconnect inductance is much more complicated to extract than resistance or capacitance because of the loop current definition of inductance. If the operating frequency is not too high or the FPAA scale is not too large, the effect of parasitic inductance can usually be neglected.

3.3.2 Performance Metrics

While the real impact of interconnect parasitics on the system performance cannot be exactly evaluated until the system is finally configured, some metrics can be used to assess the parasitics, and hence performance degradation during the routing process. Since wire width and spacing are predetermined by the channel design, the only thing the router considers is the segment length and their locations. We apply the following metrics when evaluating the soundness of assigning net *n* to wire segments available in the routing channel:

- *Number of segments*, denoted by *M*. If connection switches are needed to connect multiple segments to enable the routing, the switches introduce extra resistance.

- *Total segment length*, defined as $S = \sum_{g=1}^{M} len(s_g)$, where $len(s_g)$ is the length of segment s_g. As indicated by the parasitic models, the interconnect parasitic and cross-coupling are directly related to the length of the wire;

- *Wire wastage ratio*, defined as $U(n,t) = \sum_{g=1}^{M} len(s_g) \Big/ len(n) - 1$, where $len(n)$ is the length of net *n*. Since the unused portion of the wire segments presents as a loading capacitance, this metric shows how much the channel routing results deviates from a freely routed circuit.

- *Net Priority of net n*, denoted by $N_p(n)$. Parasitics of the critical nets in the circuits can be minimized by assigning them a relatively large weight costs compared to non-critical nets.

In a performance-bounded routing [3.14], those metrics are limited by user-defined bounds, which are set based on reasonable estimates of parasitic values extracted from the layout. The router then actually enforces the parasitic constraints during routing in order to maintain a satisfactory circuit performance. If the net's parasitic bound is violated, the net is ripped-up and the routing is retried. In a performance-driven routing, those metrics are combined properly to reflect the performance degradation caused by routing, and a routing result with the minimum overall performance degradation is obtained.

3.3.3 Congestion avoidance

Since each routed net becoming an obstacle for subsequently nets, the routing feasibility of subsequent nets can be increased significantly by avoiding the resources that are potentially needed by other nets. We use the concept of *resource demand* to help the router to be aware of the needs of future nets. The *demand* on a segment is the number of nets that subscribe to it. By incorporating the demand for the segments into the cost function, the router automatically chooses the segments with fewer potential subscribers. This increases the chance of routing future nets successfully.

The initial segment demands are computed by routing each net independently as if all routing resources are available. In the actual routing, the resource demand is updated as follows: If the net avoided a segment that it initially subscribed to, the

demand for that segment is decreased by one, or, if the net used a resource initially not subscribed to, the demand for that segment is increased by one.

At first the effect of demand on assignment cost is chosen to be small, so the router has more freedom to choose the best matching segments. If the circuit routing failed, then the routing demand is restored to its initial value and retried with tighter feasibility constraints by increasing its weight on the cost function, and hence redefining the costs of all routing resources. This gradually forces the router to avoid over-subscribed resources.

3.3.4 Cost function definition

The cost of allocating net n to M segments (s_1 to s_M) in track t is then defined as

$$C(n,t) = \left(\left[M + w_1 \cdot \sum_{g=1}^{M} len(s_g) + w_2 \cdot \left(\sum_{g=1}^{M} len(s_g) \Big/ len(n) - 1 \right) \right] \cdot N_p(n) \right)^{\alpha} \bullet \left(\sum_{g=1}^{m} Demand(s_g) \right)^{\beta} \tag{3.2}$$

where w_1, w_2, a and β are weighting factors. The object of the segment router is to minimize the total allocation costs, which can be solved in polynomial time by a weighted bipartite matching algorithm.

3.3.5 Grouped routing

Like the matching-based, timing-driven routing algorithm in [3.15], each time the router takes a maximum clique, defined as the maximum set of nets overlapping each other from the nets unassigned, and assign it to the wire segments left in the channel

using the minimum cost matching algorithm. Unlike other algorithms that route nets one by one, this algorithm utilizing routing resources more effectively.

3.4 Performance Constraints on the Routing

The goal of routing is not only to complete all the required connections without congestion, but also to satisfy a set of performance constraints. For an FPGA/digital circuit, performance is usually measured by clock speed or/and delay on the critical path. However, for an FPAA/analog circuit, signal delay is not the only concern. The system performance is usually measured by its bandwidth, gain, linearity etc. Routing parasitics can affect the performance of analog system in many different ways. For examples, in an OPAMP circuit, a small capacitive coupling may degrade the frequency response due to the Miller effect. Sometimes, stray coupling which gives rise to positive feedback may lead to oscillations. When a net travels a long distance, the parasitic capacitance to ground can introduce an extra pole (for example, a pole very close to the dominant pole) that may deteriorate the op amp's stability.

The performance constraints (tolerable variation of gain, bandwidth etc.) imposed on analog array are too abstract for the routing tools to handle directly and must be converted to a set of routing constraints, i.e. interconnect parasitic constraints. Once the routing constraints are met by the router, the performance constraints of the analog circuit should also be satisfied. There are two major kinds of constraints, namely, the bounding constraints and the matching constraints, as described below:

3.4.1 Bounding Constraints

Bounding constraints can be further divided into two classes: loading constraints, which are mainly the parasitic capacitance to ground and coupling constraints, which are coupling capacitance among a set of sensitive nets. Capacitive coupling is present whenever two nets have segments that cross or are parallel to each other. Thus, it can be further classified by crossover constraints and adjacency constraints. For FPAA, the adjacency constraints are the dominant factor because most of the capacitances induced by crossover can only occur at the intersections of horizontal and vertical channels.

3.4.2 Matching Constraints

Fully differential topology is frequently used in the FPAA circuit, which results in an additional need for the interconnect parasitics associated with appropriate nodes or branches to nominally match, for impedance matching and noise cancellation purposes. The matching constraints require: (1) For impedance matching, the capacitances to ground associated with each matched pair of nets should be equal; (2) When a casual net (the net that does not have any constraints) is close to a matched pair, the coupling capacitances between that casual net and the pair of matched nets should match; (3) When two pairs of matched nets come close to each other, it is necessary to match the direct coupling capacitances and cross-coupling capacitances. Besides having symmetrical loading, this also ensures that equal levels of noise on the two nodes of one matched pair causes the same on the other pair, if any coupling is present.

The performance-constrained routing problem can be then defined as follows:

Definition: For a set of performance functions $\{W_i\}$, i = 1, 2, . . .N_w and a set of parasitics $\{p_j\}$, j = 1, 2, . . . ,N_p, The routing constraints on a subset of $\{p_j\}$ are defined as:

• Matching constraint : $p_j = p_k$

• Bounding Constraint : $p_j \leq p_{jbound}$

and they ensure: $\Delta W_i \leq |\Delta W_{i,max}|$, where $|\Delta W_{i,max}|$ is the maximally allowed performance variation due to the parasitics.

3.4.3 Incorporating performance constraints using Lagrange multipliers

In mathematics, the Karush-Kuhn-Tucker conditions (also known as the Kuhn-Tucker or the KKT conditions) are necessary for a solution in nonlinear programming to be optimal. It is a generalization of method of Lagrange multipliers.

Considering the following nonlinear optimization problem:

minimize $f(x)$
subjected to $g_i(x) \leq 0$ (i = 1, …, m), $h_j(x) = 0$ (j = 1, …, l) (3.3)

The necessary conditions for inequality constrained problem were first published by W. Karush [3.16], and renowned by Harold W. Kuhn and Albert W. Tucker [3.17] as:

Suppose that the objective function, i.e., the function to be minimized, is $f : R^n \to R$ and the constraint functions are $g_i : R^n \to R$ and i. Further, suppose they are continuously differentiable at a point x*. If x* is a local minimum, then there exist constants $\lambda \geq 0$, $\mu_i \geq 0$ (i = 1, …, m) and ν_j (j = 1, …, l) such that

$$\lambda + \sum_{i=1}^{m} \mu_i + \sum_{j=1}^{l} |\nu_j| > 0$$

$$\lambda \nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \nu_j \nabla h_j(x^*) = 0$$
$$\mu_i g_i(x^*) = 0 \text{ for all } i = 1, \ldots, m \tag{3.4}$$

There are no analytical forms for the parasitics or cost functions, therefore it is nearly impossible to obtain μ_i and ν_j in an optimum sense. In order to incorporate the bounding constraints ($p_i - p_{jbound} \leq 0$) and matching constraints ($p_j - p_k = 0$) into the search of minimum assigning cost, net ordering is performed, then μ_i and ν_j are chosen according to the severity of each assignment case. The cost function of each nets with bounding constraint are increased by $\mu_i(p_i - p_{jbound})$. Each pair of matching nets *j* and *k* are routed at the same time, with extra assigning cost of $\nu_j(p_j - p_k)/2$ for each of them. The μ*'s* and ν*'s* are initially chosen to be relatively large so that those constraints will precede other assigning costs. The router then performs the weighted bipartite matching on the updated cost matrix and finds the minimum-cost assignment with the performance constraints enforced.

3.5 Conclusions

MakeLink^{TM} high performance programming technology has made segmented routing practical for FPAAs. We have proposed a performance-driven segmented router based on weighted bipartite matching algorithms. Performance metrics and cost functions are proposed for analog routing to properly reflect the performance degradation caused by interconnects. Resource demands are incorporated into the cost to avoid congestion, and maximum clique routing are used to enhance effectiveness in resource utilization. The next two chapters will discuss with more details on two

important electrical performance of the routing channel, namely, RC delay and cross-talk reduction.

Chapter 4

Interconnection Delay Optimization

4.1 Introduction

One of the most perceptible performance degradation caused by routing parasitics is the interconnection delay [4.1], which directly limits the signal and system bandwidth. Existing routability-driven channel segmentation algorithms have paid little consideration to it. This is acceptable for small scale FPAAs where the routing channels are short enough not to impact the system performance severely. However, as the scale of the FPAA grows, the interconnect delay becomes so significant that it must be taken into account at the earliest stage. Research on minimizing interconnection delays in array- based FPGAs shows that the routing architectures of the chips, as well as the CAD tools dramatically affects speed-performance [4.2].

There are usually three techniques to reduce the delay of an existing topology: transistor sizing, wire sizing and buffer insertion, which have been studied extensively for free channel routings. The optimum transistor sizing, metal width and metal spacing for programmable interconnect have been studied in [4.3]. Once the channel area and number of tracks are given, the optimum metal width and spacing

can be determined regardless the segmentation scheme. Therefore, in this Chapter we focus on the buffer insertion technique, which can either directly reduce the RC delay of a long wire or reduce the net delay by decoupling a large load off the critical path [4.4].

The optimality of Van Ginneken's dynamic programming algorithm [4.5] has inspired numerous variants. Among them, theoretical results have been derived and algorithm proposed for computing the optimum buffer insertion for fixed net trees [4.6], when the library contains only a single, non-inverting buffer. However, routing wires are pre-fabricated in the FPAAs, and then buffer insertions are made while the net tree topology is still unknown. This makes it very difficult to find an overall optimum buffer planning scheme for all circuits to be implemented. In this study, we adopted a greedy approach by inserting the optimum number of buffers for each wire segment in the channel. The lengths and staggering of the segments are chosen to construct the segmentation scheme that requires the fewest buffers while satisfying the delay constraints and routability requirement.

In this Chapter, we propose a design approach combining segmentation and buffer insertion in each stage of the channel design, from segment length selection to track assignment. Experiments show that, compared to a sequential segmenting-then-buffering design, our approach can significantly reduce the total number of buffers required, while achieving improved routability and maintaining the same average interconnect delay. The rest of this Chapter is organized as follows. Section 4.2 overviews some preliminary results of buffer insertion and defines our problem. Section 4.3 presents the combined channel segmentation and buffer insertion

approach, while the experiment results are presented in Section 4.4 and conclusions are given in Section 4.5.

4.2 Buffer insertion

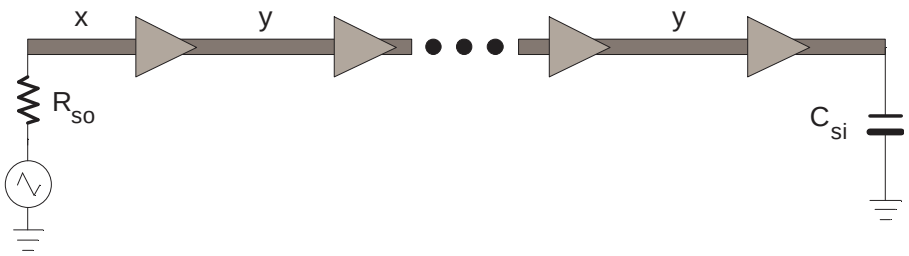


Figure 4.1 Placement of buffers for interconnection delay optimization.

We use the π-model for wire segments and the Elmore model [4.8] to compute the RC delay, which preserves the property that the Elmore delay is the same for a given wire no matter how the wire may be subdivided. For the buffer, we use a switch-level RC model with input capacitance C_b, output resistance R_b and intrinsic delay T_b. Here is a brief summary of the results of [4.6]:

Assuming a uniform-sized buffer, it has been proven that the optimal placement of k buffers is to space them at equal increments except the first one, as shown in Figure 4.1. Let x_d be the distance between the source and the first buffer, and y_d be the distance between two consecutive buffers, R_{so} be the source resistance, C_{si} be the sink capacitance, the Elmore delay caused by the interconnect is given by

$$
\begin{aligned}
D(k, x_d, y_d) = {} & R_{so}(Cx_d + C_b) + RCx_d{}^2/2 + Rx_d C_b + \\
& (k-1)\left[R_b(Cy_d + C_b) + T_b + RCy_d{}^2/2 + Ry_d C_b\right] \\
& + \left(l - x_d - (k-1)y_d\right)\left(R_b C + RC_{si}\right) + T_b \\
& + RC\left[l - x_d - (k-1)y_d\right]^2/2 + R_b C_{si}
\end{aligned}
\tag{4.1}
$$

where k is the number of buffers inserted, l is the wire length, R and C is the unit resistance and capacitance for the wire. The Elmore delay is minimized when

$$
\begin{aligned}
x_d(k) &= \frac{1}{k+1}\left(l + \frac{kR_b - kR_{so}}{R} + \frac{C_{si} - C_b}{C}\right) \\
y_d(k) &= \frac{1}{k+1}\left(l - \frac{R_b - R_{so}}{R} + \frac{C_{si} - C_b}{C}\right)
\end{aligned}
\tag{4.2}
$$

and the optimized delay is given by

$$
\begin{aligned}
D_k = {} & \frac{Rl(kC_b + C_{si}) + (Cl + kC_b + C_{si})(kR_b + R_{so})}{k+1} \\
& + kT_b + \frac{RCl^2 - \dfrac{kR(C_b - C_{si})^2}{C} - \dfrac{kC(R_b - R_{so})^2}{R}}{2(k+1)}
\end{aligned}
\tag{4.3}
$$

The optimum number of buffers for the wire is found to be

$$
k_{opt} = \left\lfloor -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{4U}{V}} \right\rfloor
\tag{4.4}
$$

where

$$
\begin{aligned}
U &= \frac{\left[RCl + R(C_{si} - C_b) + C(R_{so} - R_b)\right]^2}{2RC} \\
V &= (K_b + R_b C_b)
\end{aligned}
\tag{4.5}
$$

In a performance-constraint routing, it is usually preferable to have the interconnect delay constrained by D_c. To find the minimum number of buffers, $k(l)$, to be inserted satisfying that constraint, from (4.3) we note that

$$D_{k-1} - D_k = \frac{U}{k} - \frac{U}{k+1} - V \tag{4.6}$$

Assuming $D_c > D_{k_{opt}}$, we have

$$\begin{aligned} & D_c - D_{k_{opt}} \geq D_{k(l)} - D_{k_{opt}} \\ & = D_{k(l)} - D_{k(l)+1} + D_{k(l)+1} - D_{k(l)+2} + \cdots + D_{k_{opt}-1} - D_{k_{opt}} \\ & = \frac{U}{k(l)+1} - \frac{U}{k_{opt}+1} - V\left[k_{opt} - k(l)\right] \end{aligned} \tag{4.7}$$

Solving for $k(l)$ yields

$$k(l) = \left\lfloor \frac{Q}{2} - \frac{1}{2}\sqrt{Q^2 - \frac{4U}{V}} \right\rfloor \tag{4.8}$$

where $Q = \frac{U}{V(k_{opt}+1)} + (k_{opt}+1) + \frac{D_c - D_{k_{opt}}}{V}$, U and V is given by (4.5). If $D_c \leq D_{k_{opt}}$, we set $k(l) = k_{opt}$.

4.3 Combined segment length selection

For a combined buffer insertion and channel segmentation, to minimize the number of buffers to be inserted, as well as the number of types of segments, Λ_r is chosen as following

$$\Lambda_r = \begin{cases} \Lambda_{rm} & if \quad k(\Lambda_{rm}) = k(\Lambda_{r+1}) \\ MAX\{\Lambda, \quad k(\Lambda) = k(\Lambda_{rm})\} & else \end{cases} \tag{4.9}$$

where Λ_{rm} is the segment length given by the length selection algorithm defined by equation (2.1). The working principle here is to reduce the designated net length range to type r+1 segments, which requires more buffers than type r segments, by up-shifting Λ_r to the longest segments that requires the same number of buffers as Λ_{rm}. An example of this combined segment length selection algorithm is shown in Figure 2.2.

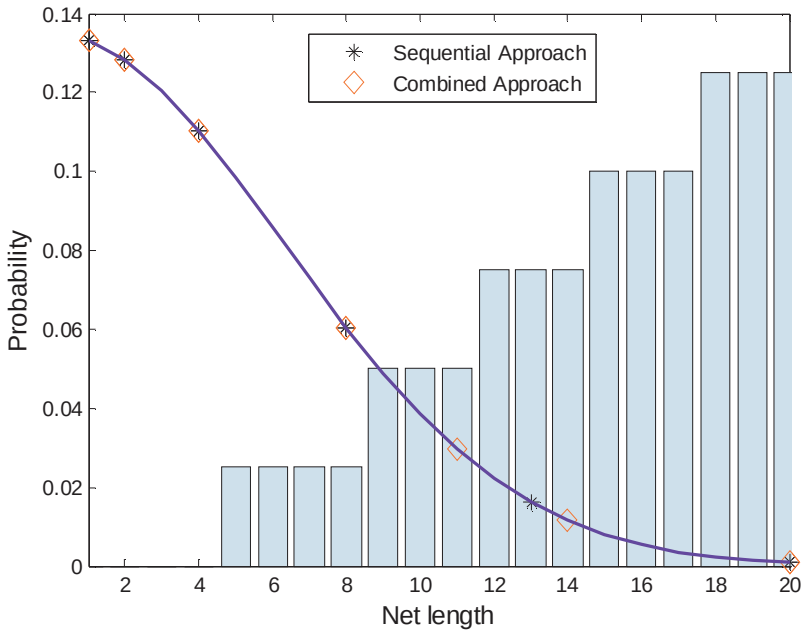


Figure 4.2 Combined segment length selection results

4.4 Delay-driven routing

When minimizing the interconnection delay is the primary object, the cost function of allocating net n to M segments (s_1 to s_M) in track t is then defined as

$$C(n,t) = \left(\sum_{g=1}^{M} Delay(s_g) \cdot N_p(n) \right)^{\alpha} \bullet \left(\sum_{g=1}^{m} Demand(s_g) \right)^{\beta} \tag{4.10}$$

where $N_p(n)$ is the priority weight of net n, and $Demand(s_g)$ is the current demand on segment s_g, as defined in Chapter 3, and a and β are weighting factors. The object of the segment router is to minimize the total interconnection delay, weighted by net priorities.

4.5 Experimental Results

To evaluate the robustness of the proposed channel segmentation and buffer insertion algorithm over different net distributions, we designed channels for six different net length distributions based on geometric, normal and Poisson distributions, the same as those used in Chapter 2. We evaluate the routability and the average interconnection delay of the generated channels using randomly generated routing instances according to these six distributions.

We set the channel length L = 20, total number of tracks T = 20. For proprietary reasons, the electrical properties of the CMOS process we used are not disclosed here. Instead, the parameters for buffer insertion are chosen from the $0.18\mu m$ technology in NTRS'97 roadmap [2.10], which is quite close to those of the actual process: the unit wire resistance $R = 0.075\Omega/\mu m$ and the unit wire capacitance $C = 0.118fF/\mu m$. The

buffer output resistance R_b = 180Ω, the buffer input capacitance C_b = 23.4fF, the intrinsic buffer delay T_b = 36.4ps. The source and sink of a wire are also assumed to be a buffer. The unit length of the channel is assumed to be 100μm.

The total numbers of buffer inserted for one and two-segmentation are shown in Figure 4.3 with respect to the channel staggering factors δ_1 and δ_2 defined in Chapter 2. It's seen that larger δ_2 generally results in more buffers because it means more tracks are allocated to long segments. Since two-segmentation channels contain wire segments much shorter than those in the one-segmentation channels, they also require much less buffers.

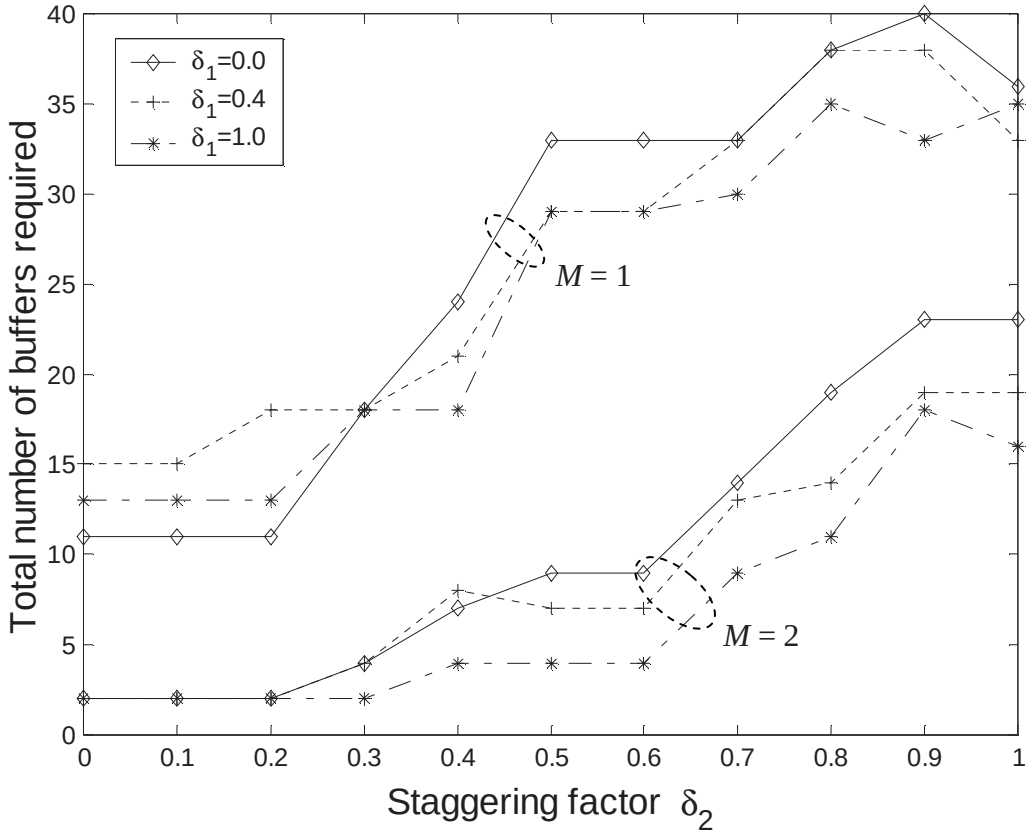


Figure 4.3 The effects of δ_1, δ_2 on buffer insertion

Figure 4.4 shows the unit-length delays, which are the average net delay per one logic block length. It's seen that for one-segmentation the unit-length delay increases with δ_2, since longer segments usually results in larger delay. Note that in Chapter 2 we found out that a staggering factor δ_2 greater than zero usually improves the routability. Therefore, there exists a tradeoff between the channel routability and interconnection delay for one-segmentation channels. On the contrary, the unit-length delay actually decreases with δ_2 for two-segmentation channels, because longer segments reduce the usage of connection switches, which contribute a considerable portion to the overall interconnect delays.

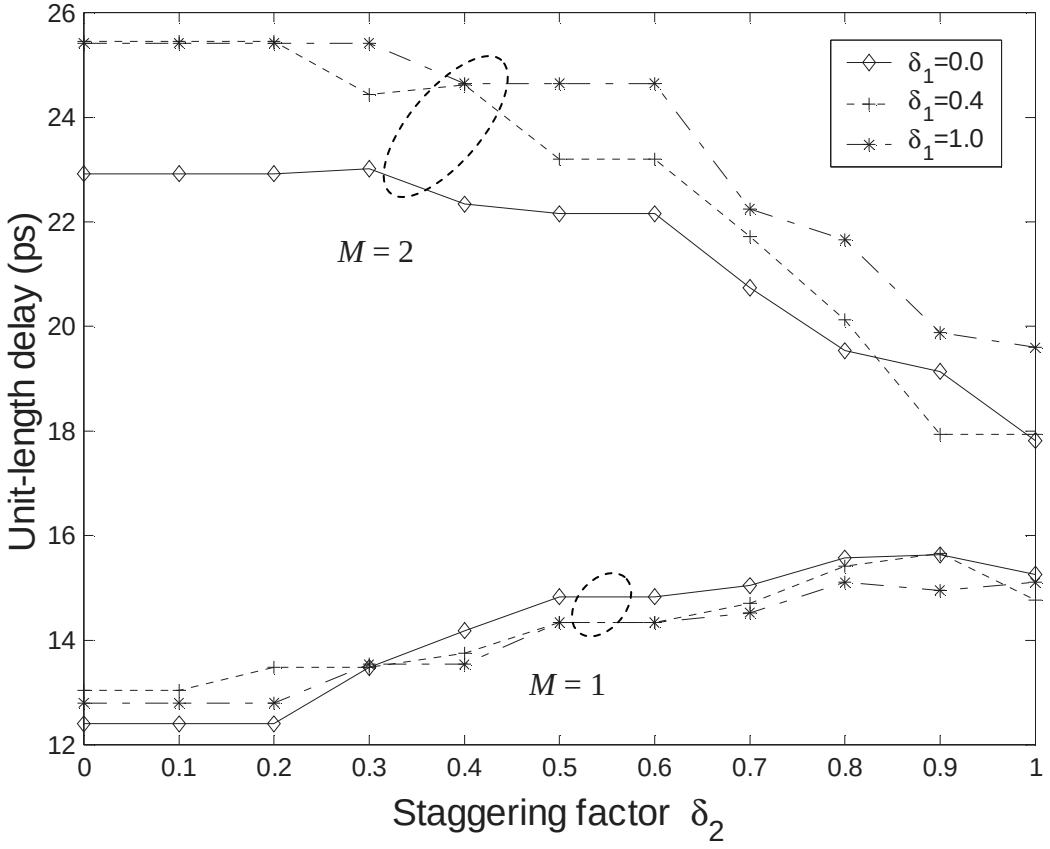


Figure 4.4 The effects of δ_1, δ_2 on interconnect delay.

Obviously the value of δ_2 can be chosen to provide a tradeoff among routability, speed and area costs. We use the metric $A = S^{\gamma} K^{-\nu} D^{-\sigma}$ to evaluate the quality of different channel segmentation and buffer insertion schemes, where *S* is the success rate, *K* is the number of buffers required and *D* is the unit-length delay. The parameters γ, ν and σ are used to trade off between routability and area/speed costs. For γ = 3, ν = 1 and σ = 3 (i.e., routability and interconnection delay are more important concerns than area increase), the experimental results for all six distributions are given in Table 4.1. Over the simple case of $\delta_1 = \delta_2 = 0$, the optimized channel segmentations have an average improvement of 66.5% (46.4%) on the routing success rate, at the cost of an increase of 16.7 (12.7) on the number of buffers and 1.8*ps* increase (4.1*ps* decrease) on the unit-length delay for one-segmentation (two-segmentation) designs.

Table 4.2 shows the comparison results of the combined approach to a sequential, i.e., buffer insertion after channel segmentation approach. It seen that the combined approach can reduce the number of buffers by an average 13.7% (27.2%) and achieve a 1.5% (2.4%) increase on the routing success rate, with only 0.4% (3.0%) increase on the unit-length delay for one-segmentation (two-segmentation) designs.

Table 4.1 Experimental results for all six net distributions

		Optimized			$\delta_1 = \delta_2 = 0$		
		S	*K*	*D* (*ps*)	*S*	*K*	*D* (*ps*)
M=1	Ge1	86.4%	30	14.51	6.2%	11	12.41
	Ge2	74.0%	69	12.91	39.8%	58	11.69
	No1	65.0%	31	16.40	2.6%	9	14.30
	No2	85.4%	71	13.87	27.0%	51	11.61
	Po1	91.4%	14	15.12	4.8%	2	13.99
	Po2	84.2%	55	12.73	7.2%	39	10.95
M=2	Ge1	98.4%	19	17.95	13.0%	2	22.91
	Ge2	84.6%	43	19.16	62.8%	32	21.27
	No1	76.4%	4	23.63	51.0%	0	25.82
	No2	86.0%	37	19.98	59.2%	26	22.57
	Po1	98.2%	0	25.11	38.2%	0	24.54
	Po2	98.8%	42	16.70	39.6%	9	30.29

Table 4.2 Comparison with a sequential approach

	M=1			*M*=2		
	ΔS	*ΔK*	*ΔD*	*ΔS*	*ΔK*	*ΔD*
Ge1	-3.6%	-38.8%	5.4%	-0.6%	-60.4%	1.5%
Ge2	1.4%	-8.0%	-1.7%	-12.2%	-32.8%	7.3%
No1	5.5%	-3.1%	2.0%	3.5%	0.0%	-9.4%
No2	-3.2%	-5.3%	0.2%	-10.4%	-32.7%	8.6%
Po1	0.0%	0.0%	0.0%	34.5%	0.0%	-14.4%
Po2	9.1%	-26.7%	-3.4%	-0.4%	-37.3%	24.1%
Avg	1.5%	-13.7%	0.4%	2.4%	-27.2%	3.0%

4.6 Conclusions

In this Chapter, we addressed the problem of minimizing the interconnection delay of FPAA routing. A combined channel segmentation and buffer insertion approach was proposed, and delay-driven analog routing developed. Staggering factors are used to provide tradeoff among routability, interconnect delay and area costs. Experiments show that the combined approach can significantly reduce the total number of buffers required, while improving the routability and minimizing the interconnect delay.

Chapter 5

Cross-coupling Noise Deduction

5.1 Introduction

As modern IC processes scale to smaller size, wire spacing diminishes and coupling capacitance increases. Furthermore, the thickness of the wires is increased to maintain the same resistance per unit length, which increased the ratio of coupling to total parasitic capacitance. As a result, crosstalk has become an increasingly important design metric to timing, noise and area in VLSI designs. Analog systems are more susceptible to noise distortion because of their continuous nature. The situation is made even worse in a routing channel for FPAAs, where many wire segments are placed adjacent to one another in parallel, as shown in Figure 5.1, while the spacing between them is stringently limited by the available die area.

A number of works have been addressed on on-chip interconnect optimization problem. Among the popular techniques, buffer insertion has gained wide acceptance in deep submicron design. Closed form solution for computing the optimum number and location of buffers that minimize the Elmore delay of a two-pin net has been found by Alpert in [5.1]. Pioneering works on buffer block planning for interconnect-driven

floorplanning were presented by Cong, Kong and Pan in [5.2], and by Sarkar et al in [5.3], which compute the feasible regions of buffer insertion for timing optimization.

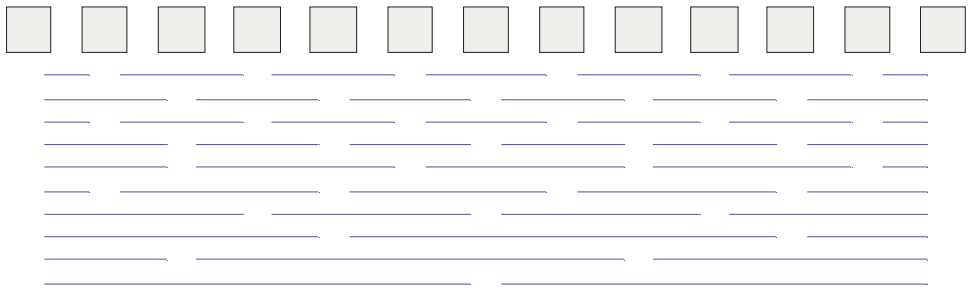


Figure 5.1 Wire segments in a FPAA routing channel

Various techniques have been proposed over the past few years on noise analysis and avoidance. Since circuit simulation techniques, such as SPICE, are not suitable for a physical design system due to the prohibitive computational cost, an effective noise metric suitable for high level CAD tools is required. Vittal and Marek-Sadowska model a coupled network as a simplified RC circuit and derived analytical expressions for noise from the resultant circuit [5.6]. The recently proposed noise metric of Devgan [5.7] considers circuit effects such as input slew rate, line resistance and coupling capacitance. Its similarity to the Elmore delay metric is particularly amenable for use within a physical design optimization tool. Using this metric, Alpert et al [5.1] presented buffer insertion techniques for noise avoidance for single sink and multiple sink trees, and simultaneous noise and delay optimization, while Li, Cherng and Chang proposed buffer insertion at the floorplanning stage instead of routing and post-layout stages [5.5].

Most of previous works on noise-avoiding buffer insertion are for digital signals, where a buffer working as a restoring stage. If an intermediate buffer is present, the noise computation begins from the output of that buffer, which is not the case for analog buffering. An analog buffer will replicate exactly what it receives, and therefore cannot distinguish the noise from the signal but pass it to the next stage. Instead, analog buffers reduce coupling noise by breaking the long wire into pieces and then preventing the downstream induced-current from adding noise to wire segments after it. Those differences make the behavior of buffer insertion quite different from it in digital systems. In this Chapter, we investigated noise and delay optimization by buffer insertion for analog systems. Similarities between the noise metric and Elmore delay model are exploited. Analytical results for both noise optimization alone and noise constrained delay optimization are derived. The results are then applied to the design of segmented routing channels for three different target net distributions. Experiments show that coupling noise optimization alone can reduce the noise by 30%. Compared to delay optimization only, the noise constrained delay optimization can eliminate the coupling noise violation with only 8% increase on the number of buffers and a delay performance penalty less than 4.5%.

The rest of this Chapter is organized as follows. Section 5.2 gives some preliminaries and Section 5.3 presents the analysis of buffer insertion for noise optimization and noise constrained delay optimization for analog signals. Analytical formulae are derived and optimal buffer insertion algorithms are proposed. The results are then applied in the design of a FPAA routing channel as described in Section 5.4. Experiment results are given in Section 5.5 and conclusions are presented in Section 5.6.

5.2 Preliminaries

The Devgan noise estimation metric, which depends on the victim net resistance, the driving gate resistance, coupling capacitances to the aggressor nets and the slops of the signals on the aggressor nets, is an upper-bound for *RC* and over-damped *RLC* circuits. The preliminary results in [5.7] and [5.1] are repeated here.

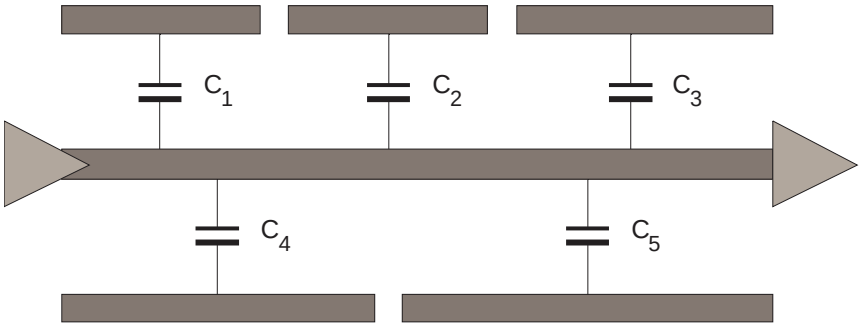


Figure 5.2 Coupling noise due to multiple aggressor nets.

For a wire *e* coupled to *t* aggressor nets, as shown in Figure 5.2, let C_i be the coupling capacitance from net *i* to wire *e*, and μ_i be the slopes (voltage changing rate) of the signal on net *i*, the total current induced by the aggressor nets is given by

$$I_e = \sum_{i=1}^{t} \mu_i C_i \tag{5.1}$$

The coupling capacitance is proportional to the coupling length l_i, and inversely proportional to d_i, the distance between the aggressor and the victim,

$$C_i = C_c \, l_i d_{\min} / d_i \tag{5.2}$$

where C_c is the coupling capacitance for unit length and spacing d_{min}, the minimum wire distance specified by the design rule.

Often, information about neighboring aggressor nets is unavailable before routing. To perform buffer insertion in an estimation mode, we assume μ as the slope for all aggressor signals. In the worst case that the victim is fully coupled from both sides, and the distance to be the minimum wire distance specified by the design rule, we have

$$I_e = \mu C_c \sum_{i=1}^{t} l_i = I l_e \tag{5.3}$$

where $I = 2\mu C_c$ is the unit length coupling current.

A path from the source node s_o to the sink node s_i may consist of multiple wires. For a wire $e = (u,v)$, let $I_{T(v)}$ be the total downsteam current seen at v. The total coupling noise perceived at the sink s_i is given by

$$Noise(so - si) = R_{so} I_{T(so)} + \sum_{e \in path(so-si)} R_e \left(I_e / 2 + I_{T(v)} \right) \tag{5.4}$$

where R_e is the resistance of wire e.

A buffer prevents the downstream induced-current from adding noise to wire segments after it. If one buffer is inserted at each node in the path, the coupling noise perceived at the sink s_i is reduced to

$$Noise(so - si) = \sum_{e \in path(so-si)} \left(R_u I_e + R_e I_e / 2 \right) \tag{5.5}$$

where R_u is the buffer output resistance at node u.

5.3 Buffer insertion

Since each wire segment can be viewed as a two-pin net, therefore, we first study the noise optimization problem and the noise constrained delay optimization problem on a two-pin net with fixed length, derive closed form formulae and then apply the results in constructing the segmented routing channel.

In buffering planning for routing channel of FPAAs, there are usually two problems similar to those in [5.1]. The first one is to find the optimum number and placement of buffers to optimize the noise only, which is useful in handling non-critical nets. For critical nets where delay optimization is necessary, the second problem formulation seeks to minimize the delay while satisfying the noise constraint. We will discuss those two problems in more details below.

5.3.1 Coupling noise constrained only

In Section II we see the Devgan noise metric works in analog buffering exactly the same way as Elmore model for delays, by replacing C with I except that there are no terms corresponding to C_b, T_b and C_{si} since buffer itself does not incur coupling current. Most of the conclusions on delay optimization can be applied directly here. For example, following the same arguments in [5.1], we conclude that the optimal placement of buffers is to space them at equal increments except the first one. Let x_n be the distance between the source and the first buffer, and y_n be the distance between two consecutive buffers, the coupling noise perceived by the sink node is given by

$$N(k, x_n, y_n) = R_{so} I x_n + R I x_n{}^2 / 2 + (k-1)\left[R_b I y_n + R I y_n{}^2 / 2\right] + R_b I\left[l - x_n - (k-1) y_n\right] + R I\left[l - x_n - (k-1) y_n\right]^2 / 2 \tag{5.6}$$

The optimum coupling noise by inserting k buffers for a given wire of length l is achieved when

$$x_n(k) = \frac{1}{k+1}\left(l + \frac{kR_b - kR_{so}}{R}\right)$$
$$y_n(k) = \frac{1}{k+1}\left(l - \frac{R_b - R_{so}}{R}\right) \tag{5.7}$$

and the optimized coupling noise is given by

$$N(k) = \frac{Il(kR_b + R_{so}) + \frac{RIl^2}{2} - \frac{kI(R_b - R_{so})^2}{2R}}{k+1} \tag{5.8}$$

From (5.8) we note that

$$N(k-1) - N(k) = \frac{U}{k(k+1)} \tag{5.9}$$

where

$$U = \frac{\left[RIl + I(R_{so} - R_b)\right]^2}{2RI} \tag{5.10}$$

Let $N(0)$ be the coupling noise perceived by the sink node when no buffer is inserted, we have

$$N(0) - N(k) = \sum_{i=1}^{k}\left[N(i-1) - N(i)\right] = \frac{k}{k+1}U \tag{5.11}$$

which shows we can obtain a noise reduction of U/2 by inserting the first buffer. By inserting more buffers, we can still get some more reduction on the coupling noise, but less significant as the number of buffers increase. Eventually, U is the maximum coupling noise reduction that can be achieved by buffer insertion.

From (5.8) we also have

$$N(0) = \frac{RI}{2} l^2 + R_{so} Il \tag{5.12}$$

which increases as the square of the wire length l. The minimum coupling noise we can achieve by inserting infinite buffers is given by

$$N(\infty) = N(0) - U = R_b Il - \frac{RI}{2}\left(\frac{R_{so} - R_b}{R}\right)^2 \tag{5.13}$$

which is linear in terms of l. therefore, buffer insertion reduces the cross-talk by decrease the order of the coupling noise with respect to the wire length. Practically, we need only insert nine buffers to achieve 90% of the maximum reduction.

To find the minimum number of buffers, k_n, to satisfy the noise constraint, $N(k_n) \leq N_c$, we have

$$N(0) - N(k_n) \geq N(0) - N_c \tag{5.14}$$

Solving for k_n yields

$$k_n = \left\lceil \frac{N(0) - N_C}{U - \left(N(0) - N_C\right)} \right\rceil \tag{5.15}$$

5.3.2 Delay optimization with noise constraint

This problem formulation integrates the delay into the solution, trying to minimize the delay while satisfying the noise constraint. Since the optimal buffer placement for both delay and noise optimization is to space them at equal increments except the first one, this conclusion still applies here. Let k be the number of buffers to be inserted, x be the distance between the source and the first buffer, and y be the distance between two consecutive buffers, we first try to find a placement (x, y) that minimize $D(k, x, y)$ while $N(k, x, y) \le N_c$.

First, if $N(k, x_d(k), y_d(k)) \le N_c$, we already find the solution because $(x_d(k), y_d(k))$ is optimum placement for delay. Otherwise, the optimal (x, y) can be found by Lagrange's method as

$$\frac{\partial D(k,x,y)}{\partial x} - \lambda \frac{\partial N(k,x,y)}{\partial x} = 0$$

$$\frac{\partial D(k,x,y)}{\partial y} - \lambda \frac{\partial N(k,x,y)}{\partial y} = 0 \qquad (5.16)$$

under the condition that

$$N(k,x,y) = N_c \qquad (5.17)$$

From (5.16) we have

$$\frac{\partial D(k,x,y)/\partial x}{\partial N(k,x,y)/\partial x} = \frac{\partial D(k,x,y)/\partial y}{\partial N(k,x,y)/\partial y} \qquad (5.18)$$

which yields

$$(C_b - C_{si})\left(x - y + \frac{R_{so} - R_b}{R}\right) = 0 \tag{5.19}$$

When $C_{si} = C_b$, any (x, y) will satisfy (23). Also from (2) and (11) we see $x_n(k) = x_d(k)$, $y_n(k) = y_d(k)$ when $C_{si} = C_b$, which means the same placement of buffers simultaneously optimize the delay and coupling noise, that is also the optimum placement for this problem.

When $C_{si} \neq C_b$, we have

$$y = x + \frac{R_{so} - R_b}{R} \tag{5.20}$$

Let $y = y_n(k) + \Delta_n$, using Taylor's expansion we have

$$\begin{aligned} N_c - N(k) &= N(k, x, y) - N(k, x_n(k), y_n(k)) \\ &= \left.\frac{dN}{dy}\right|_{y_n(k)} \Delta_n + \frac{1}{2}\left.\frac{d^2N}{dy^2}\right|_{y_n(k)} \Delta_n^{\,2} \\ &= \frac{RI}{2} k(k+1)\Delta_n^{\,2} \end{aligned} \tag{5.21}$$

Note that $(x_d(k), y_d(k))$ and $(x_n(k), y_n(k))$ also satisfy (24), therefore those three points are actually in a line on the XY plate. Since we know (x, y) must lie somewhere between $(x_d(k), y_d(k))$ and $(x_n(k), y_n(k))$, we have

$$\Delta_n = \kappa\sqrt{2\frac{N_c - N(k)}{k(k+1)RI}} \tag{5.22}$$

where $\kappa = sign(C_{si} - C_b)$.

Let $y = y_d(k) - \kappa\Delta_d$, we have

$$\Delta_d = \left|\frac{C_{si} - C_b}{(k+1)C}\right| - \sqrt{2\frac{N_c - N(k)}{k(k+1)RI}} \tag{5.23}$$

When $\Delta_d > 0$, the optimum delay is given by

$$\begin{aligned} D_p(k) &= D(k) + \left.\frac{dD}{dy}\right|_{y_d(k)} \kappa\Delta_d + \frac{1}{2}\left.\frac{d^2N}{dy^2}\right|_{y_d(k)} {\Delta_d}^2 \\ &= D(k) + \frac{RC}{2}\left(\left|\frac{C_{si} - C_b}{C}\right| \cdot \sqrt{\frac{k}{k+1}} - \sqrt{2\frac{N_c - N(k)}{RI}}\right)^2 \end{aligned} \tag{5.24}$$

When $\Delta_d \le 0$, which means $N(k, x_d(k), y_d(k)) \le N_c$, the optimum solution is ($x_d(k)$, $y_d(k)$) and the optimum delay is $D(k)$. Note that $(k+1)\Delta_d$ is a decreasing function of k. Let

$$k_p = \left\lceil \frac{N(0) - N_C}{U - (N(0) - N_C) - RI(C_{si} - C_b)^2 / 2C^2} \right\rceil \tag{5.25}$$

(If $U - (N_0 - N_C) \le RI(C_{si} - C_b)^2 / 2C^2$, $k_p = \infty$), then for all $k \ge k_p$, $\Delta_d \le 0$. Compare (5.25) and (5.15) we see that k_p is always no less than k_n, the minimum number of buffers required to satisfy the noise constraint, which means the noise constraint is satisfied for all $k \ge k_p$.

When $k_p \le k_d$, the optimum number of buffers for delay optimization only, the case is simple: we have $k_{opt} = k_d$. When $k_p > k_d$, the optimum k is the first one in the range [k_n, k_p] which satisfies

$$D_p(k+1) - D_p(k) \ge 0 \tag{5.26}$$

which can be found by numerical methods. Figure 5.3 presents the algorithm, which returns an optimal solution to the noise constrained delay optimization problem for a two-pin net.

Algorithm:	Buffer Insertion for Noise Constrained Delay Optimization
Input:	$\{l, R, C_0, R_{so}, C_{si}\} \equiv$ two-pin net $\{R_b, C_b, T_b\} \equiv$ buffer type $\{C_c, \mu\} \equiv$ coupling parameters $N_c \equiv$ Coupling noise constraint
Output:	$k \equiv$ number of buffers $\{x, y\} \equiv$ placement of buffers

$C = C_0 + 2C_c$; $I = 2\mu C_c$;

1. Compute the solution to delay optimization only, k_d and $\{x_d(k), y_d(k)\}$, from (4) and (2) respectively;
2. Compute the solution to noise optimization only, k_n and $\{x_n(k), y_n(k)\}$, from (5.15) and (5.7) respectively;
3. if $C_{si} = C_b$ then
   $k = MAX(k_d, k_n)$;
4. else
   compute k_p from (5.25);
   if $k_p \le k_d$ then $k = k_d$;
   else
   find k_m in $[k_n, k_p]$ which minimize $D_p(k)$ defined by (5.24);
   $k = k_m$;
5. $x = x_n(k) + \Delta_n$; $y = y_n(k) + \Delta_n$;
   where Δ_n are given by (5.22).

Figure 5.3 Algorithm for noise-constrained delay optimization.

5.4 Experimental Results

The parameters for buffer insertion are chosen from the 0.18μm technology in NTRS'97 roadmap [2.10]: the unit wire resistance R = 0.075Ω/μm, the unit plate wire capacitance C_0 = 0.118*fF*/μm and the coupling capacitance C_c = 0.09*fF*/μm. The buffer output resistance R_b = 180Ω, the buffer input capacitance C_b = 23.4*fF*, the intrinsic buffer delay T_b = 36.4ps. Same as [5.1], we assume a maximum aggressor voltage changing rate of 7.2*V*/ns. Let N_∞ be the limit on noise achievable by inserting buffers on the longest wire segment, the noise constraint is set to be 1.3 times N_∞.

The buffer insertion algorithms proposed in Section IV are used to obtain buffer requirement information that are incorporated in designing channels for three different net length distributions, namely, geometric, Poisson and bi-peak distributions, the same as in Chapter 4. We set the channel length to be 40 analog blocks, each having a width of 300μm. There are totally 40 tracks in each channel.

For each of the three channel constructed, we randomly generated 200 instances, each with 50 nets using the target length distribution, and assigned them to the routing segments in the channel. Table 5.1 and Table 5.2 show the number of buffers required, average/maximum delay, average/maximum noise and the number of noise violation detected for the case $C_{si} = C_b$ (for example, the sink is also a buffer) and $C_{si} = 10C_b$, respectively.

It is seen that though the average noise resulted by delay optimization may be lower than the noise constraint, delay optimization alone cannot eliminate noise violations.

Noise optimization alone can fix the violations with an increase of 10% on delay. Note that it requires much less buffers than the delay optimization. The noise constrained delay optimization requires an average 8% more buffers than delay optimization alone, and cause a delay increase of only 0.06% on average and 4.5% maximum.

Table 5.1 Experimental results when $C_{si} = C_b$

		# Buffers	Delay(ps)		Noise (V)		Noise Viol.
			Avg.	Max.	Avg.	Max.	
Delay Opt only	Ch1	24	65.08	1151.8	0.25	4.01	18
	Ch2	119	380.28	1151.8	1.43	4.01	1
	Ch3	94	175.54	1151.8	0.66	4.01	11
Noise Opt. only	Ch1	13	65.65	1202.7	0.26	3.53	0
	Ch2	15	456.74	1202.7	1.93	3.53	0
	Ch3	14	191.98	1202.7	0.80	3.53	0
Noise Const. Delay Opt.	Ch1	28	65.17	1202.7	0.25	3.53	0
	Ch2	123	380.28	1202.7	1.43	3.53	0
	Ch3	98	175.60	1202.7	0.66	3.53	0

Table 5.2 Experimental results when $C_{si} = 10C_b$

		# Buffers	Delay(ps)		Noise (V)		Noise Viol.
			Avg.	Max.	Avg.	Max.	
Delay Opt	Ch1	81	213.45	1269.9	0.39	3.94	1
	Ch2	222	547.52	1269.9	1.53	3.94	61
	Ch3	198	284.20	1269.9	0.64	3.94	38
Noise Opt.	Ch1	14	227.91	1324.3	0.42	3.53	0
	Ch2	122	661.87	1324.3	1.93	3.53	0
	Ch3	10	339.41	1324.3	0.79	3.53	0

Noise Const. Delay Opt.	Ch1	85	213.45	1323.3	0.39	3.53	0
	Ch2	266	548.03	1323.3	1.52	3.53	0
	Ch3	202	284.40	1323.3	0.64	3.53	0

5.5 Conclusions

Interconnect optimization is an important step in design of field programmable devices. In this Chapter, we investigated the coupling noise avoidance problem for analog signals using the Devgan noise metric. Analytical results of buffer insertion for noise optimization and delay optimization with noise constraint for two-pin net are derived, and applied in the design of routing channels for a field programmable analog array. Experiments show that, compared to optimizing delay only, optimizing both the noise and delay only causes increase of 8% on the number of buffers required and only 0.06% on the average interconnection delay.

Chapter 6

Configurable Analog Block Design

6.1 Introduction

The design of the Configurable Analog Block the basic cell used in FPAAs, is essential in the FPAA design. It is usually influenced by a number of factors, including the class of circuits to be implemented, area-efficiency, routing resources available and versatility requirements. There are several important choices need to be made before the routing architecture can be determined. Those choices will strongly influence the area, performance and variety of circuits that can prototyped by the device, which are discussed in details below.

The first issue to be considered is the level of granularity. Fine-grained architectures (reconfigured at the transistor level) will be versatile than a coarse-grained architecture (reconfigured at a macro-block level. e.g. amplifiers), but also require more routing resources and will have more switches in the signal path, which prohibit them to realize high-complexity, high-speed circuits. Except in some research on evolvable hardware [6.1], coarse-grained CABs are adopted by the

majority of current commercial and academic FPAAs. Therefore, we chose to design a coarse-grained CAB in our design.

The operation mode is also a key choice when design the CAB. Discrete-time approaches, such as switched-capacitor circuit techniques, are more robust to noise and offset and hence do not require the use of on-chip tuning circuitries. However, such sampled-data techniques require that input signals be band-limited to at least one-half the sampling frequency, and hence anti-aliasing and reconstruction filters must be used. On the other hand, continuous-time circuits do not need band-limited input signals, but are more sensitive to noise, mismatch and process variations. They also require more complicated implementations to have circuit components programmable over a large dynamic range. Existing FPAAs can only implement circuits in either discrete-time mode or continuous-time mode. However, it is highly desirable to have a CAB with the capability of implementing both kinds of circuits, and therefore provide the customer more ways to optimize the system performance.

The signal parameter is another important choice: voltage or current. The simple implementation of some operations (e.g., algebraic addition) and possible larger linearity range with low power supply make current-mode circuits very attractive. However, voltage-mode circuit techniques are well-developed, and voltage signals have a high fanout. Moreover, discrete-time approaches such as switched-capacitor circuits have been predominantly voltage mode. Based on those considerations, we chose voltage as the signal parameter.

Other issues include whether to make the distinct CABs for different circuits, or make them all identical but programmable to implement different functions. While

some commercial devices designed for a targeted application have different CABs for specific functions (like digital, analog and I/O), others and most academic devices assume a uniform CAB structure, which was also considered in our study.

The rest of this chapter is organized as follows. Section 6.2 gives some literature review of existing CAB topologies, and Section 6.3 presents our high flexibility CAB and its internal routing architecture design. Routing channels with different segmentation schemes are investigated, and experimental results are given in Section 6.4. The conclusions are presented in Section 6.5.

6.2 Existing CAB Topologies

Various topologies have been proposed for coarse-granularity CABs. Most of them fall in either of the two categories: continuous-time mode and discrete-time mode.

Lee and Gulak presented a CAB in there pioneering work on FPAAs [6.2], where pass transistors controlled by SRAM based memory elements, were used as the active switch elements that connected basic resources such as differential pairs, current mirrors and transistors. A transconductor-Based FPAA described later [6.3] consists of operational amplifiers and programmable capacitors linked by a transconductor based interconnection array. A more general, continuous-time FPMA prototype IC that allows analog and digital signals to be exchanged on-chip was described in [6.4]. For CABs using other core building blocks, a CAB consisting of the programmable OTA, capacitor and MOSFET switches was proposed in [6.9]. A digitally controllable Gm-cell achieved by a set of binary weighted unit-

transconductor is presented [6.10]. Recently, floating–gate pFET switches have been explored to help FPAAs to enter the realm of large–scale reconfigurable devices such as modern FPGAs [6.11]. Instead of using voltage as the signal parameter, current-mode bipolar FPAA was presented in [6.5], [6.6], [6.7], [6.8], which employed current-mode techniques to avoid the penalty of switches in the signal path to achieve high performance.

CABs based on discrete-time approaches, such as switched-capacitor circuit techniques have been announced by IMP [6.12], and many other researchers [6.13], [6.14], [6.15], aimed at general-purpose signal conditioning tasks in medical, industrial, or other instrumentation and control systems. Switch capacitor techniques show good promise for further development for applications below 1MHz. The technology is quite mature and well understood. However, inherent limitation exists. Namely, switched capacitor technique is a sampled data technique which requires continuous time filters for anti-aliasing and reconstruction. To minimize continuous time filtering, it is usually required to have the switched-capacitor filter cutoff frequency one order of magnitude lower than the sampling frequency. A similar CAB using pulse-width modulated digital signals to convey analog signal information between programmable analog cells is presented in [6.16], which faces the same shortcoming.

6.3 High flexibility CAB design

As mentioned above, all existing CAB are designed to implement either continuous-mode or discrete-mode circuits, and are usually optimized for certain

kind of target circuits like filters. In each implementation, the allowable configurations are usually very limited, so are the analog functions that can be realized. To take advantages of both circuit techniques, a CAB that can be configured to implement both continuous-time and switched-capacitor circuit is very desirable, which essentially requires a high flexibility internal CAB routing architecture allowing arbitrary connections of its components.

6.3.1 CAB Topology

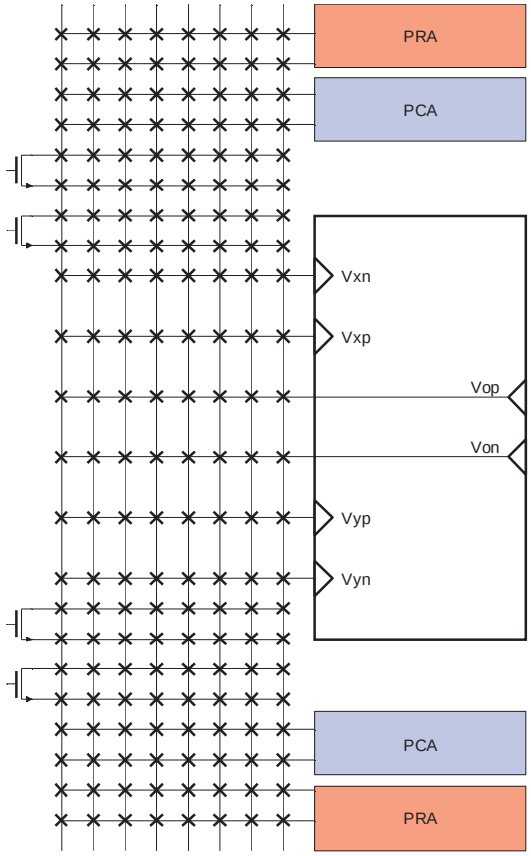


Figure 6.1 Illustration of a high flexibility CAB topology

In our study, a general routing approach was adopted in the internal CAB topology. All terminals of the CAB components have access to a routing channel,

which allows any two or more of them to be connected. The number of configurations is limited only by the available routing resources, namely, the number of routing tracks in the channel. Figure 6.1 shows an Illustration of a high flexibility CAB.

6.3.2 Components

Though the resources available in a CAB vary widely between different devices commercially available and those still in research, it usually contains an operational amplifier (or an operational transconductor), some passive elements like programmable capacitor arrays (PCAs) and programmable resistor arrays (PRAs), plus a set of pass transistor switches if switched-capacitor circuit techniques are to be utilized.

Operational Amplifier

The ideal operational amplifier is one with high gain and bandwidth, a wide dynamic range, a low power dissipation and good power supply rejection ratio. A fully differential configuration is desired because it is less susceptible to common-mode/coupling noise, providing larger output swing and better linearity by reducing even-order harmonics. The OPAMP used in out design features a telescopic cascoding input stage that can work alone as a high speed OTA or preamp, and a detachable Class AB second stage for resistive load, larger output swing and slew rate. Its schematic is shown in Figure 6.2

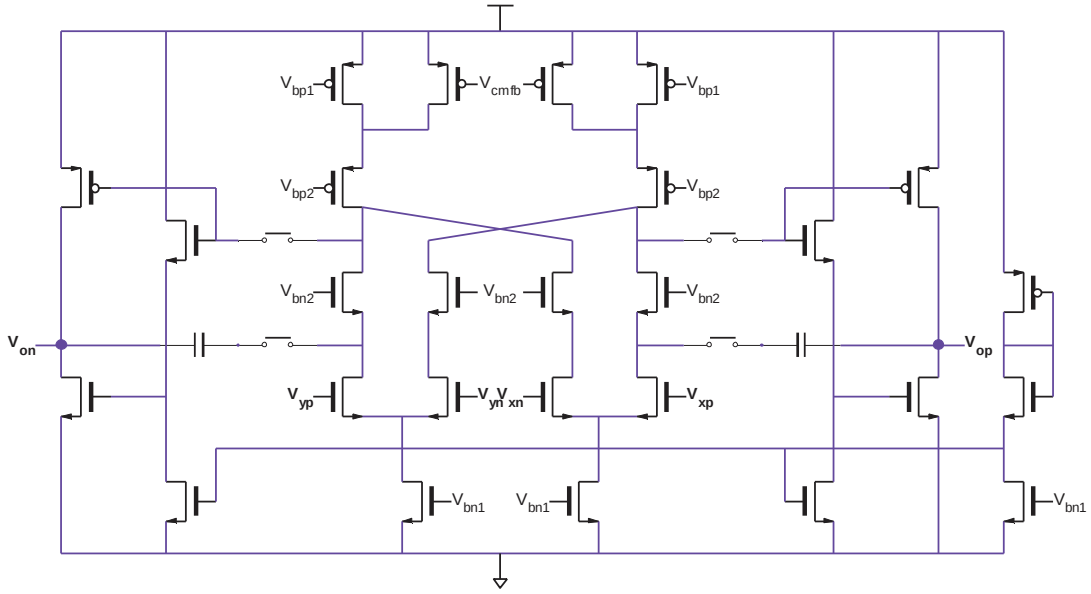


Figure 6.2 Schematic of the differential OPAMP

Programmable capacitor array

Usually PCA are made of capacitors adding in parallel, each of which must have a capacitance significantly larger than the parasitic capacitance associated with the substrate and interconnect. Therefore, PCAs usually occupy a large amount of area of the CAB. The way to realize a PCA with 4-bits precision by 31 equal parts is shown in Figure 6.3.

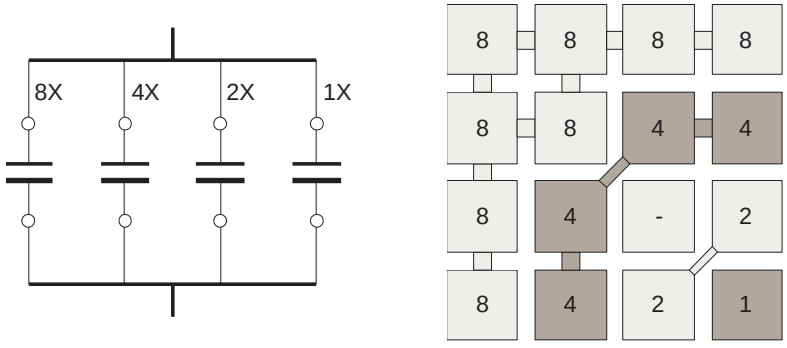


Figure 6.3 Layout of a PCA with 4-bit precision

Programmable resistor array

Various methods may be used to implement programmable resistors, including the use of polysilicon resistors switched into circuits, complementary MOS transistor pairs with controlled gate voltages and more complex transistor implementations of programmable resistive elements such as MOS transconductors.

Controllable switches

In order to realized dynamic circuits like sample-and-hold, and to facilitate the implementation of switched-capacitor circuits, pass-transistors or transmission gates are included not for configuring the circuits, but as routable components themselves whose terminals can be connected to other components to form the circuit wanted.

6.3.3 Symmetrical CAB

There are some other considerations that need to be taken care of in designing the routing channel. The first one is symmetry. To take full advantage of the differential OPAMP in combating common-mode noise, a totally symmetrical routing structure between the differential input and output terminals of the OPAMP is used. Considering the fact that input signals to those two terminals are usually independent to each other in a fully differential circuit topology, the routing channels can be divided in to two identical branches, which ensure the symmetry while reducing the area cost by half. Since some global nets like the reference voltages that need to access both branches, global nets are provides which are usually long lines. The improved CAB diagram is shown in Figure 6.4.

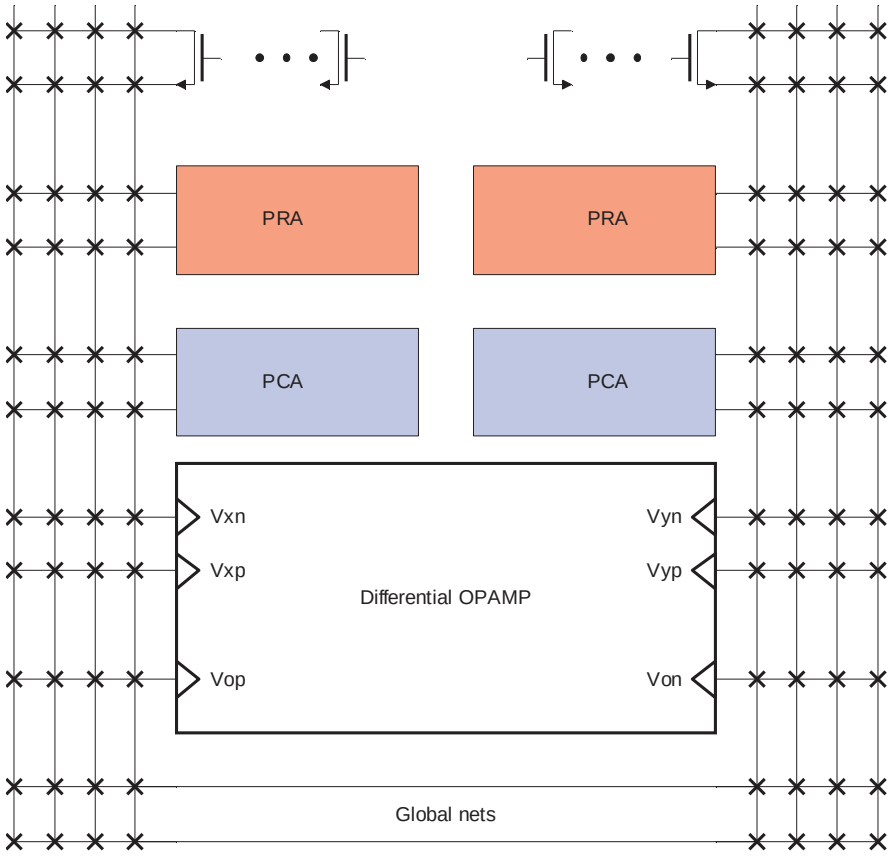


Figure 6.4 Diagram of a symmetrical CAB

6.3.4 CAB layout

The layout of the whole CAB is shown in Figure 6.5. The digital portion (latch, inverter and buffers) are confined in the upper comer and surrounded by guard ring. The programmable resistor arrays are placed under the routing channel.

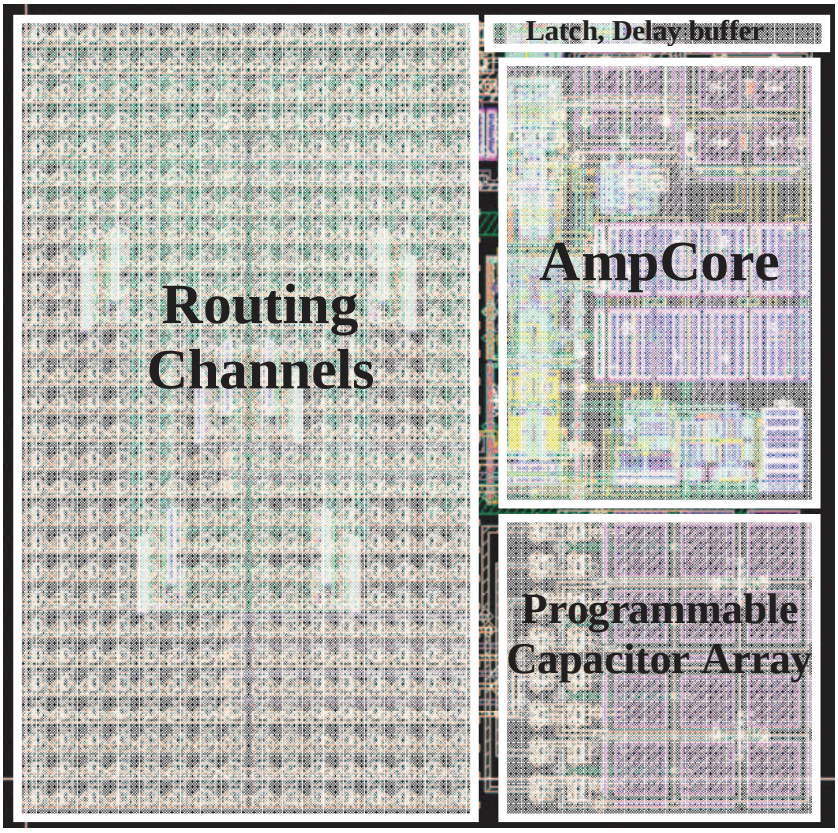


Figure 6.5 Layout of the CAB

6.4 Channel Segmentation

The main design problem in this universal CAB routing architecture is obviously the channel design. We compare three channel segmentation schemes in this study. One is used by almost all existing FPAAs, where the channel consists of long tracks running through the whole chip. We called it single segment scheme. The other scheme is widely used by FPGAs, where the channels consist of unit-length segments (only connecting two neighboring terminals), and connection switches are needed to route longer nets. The third is a staggered non-uniformly segmented

channel generated by the algorithm described in Chapter 2, where the net distribution was obtained empirically.

To save the silicon area, we always want to design a routing channel with the minimum length and width while satisfying the requirements. By limiting the components to one differential OPAMP, one programmable capacitor array and six controllable switches for each input branch, we need a channel length of 23. In a trade off between the flexibility and area cost, a channel width of 8 was chosen. The three channels are shown in Figure 6.6.

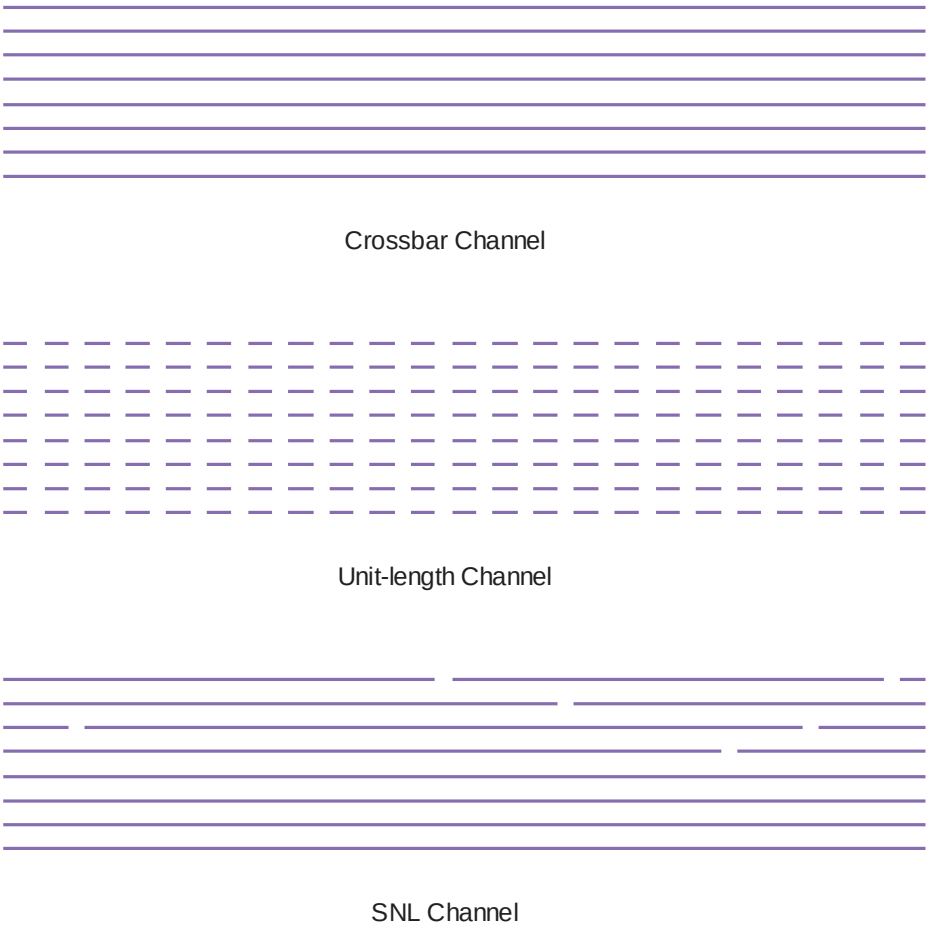


Figure 6.6 Three channel segmentation schemes

6.5 Experimental results

The improvements of SNL channel on routability over crosbar channel have been investigated in Chapter 1 and 2. It's effectiveness in reducing parasitic capacitance and cross-coupling was also studied in Chapter 3 and 4. Here we show the improvements of SNL channel on performance of real circuits in terms of accuracy and speed. The test vehicle is a switched-capacitor circuit block called multiplying DAC (MDAC) shown in Figure 6.7.

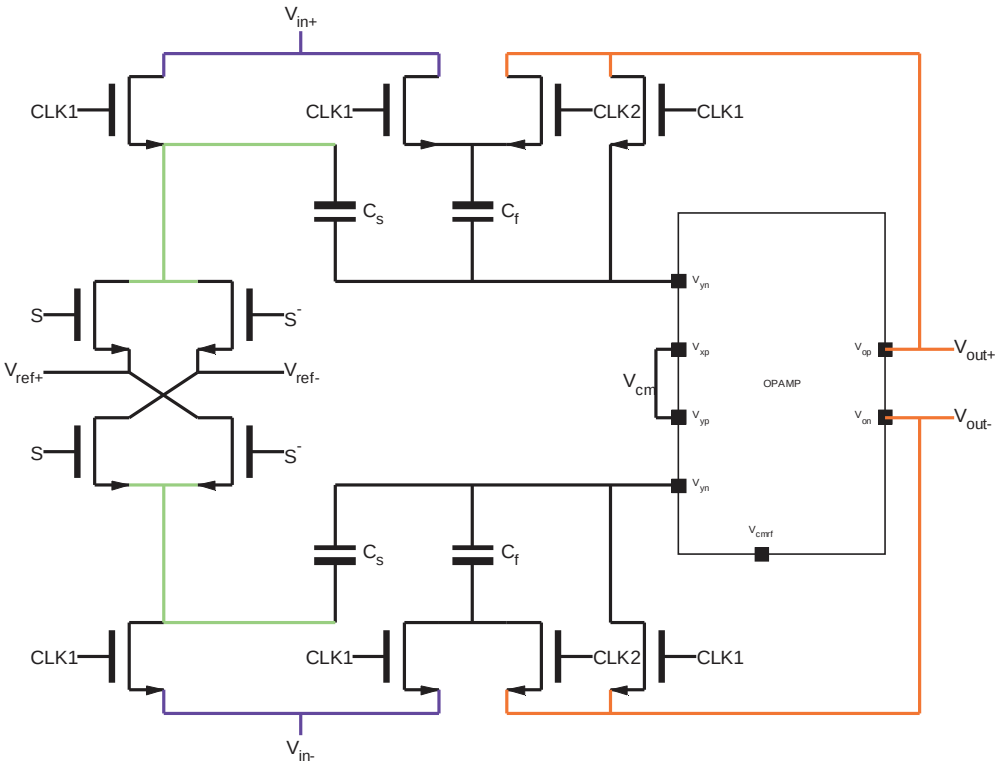


Figure 6.7 Schematic of the MDAC circuit

During phase 1, the bottom plates of both the sample capacitor C_s and the feedback capacitor C_f are connected to the input, while the OPAMP is in a unit-gain setting, which provides input-offset cancellation. In phase 2, the bottom plate of the

C_f is connected to the OPAMP output, while the bottom plate of the C_s is connected the DAC input, V_{ref+} or V_{ref-} determined by S and $\overline{S}$. Ideally the output of the MDAC at the end of CLK2 will be

$$V_{out+} - V_{out-} = \left(1 + \frac{C_s}{C_f}\right)(V_{in+} - V_{in-}) - (S - \overline{S}) \cdot \frac{C_s}{C_f}(V_{ref+} - V_{ref-}) \qquad (6.1)$$

A simulation result is shown in Figure 6.8.

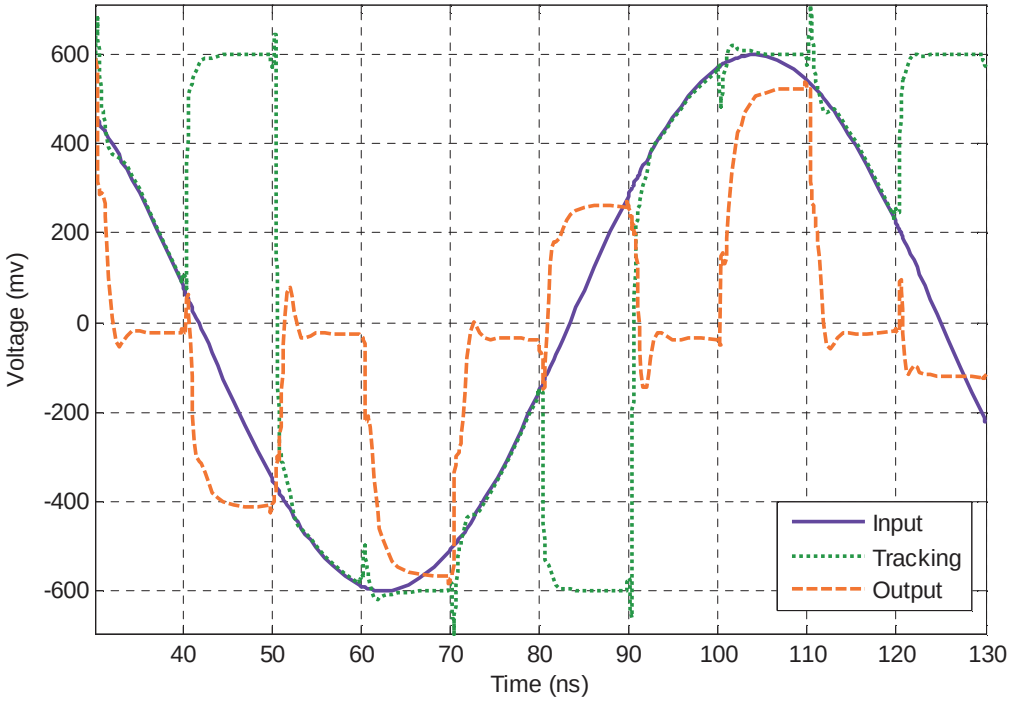
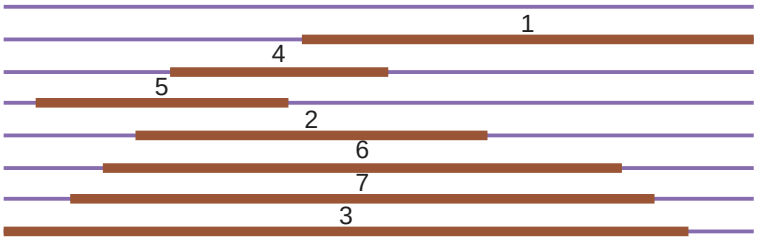
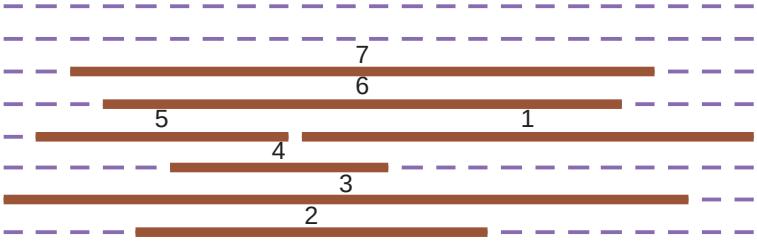


Figure 6.8 MDAC output waveforms

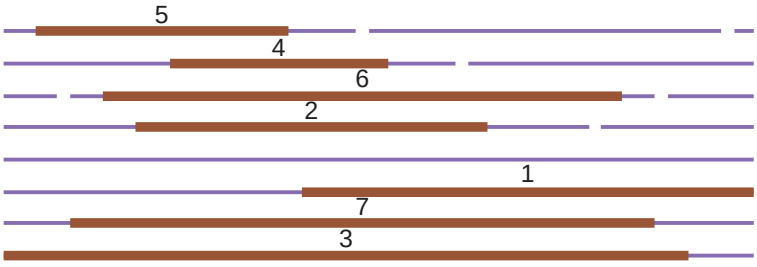
To implement this circuit, two capacitors and six pass-transistors are required for each of the differential input. There are seven nets to be routed for each input branch. The routing results for the three channels described above are shown in Figure 6.9.



Crossbar Channel



Unit-length Channel

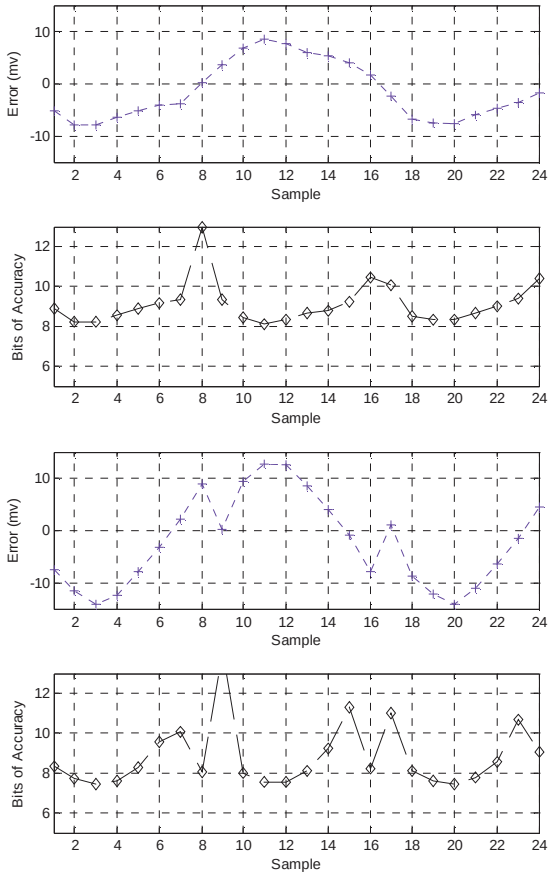


SNL Channel

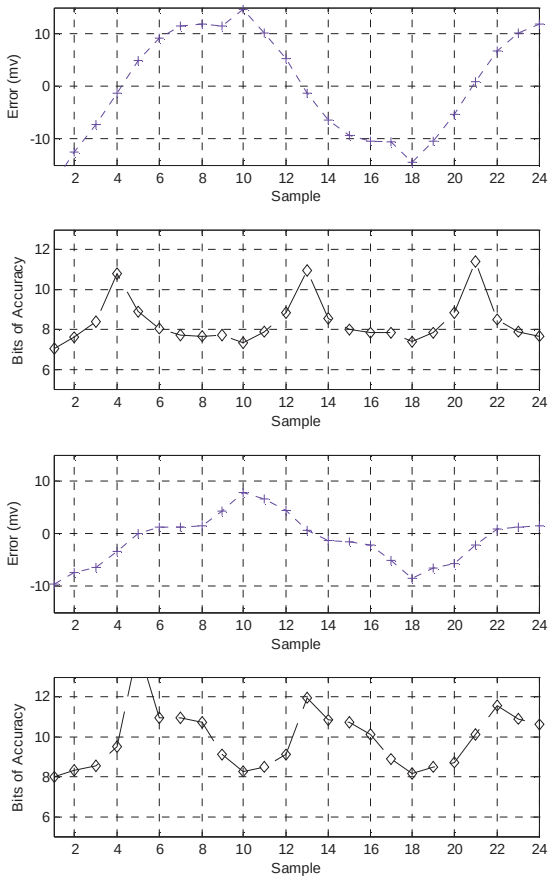
Figure 6.9 Channel routing results of the MDAC circuits

First, we compared the accuracy of the MDAC with clock frequency of 50MHz and a 3MHz sinusoid wave as the input. The output errors and corresponding bits of accuracy of 24 samplings are shown in Figure 6.10. As a reference, the output errors and bits of accuracy with schematic are also given, in which no effects of interconnect parasitics were taken in to account, and therefore stands for the best

performance of any channel segmentation scheme can ever achieve. It is seen that a unit-length channel results in the biggest performance degradation, with an average output error of 9.02mv, corresponding to only 7.36 bits of accuracy, which is expected because the large number of connection switches it requires introduce significant series resistance. The crossbar channel performs no better, with an average output error of 7.65mv and 7.80 bits of accuracy. The SNL channel achieves an average output error of 3.78mv and 8.90 bits of accuracy, which is only 5% degradation to schematic results.



Ideal channel Crossbar Channel



Unit-length Channel SNL Channel

Figure 6.10 MDAC accuracy with various routing channel

With increased input signal frequency, the interconnect parasitics degrade the performance more because of the sampling jitter. The accuracy of the three MDAC implementations with respect to the frequency of the input sinusoid signal is given in Figure 6.11. Once again we see that a unit-length channel yields the worst performance because the larger RC delay worsen the sampling jitter. And as the input frequency increases, the degradation becomes more obvious. SNL channel remains one bit more accurate than the single segment channel for input frequency up to (larger than the Nyquist frequency). Beyond that, their difference diminishes because the sampling jitter dominates the output error.

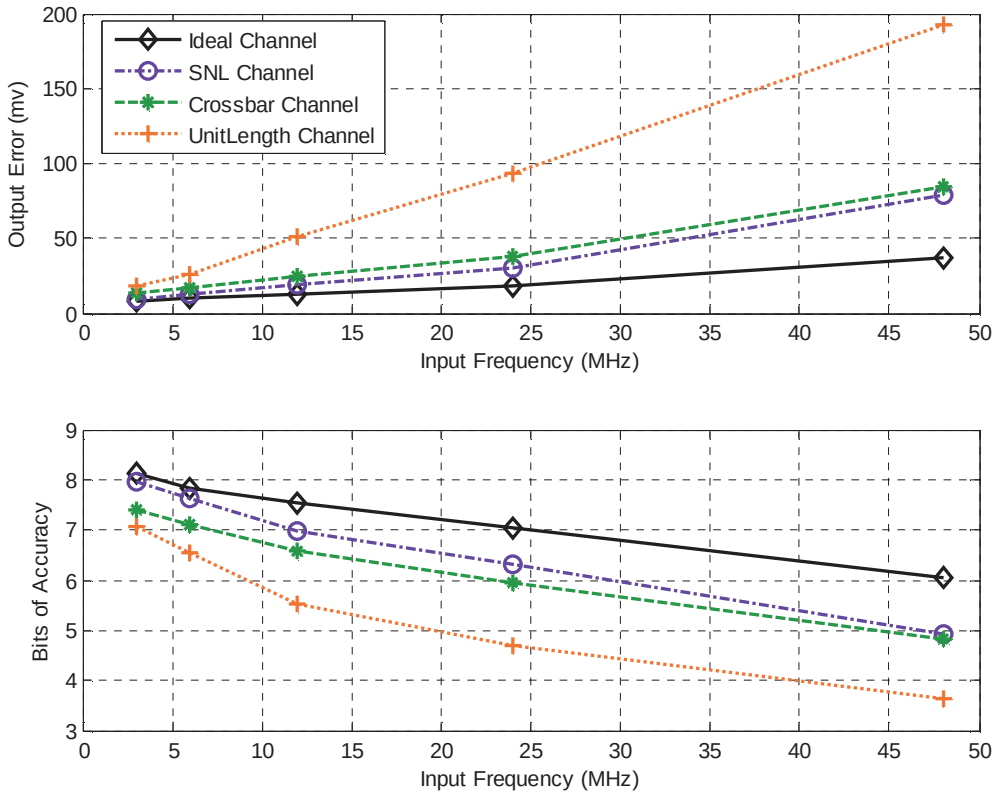


Figure 6.11 MDAC accuracy with respect to input frequency (50MHz Clock)

The speed performance of the three MDAC implementations was found by varying the clock frequency. The results are given in Figure 6.12. It shows that for the whole clock frequency range, the SNL channel remains approximately one bit more accurate than the single-segment channel, which is one bit more accurate than the unit-length channel. Interestingly, the simulation results show that the SNL channel performs even better than pure schematic at high clock frequencies. This phenomenon can be explained as follows: the parasitic capacitance loads the OPAMP and reduces its phase margin, and therefore causes overshooting at the output. When the clock is so fast that the sampling ends before the output reaches its settling value the first time, this overshooting actually helps to improve its accuracy.

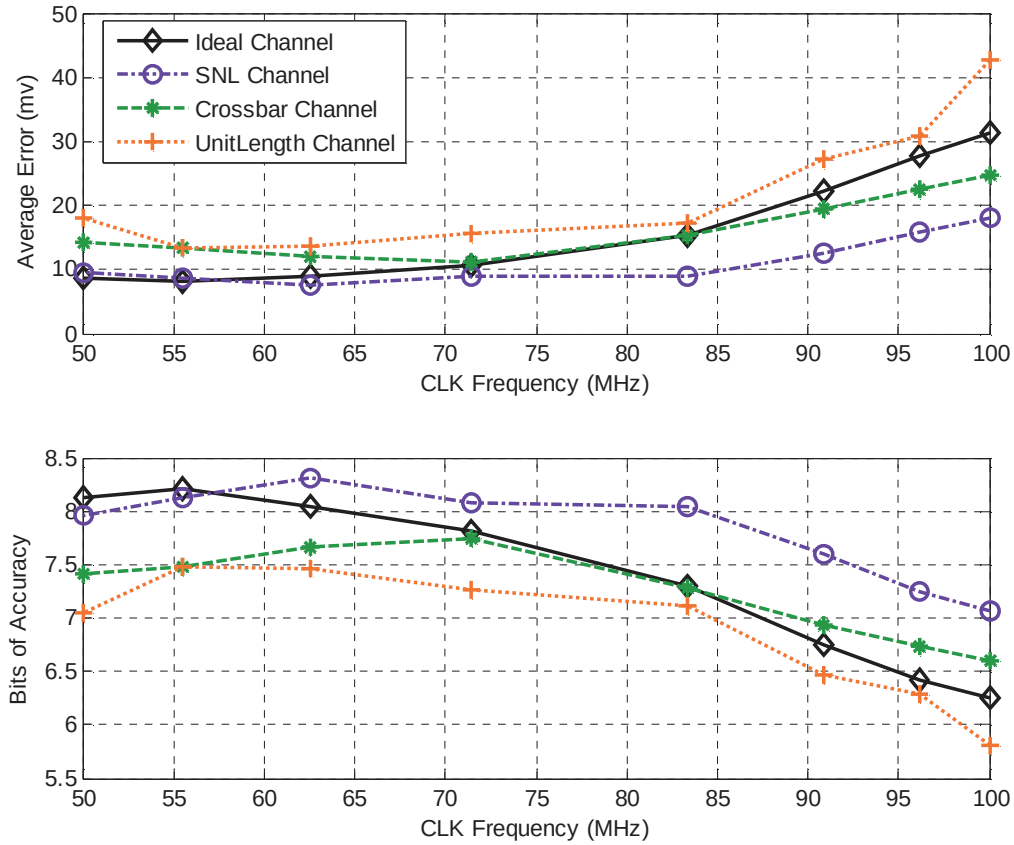


Figure 6.12 MDAC accuracy with respect to clock frequency (3MHz input)

6.6 Conclusions

As one of the essential parts of the high-performance FPAA, a highly flexible CAB designated to implement both continuous-time and discrete-time circuits is proposed. Containing a configurable two-stage differential OPAMP, programmable passive elements and controllable switches, the CAB can implement various high-level analog functions, like sample-and-hold amplifier, comparator, precision amplifier and first-order filters. An internal routing architecture enables arbitrary connections among all its components, while the total number of configuration limited only by the routing resources available. To reduce the effects of interconnect

parasitics, channel segmentation techniques described in previous chapters are applied in designing the routing channel. The results show that a proper channel segmentation scheme is shown the achieve significantly less performance degradation than those commonly used in existing FPAAs or FPGAs, while requiring much less routing resources and processing efforts.

Chapter 7

Hierarchical Implementation of An 8-bit Pipelined A/D Converter

7.1 Introduction

Providing the link between the real analog world and digital signal processing and data storage, analog-to-digital conversion is one of the most widely needed analog functions. High-speed A/D converters find applications in digital oscilloscopes [7.1], disk drive read channel [7.2] and wireless communication systems [7.3], while high-resolution A/D converters enable digital audio [7.4] and video-imaging systems [7.5].

There are different ADC architectures targeting at different applications, each has its own advantages and disadvantages [7.6]. While the low-sampling-rate, high-resolution applications are still the domain of successive approximation register (SAR) and integrating architectures (more recently oversampling/sigma-delta ADCs), and the high-sampling-rate (Giga sample per second or higher) but low resolution are still obtained using flash ADCs and their variants, it is safe to say that pipelined ADCs of various forms have become the most popular ADC architecture for anything between them [7.7], with sampling rates from a few MSPS up to

several hundred MSPS [7.8], and resolutions from 8 bits up to 16 bits [7.9], covering a wide range of applications including CCD imaging, ultrasonic medical imaging, digital communications, digital video, xDSL, cable modem and fast Ethernet.

The rest of this chapter is organized as follows. Section 7.2 provides some preliminaries on pipelined A/D converters, and Section 7.3 describes the detailed hierarchical design flow, including circuit partition, CAB configuration, sub-block placement and routing. Experimental results are presented in Section 7.4, focusing on the impacts of routing parasitics on the static and dynamic performance of the ADC. The conclusions are given in Section 7.5.

7.2 Pipeline A/D Converter

A typical block diagram of a pipeline A/D converter is shown in Figure 7.1. It consists of a cascade of N identical stages. Each stage samples the output of the previous stage and quantizes it coarsely into B+1 bits (effective per-stage resolution is B, and one extra bit is used foe digital correction). The quantized signal is then converted back to analog signal using a DAC and subtracted from the sampled signal. The residue is amplified by the interstage amplifier with a factor of 2^B and fed to the subsequent stage. The same procedure is repeated by each stage down along the pipeline to finish the whole conversion, yielding a total resolution of $N \cdot B$ bits. All the bits corresponding to the same sample are time-aligned with shift registers before being fed to the digital-error-correction logic. Since the S/H function in each stage allows all stages to operate concurrently, the pipelined ADC achieves a throughput of one output per clock cycle, with a latency of N clock cycles.

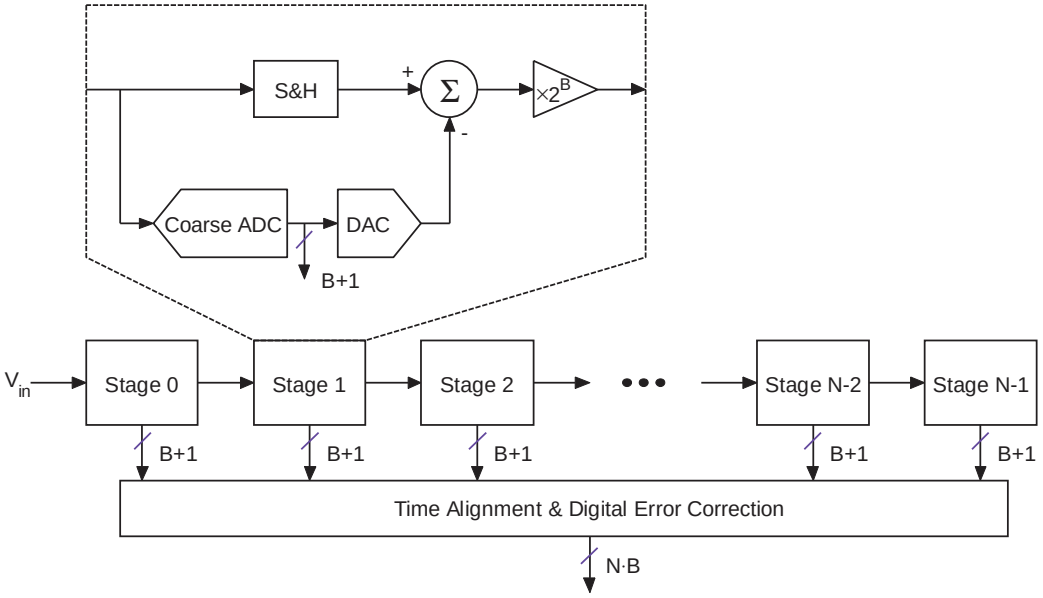
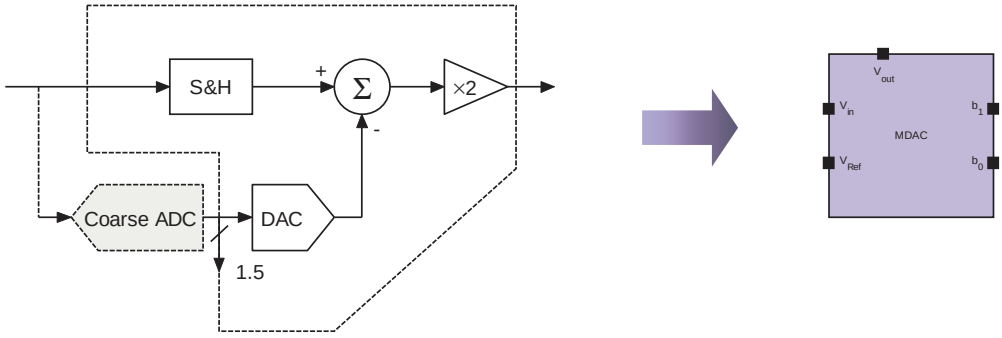


Figure 7.1 Diagram of a pipelined A/D converter

Depending on how many bits each stage resolves and the number of bits in the LSB flash ADC, there can be many variations of pipelined ADC. The partition of bits per stage is determined in part by the targeting speed and resolution. In general, higher-speed CMOS pipelined ADCs tend to favor a lower number of bits per stage because it is difficult to realize wideband amplifier of very high gain. On the contrary, lower-speed CMOS pipelined ADCs tend to favor more bits per stage, which reduce the accuracy requirement of the coarse flash ADC in each stage and results in less latency. To maximize the interstage amplifier bandwidth, the dominant limiting factor of the conversion speed, a resolution of 1.5 bits per stage [7.10] (i.e., 2 decision levels) is chosen in this pipeline implementation, which also allows a correction range for comparator offsets up to $\pm V_{REF}$ /4 and eliminate a dedicated front-end S/H circuit [7.11].

7.3 Implementation of One Stage

The objective of partition is transforming the circuit to be implemented into an interconnection of components of the given target library with a minimum covering cost. In the case of 1.5-bit per stage, the flash ADC section is just two comparator, and the DAC is simply a multiplexing of voltages $+V_{REF}$ and $-V_{REF}$, controlled by the comparator output *S*. In practice, a single switched-capacitor circuit block called multiplying DAC (MDAC), as shown in Figure 7.2, performs the functions of sample-and-hold, subtraction and interstage amplification. The circuit was already presented in Chapter 6 as a test vehicle of the CAB, and its working principles and implementation was described there too.



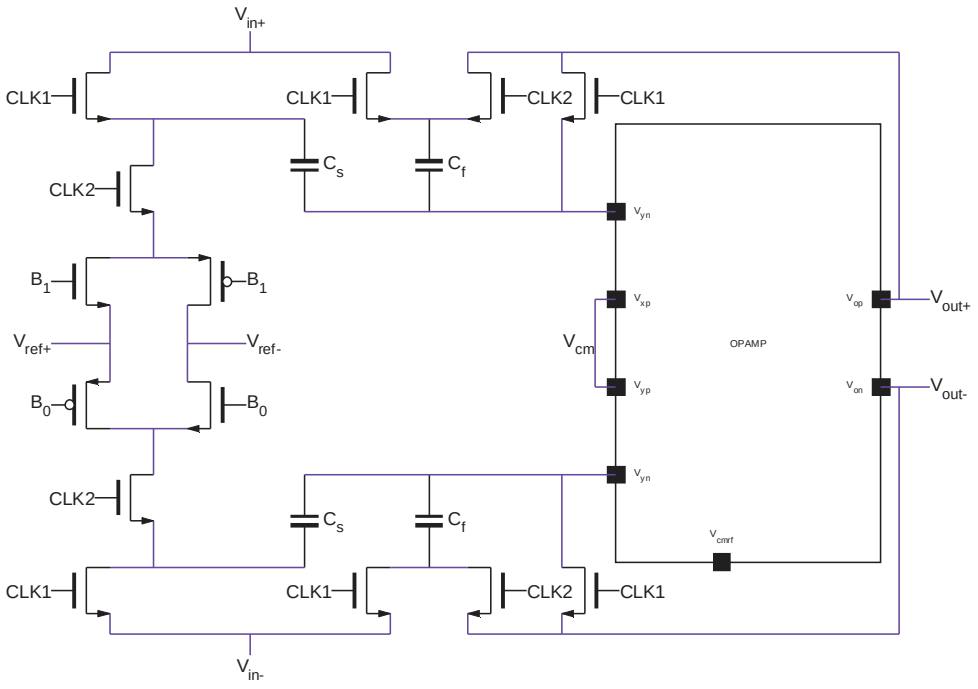


Figure 7.2 Schematic of the MDAC circuit

The coarse ADC in each stage is a flash ADC comprised of two comparators. Figure 7.3 shows the circuit of a high-speed latched comparator with offset cancellation. It consists of a preamplifier followed by a track-and-latch stage. The preamplifier typically has some gain to improve the resolution, but usually no greater than 10 because the time constant would be too large and the speed is limited. It can be simply a unit-gain buffer if very high speed but only moderate resolution is required [7.12]. The preamplifier also prevents kickback, the charge transfer either into or out of the inputs when the track-and-latch stage goes from track mode to latch mode, caused by the charge needed to turn the transistors in the positive-feedback circuitry on and turn the transistors in the tracking circuitry off. Without the preamplifier, this kickback will enter the driving circuitry and causes very larger glitches.

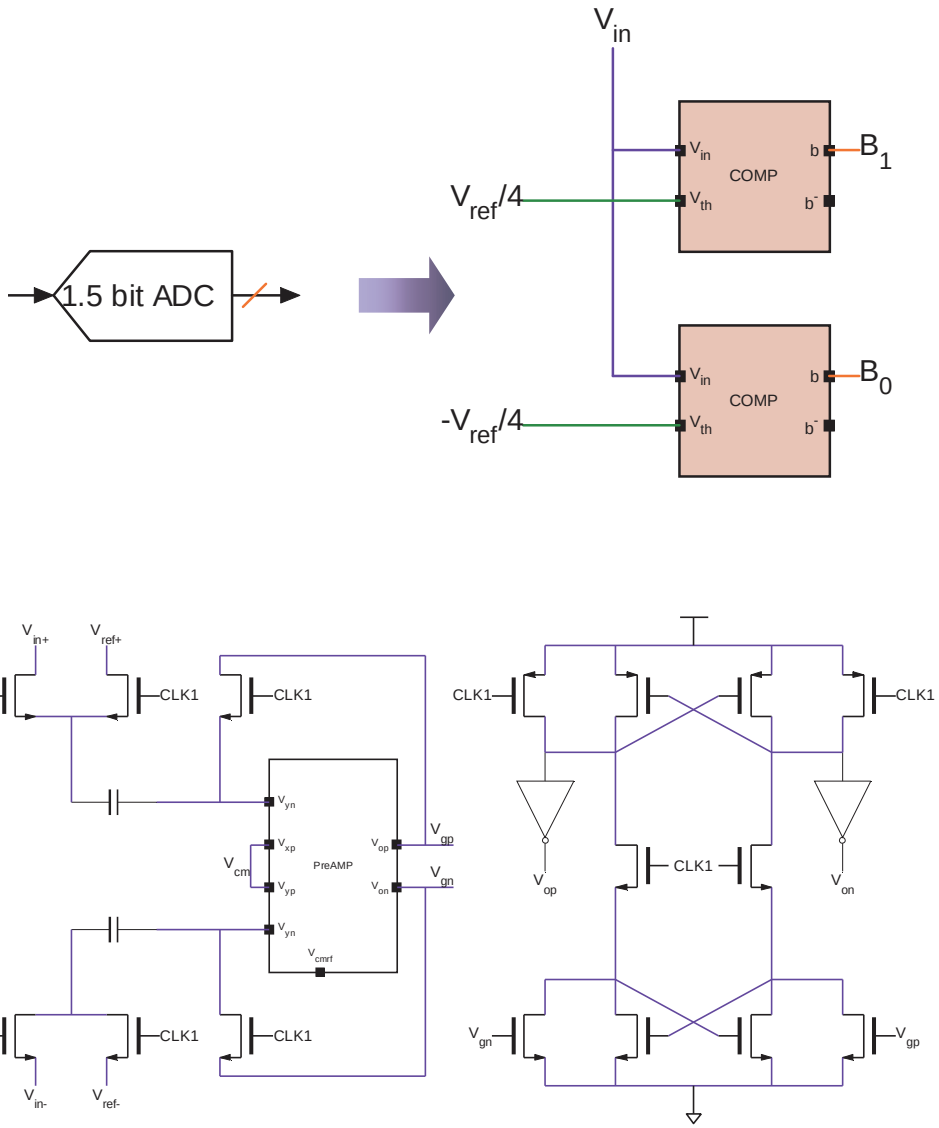


Figure 7.3 Schematic of the comparator

7.4 Placement and Routing

The objective of the placement is assigning each sub-circuit to a CAB in the FPAA, ensuring no overlap, as well as 100% routability and minimum performance degradation. Usually the placement should work closely with the routing. In case the routing results are unsatisfactory, the placement maybe rearranged and rerouted.

Automatic placement algorithm for VLSI synthesis has been studied extensively [7.13] and will be a good topic of our future study, but is beyond the scope of this work. The sub-circuit placement is quite straightforward here and hence done manually. Each stage of the ADC is conveniently placed into three CABs near to each other, and the stages are placed one by one following the signal flowing path. The whole 8-bit ADC is partitioned and placed into a 6×4 array of CABs, as shown in Figure 7.4, where the expected connections are made with dashed lines. For simplicity, differential signals are represented by a single line, and global nets like power supplies and biases are not shown.

The input to the router is a netlist of CABs, which describes how the terminals should be connected. The format of a netlist used in this work is described as follows:

- A terminal is represented as a three-component vector that defines its coordinates in the array: [row, column, pin].
- A net is a connection with multiple terminals. It will be represented as a series of terminals. For example, [1 2 3; 1 3 3; 1 4 3] represents a net that connects three terminals.

There are totally 19 nets in our placement of the circuit to be routed, and the netlist of CABs is given as

```
net 1.   [6 1 1; 3 1 1;2 2 1;5 2 1;6 3 1; 3 3 1;2 4 1;5 4 1] % Vref/4
net 2.   [5 1 1; 2 1 1;1 2 1;4 2 1;5 3 1; 2 3 1;2 4 1;4 4 1] % -
         Vref/4
net 3.   [4 1 1;1 1 1;3 2 1;6 2 1;4 3 1;1 3 1;3 4 1;6 4 1] % Vref
net 4.   [6 1 2;5 1 2;4 1 2]  % input
net 5.   [6 1 3;4 1 4] [5 1 3;4 1 3]  %comparator output
net 6.   [4 1 5;3 1 2;2 1 2;1 1 2]  % stage2 input
```

```
net 7.   [3 1 3;1 1 4] [2 1 3;1 1 3]  %comparator output
net 8.   [1 1 5;1 2 2;2 2 2;3 2 2]  % stage3 input
net 9.   [1 2 3;3 2 3] [2 2 3;3 2 4] %comparator output
net 10.  [3 2 5;4 2 2;5 2 2;6 2 2]  % stage4 input
net 11.  [4 2 3;6 2 3] [5 2 3;6 2 4] %comparator output
net 12.  [6 2 5;4 3 2;5 3 2;6 3 2] % stage5 input
net 13.  [6 3 3;4 3 4] [5 3 3;4 3 3] %comparator output
net 14.  [4 3 5;3 3 2;2 3 2;1 3 2] % stage6 input
net 15.  [3 3 3;1 3 4] [2 3 3;1 3 3] %comparator output
net 16.  [1 3 5;1 4 2;2 4 2;3 4 2] % stage7 input
net 17.  [1 4 3;3 4 3] [2 4 3;3 4 4] %comparator output
net 18.  [3 4 5;4 4 2;5 4 2;6 4 2] % stage8 input
net 19.  [4 4 3;6 4 3] [5 4 3;6 4 4]} %comparator output;
```

The routing result with a crossbar routing channel is shown in Figure 7.5, where the routing segments run through the whole length and width of the array. The used routing segments are plotted as solid lines, while the unoccupied resources are plotted as dotted lines. A solid circle at the intersection of two tracks means they are connected by the router. It shows a minimum of nine tracks per channel for this architecture are required to complete the routing. Most of the time a short connection is forced to use a long line, which not only results in ineffective resource utilization but also deteriorates the performance, because the unused portion of the segment presents as a loading capacitance and increases cross-coupling.

The routing result with a segmented channel is shown in Figure 7.6. It is seen that without increasing the number of switches, the routing can be completed with only six tracks, while the wire wastage and the chance of cross-coupling are also greatly reduced.

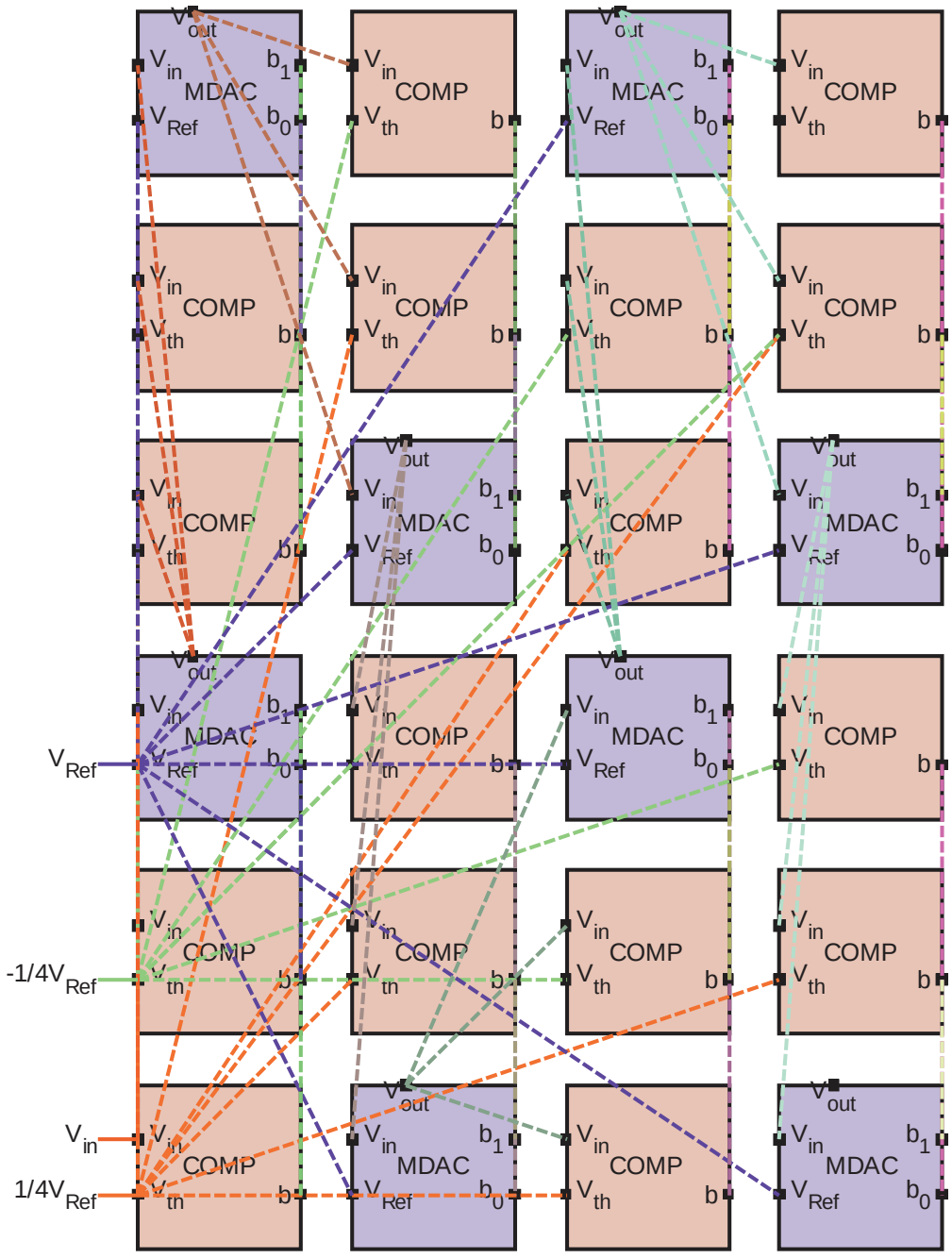


Figure 7.4 Placement of the 8-bit A/D converter

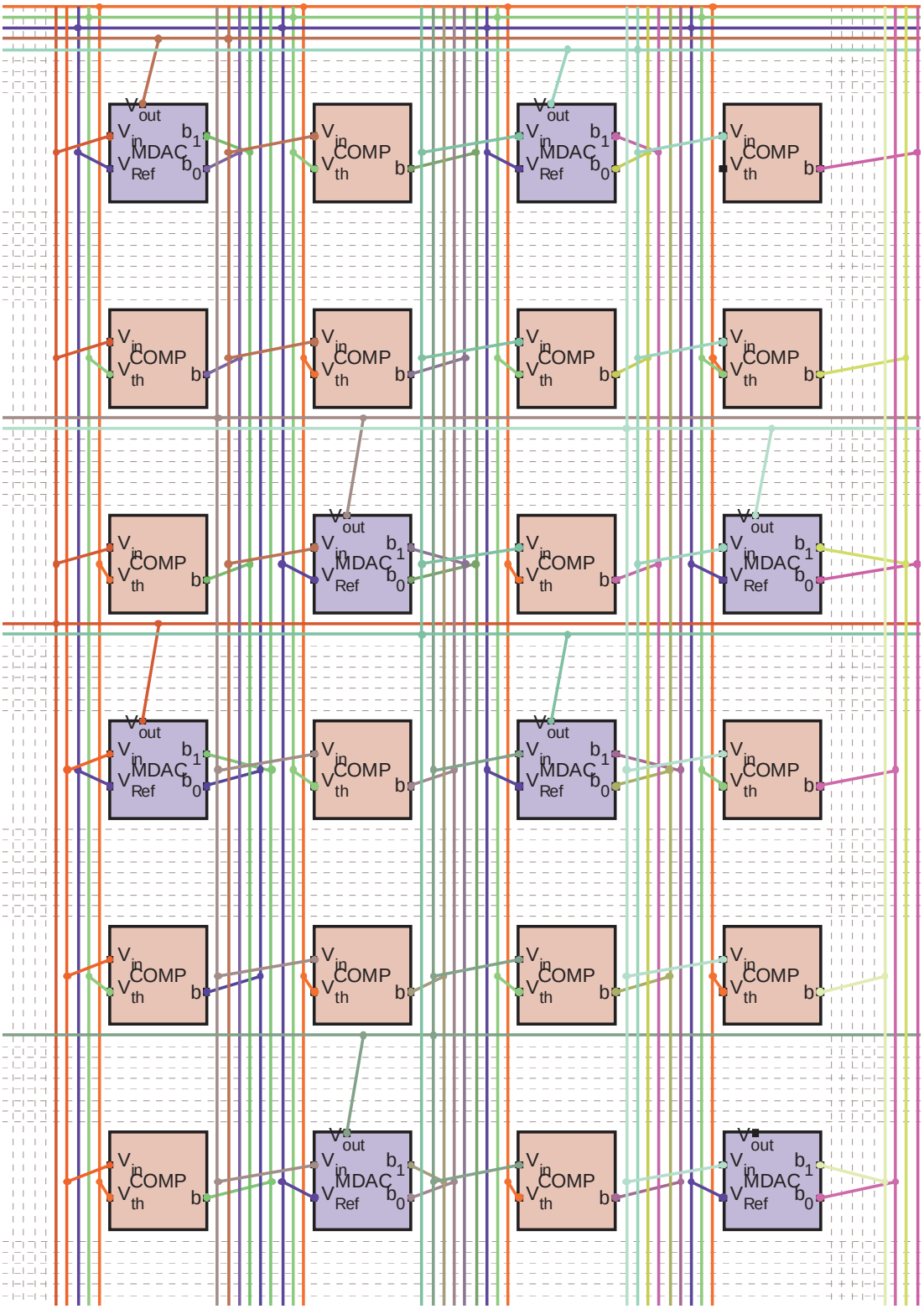


Figure 7.5 Routing results with crossbar channels

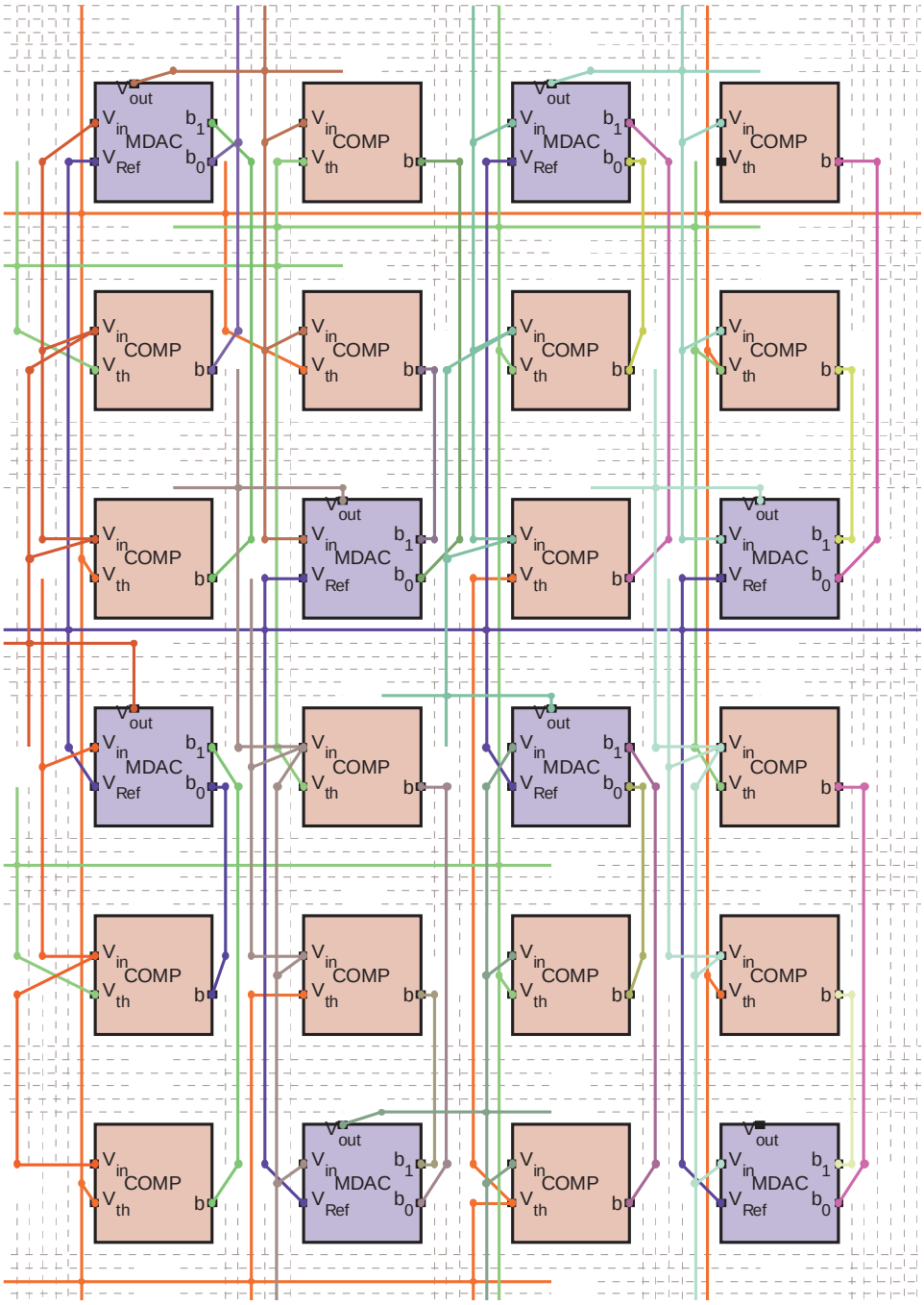


Figure 7.6 Routing results with segmented channels

7.5 Experimental results

To investigate the performance penalty caused by the routing interconnections, the whole array was laid out and the interconnect parasitics were extracted, while a programmed switch is simply modeled as a single via between metals. In our experiments, three ADCs are implemented, using the crossbar, unit-length and a segmented routing channel, respectively. Figure 7.7 shows the layout of the ADC chip.

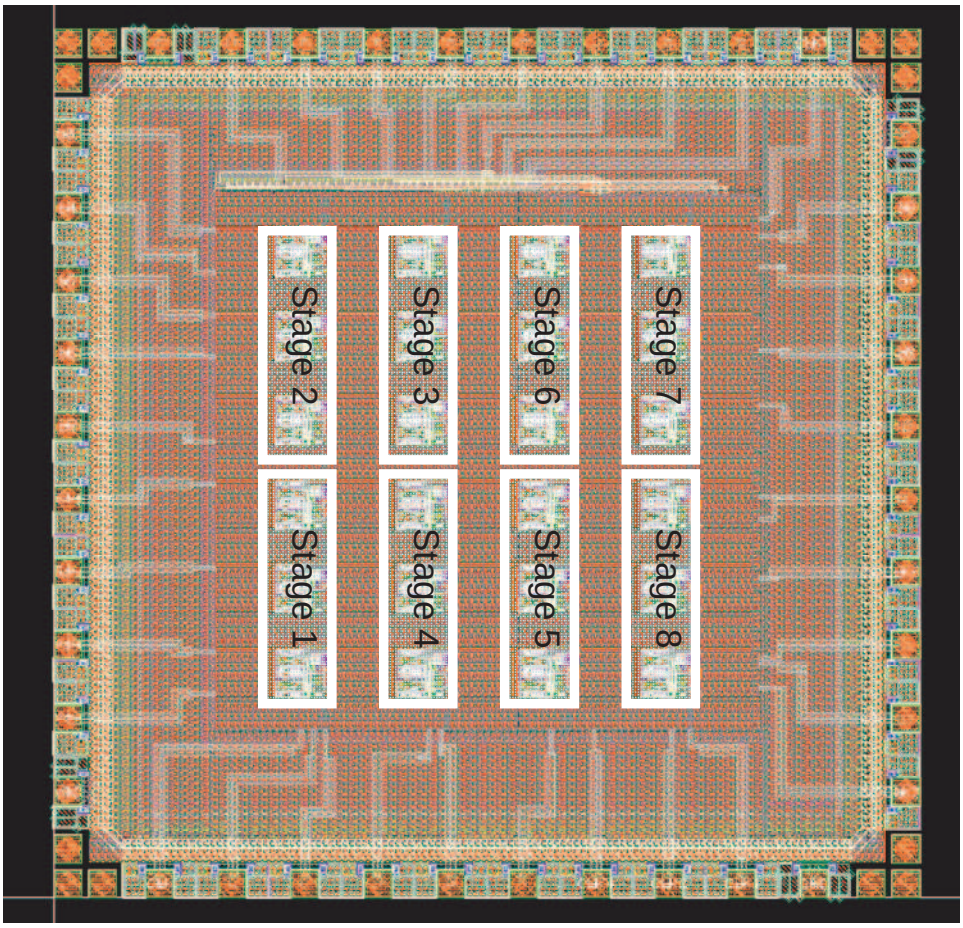


Figure 7.7 Pipelined ADC: Chip Layout

7.5.1 ADC static accuracy specifications

The absolute accuracy of an A/D converter includes the offset, gain and linearity errors. The relative accuracy is the accuracy after the offset and gain errors have been removed. It is also referred to as the *integral non-linearity* (INL), which is defined to be the deviation of the output signal from a straight line drawn through zero and full scale. A conservative measure of non-linearity is to use the endpoints of the converter's transfer response to define the straight line, while an alternative definition is to find the best-fit line such that the maximum difference is minimized. The best straight-line approach is generally preferred, because it produces better results. The INL specification is measured after both static offset and gain errors have been nullified, and can be described as follows:

$$INL = \left(V_D - V_{zero}\right)/V_{LSB} - D, \tag{7.1}$$

where $0 < D < 2^N$-1 is the digital output code, N is the ADC's resolution, V_D is the analog value represented by D, V_{zero} is the minimum analog input corresponding to an all-zero output code, and V_{LSB} is the ideal spacing for two adjacent output codes. If the maximum INL error is less than 0.5 LSB, the A/D converter is guaranteed not to have any missing codes [7.6].

Another term, *differential non-linearity* (DNL) is defined as the variation in analog step sizes away from 1 LSB. DNL is specified after the static gain error has been removed. It is defined as follows:

$$DNL = \left(V_{D+1} - V_D\right)/V_{LSB} - 1, \tag{7.2}$$

where $0 < D < 2^N$-2. Similarly, an A/D converter is guaranteed not to have any missing codes if its maximum DNL error is less than 1 LSB. In Figure 7.8 the transfer curve of an 8-bit A/D converter is shown. The drawn line shows the ideal transfer characteristic, while a dashed line indicates the measured transfer curve of a practical converter. LNL and DNL are shown partly as a function of the LSB error between the drawn line and the dashed line.

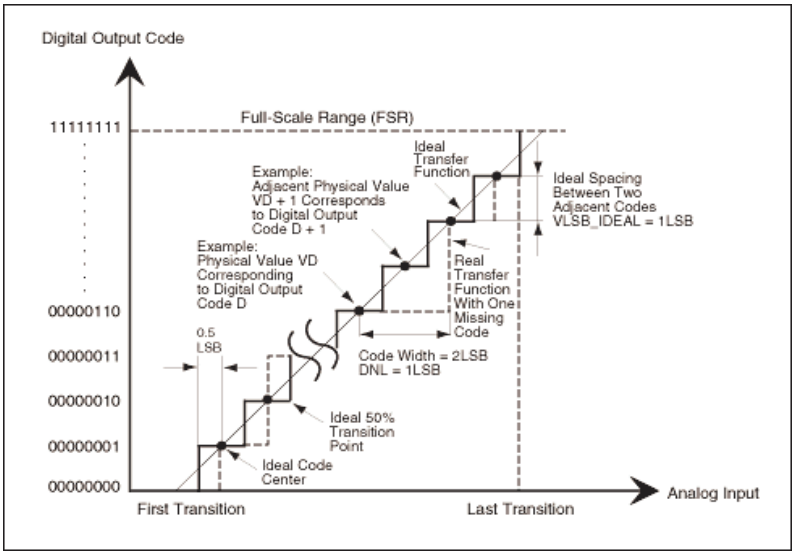


Figure 7.8 Transfer curve of an 8-bit A/D converter

In the applications where it is necessary to distinguish the slight difference between adjacent values like color densities in imaging processing, DNL is the important measurement of linearity. However, in an application in which widely varying parameters like speed must be continuously monitored, INL is usually more important. INL and DNL can be measured with either a quasi-DC voltage ramp or a low-frequency sine wave as the input [7.14]. A simple DC (ramp) test can

incorporate a logic analyzer, a high-accuracy DAC (optional), and a high-precision DC source for sweeping the input range of the device under test (DUT).

There are many factors in defining INL and DNL of an pipelined ADC. The following are the dominant ones:

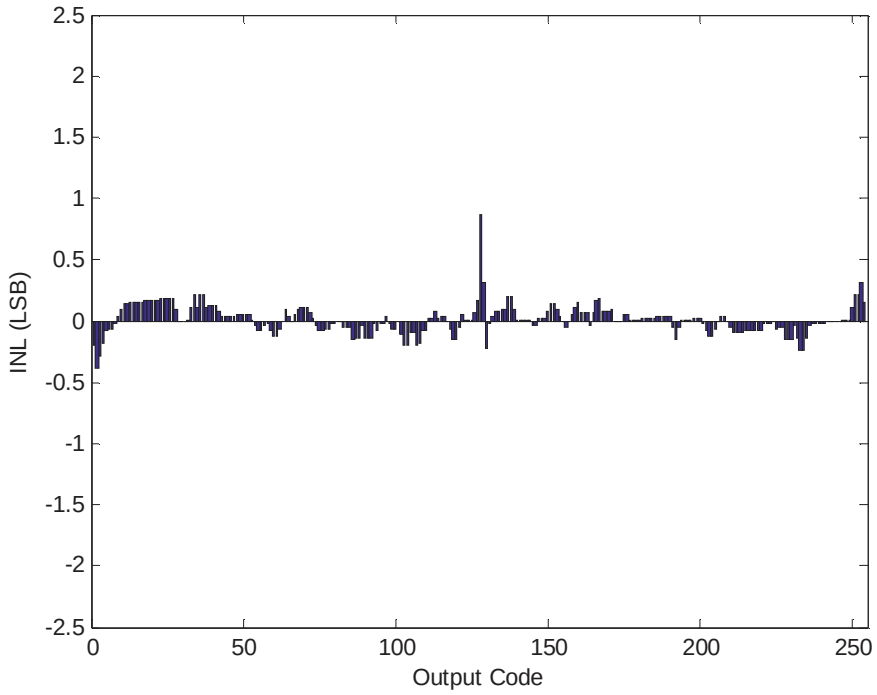
- Reference voltage variations: As the ADC output is the ratio between the analog signal voltage and the reference voltage, any variation or noise on the analog reference will cause a change in the converted digital value. As the comparators are placed all around the FPAA, it is essential to make sure the IR drops caused by the routing wires are low enough, or at least the reference voltage at all blocks are affected in the same direction. It's also important to keep the wire track used to route reference voltage far away from the quasi-digital nets like the comparator outputs to avoid excessive switching noises.
- Non-ideal OPAMP: Limited gain, CMRR and PSRR of the OPAMP cause conversion errors at each stage of the pipelined ADC, and therefore need to be optimized. OPAMP non-linearity generates harmonics which directly add to the non-linearity of the ADC. In the aspect of routing, unbalanced input/output loading deteriorates it and therefore need to be avoided as long as possible.
- Routing parasitic resistance and capacitance: The impedance of the analog signal source or series resistance between source and pin will cause a voltage drop across it because of current flowing into the pin. When there is parasitic resistance in series, the sampling time for each stage will be changed, which is not uniform across the chip and contribute to non-linearity too. The parasitic capacitance will not allow the sampling capacitor to be charged to exactly to the

input. If the analog input signal varies, the sampling errors at different stages are non-uniform either and increase the DNL and INL of the ADC. Therefore, a router minimizes the routing parasitics naturally gain better INL and DNL performance.

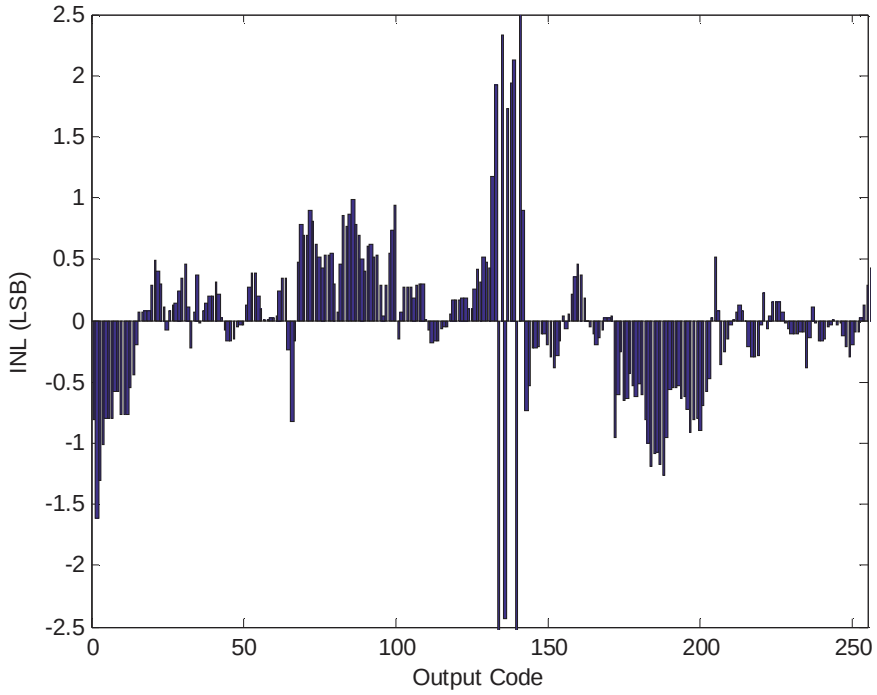
- Signal cross-talk: cross-talk between nets cause changes in the input level and timing in an obviously no-linear way. A router with cross-coupling noise deduction will possibly improve the linearity of the ADC as well.

For comparison, the simulation results of INL of the 8-bit ADCs implemented using those three routing channels are plotted in Figure 7.9, at a sampling rate of 50MSPS. Also plotted is the results of an ideal routing channels (the interconnect parasitics are not extracted in the simulation). It is seen that while the ideal routing channel will yield a maximum INL less than 0.5 LSB, which guarantees the monotonicity, the interconnect parasitics in other three channels results in larger INL and therefore worse linearity. Among them, the conventional crossbar routing channel results in a maximum INL about 2.5 LSB, which may exclude it from many high-linearity applications. However, a segmented channel can reduce the maximum INL to about 0.8 LSB, meeting specification for most applications on an 8-bit ADC.

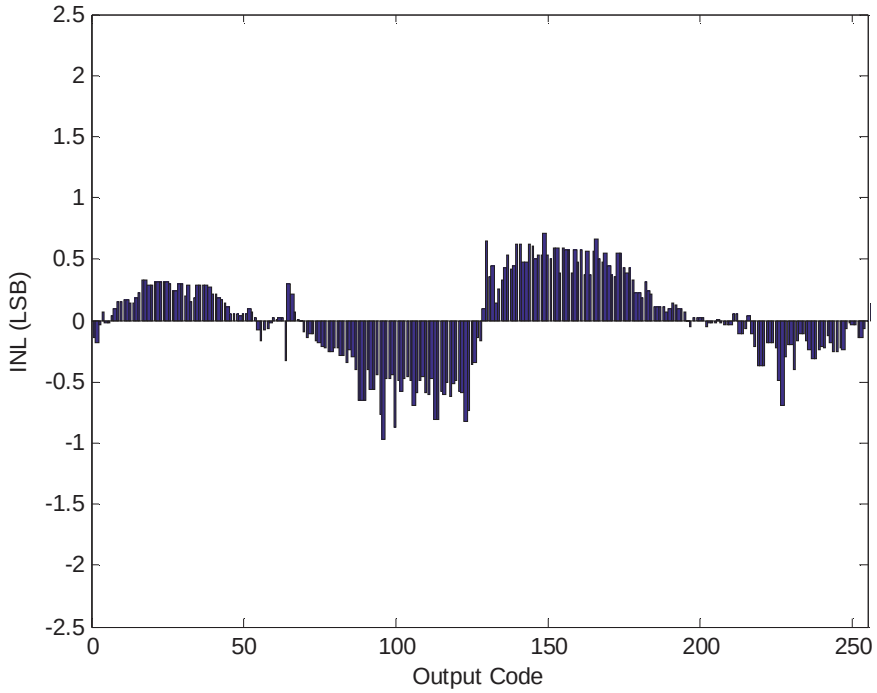
The DNL of the ADCs implemented with different routing channels are plotted in Figure 7.10, which also demonstrates the effectiveness of the segmented channel in reducing parasitics over the crossbar channel. It is seen that a segmented channel not lonely reduces the maximum value of the DNL but also reduces the frequency when DNL is greater than 0.5 LSB.



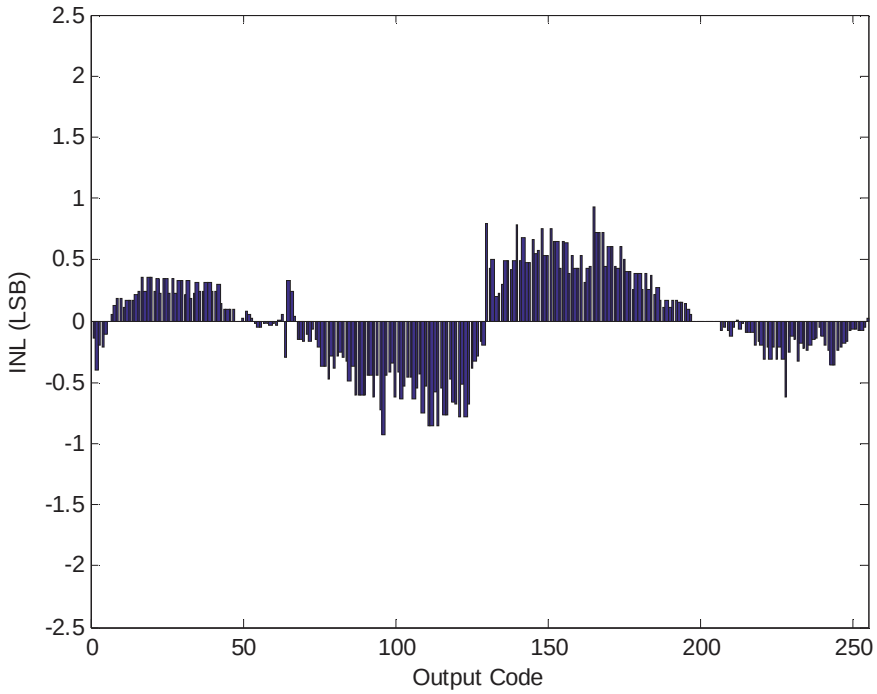
Ideal Channel



Crossbar Channel

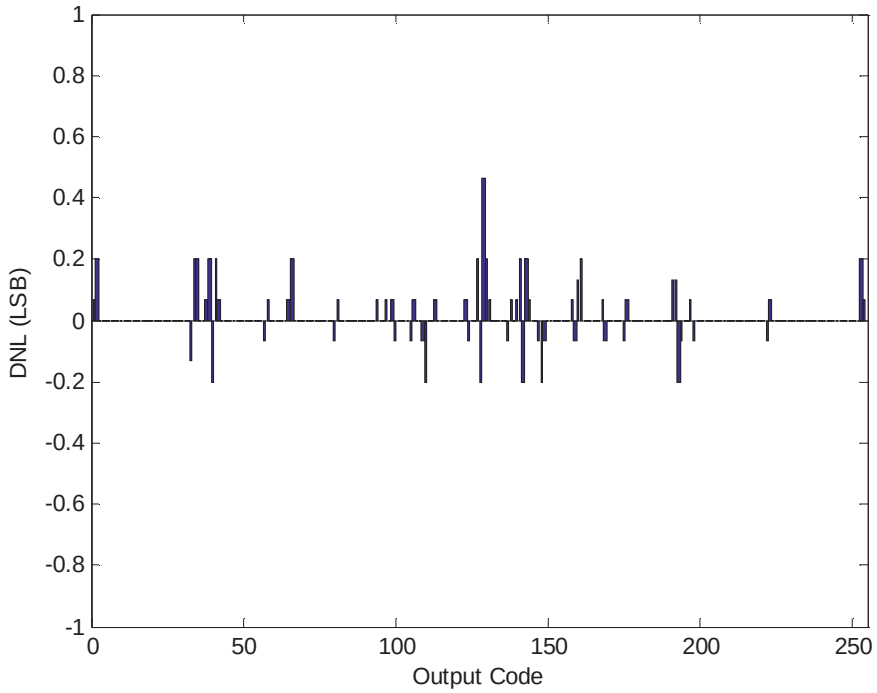


Unit-length Channel

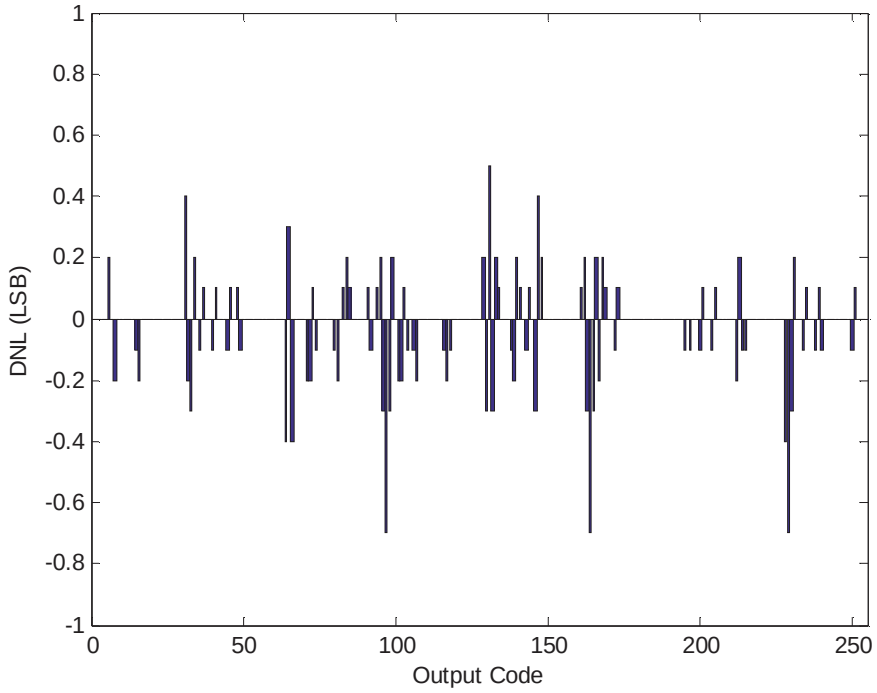


Segmented Channel

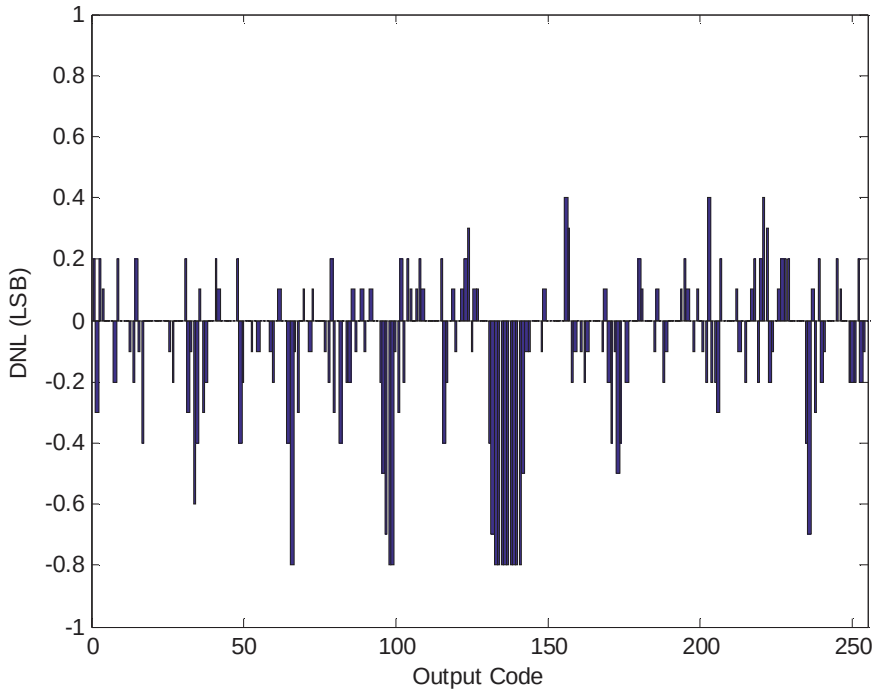
Figure 7.9 INL error plot of three implementations at 50MSPS



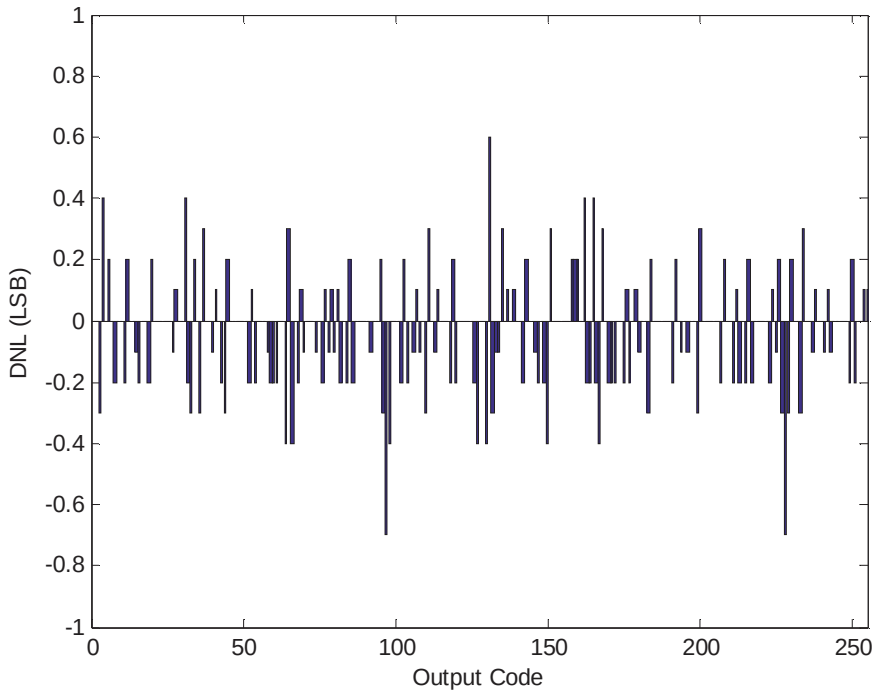
Ideal Channel



Segmented Channel



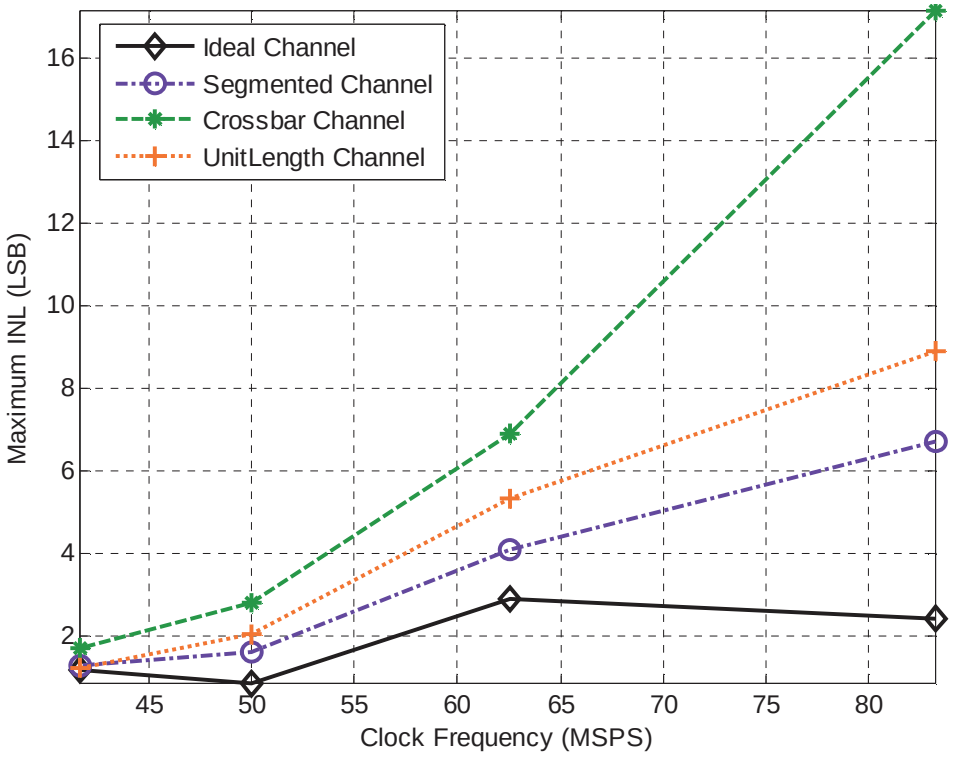
Crossbar Channel



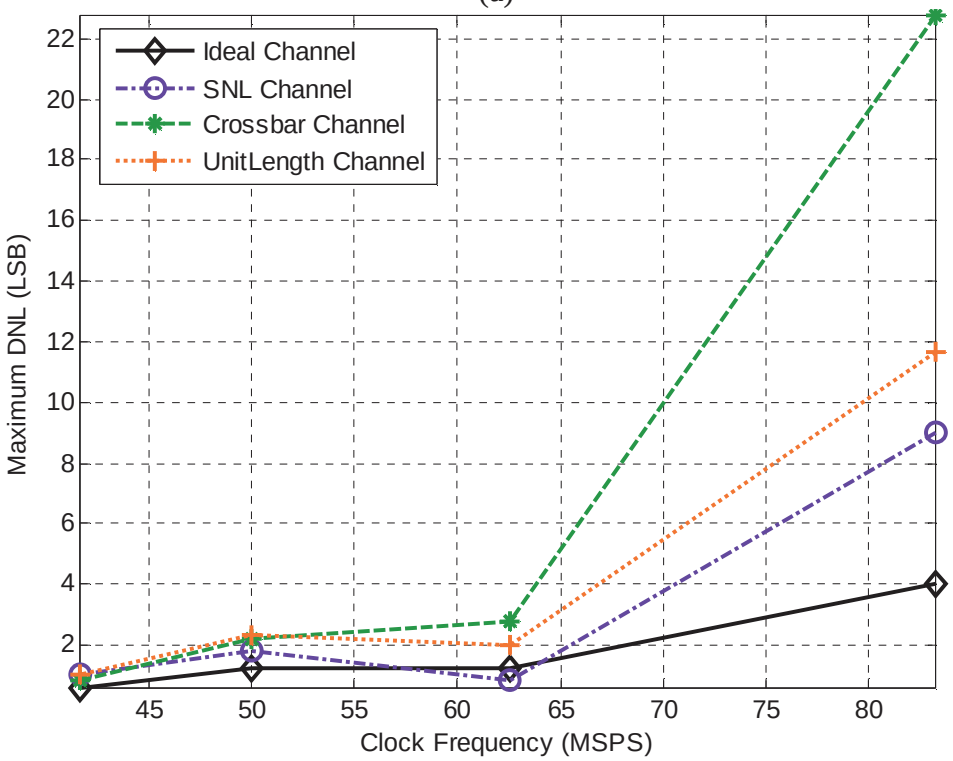
Unit-length Channel

Figure 7.10 DNL error plot of three implementations at 50MSPS

The above plots may indicate that the unit-length routing channel is comparable to a segmented channel in accuracy performance. However, since a unit-length channel requires much more connection switches in the signal path, its processing cost is much higher while the system reliability is significantly lower because each switch setting does come with a failure rate, though extremely small. Moreover, though the series resistance of the connection switches implemented by the Makelink^{TM} technology is very low, and therefore doesn't cause significant degradation at low frequency, its impact on the system performance will be revealed at high frequencies. The INL and DNL performance of the three channels with respect to the sampling rate is plotted in Figure 7.11. It shows that as the clock frequency increases, the improvement of segmented channel over the unit-length channel become more obvious. When the clock frequency is quite high, the DNL performance of the unit-length channel is even worse than that of the crossbar channel.



(a)



(b)

Figure 7.11 (a) INL and (b) DNL versus sampling rate

7.5.2 ADC dynamic accuracy specifications

While the DC-specifications are for the static linearity, dynamic specifications of A/D converters give a better insight into the applicability of a converter in a high frequency system, where linearity and spectral purity are essential. Popular specifications for quantifying ADC dynamic performance are signal-to-noise ratio (SNR), total harmonic distortion (THD) and spurious free dynamic range (SFDR).

The quantization process introduces an irreversible error, which sets the limit for the dynamic range of an A/D converter. Assuming that the quantization error of an ADC is evenly distributed, the SNR for a single-tone sinusoidal signal can be obtained to be

$$SNR = (6.02N + 1.76)dB \tag{7.3}$$

Any nonlinearity in an A/D converter creates harmonic distortion. In differential implementations, the even order distortion components are ideally canceled. However, the cancellation is not perfect if any mismatch or asymmetry is present. The THD describes the degradation of the signal-to-distortion ratio caused by the harmonic distortion. By definition, it can be expressed as an absolute value with

$$THD = \frac{\sqrt{\sum_{j=2}^{N_d} P(j \cdot f_{in})}}{P(f_{in})} \tag{7.4}$$

Where N_d is the number of harmonics to be considered, and $P(f)$ is the power spectrum density of the ADC output at frequency f. When large oversampling ratios are used and the spectral purity of the ADC is important, like wireless

telecommunication applications, a proper specification is the ratio between the powers of the signal component and the largest spurious component within a certain frequency band. The SFDR is usually expressed in dBc as

$$SFDR = 10 \cdot \log \frac{P(f_{in})}{P(f_{spur})} \tag{7.5}$$

For an exact SFDR definition, the power level of the fundamental signal relative to the full-scale must also be given. Normally the limiting factor of the SFDR in ADCs is harmonic distortion. In most situations, the SFDR should be larger than the signal-to-noise ratio of the converter [7.15].

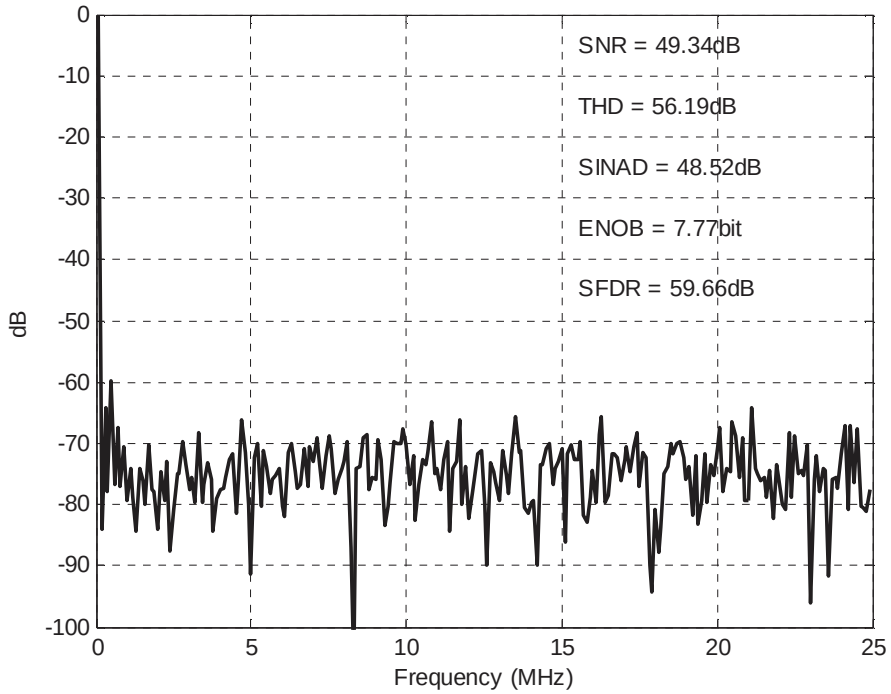
A more realistic figure of merit for an ADC is the signal-to-noise and distortion ratio (SINAD), which is the ratio of the signal energy to the total error energy including all spurs and harmonics. SINAD is determined by employing the sine-fit test, in which a sinusoidal signal is fitted to a measured data and the errors between the ideal and real signal are integrated to get the total power of noise and distortion [7.16]. Effective number of bits (ENOB) is defined as

$$ENOB = \frac{SINAD - 1.76dB}{6.02dB} \tag{7.6}$$

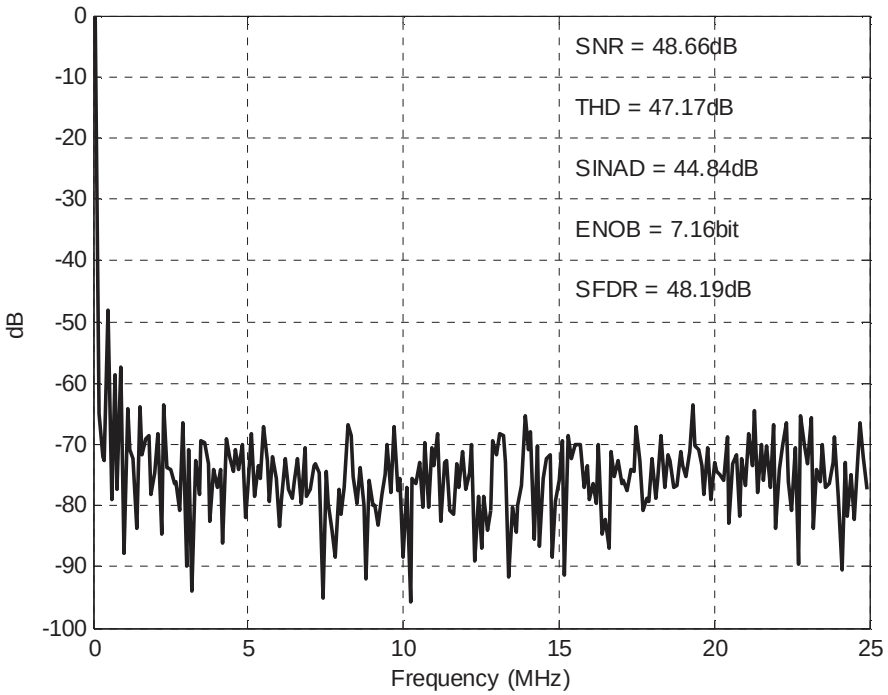
And the effective resolution bandwidth (ERB) is defined as the maximum analog frequency for which the ENOB is 1/2 LSB lower with respect to the theoretical value.

The spectral outputs of the four ADCs with 100 KHz sinusoidal input and 50 MSPS clock is plotted in Unit-length Channel

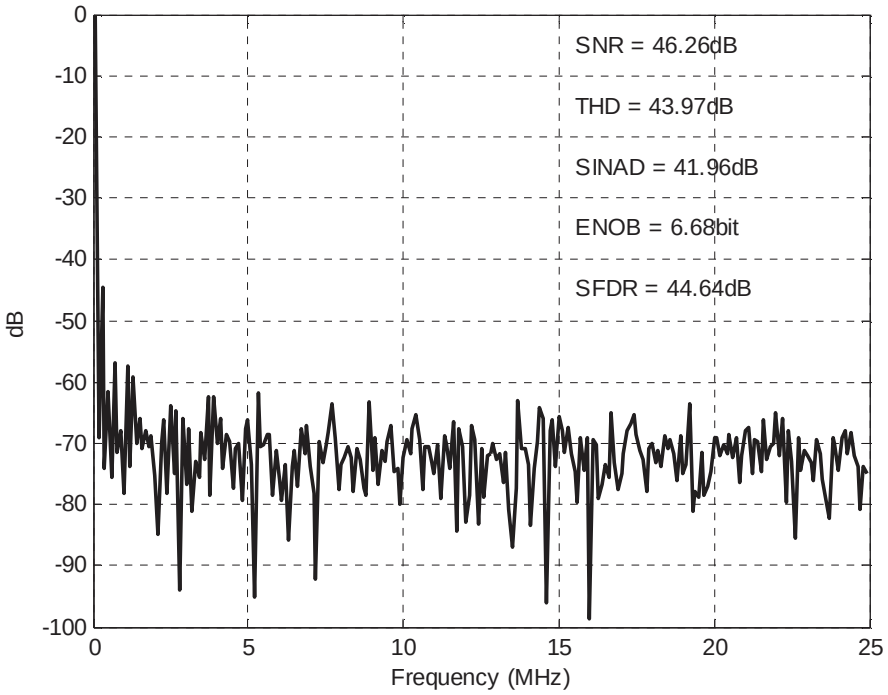
Figure 7.12. It shows that an ideal routing channel achieves an ENOB of 7.77 bits, but the parasitics associated with non-ideal routing channel reduces it from that value. Compared to a crossbar channel, the segmented channel improves the SNR by 2.4 dB, while reducing the THD by 3.2 dB, and therefore achieves a 0.5 bit higher ENOB. The segmented channel also achieves a 3.55 dB higher SFDR.



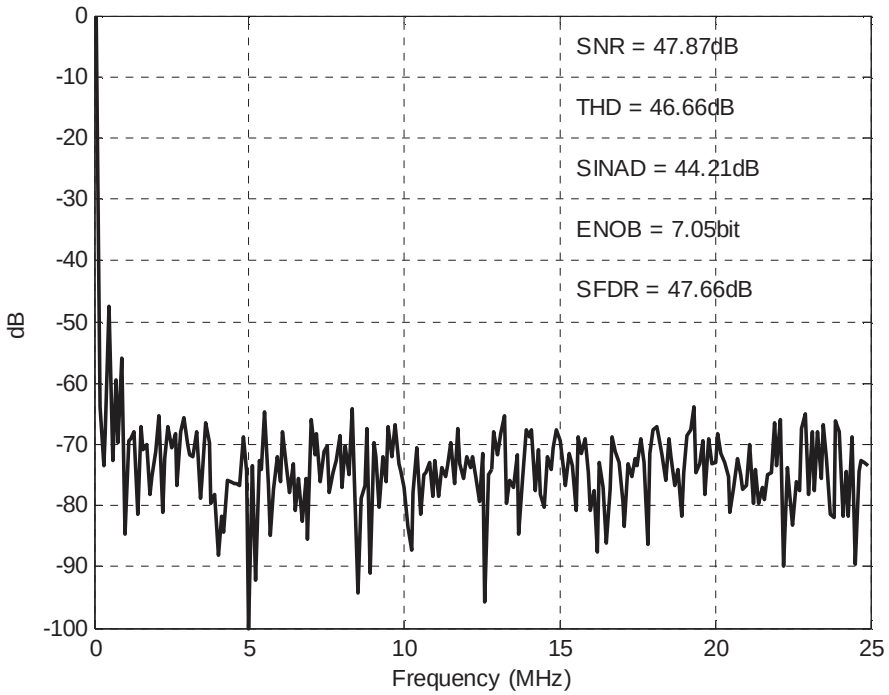
Ideal Channel



Segmented Channel



Crossbar Channel



Unit-length Channel

Figure 7.12 FFT plot of the ADC output (Fin = 100 KHz, Fs = 50MSPS)

The ENOB of the ADCs versus input frequency (50 MSPS) is plotted in Figure 7.13. While he plot shows that the ac performance of all ADCs degrades due to sampling uncertainties caused by high-frequency distortion, the ADC implemented using the segmented channel maintains a higher ENOB over ADCs implemented with other two channels over a wide range of input signal bandwidth. It also has the widest ERB of 11MHz. At very high input frequency, the interconnect parasitics on longer dominates the noise and distortion, and the ENOB of all ADCs deteriorates rapidly and falls to about the same level without regards to which routing channel they are implemented with.

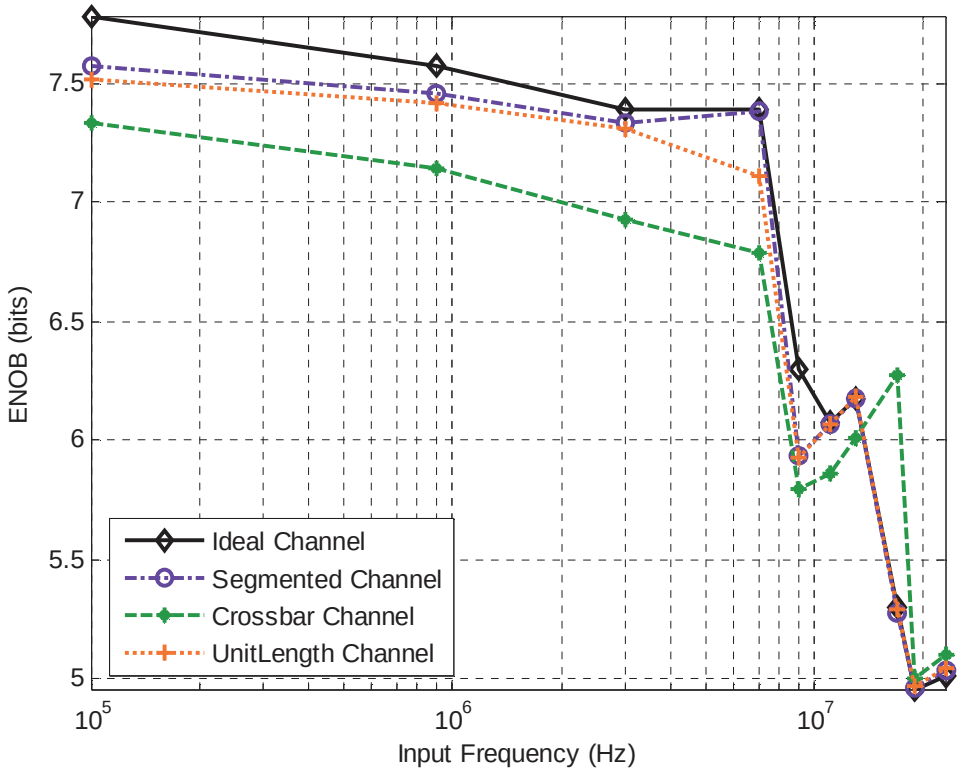


Figure 7.13 ENOB versus input frequency (50 MSPS)

The ENOB of the ADCs versus clock frequency (with 3MHz input frequency) is plotted in Figure 7.14. Unlike the previous plot, it shows that the impact of interconnect parasitics become more obvious as the clock frequency increases, since the loading capacitance and series resistance directly affect the settling time of the OPAMP. Since the crossbar channel introduces the largest interconnect capacitance, the ADC implemented with it has the worst performance. At low frequencies, the ADC implemented with the unit-length channel performs about the same as that implemented with the segmented channel, but become inferior at high frequencies due to its high interconnection series resistance.

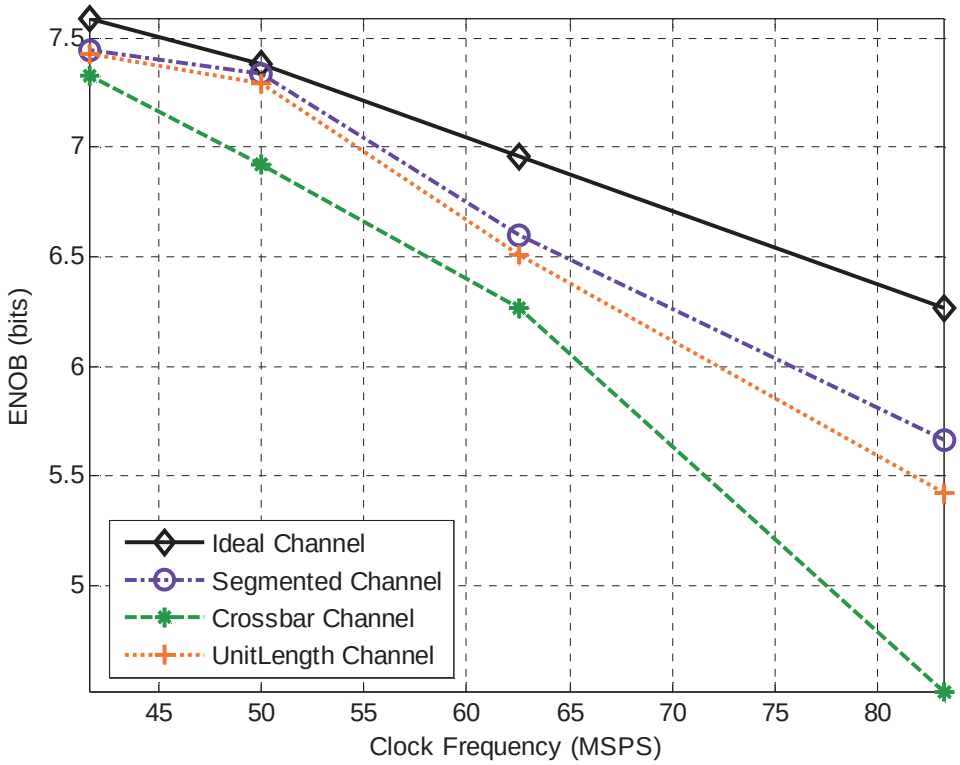


Figure 7.14 ENOB versus clock frequency (with 3MHz input)

7.6 Conclusions

In this Chapter, an example of hierarchical analog system design is presented. An 8-bit pipelined ADC is implemented using the FPAA developed. The design flow (circuit partition, placement, routing and post-layout simulation) are illustrated, and detailed configurations of the CABs to realize sub-circuits like the MDAC and comparators are described. The key performance (static and dynamic) of the ADCs implemented with different routing channels are compared and discussed. The results show that a well designed routing architecture can not only greatly improves

the routability but also reduce the interconnect parasitics and therefore the performance degradation.

Chapter 8

Conclusions and Future Work

In this dissertation, design methodologies of high-performance FPAA for hierarchical implementation of analog and mixed-signal systems are examined. Key aspects like programming technology, routing architecture optimization, analog performance-driven routing, flexible CAB circuit and topology are addressed in details. The main contributions of this work can be summarized as follows:

1. Innovative programming technology, laser Makelink^{TM} is used to minimize the non-ideality of programmable switches. Providing metal-to-metal links with extremely low resistance and negligible capacitance, it is the enabling technology of configurable analog devices with high-performance, which will be severely limited by popular programming technologies like SRAM-controlled pass transistor used in configurable digital devices.

2. Channel Segmentation schemes are investigated to improve the routability while reducing interconnect parasitics and cross-talk. Compared to the crossbar routing channel employed by the majority of existing academic and commercial FPAA devices, a well segmented routing channel consisting of wire segments of various lengths and staggering locations can better match the actual net distribution, and

therefore allow more nets to be routed without increasing the channel area, or require less channel area at the same capacity At the same time, since a better matched routing greatly reduces wire wastage, which causes unnecessary parasitic capacitance and cross-coupling, a segmented channel is shown to significantly reduce interconnect parasitics and then enhance both the speed and accuracy of the system.

3. In large scale arrays where long connections are expected, the channel segmentation alone cannot guarantee the interconnect delays and cross-couplings are within the performance bound. Buffer insertion algorithms are studied for several scenarios, namely, delay optimization, cross-coupling noise optimization and noise-constrained delay optimization. Analytical results are derived for a single net, and combined channel segmentation and buffer insertion algorithms are proposed for each scenario.

4. A high-flexibility CAB is built using a fully differential internal routing architecture. Consisting of an high-gain, high bandwidth two-stage OPAMP, whose second stage is detachable, programmable capacitor array and resistor array, and controllable pass-transistors, the coarse-granulized CAB can realized a wide range of commonly used analog functions like sample-and-hold, comparator, precise gain amplifier and analog filters, in both continuous-time mode and switched-capacitor circuits.

5. As a demonstration, an 8-bit pipelined A/D Converter is hierarchically implemented via the proposed FPAA. Detailed design flow is described, including circuit partition, sub-block placement and routing results. Key performances of

the ADCs implemented with different routing architectures are presented and discussed.

However, for FPAAs to play more vigorously in analog design automation, and eventually achieve a similar role as FPGAs in the digital world, the work done in this dissertation is still in the infant stage. The following work is expected to be continued in the future:

1. Analog IP Library Development: To properly drive the off-chip load, various application specific circuit functions at a higher design level should be added into the IP module library as the pre-qualified design for the end users. Those may include but not limited to: ADCs, PLLs, high-order filters, control circuits and I/O block
2. Automatic circuit partition and placement, which takes the block-level signal flow diagram or even the design descriptions and specifications as the input, chooses the necessary modules from the IP library with proper parameters, and place them in the CABs available in the device.
3. High-level Design Methodology: Instead of a traditional bottom-up design, a top-down process can be employed. Design entry can start from description languages like Verilog-A or AHDL. The overall system performance can be estimated at the early design stage thus preventing the risk of insufficient design or over-design.

References

[1.1] Cofler, A.M., *et al.*, "A reprogrammable EDGE baseband and multimedia handset SoC with 6-mbit embedded DRAM," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 97 – 106, January 2006.

[1.2] "Innovation for WiMAX Fixed Wireless Broadband Platforms," http://www.intel.com/network/connectivity/products/wireless/prowireless_5116.htm

[1.3] Okamoto, K., *et al.*, "A fully integrated 0.13 μm CMOS mixed-signal SoC for DVD player applications," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 1981 – 1991, November 2003.

[1.4] Behzad Razavi, *Design of analog CMOS integrated circuits*, McGraw-Hill, 2001

[1.5] John V. Oldfield and Richard C. Dorf, *Field-programmable gate arrays: reconfigurable logic for rapid prototyping and implementation of digital systems*, John Wiley & Sons, 1995

[1.6] Carley, L.R., *et al.*, "Synthesis tools for Mixed-Signal ICs: progress on frontend and backend strategies," *Design Automation Conference* 1996, pp. 298 – 303

[1.7] Geert Van der Plas and Georges Gielen, *A Computer-Aided Design and Synthesis Environment for Analog Integrated Circuits*, Kluwer Academic Publishers, 2002

[1.8] Nicolas Williams, *Facing the challenges in analog design*, http://www.eet.com/news/design/showArticle.jhtml?articleID=181502541

[1.9] E. K. F. Lee and P. G. Gulak, "A CMOS field programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1860–1867, December 1991.

[1.10] R. T. Edwards, K. Strohbehn and S. E. Jaskulek, "A Field-Programmable Mixed-Signal Array Architecture Using Antifuse Interconnects," *ISCAS* 2000, pp. 319-322

[1.11] Bogdan Pankiewicz, *et al*., "A Field Programmable Analog Array for CMOS Continuous-Time OTA-C Filter Applications," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 125-136, February 2002.

[1.12] "MPAA020 Field Programmable Analog Array Datasheet," Motorola, 1997.

[1.13] "New Dimensions in ISP Programmable Analog Circuit," Lattice Semiconductor Corp., Hillsboro, OR, 1999.

[1.14] The AN10E40 Field Programmable Analog Array. Anadigm Co, Crewe, U.K. [Online]. Available: http://www.anadigm.com.

[1.15] S. Trimberger, *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994

[1.16] http:// www.actel.com

[1.17] M. John and S. Smith, *Application-Specific Integrated Circuits*, VLSI Systems Series, 1997

[1.18] Zhuo Gao, Ji Luo, Hu Huang, Wei Zhang and J. B. Bernstein, ”Reliable laser programmable gate array technology,” *Proc. International Symposium on Quality Electronic Design* 2002, pp. 252 –256.

[1.19] J. B. Bernstein, W. Zhang and C. H. Nicholas, “Laser Formed Metallic Connections”, *IEEE Trans. on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, Vol. 21, No. 2, pp. 194, May 1998.

[1.20] Glenn Gulak and Dean R. D’MJZLLO, “A Review of Field Programmable Analog Arrays,” *SPIE* Vol. 2914, pp. 152-169

[1.21] V. Betz, J. Rose and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1999.

[1.22] J. Rose and D. Hill, “Architectural and physical design challenges for one-million gate FPGA’s and beyond,” in *ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 1997, pp. 129–132.

[2.1] V. betz and J. Rose, “FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density,” FPGA’99, pp. 59-68

[2.2] Rose, J. and Brown, S., “Flexibility of interconnection structures for field-programmable gate arrays,” *IEEE Journal of Solid-State Circuits*, vol. 26, pp.277 – 282, March 1991

[2.3] Kaushik Roy and Sudip Nag, "Automatic Synthesis of FPGA Channel Architecture for Routability and Performance," *IEEE Tran. on VLSI Systems*, vol. 2, pp. 508-511, December 1994.

[2.4] El Gamal, J. Greene, and V. Roychowdhury, "Segmented channel routing is nearly as efficient as channel routing (and just as hard)," *Proc. Advanced Research VLSI*, Santa Cruz, CA, pp. 193-221, 1991.

[2.5] K. Zhu and D.F. Wong, "On channel segmentation design for row-based FPGAs," *Proc. ICCAD*, pp. 26-29, 1992.

[2.6] Roy, K. and Mehendale, M., "Optimization Of Channel Segmentation For Channeled Architecture FPGAs," *CICC*'1992, pp. 4.4.1 - 4.4.4

[2.7] M. Pedram, B. S. Nobandegani, and B. T. Preas, "Design and analysis of segmented routing channels for row-based FPGAs," *IEEE Trans. on Computer Aided Design*, vol. 13, no. 12, pp. 1470-1479, Dec. 1994.

[2.8] W. K. Mak and D. F. Wong, "Channel segmentation design for symmetrical FPGAs," *ICCD*97, pp. 496–501.

[2.9] Jai-Ming Lin, Song-Ra Pan and Yao-Wen Chang, "Graph matching-based algorithms for array-based FPGA segmentation design and routing," *Proc. ASP-DAC*, pp.851-854, 2003

[2.10] Burman, S., Kamalanathan, C. and Sherwani, N., "New channel segmentation model and associated routing algorithm for high performance FPGAs," *ICCAD*'92, pp. 22 -25

[3.1] Semiconductor Industry Association, National technology Roadmap for Semiconductors, 1997.

[3.2] S. Ganesan and R. Vemuri, "FAAR: A Router for Field-Programmable Analog Arrays", 12th *Intl. Conf. VLSI Design* '99, pp.556-563.

[3.3] Kaushik Roy and Sudip Nag, "Automatic Synthesis of FPGA Channel Architecture for Routability and Performance," *IEEE Tran. on VLSI Systems*, vol. 2, pp. 508-511, December 1994.

[3.4] Roychowdhury, V.P., Greene, J.W. and El Gamal, A. "Segmented channel routing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 79-95, Jan 1993.

[3.5] Yachyang Sun, Ting-Chi Wang, Wong, C.K and Liu, C.L., "Routing for symmetric FPGAs and FPICs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 20-31, Jan 1997.

[3.6] Kai Zhu, Yao-Wen Chang and Wong, D.F., "Timing-driven routing for symmetrical-array-based FPGAs," *ICCD*98, pp. 628-633.

[3.7] El Gamal, J. Greene, and V. Roychowdhury, "Segmented channel routing is nearly as efficient as channel routing (and just as hard)," *Proc. Advanced Research VLSI*, Santa Cruz, CA, pp. 193–221, Mar. 1991.

[3.8] C. Lee, "An Algorithm for Path Connections and its Applications", IRE Trans. Electron. Comp, vol. 10, 1961.

[3.9] T. Cormen, C. Leiserson et. al, *Introduction to Algorithms*, McGraw-Hill, 2001.

[3.10] L. McMurchie, C. Ebeling, "Pathfinder: A Negotiation-based Performance-Driven Router for FPGAs", Univ. of Washington, 1996.

[3.11] Bernhard Korte, Jens Vygen, *Combinatorial Optimization: theory and algorithms*, Springer, 2006.

[3.12] W. Kao, C. Lo, M. Basel and R. Singh, "Parasitic Extraction: Current State of the Art and Future Trends," *Proceedings of the IEEE*, vol. 89, No. 5, May 2001.

[3.13] N.D. Arora, K.V. Raol, R. Schumann and L.M. Richardson, "Modeling and Extraction of Interconnect Capacitances for Multilayer VLSI Circuits," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 15(1):58-66, Jan. 1996.

[3.14] U. Choudhury, and A. Sangiovanni-Vincentelli, "Constraint-Based channel routing for analog and mixed analog/digital circuits" *IEEE Tans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 4, 1993.

[3.15] Jai-Ming Lin, Song-Ra Pan and Yao-Wen Chang, "Graph matching-based algorithms for array-based FPGA segmentation design and routing," *Proc. ASP-DAC*, pp.851-854, 2003.

[3.16] W. Karush, "Minima of Functions of Several Variables with Inequalities as Side Constraints," *M.Sc. Dissertation*. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois. 1939.

[3.17] Kuhn, H. W. and Tucker, A. W., "Nonlinear programming," *Proceedings of 2nd Berkeley Symposium*: 481-492, Berkeley: University of California Press, 1951.

[4.1] Zhou, D., Preparata, F.P., Kang, S.M., “Interconnection delay in very high-speed VLSI,” *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 779 - 790, July 1991

[4.2] Khellah, M., Brown, S., Vranesic, Z., “Minimizing interconnection delays in array-based FPGAs,” *CICC*, pp.181 - 184, 1994

[4.3] V. Betz and J. Rose, “Circuit design, transistor sizing and wire layout of FPGA interconnect,” *CICC*, pp. 171 -174, 1999.

[4.4] Pamunuwa, D., Tenhunen, H., “Repeater insertion to minimise delay in coupled interconnects,” *International Conference on VLSI Design*, pp. 513 - 517, January 2001

[4.5] L. P. P. P. van Ginneken, “Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay”, *Proc. International Symposium on Circuits and Systems*, 1990, pp. 865-868.

[4.6] C. Alpert and A. Devgan, “Wire Segmenting for Improved buffer Insertion,” *Proc. DAC*, pp. 588-593, 1997.

[4.7] C.J. Alpert, A. Devgan and S. T. Quay, “Buffer Insertion for Noise and Delay Optimization,” *IEEE Trans. on Computer-Aided Design*, vol. 18, pp. 1633-1645, November 1999.

[4.8] W. C. Elmore, “The Transient Response of Damped Linear Networks with Particular Regard to Wide Band Amplifiers,” Journal of Applied Physics, vol. 19, pp. 55-63, 1948.

[4.9] Semiconductor Industry Association, National technology Roadmap for Semiconductors, 1997.

[5.1] C. Alpert and A. Devgan, “Wire Segmenting for Improved buffer Insertion,” *Proc. DAC*, 1997, pp. 588-593.

[5.2] J. Con, T. Kong and D. Z. Pan, “Buffer Block Planning for Interconnect-Driven Planning,” *Proc. ICCAD*, 1999, pp. 358-363.

[5.3] P. Sakar and C. K. Koh, “Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning,” *IEEE Trans. on Computer-Aided Design*, vol. 20, pp. 660-671, May 2001.

[5.4] C.J. Alpert, A. Devgan and S. T. Quay, “Buffer Insertion for Noise and Delay Optimization,” *IEEE Trans. on Computer-Aided Design*, vol. 18, pp. 1633-1645, November 1999.

[5.5] S. M. Li, Y. Cherng and Y. Chang, “Noise-Aware Buffer Planning for Interconnect-Driven Floorplanning,” *Proc. ASP-DAC*, 2003, pp. 423-426.

[5.6] Vittal and M. Marek-Sadowska, “Crosstalk Reduction for VLSI,” *IEEE Trans. on Compute-Aided Design*, vol. 16, pp. 290-298, March 1997.

[5.7] Devgan, “Efficient Coupled Noise Estimation for On-Chip Interconnects,” *Proc. ICCAD*, 1997, pp. 147-151.

[5.8] W. C. Elmore, “The Transient Response of Damped Linear Networks with Particular Regard to Wide Band Amplifiers,” *J. Appl. Phys.*, vol. 19, pp. 55-63, 1948.

[5.9] Hu Huang, J. B. Bernstein, M. Peckerar and J. Luo, “Combined Channel Segmentation and Buffer Insertion for Routability and Performance Improvement of Field programmable Analog Arrays,” *Proc. ICCD*, 2004, pp. 490-495.

[5.10] Bogdan Pankiewicz, *et al.*, "A Field Programmable Analog Array for CMOS Continuous-Time OTA-C Filter Applications," *IEEE J. Solid-State Circuits*, vol. 37, pp. 125-136, Feb. 2002.

[5.11] The AN10E40 Field Programmable Analog Array. Anadigm Co, Crewe, U.K. [Online]. Available: http://www.anadigm.com.

[5.12] K. Zhu and D.F. Wong, "On channel segmentation design for row-based FPGAs," *Proc. ICCAD*, pp. 26-29, 1992.

[5.13] Jai-Ming Lin, Song-Ra Pan and Yao-Wen Chang, "Graph matching-based algorithms for array-based FPGA segmentation design and routing," *Proc. ASP-DAC*, pp.851-854, 2003

[5.14] Semiconductor Industry Association, *National technology Roadmap for Semiconductors*, 1997.

[6.1] R. Zebulum, et al, "A reconfigurable platform for the automatic synthesis of analog circuits," *The Second NASA/DoD Workshop on Evolvable Hardware*, 2000, pp. 91 - 98

[6.2] E. Lee and G. Gulak, "A CMOS field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, pp.1860 – 1867, Dec. 1991

[6.3] E. Lee and G. Gulak, "A Transconductor-Based Field-Programmable Analog Array", *ISSCC Digest of Technical Papers*, Feb. 1995, pp. 198-199.

[6.4] P. Chow, P. Chow, P.G. Gulak, "A Field-Programmable Mixed-Analog-Digital Array," *ACMISIGDA FPGA'95*, Monterey, CA, Feb. 12-14,1995, pp. 104-109.

[6.5] E. Pierzchala, M. Perkowski, Paul Van Halen Rolf Schaumann, "Current-

Mode Amplifier-Integrator for a Field-Programmable Analog Array," *ISSCC Digest of Technical Papers*, pp. 196-197, Feb. 1995.

[6.6] C. Premont, R. Grisel, N. Abouchi and J.-P. Chante, "Current-conveyor based field programmable analog array," *MWSCAS'96*, pp.155 – 157.

[6.7] S.T. Chang, B.R. Hayes-Gill and C.J. Paull, "Multi-function block for a switched current field programmable analogue array," *MWSCAS'96*, pp.158 – 161.

[6.8] X. Quan, S.H.K. Embabi and E. Sanchez-Sinencio, "A current-mode based field programmable analog array architecture for signal processing applications," *CICC'98*, pp. 277 – 280

[6.9] Bogdan Pankiewicz, *et al.*, "A Field Programmable Analog Array for CMOS Continuous-Time OTA-C Filter Applications," *IEEE J. Solid-State Circuits*, vol. 37, pp. 125-136, Feb. 2002.

[6.10] Joachim Becker and Yiannos Manoli, **"**A Continuous-Time Field Programmable Analog Array (FPAA) consisting of digitally reconfigurable Gm-cells.," *ISCAS*'04, pp. 1092-1095.

[6.11] J.D. Gray, C.M. Twigg, D.N. Abramson and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossbar network," *ISCAS'05*, pp. 468 – 471.

[6.12] F. Goodenough, "Analog Counterparts of FPGAs Ease System Design", *Electronic Design*, Oct. 14, 1994, pp. 63-73.

[6.13] Bratt and I. Macbeth, "Design and Implementation of a Field Programmable Analogue Array," *FPGA '96*, pp.88 – 93.

[6.14] H. Kutuk and Sung-Mo Kang, "A field-programmable analog array (FPAA) using switched-capacitor techniques," *ISCAS '96*, pp. 41 – 44.

[6.15] D. Anderson, *et al.*, "A field programmable analog array and its application," *CICC'97*, pp. 555 – 558.

[6.16] K. Papathanasiou, T. Brandtner and A. Hamilton, "Palmo: pulse-based signal processing for programmable analog VLSI," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, pp. 379 – 389, June 2002.

[6.17] R.T. Edwards, K. Strohbehn and S.E. Jaskulek, "A field-programmable mixed-signal array architecture using antifuse interconnects," *ISCAS'00*, pp.319 – 322.

[6.18] Ji Luo, J.B. Bernstein, J.A. Tuchman, Hu Huang, *et al.*, "A high performance radiation-hard field programmable analog array," *ISQED'04*, pp.522 – 527.

[7.1] http://www.agilent.com/labs/news/2003features/fea_adc03.html

[7.2] http://www.planetanalog.com/features/showArticle.jhtml?articleID=20900003

[7.3] http://lib.tkk.fi/Diss/2002/isbn9512262231/

[7.4] http://audacity.sourceforge.net/manual-1.2/tutorial_basics_1.html

[7.5] http://www.videsignline.com/showArticle.jhtml?printableArticle=true&articleId=181502565

[7.6] Plassche, Rudy J. van de., *CMOS integrated analog-to-digital and digital-to-analog converters*, Kluwer Academic Publishers, 2003

[7.7] http://www.maxim-ic.com/appnotes.cfm/appnote_number/634

[7.8] Varzaghani, A.; Yang, C.-K.K., "A 600-MS/s 5-bit pipeline A/D converter using digital reference calibration," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 310 - 319, February 2006

[7.9] Zanchi, A.; Tsay, F., "A 16-bit 65-MS/s 3.3-V pipeline ADC core in SiGe BiCMOS with 78-dB SNR and 180-fs jitter," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 1225 – 1237, June 2005

[7.10] S. H. Lewis, et al., "10-b 20-Msample/s analog-to-digital converter," *IEEE journal of. Solid-State Circuits*, vol. 27, pp. 351-358, March 1992.

[7.11] Cho, T.B., Gray, P.R., "A 10 b, 20 Msample/s, 35 mW pipeline A/D converter," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 166 - 172, March 1995

[7.12] David Johns, Ken Martin, *Analog integrated circuit design*, John Wiley & Sons, 1997

[7.13] Nam, G.-J. , et al., "A Fast Hierarchical Quadratic Placement Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 678 - 691, April 2006

[7.14] http://www.maxim-ic.com/appnotes.cfm/appnote_number/283

[7.15] B. Razavi, *Principles of Data Conversion System Design*, IEEE Press, New York, 1995.

[7.16] "IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters, Standard, Measurements," *IEEE Standard 1241-2000*, December 2000.