Time-interleaved SAR ADC Design Using Berkeley Analog Generator

Zhaokai Liu
Borivoje Nikolic, Ed.
Vladimir Stojanovic, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-109
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-109.html

May 29, 2020

Copyright © 2020, by the author(s).
All rights reserved.

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

Time-interleaved SAR ADC Design Using Berkeley Analog Generator

by Zhaokai Liu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Borivoje Nikolic
Research Advisor

May 28, 2020.
Date

* * * * * * *

Vladimir Stojanovic
Second Reader

5/27/20
Date

Abstract

Among different ADC architectures, the successive approximation register (SAR) ADC has flexible architecture, high power efficiency and is suitable for the digital CMOS process. Its building blocks rely on MOS switches and latches, which makes it strongly benefits from technology scaling. Time-interleaving (TI) architectures can provide a higher sampling rate because they help relax the power-speed trade-offs of ADCs. Therefore, combining SAR with time-interleaving becomes a good solution to many digital signal processing applications that require power-efficient analog-to-digital conversion. Based on Berkeley Analog Generator (BAG), a time-interleaved SAR ADC generator has been implemented in different technologies. To explore the design flow using circuit generators, this report discusses the working principle and implementation of time-interleaved SAR ADC. A test chip has been taped out in Intel22nm FFL process, containing 6 different versions of ADCs. In each design, a 9-bit 16-way TI-SAR ADC samples at 10GS/s with a memory block storing the digitized result from ADC.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Analog-to-digital converter (ADC) has been one of the most commonly used building blocks of mixed-signal circuit as they act as the interface between analog and digital realm. It is used to acquire analog signals from different sources and convert them into digital form for analysis or transmission. Therefore, one of the keys to the success of different digital systems which operate at a wide range of continuous-time signal has been the advance in ADC design. The speed and performance of ADC are often the bottleneck when building modern systems.

New applications have continuously been driving research in the ADC targeting at higher speed and resolution. For the application that benefits from the fast evolution of the digital integrated circuit, the steady increase in the performance naturally leads to more sophisticated signal processing in the digital domain and moves ADC closer to the input of chip in order to capture more analog information. Communication systems that enable higher data rates will demand ADC with higher sampling rates. And higher resolution is also needed when complex modulation is applied.

Figure 1.1 shows the Walden figure of merit of ADCs published at ISSCC and VLSI Symposium [1]. The standard Walden figure of merit here is defined as

$$FoM = \frac{P}{2 \cdot min(f_s/2, BW_{eff}) \cdot 2^{ENOB}} \tag{1.1}$$

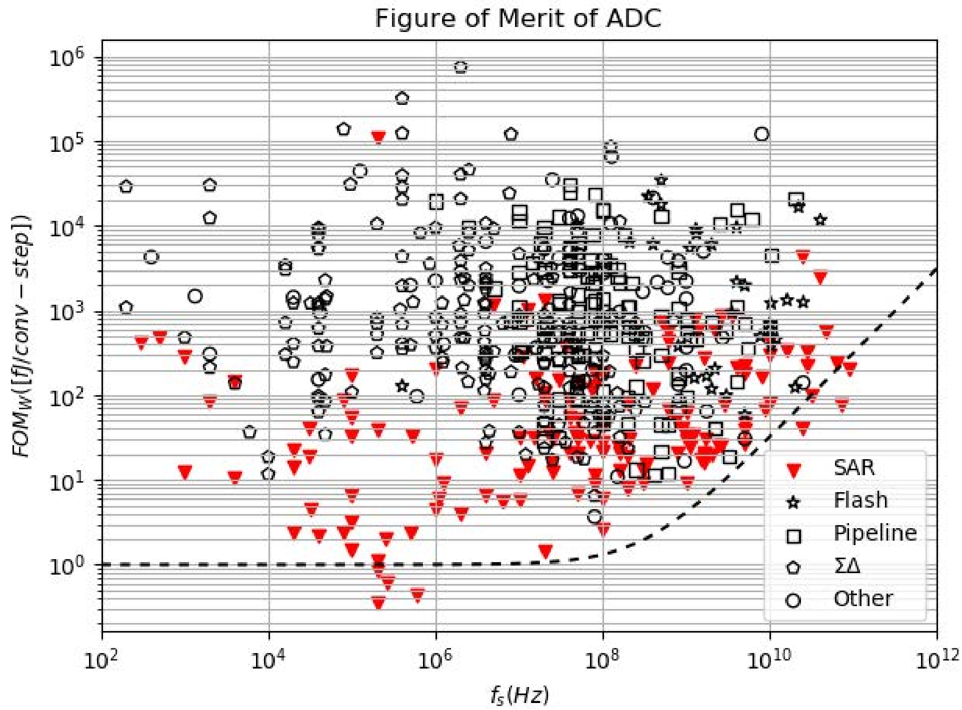

Figure 1.1: Figure of merit of all ADCs published at ISSCC and VLSI Symposium from 1997 to 2019 [1].

Where P is the sampling frequency, f_s is the sampling frequency and BW_{eff} is the effective resolution band width.

It is evident from the plot that different architectures ADC has been adopted for different applications with specific speed and performance requirements. Specifically, ADCs work at multi-gigasample per second with moderate resolution is widely used in high-performance electronic/optical link and radar/lidar sensing system([2], [3]). As stated above, these systems naturally scale to advanced technology nodes for both technological and economic benefits. Therefore, ADCs that enable SoCs to use digital power reduction brought by technology scaling should adopt architectures that benefit scaling as well.

One problem for circuit design in an advanced technology node is that as devices size has shrunk exponentially, the number of design rules also increases exponentially, making it difficult to quickly prototype design in modern processes. A generator-based design methodology can help to deal with stringent and complicated rules. Circuit generators can shorten the time spent in post-layout verification, help accelerate the design cycle, and enable designers to explore circuits under different technologies.

1.2 Research goals

The goal of this research is to develop a circuit generator based on SAR architecture for energy-efficient data conversion for moderate resolution and working at moderate to high frequency. SAR architecture can be combined with time-interleaving technique to achieve a flexible range of configuration.

The function of an ADC is to generate a N-bit digital output such that the analog signal can be approximated as $V_{DAC} = D/2^N \cdot V_{ref}$, where V_{ref} is the reference voltage. Depending on the approach of getting the final value, there are different categories of ADCs. An ADC's sample rate (f_s) can be chosen either for Nyquist rate operation ($f_s = 2 \times f_{BW}$) or for oversampled operation ($f_s >> 2 \times f_{BW}$). ADC architectures like flash, pipeline, successive approximation register (SAR) sample input signal at Nyquist frequency while sigma-delta ($\Sigma\Delta$) ADC working with a higher sample rate.

Each of these typologies has their own unique advantages in terms of power, speed and resolution that makes them suitable for a certain particular use scenario. For example, pipeline ADCs perform analog-to-digital conversion by cascade low-resolution stages that sample, coarse quantizing and amplify residue for the next stage. While this architecture is suitable for high-speed applications, the requirement of precise active amplifiers in each stage make it analog-intensive and take much power. In oversampled ADCs, which typically implemented as $\Sigma\Delta$ converter, higher resolution is provided through oversampling and noise shaping. A broad range of $\Sigma\Delta$ converters can also be implemented in advanced technology nodes. But it still not able to benefit from power reduction with CMOS scaling because op-amps usually are needed to construct analog integrator in loop filter.

In SAR architecture, a binary search algorithm is performed. It typically uses capacitor digital-to-analog converter to substract a quantized amount of reference voltage from sampled voltage. In most cases, only a clocked digital-like comparator and digital logic are needed. This results in a primary reason for modern SAR ADCs to thrive. The MOS switches and latches strongly benefit from aggressive technology scaling. The digital-intensive highly efficient operation makes SAR ADC a strong candidate in modern systems. It is shown in Figure

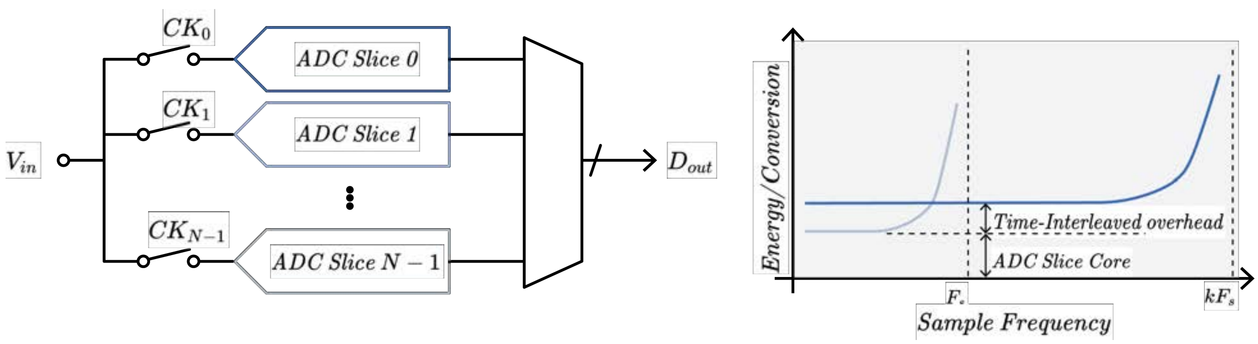

Figure 1.2: Diagram of time-interleaved concept.

1.1 that SAR-based design (marked in red) achieve leading-edge performance for sampling rate (f_s) range from tens of kilohertz to tens of gigahertz [4]. In the low-frequency regime, there are designs like [5], [6] for medical application. For moderate frequency, SAR or SAR assisted ADCs such as [7] and [8] can offer moderate resolution at low power level(1mW). At the ultra-high-speed region, a 90 GS/s design [9] is demonstrated to be suitable for optical and electrical data link applications.

Time-interleaving originally was used as an effective method with area and power penalty[10]. But it turns out to be a good solution for power-speed trade-off and will relieve many problems even when speed is not of primary limitation. The basic working principle is shown in Figure 1.2. As the speed of a single-channel ADC approaches the limits of the technology, the power-speed trade-off becomes nonlinear and demanding a disproportionately higher power for the desired increase in speed. Time-interleave relaxes the trade-off and makes pushing to higher conversion speed possible. And this benefit comes with the overhead of sampling clock phase generation, which makes the energy efficiency of equivalent single channel conversion worse. And also the multi-channel structure suffers from mismatches in gain, offset and bandwidth, which usually requires some calibration techniques.

1.3 Report organization

Overall, this report evaluates the design methodology and generator-based implementation of high-speed time-interleaved SAR ADC. Also, both the generator-based design flow and the LAYGO layout generation engine in BAG are introduced

In Chapter 2, the design considerations in various building blocks of SAR ADC is explained. By exploring the design space of each building block, especially for those well-understood parts, an automated generator-based design flow can be applied to accelerate design iterations.

Chapter 3 explains the working principles of generator-based design flow and the layout generation engine. Also, some example code are presented to show the layout and schematic generation steps.

Chapter 4 talks about circuit and generator details. The implementation of generation in Intel 22nm FFL is presented. Chapter 5 discusses the future work and possible improvement on the speed and resolution of ADC generator.

Chapter 2

SAR ADC Design Considerations

Generally, successive approximation register (SAR) ADCs don't reply on high-performance analog circuits such as precise amplification circuits. The performance of SAR ADC depends on the precision/speed of voltage comparison and digital logic gates. The major power consumption of SAR ADCs comes from charging and discharging capacitive DAC. Therefore, as the technology scales, the speed of SAR ADC improves and the power scales according to $C_{tot}V_{dd}^2$.

To better adopt automatic generator-based design flow and develop the SAR ADC generator, this chapter review major aspects of design in SAR building blocks. Also, time-interleaved (TI) topology that applied to ADCs is further examined to understand the advantages and limitations of it. First, this chapter will review the basic operation of SAR ADC, then discuss major building blocks including comparators, capacitive DAC and SAR logic. Lastly, the speed benefits in TI-SAR architecture, effects on metastability and major challenges arise from multiple channel mismatches are explained.

2.1 General working principle of SAR ADC

Figure 2.1 shows a conceptual diagram of SAR ADC.It consists of capacitor DAC, digital logic, comparator, and switches and the plot on the right shows an example conversion waveform. The Algorithm 1 describes the binary search operation in SAR.

At the beginning of each conversion, the input voltage is sampled on the capacitor array (top-plate in this example). Then the sampled voltage is compared with a reference voltage

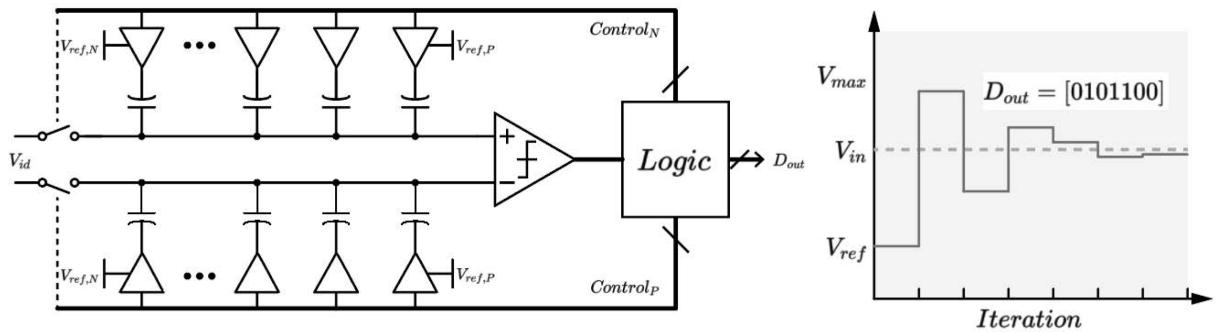

Figure 2.1: Conceptual diagram of SAR ADC with top-plate sampling.

and the comparator generates the result. Based on the first result, half of the total capacitor flip makes the reference voltage change. This operation repeats and gradually drives the differential input of the comparator toward zero. Then the conversion finishes with only the quantization error ($< \frac{1}{2^N}$) left.

This conversion process can benefit from modern nano-CMOS technology because new technology is optimized for digital operation. So logic gates and MOS switches all naturally become faster and consume less power. Also, some technology provides tall metal stack and advanced lithography which makes metal fringe capacitor can be smaller while keeping accuracy. Lastly, the voltage comparator in design can be implemented as a regenerative dynamic latch. So the entire design can be implemented in a dynamic operation manner and doesn't consume static currents. For low-speed sensor applications, it works well even at kilohertz and the power of the entire system is reduced by turning it into a "sleep" state. For aggressively high-speed application time-interleaved architecture that combines tens of ADCs is also a good choice even a single-channel of ADC can only deliver moderate speed.

There are also non-idealities in each building block. First, the noise will be collected from various sources and impact the general signal-to-noise (SNR) of design. Besides quantization noise, thermal noise sampled onto capacitive DAC and input-referred noise from the comparator also degrade the performance. Distortions caused by limited bandwidth and non-linearity in track and hold circuit limit the SNDR it can achieve. Systematic errors arise from the mismatch between DAC units and transistors in comparator cause trade-off with conversion speed. As for the power consumption, the total power can be estimated by sum

Algorithm 1 Successive approximation algorithm

Require: $V_{in} \in [V_{ref,N}, V_{ref,P}]$

$V_{max} = (V_{ref,P} - V_{ref,N})/2$
$V_{ref}[0] = V_{max}/2$
for $k = 1 \rightarrow N - 1$ **do**
  if $V_{in} > V_{ref}[k]$ **then**
    $V_{ref}[k+1] = V_{ref}[k] + V_{man}/2^k$
    $D_{out}[k] = 1$
  else
    $V_{ref}[k+1] = V_{ref}[k] - V_{man}/2^k$
    $D_{out}[k] = 0$

of contributions from each block: as the number of bits (N_{bit}) and sample frequency (f_s) increase logic stages energy (E_{logic}) as well as comparator energy (E_{cmp}) linearly increase. And capacitor array consumes the major part of the power which proportional to the reference voltage (V_{ref}) and supply (V_{dd}). Total power would be:

$$P_{tot} = (N_{bit}E_{cmp} + N_{bit}E_{logic} + C_{DAC,tot}V_{ref}V_{dd}) \cdot f_s \tag{2.1}$$

Here it is assumed that capacitor array are fully charged and discharged in each conversion, which would be modified by a switching factor α if use more sophisticated switching scheme.

2.2 Comparator

Comparators in SAR ADCs are used to generate a result of voltage comparison. It is a critical block in building high-speed converters. The target of comparator design focus on making a fast and accurate decision, reducing noise as well as lowering the meta-stability rate. The specific implementation of the comparator may vary in different applications. However, because of the reasons stated above, generally comparators are designed as a clocked dynamic circuit. This section uses strong-arm comparator as an example to illustrate detail considerations in this block.

Strong-arm comparator

Figure 2.2 shows an example of using strong-arm latches as a comparator. The pre-amplifier in front of it is optional. The comparator is essentially a regenerative latch controlled by clock phase Φ. When the clock is high, the tail current turns on and the input pair begins

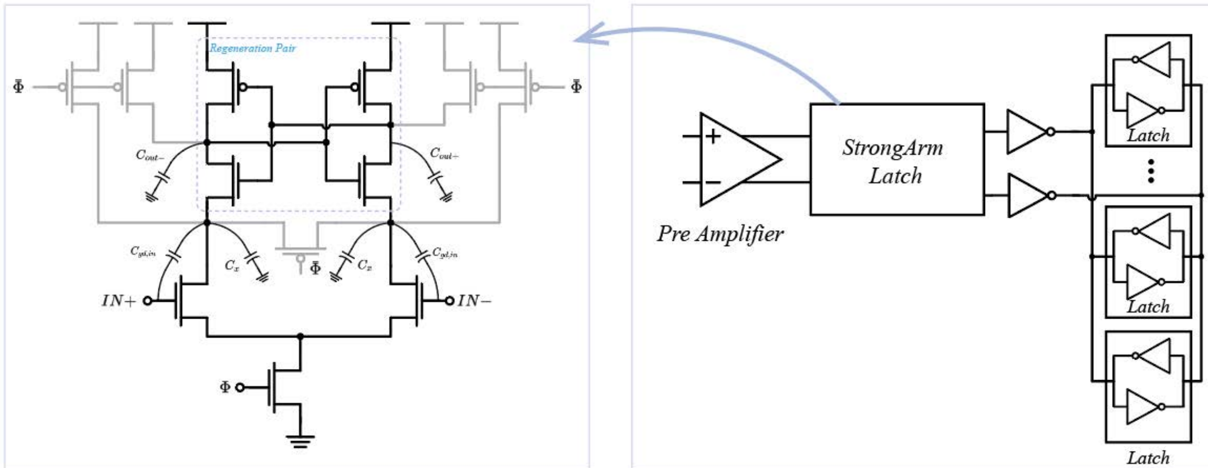

Figure 2.2: Example of comparator circuit in SAR ADCs.

to amplify input voltage difference. As the difference accumulates, the regenerative cross-coupled inverters on the top form positive feedback to generate a digital output. And this signal is buffered and drives latches in SAR logic. During $\bar{\Phi}$ phase, the current tail is off and multiple reset transistors are shown in gray color turn on and pull corresponding nodes to well-defined values. This operation helps eliminate both common-mode and differential hysteresis impacts on the comparator.

Details on working phases

This kind of dynamic circuit is different in terms of analysis method from circuits in which small-signal analysis can apply. To better understand each transistors' working states and sources of non-idealities and to form a scripted design method, the detailed working phases are explained in this part. It is possible that we can divide the working process of strong-arm into several phases in different ways. Here, Figure 2.3 shows equivalent circuits corresponding to three different phases during decision making [11]. It is assumed that before phase A start, internal nods are filly precharged to supply.

- **Amplification:** After the tail current is turned on by the clock, the input pair amplifies input voltage. And in this phase, the current tail is fairly constant because input transistors act as a differential pair. As the nodes P and N are being pulled down to different values, the comparator will enter phase B. During this period, the input pair produces an input gain:

$$A_{v,A} \approx \frac{V_{th} \cdot g_{m,in}}{I_{cm}}$$

This current mode amplification period takes τ_{int} to finish.

- **Turns on NMOS:** After entering this phase, the cross-coupled transistors form a feedback loop and exponentially split the output nodes apart. In this period, the circuit is working in a positive feedback loop and can be quantified by the time constant τ_{reg}. After several time constants, the output continuously falls under $V_{dd} - V_{th,p}$ and turns on the PMOS pair.

- **Regeneration:** After PMOS's are on, circuit it performing integration and finally have a large enough output swing for output buffer to produce a digital output. And the positive feedback will eventually pull one output to supply and another one to ground.

During each iteration of SAR operation, the amount of time used for comparison is critical since it will directly limit the ADC speed. On the evaluation of comparator design, both τ_{int} and τ_{reg} are important. For different input common-mode voltage, these two time constants will vary accordingly. And one simple approach to estimate them is the approximate total time it takes to produce a valid output as:

$$T_{tot} = \tau_{int} + \tau_{reg} \cdot log(\frac{V_{diff}}{V_{in} + V_{os}}) \tag{2.2}$$

where V_{diff} is a pre-defined output difference and V_{os} is the input-referred offset voltage. For certain input common-mode voltage, three unknown variables can be estimated by simulations with different values of V_{diff}.

Sources of noise and distortion

Besides speed, the comparator must be able to accurately make a decision. Non-idealities like noise, offset and metastability limit the performance of CMOS comparator in advanced technology.

- **Input-referred thermal noise** One of the main limits of accuracy comes from the input-referred thermal noise of comparators. The noise analysis involved in the circuit with non-linear and time-varying nature. Therefore, the small-signal analysis is not quite suitable

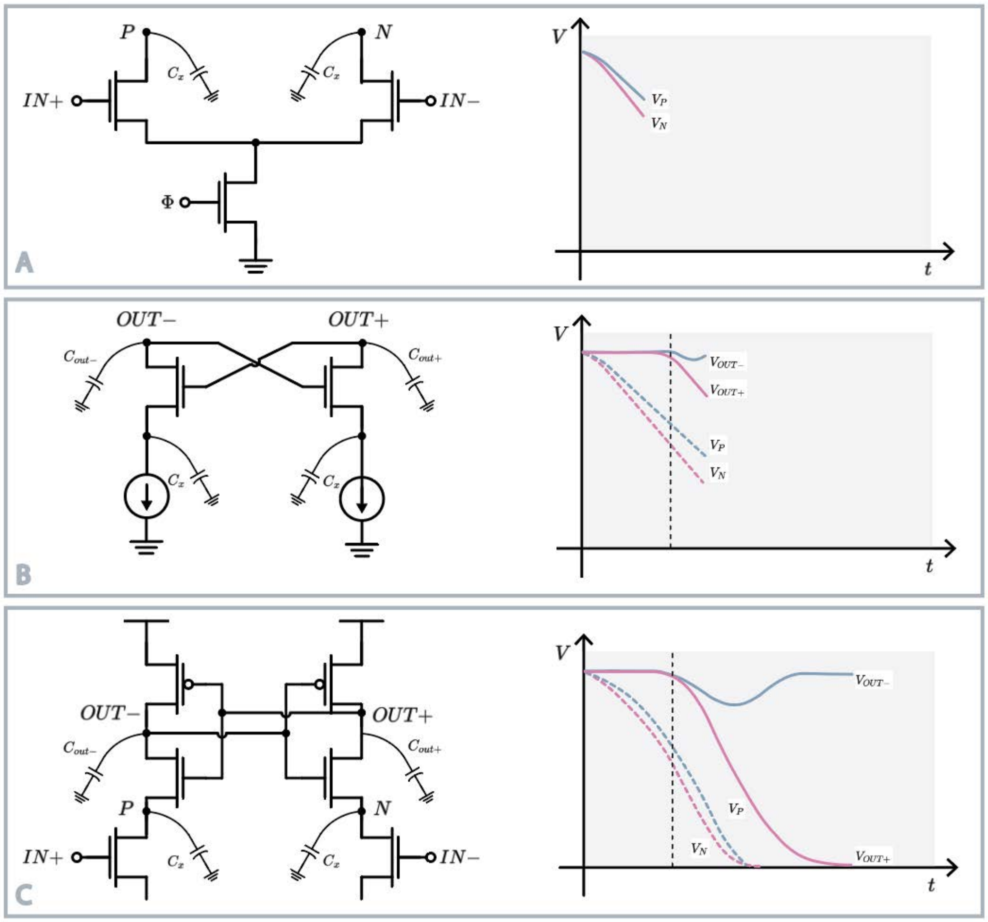

Figure 2.3: Strong arm comparator working phases.

for dynamic comparators. Intuitively speaking, the input pair will be a major source of noise. It is simply because during phase A and B in Figure 2.3 input pair transistors works as a current integrator to amplify the input signal. When the transconductance (g_m) of input pair increases, the noise current will be reduced. Also, thermal noise is averaged on the parasitic capacitance C_x in Figure 2.2. A wide device with more parasitic can also reduce the noise contribution from the input pair. Statistically speaking, with longer integration time, because of the random nature of thermal noise, the noise tends to average out on the parasitic capacitance. In summary, the input device should be wide and have g_m/I_d value in order to optimize noise performance. And comparator's input should be sized as big as allowed by speed specification. The second largest noise contributor is NMOS's in the cross-coupled inverter because they are involved in amplifying output nodes when the difference is still small.

To simulate the noise from comparators, instead of using transient simulation which takes a long time and has slow simulation speed, PSS noise analysis is a better method. Since comparator is manifesting in a periodic way in system, assuming it is working at steady-state, noise contribution during decision-making period can be integrated.

- **Input-referred offset:** Mismatches between input devices and regenerative inverter as well as the capacitive loading bring input-referred error to comparators. The contribution of offset can also be understood in a way similar to noise. In fact, if we consider the flicker noise of transistors at a low frequency, the transfer function of low-frequency noise to output will be very similar to mismatch. From this simulation, the main sources of offset can be estimated. Mismatches can be modeled as zero-mean Gaussian random variables, so increasing area and carefully layout can both reduce mismatch. Calibration is usually necessary to correct the offset error, it can be implemented as either transistor paralleled with input pair or capacitor arrays tied to the output node.

- **Metastability:** Metastability happens when differential input at comparator falls into a small range that comparator is not able to resolve within an allocated time. Along with the noise, metastability shapes the output code distribution of ADC [12]. While noise is an error source that always happens at each iteration with tiny random error, metastability happens very rarely but cause large mistake at the output digital code. Applications sensitive to metastability error need to use specific calibration to correct it. If the time pegriod allocated to the comparator is T_{cmp} and time constant is τ_{reg}, the metastability rate can be modeled as:
$$MR = \alpha \cdot exp(-\frac{T_1}{\tau_{reg}}) \tag{2.3}$$
where α is a proportionality factor. Metastability is an important issue in measurement applications, where requires a low error rate of ADC. In [13], additional logic is used to detect the occasion of metastability and make corresponding correction.

- **Kickback:** Kickback occurs because the output of the comparator always has one side is pulled down while the other side is pulled up. The kickback effect which couples input nodes P and N in Figure 2.2 with C_{GD}, will be signal dependent and therefore should be minimized in high-resolution design. It can be modeled as capacitive divided voltage on

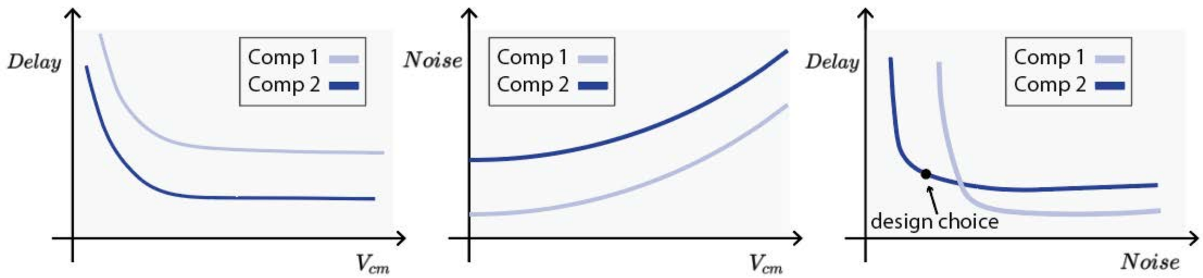

Figure 2.4: Comparator design strategy.

input gates:

$$\delta V_{kickback} = \frac{C_{GD}}{C_{GD} + C_{DAC}} \Delta V_{PN} \tag{2.4}$$

With larger DAC array, the amplitude will be reduced. Also when circuit is sensitive to kickback effect, an additional pre-amplifier can help relieve it.

From the analysis above, both noise and speed performance are closely related to input common-mode voltage of the comparator. As shown in Figure 2.4, delay tends to reduce and noise will increase with higher V_{cm}. It can be intuitively understood that the input pair is stronger and reduce integration time, make noise have less time to average out. The third plot shows the delay and noise trade-off for a specific design. It is possible to make design choice simply from this plot to decide comparator sizing and corresponding common-mode voltage it works at.

This analysis will oversimplify other non-idealities discussed above, but it shows that by combining design algorithm with generator it is possible to make quick iterations on circuit design and also automate design of some very well-understood circuits.

2.3 SAR Logic

The SAR logic often consists of two sets of registers. First one is used to monitor the current conversion bit and the second one records the decision result for each bit and drives the capacitor array. The diagram of the logic loop is shown in Figure. 2.5. In synchronous SAR design, time for each evaluation is fixed and decided by worst-case delay. If the worst-case

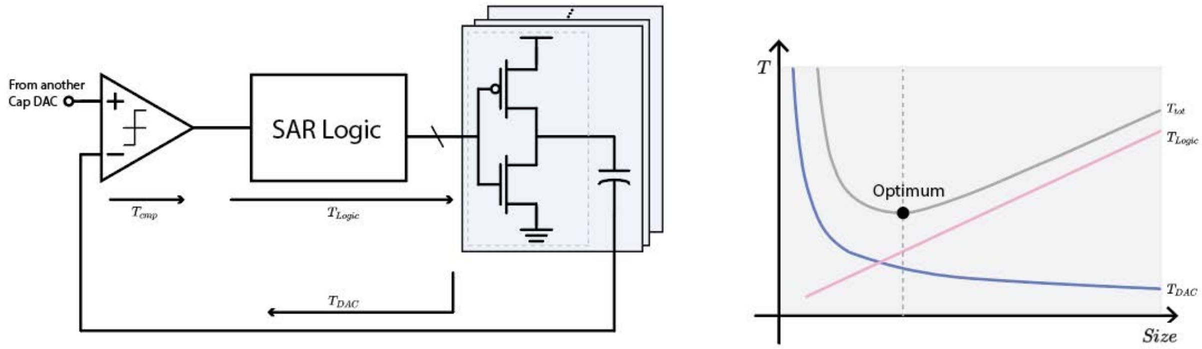

Figure 2.5: Diagram of logic loop and delay optimization.

time for resolving a certain bit is T_{conv}, total conversion time in a synchronous SAR at least should be $NT_{conv}+T_{samp}+T_{rst}$. In asynchronous SAR design [14], additional logic is inserted to assert finish when bit is resolved and trigger the next conversion.

- **Speed:** To improve the speed of SAR ADC, usually asynchronous architecture is used, because it is not limited by the worst-case delay. From an optimization perspective, the loop delay can be analyzed as Figure 2.5 shows. The delay of going through a comparator, SAR logic and settling the capacitor DAC is defined as T_{cmp}, T_{logic} and T_{DAC} separately. First of all, a larger capacitor requires larger drive strength. The capacitor driver should be sized up for different weights, and the delay is inversely proportional to driver gate (as well as corresponding latch) Assumed that only one comparator is used, with different loading at different bits, the logic delay increases linearly. As the plot shows that when combining two delays, there will be a minimum point that overall loop speed can be optimized. This result can be interpreted as a different strategies of architecture selection. The main benefit of the asynchronous operation is getting rid of wasted extra time waiting for bit that needs less time. But it is possible to carefully match delays in each iteration so that they take approximately similar amount of conversion time. The advantage of doing this is in decreasing number of gates inside the loop. Because in asynchronous design, the additional logic used for the asynchronous clock generation will introduce tens of picosecond delay even in an advanced technology.
- **Power:** The power consumption is fairly a fixed overhead, meaning that is not very

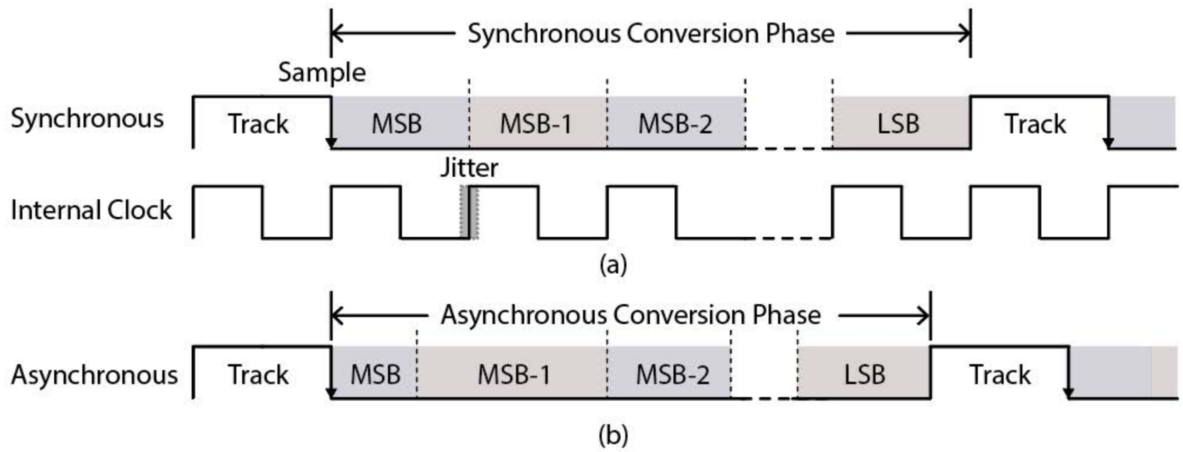

Figure 2.6: (a) Synchronous operation (b) Asynchronous operation.

sensitive to different architectures and will directly benefit from technology scaling. The power consumption of logic gates has a linear dependence on the different number of bits N_{bit}, if the switching energy per iteration is fixed. But The switching energy can also scale with N_{bit} because the number of logic gates scale with N_{bit} as well. In this case the power consumption of digital logic is quadratically depends on number of bits.

The design space of SAR logic is relatively limited and not many non-idealities are involved. However, those delays are very sensitive to layout and matching delays in each conversion step and require post-extraction simulation. By implementing these circuits using layout generators makes it possible to quickly verify the delay and reduce design time cost. The difference in timing diagram for asynchronous [14] and synchronous operations is shown in Figure 2.6. The synchronous operation relies on an internal clock that divides the conversion phase into a uniform time interval for each bit. In relatively low-speed SAR ADC design, clock generation is less critical. But for high-speed application, conversion time is composed of maximum DAC settling time, comparator resolve time and margin for worst-case clock jitter. The last part either elongates conversion time or imposes stringent requirements on clock generation. Therefore the power and speed limitation will also be limited by the internal clock circuit. It is obvious that in synchronous operation, the time needs to accommodate the worst case, which usually is the last bit since it takes a much longer time for the comparator to resolve voltage difference to around $\frac{1}{2}LSB$.

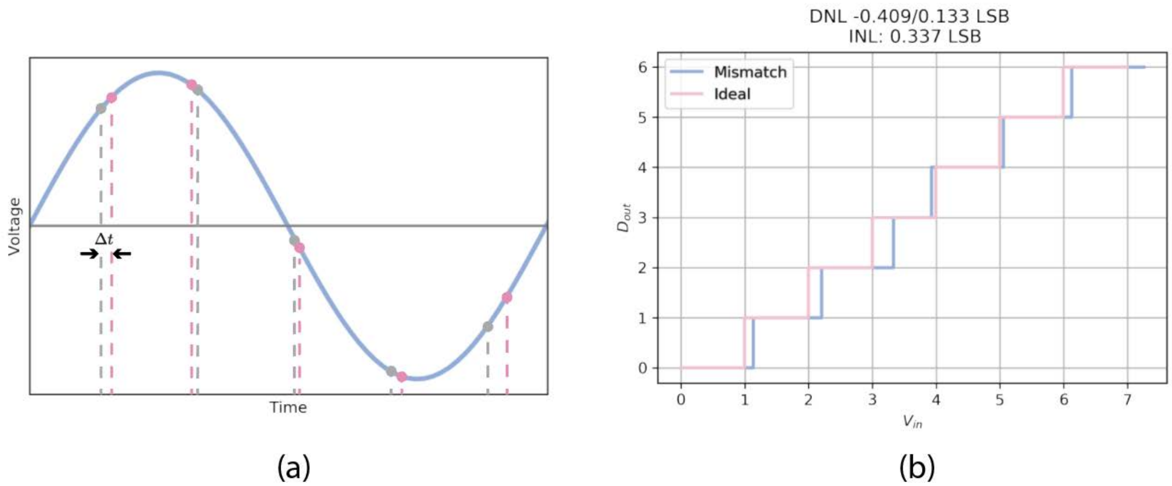

Figure 2.7: (a) Clock jitter (b)DNL and INL of a 3-bit ADC with unit capacitor mismatch.

In asynchronous operation, there will still be a global clock drive the ADC into different phases. But during the conversion phase, the asynchronous processing is triggered internally from MSB to LSB. After the comparator resolves the current bit, it produces a `DONE` signal to trigger the next bit's conversion. This operation takes advantage of different conversion bits have different times and each conversion time is not limited by the worst-case anymore. A numerical analysis in [14] shows that worst-case for total conversion time happens when input is $\frac{1}{3}V_{FS}$ or $\frac{2}{3}V_{FS}$. And if only the comparator resolve time is considered, T_{async}/T_{sync} approaches 1/2 as the number of bits increases, where T_{async}, T_{sync} are the total time for asynchronous and synchronous operation respectively.

2.4 DAC and sampling

Sampling error: The error during sampling comes from several sources: thermal noise during sampling, clock jitter, and distortion caused by switch non-linearity [15]. Those factors demonstrate different effects when implementing ADCs in a time-interleaved fashion.

First, the impact of clock jitter is shown in Figure 2.7 (a). It is assumed that the input voltage is sampled at fixed intervals. However, the uncertainty of the clock edges makes the exact time of sampling uncertain. The difference between actual sample time and theoretical sample time is shown as Δt in the plot. Clock jitter is critical in high-speed ADC design while applications at low speed don't have stringent requirements.

Thermal noise sampled on the capacitor DAC array comes from the resistance of the sampling switch. The sampled thermal noise is inversely proportional to DAC sizing C_{DAC}. The total differential noise is $4kT/C_{DAC}$. Even without any other non-idealities, the resolution of ADC still directly influenced by thermal noise. To get one more effective bit, it is required improve the signal-to-noise ratio by 6dB, which equivalently is $4 \times C_{DAC}$. As discussed above, power consumption in SAR ADCs mainly comes from charging capacitor. So thermal noise leads to the trade-off between power and resolution.

The distortion of sampling comes from the non-linearity of the sample-switch as well as the resistance associated with the switch. To get enough sample switch bandwidth, the switch needs to be sized up: while increasing the switch size, it loads itself and reach a point that further improving size doesn't help. Also, if we want to settle the input signal within half LSB, it needs sampling time:

$$t_{sample} = (N_{bits} + 1) \cdot ln(2) \cdot \tau_{sample} \tag{2.5}$$

to finish, where τ_{sample} is time constant of sampling switch, and that also means the input signal frequency is limited to

$$f_{in} < \frac{1}{2\pi \cdot \tau_{sample}} \tag{2.6}$$

The distortion comes from the fact that MOS switch doesn't maintain a fix V_{GS} during the entire sampling period. reducing the requirements on SFDR dramatically reduces the minimum sizing needs in terms of 2^{nd} and 3^{rd} harmonics.

Capacitor DAC mismatch: Ideally the digital output of ADC should be:

$$D_{out} = \left\lfloor (2^N - 1) \cdot \frac{V_{in} - V_{ref,N}}{V_{ref,P} - V_{ref,N}} \right\rfloor \tag{2.7}$$

That means each step should have a uniform width like the red line shown in Figure 2.7. However, the mismatch of each capacitor unit creates a systematic error to $V_{in} - D_{out}$ transfer curve. At low frequency, the deviation from an ideal curve will also show up as harmonic components in low-frequency simulation.

Assume unit capacitor C_L has normal distribution with variance σ_u^2. The DNL variance comes from switching from $2^{N-1}-1$ to 2^{N-1}, it has variance $\sigma_{DNL} = (2^N-1)\sigma_u$. The variance of INL can be approximated by $\sigma_{INL} = 2^{N-2}\sigma_u$. Therefore, careful layout is required during DAC design. Otherwise, the matching problem will directly limit the best performance of ADC. Figure 2.7(b) shows the transfer curve of a 3-bit binary DAC model in python, each unit capacitor has the same σ_u. The blue line shows how it deviates from the ideal curve with mismatch added. The thermometer code can also be used in capacitor DAC design. The penalty from that is the number of switches exponentially increase with number of bits. A combination of binary and thermometer code can have a good trade-off between number of switches and mismatch.

The power consumption of capacitor DAC is a major part of the SAR ADC. Some parts of power are technology scaling friendly, and reducing unit capacitor size utilize metal fringe capacitors while maintaining good matching, can also reduce power. Some fixed overhead, like thermal code to binary code decoding, should also be considered.

2.5 Time-interleaved ADC

Figure 2.8 shows the diagram of time-interleaved ADC. To get more acquisition and conversion time, multiple channels are running in parallel with their clock signal phase shift by $2\pi/N$ for adjacent slices. What is not shown here is that usually a clock generation circuit is required to span the clock signal evenly across 2π. The input is sampled consequently and digital output is multiplexed to construct digital output. Some non-ideal effects are annotated in the diagram which will be discussed later. The benefits and penalties of time-interleaved architecture are briefly explained in the rest of this chapter.

Advantages of time-interleaving

Speed and Power-speed trade-off: As a single-channel ADC's sampling frequency increases, the acquisition and conversion time both shrink. For acquisition, there are two aspects that set the lower bound. First, the voltage on the sampling capacitor needs enough time to settle. Generally, for an N-bits ADC settle to half LSB:

$$T_{settle} = \tau_{sample} \cdot ln(N_{bits} + 1) \tag{2.8}$$

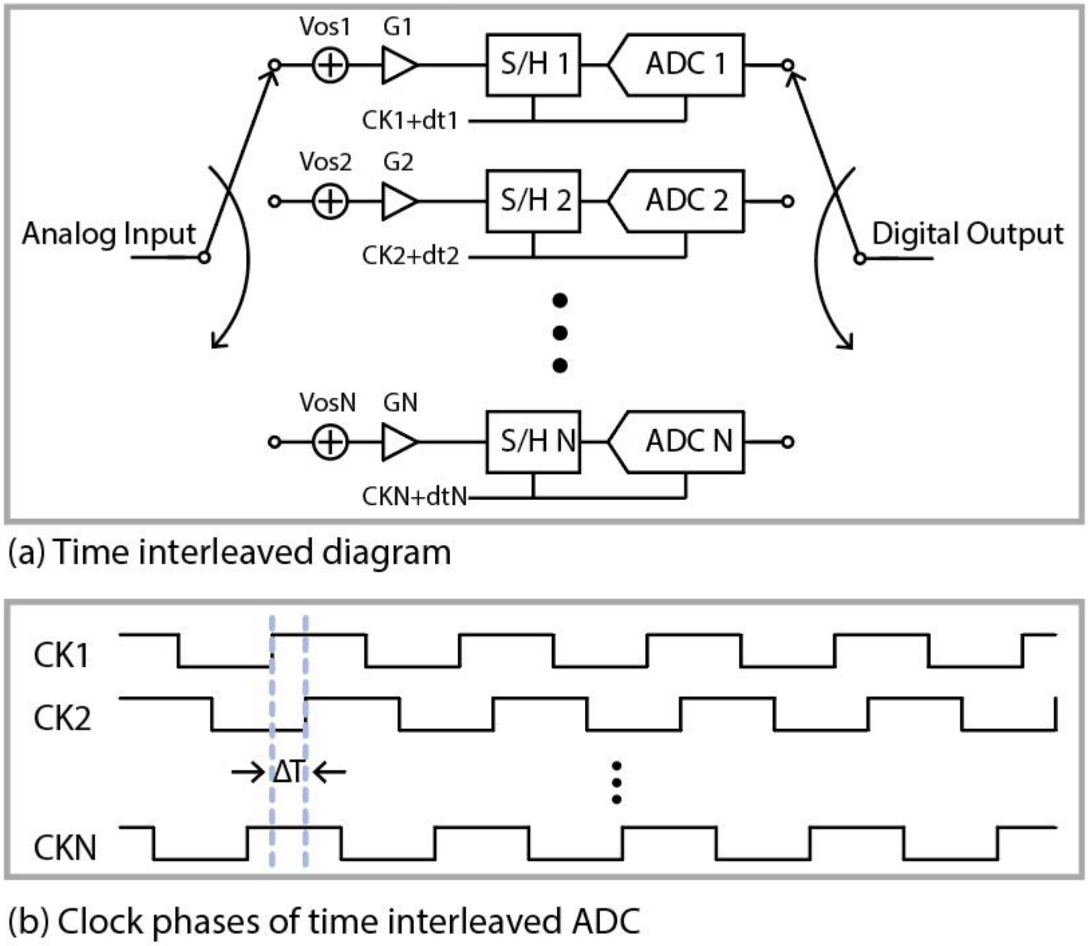

Figure 2.8: Detail diagram of time-interleaving operation.

If a ADC uses a half of clock cycle to sample, $T_{settle} < T_{clk}/2$. Also $T_{clk}/2 = 1/2f_s = 1/4f_{in,max}$ Thus

$$f_{in,max1} \leq \frac{1}{4\tau_{sample} \cdot ln(N_{bits}+1)} \tag{2.9}$$

As f_{in} grows it will ultimately be limited by the above value. While with N-way time-interleaved, the acquisition time is relaxed by N. However, the sampling switch and capacitor will have an associate time constant that limits the bandwidth.

$$f_{in,max2} \leq \frac{1}{2\pi\tau_{sample}} \tag{2.10}$$

With N increasing and $f_{in,max1} > f_{in,max2}$, this doesn't bring benefit to sampling speed any more. Also, other limitations may also limit the frequency even becomes larger than $f_{in,max}$.

From the power-speed trade-off standpoint, time-interleaved architecture is a very useful method. It can be better illustrated by considering and SAR ADC as an example. In SAR ADCs, the conversion time is mainly decided by delay of digital gates, if we consider the capacitor DAC as the final loading for gates. For a differential topology shown in 2.1, the loading is $C_{DAC}/2$. So each conversion takes time that proportional to $\tau \cdot ln(\frac{C_{DAC}}{2C_u})$. The C_u and τ are parameters that highly depend on the process. For an N-bit ADC the conversion time is roughly:

$$T_{conv} = N_{bit} \cdot \tau \cdot ln(\frac{C_{DAC}}{2C_u}) \tag{2.11}$$

Although many approaches can be taken to optimize this value, it still highly depends on technology and won't be possible to fundamentally break this limitation. It is still assumed that there is a fixed overhead and some design parameters that can be optimized:

$$T_{conv} = T_{overhead} + T_p \tag{2.12}$$

Where T_p is the delay time that depends on logic gates sizing. In modern process the power-speed are traded off in an inverse proportional dependency. The design space we have to further improve ADC speed focuses on those part that can be improved, for single ADC:

$$P = P_{overhead} + \frac{k}{T_{conv} - T_{overhead}} = P_{overhead} + \frac{k}{T_s - T_{overhead}} \tag{2.13}$$

For time-interleaved ADC, the second part becomes

$$P_{scale} = \frac{Nk}{NTs - T_{overhead}} \tag{2.14}$$

And as the N increase the power is benefit from interleaving.

Metastability rate: Use the same approximation in comparator metastable rate analysis, the metastablity is proportional to

$$P = \alpha \cdot exp(-\frac{T_{compare}}{\tau_{reg}}) \tag{2.15}$$

With longer time for conversion as stated above, the timing for comparator is relaxed. For interleaving ratio N, the result becomes exponentially better.

$$P = \alpha \cdot exp(-\frac{NT_{compare}}{\tau_{reg}}) \tag{2.16}$$

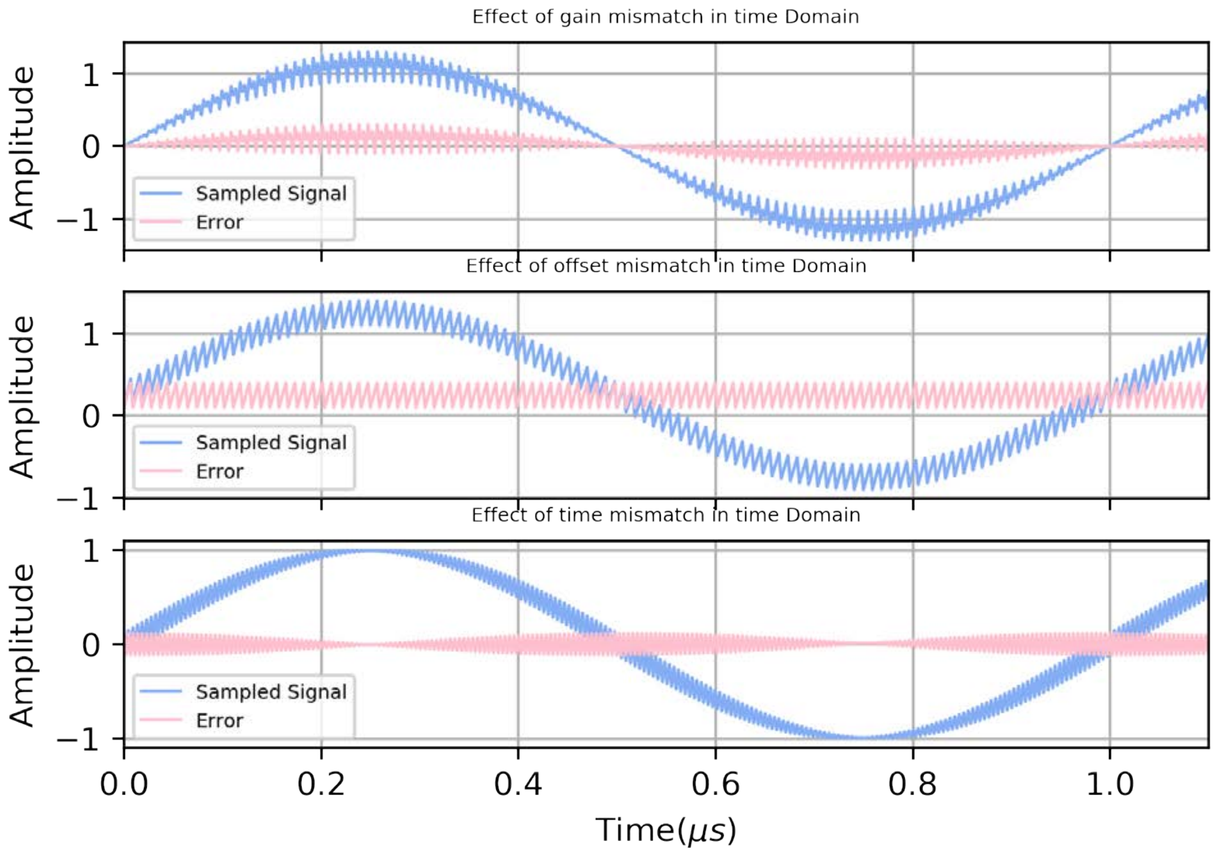

Figure 2.9: Effects of errors in time domain.

Error sources in time-interleaved ADCs

Time-interleaved ADCs suffer from mismatches among different channels such as offset, gain and clock mismatch. This part analyses the impact of those effects in time and frequency domain [16]. The time-interleaved ADC is modeled in Python script that includes different non-idealities.

Gain mismatch: In Figure 2.8, the gain mismatch is annotated as different gain G_i. Gain mismatch exhibits itself as the slopes difference in different sub-ADC's transfer curve. It comes from different sources, such as the sampling process and reference voltage change. The first plot in Figure 2.9 shows the time domain waveform illustrating the influence of gain mismatch and the frequency response is shown in Figure 2.10. For an N-way time-interleaved

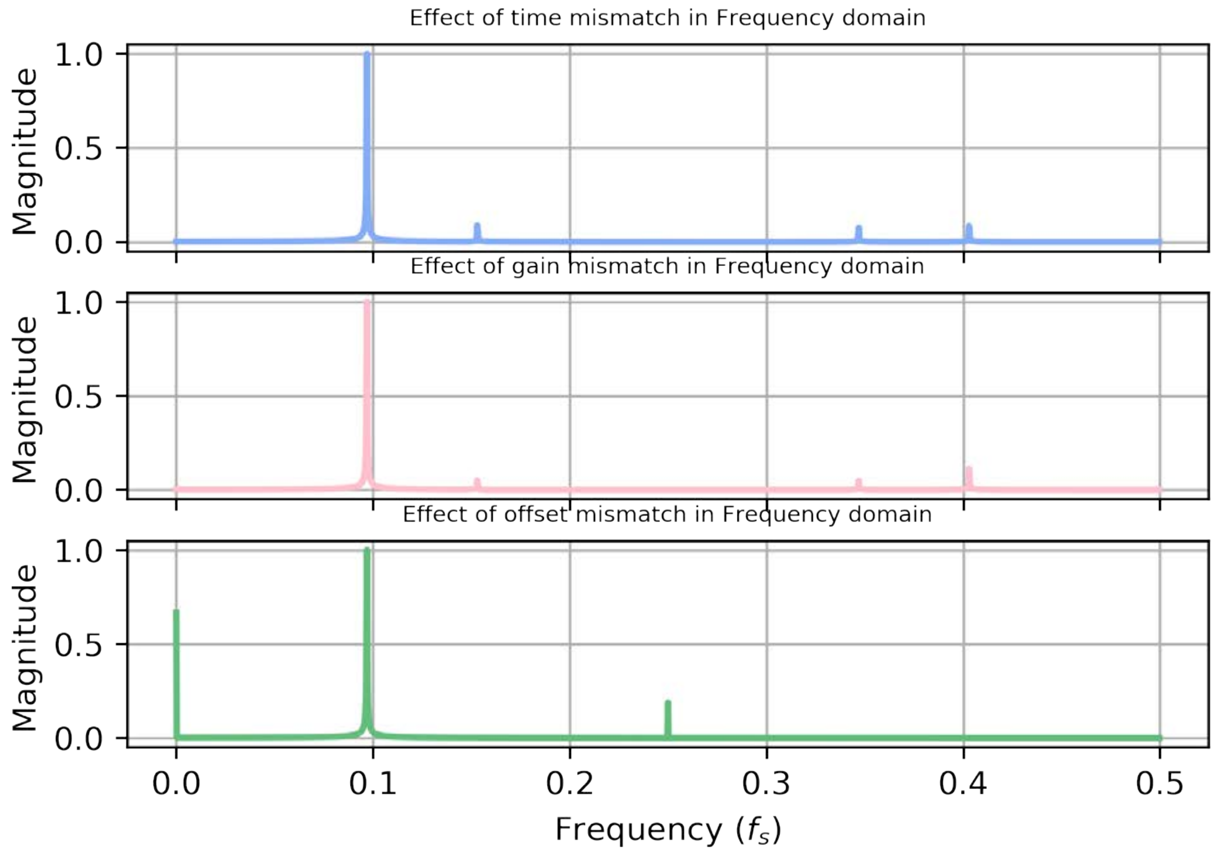

Figure 2.10: Effect of different mismatch in frequency domain.

ADC, the error in frequency domain happens repetitively every N/f_s and the error amplitude is modulated by the input signal. Therefore, this effect exhibits as an amplitude-modulated noise at frequency peaks at

$$f_{gain,noise} = \pm f_{in} \pm \frac{k}{N} f_s (k = 1, 2, ..., N) \tag{2.17}$$

The gain mismatch degrades the SNR and the extent to which performance is influenced also depends on the amplitude of the input signal.

Offset mismatch: Offset sources in the signal chain like the offset of the comparator can be moved forward to input as an input offset for ADC like $V_{os,i}$ shown in Figure 2.8. The time-domain waveform shows a repetitive error that happens every N/f_s. The average offset in each channel generates a DC component. The noise frequency response in Figure

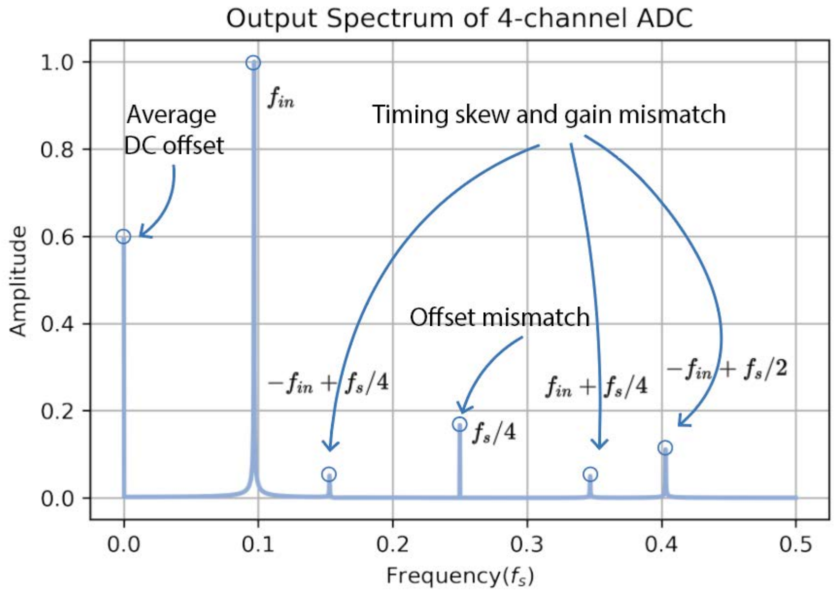

Figure 2.11: Output spectrum of an 4-channel ADC influenced by channel mismatches.

2.10 peaks at

$$f_{offset,noise} = \frac{k}{N} f_s (k = 1, 2, ..., N) \tag{2.18}$$

Timing mismatch: The mismatch caused by different sampling edges for each ADC is composed of both clock skew (systematic error) and clock jitter (random error), it is shown in Figure 2.8. This effect causes the largest error when the slope of the input signal is steepest. Therefore in Figure 2.9 the phase of the error waveform is shifted by $\pi/2$. It is essentially a phase-modulated noise and the noise frequency peaks also locate at

$$f_{time,noise} = \pm f_{in} \pm \frac{k}{N} f_s (k = 1, 2, ..., N) \tag{2.19}$$

In the frequency domain, this error will overlap with error caused by gain mismatch and the phase of the error is shifted.

Bandwidth mismatch: Different sampling bandwidth for different channels will also cause a error. The switches can be modeled by the R-C model and because the frequency response of the gain and phase varies during sampling are different for each channel.

$$V_{sampled} = G_i A \cos\left(2\pi f_{\text{in}} \frac{N}{f_s} + \theta_i\right) \tag{2.20}$$

Where G_i and θ_i are different different gain and phase shift caused by sampling bandwidth mismatch. It is both amplitude- and phase-modulated noise and that will show its effect similar to gain and time mismatch. The noise frequency peaks locate at

$$f_{BW,noise} = \pm f_{in} \pm \frac{k}{N} f_s (k = 1, 2, ..., N) \tag{2.21}$$

To summarize, the frequency response of a 4-channel time-interleaved ADC model is shown in Figure 2.11. Different frequency components are annotated in the plot. The gain and timing skew mismatch effects are added together. It is clear that those non-ideal effects need to be calibrated, otherwise they will greatly degrade the performance of time-interleaved ADCs.

Chapter 3

Generator-Based Design

3.1 Design-based design methodologies

The performance of circuit in advanced technology nodes relies on device characterization, considering the post-layout effects. Especially in high-speed circuit design, the layout quality tends to influence the speed significantly. Also, device matching needs to be carefully simulated with the actual layout parasitic. On the other hand, porting circuit design to different technologies is time-consuming, even if the changes are as simple as a metal stacks will require completely re-design. Therefore, generator-based design methodology can help reduce the design cost. It improves designers' ability to explore design space and make it possible to realize more performance optimization.

A typical generator-based design flow is shown in Figure 3.1. It starts from a design script which handles circuit specifications and translates them into circuit chosen by designer, device sizing and layout strategies. Those information are organized in a structural way that can be taken by some script-based generators to implement actual schematic, layout and testbench. A schematic generator will take a pre-defined template and map the input parameters to actual sizing of each components inside. And layout generator needs to follow the scripted layout strategies that are able to handle layout in different situations. The input to layout generators are super-set of schematic input that include other parameters like wire space and width. Designers need to construct generator scripts in a way that they are DRC/LVS clean for a reasonable combinations of devices' size in different technology. Each circuit will have specific requirements. So designers also need to construct testbench generators that instantiate testbench for the generated circuit instance, run simulation and

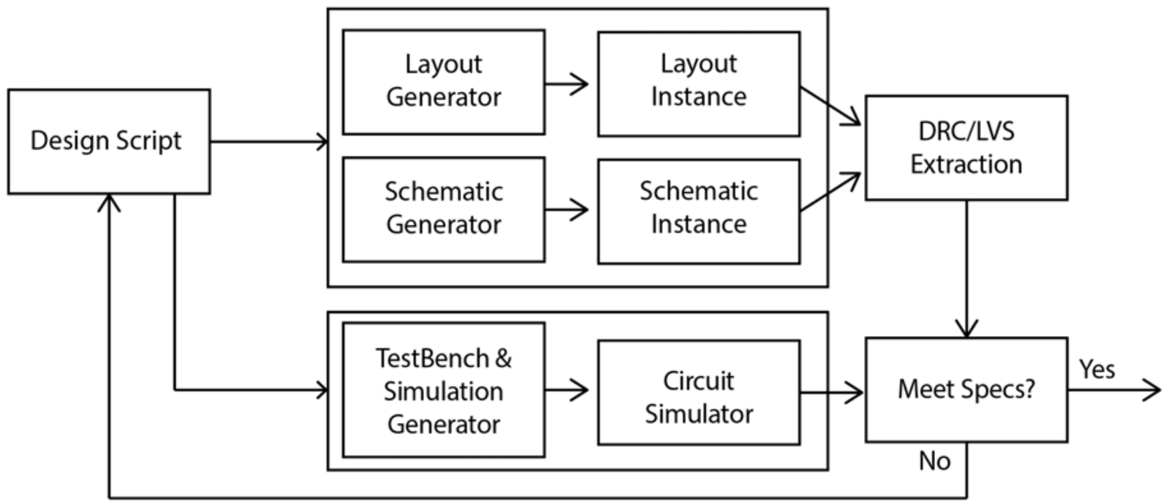

Figure 3.1: Diagram of generator-based design.

process data. If the simulation results show that the generated instance fails to meet specs, the design script should be able to handle the result returned from the simulator and make iteration based on information acquired. This agile approach makes sure that each time the circuit can be verified with post-layout effects.

The time-interleaved SAR ADC generator in this work uses Berkeley Analog Generator (BAG) [17] for layout and schematic generation. The rest of this chapter will introduce implementations of schematic and layout generator.

3.2 LAYGO layout generator

Introduction to LAYGO

The ADC generator uses LAYGO (LAYout with Gridded Objects) layout generation engine to automatically generate layout based on given parameters [18]. LAYGO is one of the layout generator engines in BAG that uses hand-crafted primitives and build layout based on that.

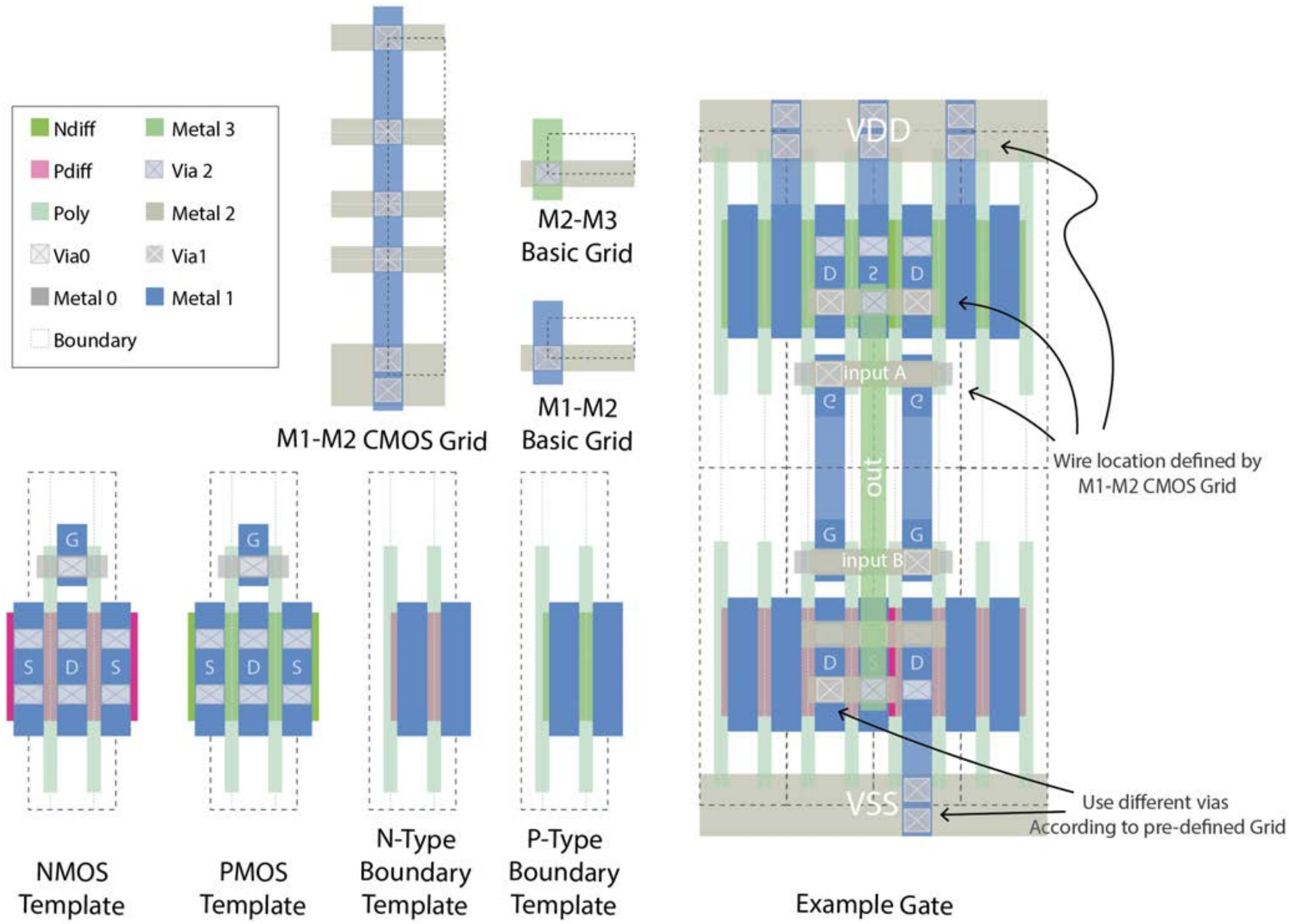

Figure 3.2: Example of LAYGO layout generation flow.

To make a process portable layout generator, LAYGO adopts the approach similar to the Lego block. The layout methodology of LAYGO is shown in Figure 3.2. It handles complex design rules by hand-made primitives and pre-defined routing grids. The unit blocks such as different unit transistors are equivalent to the Lego blocks, and the routing grid is the Lego bump. Imagine what happens if the size of Lego blocks become smaller, as long as they are still assembled according to smaller Lego bumps, we will still get the same result except the difference in dimension.

Similarly, different templates are constructed in different technology. They are assembled by the same script to generate layout. In advanced technology nodes, DRC rules becomes increasingly complex, but as long as the pre-defined unit blocks can capture different rules, the generated layout will still be DRC clean. Generally speaking, the most difficult front-end design rules are captured by different categories of hand-crafted cells. Going up to higher-level the metal patterns are wired up following pre-defined routing grid with specific spacing, width and via types. This approach is similar to digital circuit design that uses standard cells. But different device types are free to choose as long as corresponding unit block templates are implemented.

Primitives and layout example

In Figure 3.2, the left bottom side shows NMOS and PMOS templates with two fingers and pre-defined width. Inside these templates, one unit transistor is implemented with a bounding box which conveniently makes it easy to put in an array. The pins' name G, D, S stand for gate, drain, and source respectively, they are defined in coordinates that compatible with corresponding routing grid (M1-M2 CMOS grid in this case). There are other types of transistor templates available in order to handle different situations. For example, one finger transistors with gate connections at left/right side are used to implement minimum logic gates. When writing layout generators in python script, transistor templates are placed in an array to implement different device size. It is guaranteed that there will be no DRC issue in the middle. As for the boundary rules at two sides, it should be handled by N-type and P-type boundary templates. For example, it left enough space at sides to make sure it will not conflict with other primitives. Similar to transistor templates, there are also various types of boundary cells can handle different situation.

Several routing grid examples are shown in the top middle of Figure 3.2. Routing grid templates are less stringent as long as they meet metal width/space and via requirements that specified by design rules. So designers have the flexibility that define different routing grids for different purpose of layout. The M1-M2 CMOS grid is an example of an unevenly spaced grid. In this case, the grid is only valid when putting complementary transistors with their gates face each other. In this example, two routing tracks are used for source/drain routing, one routing track is used to connect gates and there is one more track left between gates of two transistors. Supply routing uses wider metal and double via. The M2-M3 basic grid and M1-M2 basic grid in the figure are example of evenly spacing grid. They can be used when multiple rows of same type of transistors are placed together.

The right side in the figures shows an example of NAND gate using the templates ex-

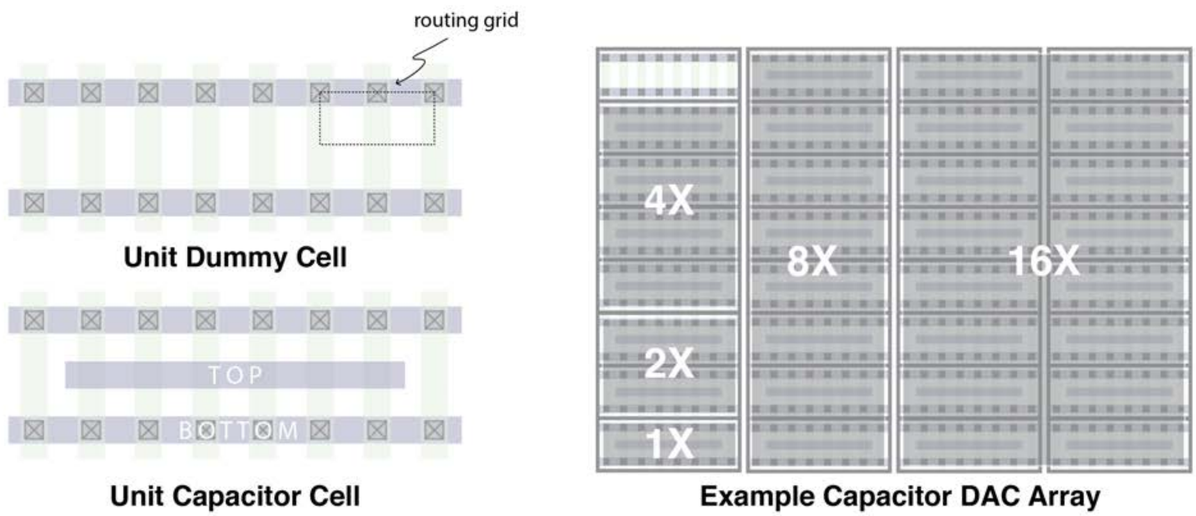

Figure 3.3: Implementation of capacitor DAC generator in LAYGO.

plained above. There are two transistor templates placed in the middle use CMOS grid. And boundary templates are aligned at two sides, making sure this layout can be tiled with other gates without violating design rules. Inputs A and B pin are placed at Metal 2, output signal is connected to Metal 3. All these connections use pre-defined CMOS routing grid, meaning that the absolute coordinate is calculated from template primitives.

Besides transistors, other passive devices such as capacitor, diode and resistor are also supported in LAYGO. A capacitor DAC layout example is shown in Figure 3.3. Unit capacitor cells and dummy cells are implemented manually. The size of unit block and the location of pins also need to be compatible with the transistor routing grid in order to be integrated with transistor in higher level layout. This example shows a simple 5-bit capacitor DAC on the right side of Figure 3.3.

The example above shows that building device primitives in LAYGO relies on correct size and pin location. Specifically, a device primitive needs to have quantized dimension. In our implementation, a `PlacementGrid` is used as the minimum grid for primitive block. And all the pins in primitive blocks should compatible with at least one routing grid defined in LAYGO. Also, LAYGO can take black-box and integrate that into layout in a similar way. As long as the boundary of block handle the design rules properly and pins of block are on-grid, it can be very flexible in specific implementation.

Example code:

After completing the templates library, a python script is used to place templates and generate layout. And all the implementations of layout design such as the floorplaning, sizing adjustment and routing need to be coded. Some example commands are listed below to show how generators are implemented.

```python
# Templates placement:
# (x0, y0), (x1, y1) are the origin points insts are placed
inst0 = laygen.relplace(cellname='cellName0',
                        gridname='gridName',
                        xy=['x0', 'y0'])
inst1 = laygen.relplace(cellname='cellName1',
                        gridname='gridName',
                        xy=['x1', 'y1'])
# Signal Routing:
# connect from one inst0's pin to one inst1's pin
laygen.route(gridname0='gridName0',
             refobj0=inst0.pins['pinName0'],
             gridname1='gridName1',
             refobj1=inst1.pins['pinName1'])

# Supply Routing:
# connect sources of inst to vdd/vss
for devName in [inst0, inst1]:
    for pinName in ['S0', 'S1']:
        laygen.route(gridname0='gridName',
                     refobj0=devName.pins[pinName],
                     refobj1=dev.bottom,
                     direction='y', via1=[0, 0])
# power and groud rails
rvdd = laygen.route(gridname0='gridName',
                    refobj0=inst0.bottom_left,
```

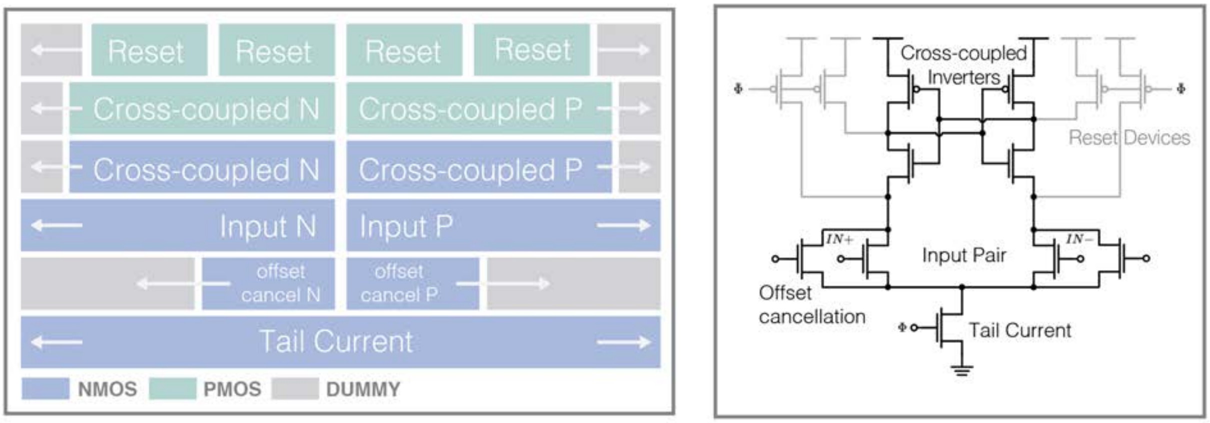

Figure 3.4: Example of strong arm comparator generator floorplan.

```
                          refobj1=inst0.bottom_right)
rvss = laygen.route(gridname0='gridName',
                          refobj0=inst1.bottom_left,
                          refobj1=inst1.bottom_right)

# Export pin
laygen.pin(name='pinName', gridname='gridName', refobj=inst0)
```

The typical layout design flow is very similar to the manual layout flow that requires a floorplan first. And device arrangement and routing need to be considered carefully by designers. However, the difference comes from the fact that a good layout generator generates DRC- and LVS-clean layout with many different input parameters. The example in Figure 3.4 shows some considerations when making a strong-arm layout generator. The arrows show the direction in which transistors expand. Each row is assigned to put transistors with a specific function. In this way, the size can be easily adjusted. For example when changing the size of input pair, layout can be easily extended toward two sides. Also, the schematic shows an offset cancellation pair to adjust the input-referred offset of the comparator. It can be implemented as an option in the layout generator. Because those transistors are put in a different row, the option can be turned off when not needed.

This example shows a very simple case but highlights some points when implementing generators. First, since the generator itself doesn't have any build-in algorithm for auto-routing and placement. More complex layout strategies always need more algorithms implemented in script. In this case, the offset cancellation transistor should be very small for resolution consideration, but they are put in a separate row, which is definitely an area inefficient strategy. Those transistors can be inserted into other rows but that might make sizing adjustment harder. Second, this floorplan assumes the transistor can be infinitely expanded to two sides. But it is very often that the layout has a width limitation or aspect-ratio requirement. If so, the generator should be coded in a way that can fold transistors into multiple rows when size exceed certain value. Third, when drawing layout it is very often the case that some special wiring and placement are implemented. But in a generator-based layout design, the designer prefers to some repetitive and regular items so it will usually lead to different layout strategies when compared with manual layout. From this example. Therefore, it is important to design the circuit in an efficient way that can utilize a reliable generator to make iterations faster.

3.3 Schematic generator

The schematic generation is relatively simple. First, the designer creates schematic templates in Virtuoso™ by using instances from generator primitive library. The schematic primitive library includes some wrappers that take the input of generator to set the parameters of each device. Then the templates are imported from Virtuoso™ library to Python. Base on the imported script, the designer can implement schematic design method in Python.

Some example codes for implementing schematic generators are listed below. The instances in the templates can be accessed using `self.instances[instance name]` and the generator configures instances using the `design` method. Schematic generators also offer the flexibility that designers can make changes based on schematic templates. Instances can be re-connected, arrayed or deleted. Dummy transistors can automatically be generated from the information returned by layout generator. Since different number of dummies will be added in layout to ensure matching and alignment, the connection and properties of dummies are calculated by the layout generator when actual calculating the layout.

The testbench generator is similar to schematic generator. Testbenches are implemented by copying and modifying testbench templates. The simulations are configured in Virtuoso™

, simulation parameters can be set by BAG. Simulation results are returned after simulation finish, so all the data can be processed using BAG.

Example code:

```
# set transistor parameters
self.instances['XP'].design(w=wp, l=lch, intent=intentp, nf=fg_load)
self.instances['XN'].design(w=wn, l=lch, intent=intentn, nf=fg_amp)

# handle dummy transistors
self.design_dummy_transistors(dum_info, 'XDUM', 'VDD', 'VSS')

# some other functions
Change pins name: rename_pin()
Remove instance: delete_instance()
Replace instance templates: replace_instance_master()
Reconnect instance terms: reconnect_instance_terminal()
Create an array of instance: array_instance()
```

Chapter 4

ADC Generator Implementations

4.1 Overview

This chapter describes the implementation of the time-interleaved SAR ADC generator based on the design methodology that introduced above and using LAYGO layout generation engine to for layout generation. And the implementation of this generator in Intel 22nm FFL is presented.

Figure 4.1 shows the overview of generator architecture. This generator can provide 4-10

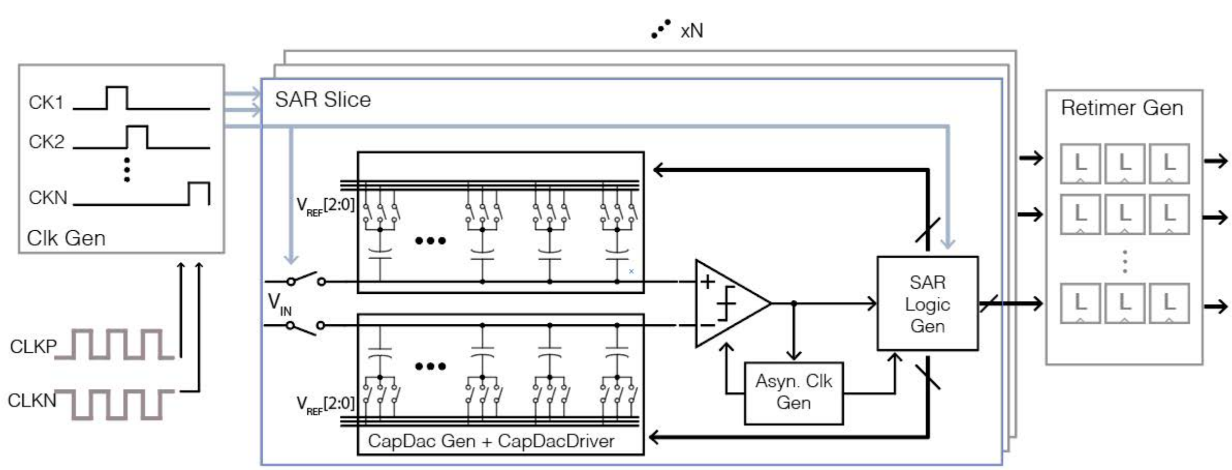

Figure 4.1: Architecture of time-interleaved ADC generator.

bits resolution and use time-interleaving architecture to achieve higher sampling rate. It is composed of four main blocks: multi-phase clock generation, main SAR slice array, retimer at the output, and the bias circuit(not shown here, which provides V_{REF} for capacitor DAC array) The blocks shown here are explained as the following:

- **Clock Generation:** It takes a half-rate differential clock input and uses a chain of delay cells to generate N different clock signal phases in parallel that trigger each SAR slice to sample input signal.
- **ADC Core:** The main part of SAR ADC that takes input signal, reference voltage as well as the clock and outputs digital result. It consists of a capacitor DAC, comparator, SAR logic and asynchronous clock generator. A fixed amount of time is allocated for sampling. And it works in an self-timed, asynchronous way to convert sampled signal at frequency $f_{s,slice} = f_s/N$.
- **Retimer:** Because each ADC slice is timed to different clock phase, a retimer block takes digital output from each slice is necessary. It aligns output from each slice to the same clock phase such that the ADC can transfer data to the digital interface.

The rest of part of this chapter will introduce several key generators in this ADC generator. Some design details, floorplan and generator options will be explained.

4.2 Capacitive DAC

The performance of SAR ADC is greatly influenced by the capacitor ADC. A simplified diagram of conversion steps is shown in Figure 4.2. The capacitors are radix-α weighted, each capacitor is α times as larger as the previous one, where α is the radix of DAC.

$$\begin{aligned} C_0 &= C_u \\ C_i &= \alpha C_{i-1} \end{aligned} \tag{4.1}$$

The total capacitor is $C_{tot} = \Sigma_i^{n-1}\alpha^i C_u + C_u$ For a 4-bit conversion, first the input signal is sampled on the capacitor DAC. In the consecutive steps, switches connect to either positive (V_{REFP}) or negative (V_{REFN}) reference voltage depending on the comparator's decision. When a bit is connected to positive reference voltage the compared reference voltage changes by $\frac{C_i}{C_{tot}}(V_{REFP} - V_{REFN})$.

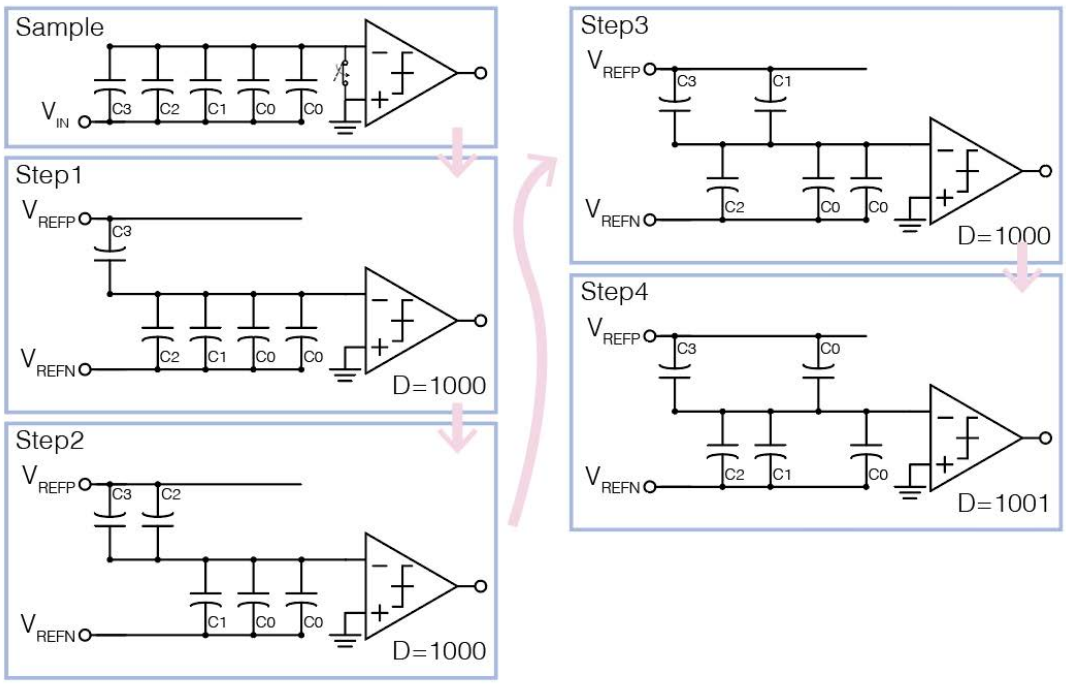

Figure 4.2: Example of 4-bit capacitor DAC switching.

The transfer function of ADC is 2^N steps from V_{REFN} to V_{REFP} for a N-bit radix-2 capacitor DAC, all the step sizes are uniform. In reality, each capacitor does not match perfectly and mismatch makes the transfer curve deviate from the ideal one. The random variance of single unit matching σ_C is inversely proportional to unit capacitor area $\sqrt{A_{unit}}$. As the area increases, the variance can be reduced. Also, from a thermal noise perspective, a larger sampling capacitor makes the total thermal noise $\frac{k_B T}{C}$ smaller, where k_B is the Boltzmann constant and T is the absolute temperature. However, this approach dramatically increases the total power consumed by charging capacitor and increase the area as well.

When capacitor DAC that uses radix not equals to two, there are different effects on the transfer function for ADC and DAC. Figure 4.3 shows transfer functions of DAC and ADC for radix = 2.2 and radix = 1.8 separately. The left plot shows an extend horizontal level which is interpreted as missing decision level and the right plot shows missing code. The former effect can only be adjusted by reducing analog input while the last one can be adjusted digitally by assigning different weights to each capacitor. From [19], it is sufficient

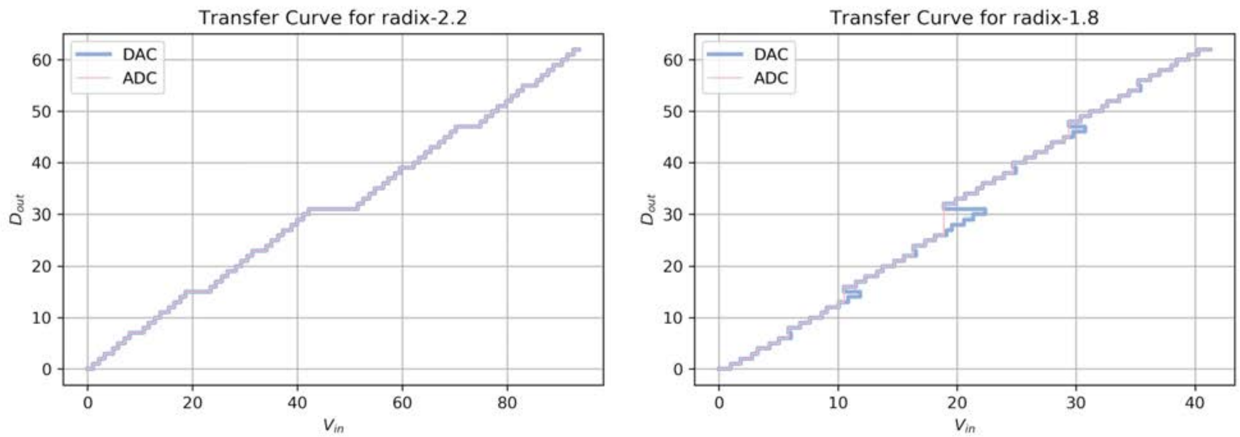

Figure 4.3: Effect of different radix on ADC and DAC transfer curve.

that the transfer function can be adjusted to be free of missing decision when

$$C_i < C_0 + \Sigma_{k=0}^{i-1} C_k = C_0 + \Sigma_0^{i-1} \alpha^k C_0 \tag{4.2}$$

The capacitor DAC generator is implemented in the way that the number of bits and radix can both be adjusted. The conceptual diagram of the capacitor DAC generator is shown in Figure 4.4. Unit capacitor and capacitor dummy primitives are hand-drawn. The schematic template defines the connections and ports. With different input parameters listed in the Figure 4.4, both schematic and layout of the capacitor are generated.

4.3 Comparator

The comparator generator implements both traditional strong-arm latch comparator and the dual strong-arm latch comparator [20]. The additional second regenerative latch in dual strong-arm latch helps reduce both offset sensitivity and offset while maintaining comparable performance as the conventional one. The diagram of comparator generator is shown in Figure 4.5. Two different schematic templates are necessary since these two circuits have enough differences.

The generator selects different topology by changing option `doubleSA`. All the other options are setting the sizing of each transistor. From the input parameters, the layout is

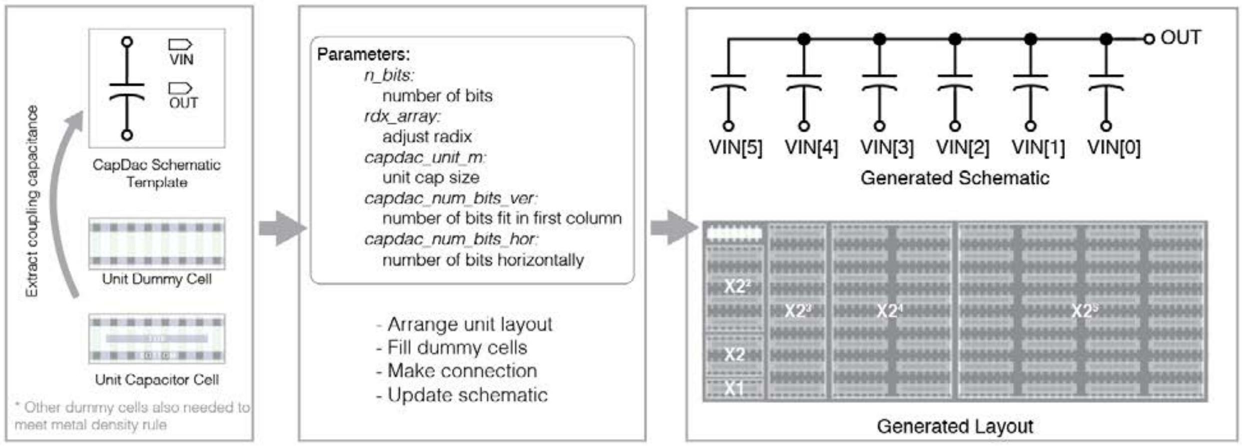

Figure 4.4: Capacitor DAC generator.

generated used the selected topology and dummy transistor information is calculated from the generated layout. The schematic generator map device sizing from the input parameter to the schematic template, all the dummies are also added from layout information as explained in previous chapter.

The floorplan of the conventional strong-arm comparator is shown in the figure as an example. The input pair and tail current are put in two rows. Substrate connections are put at the top and bottom of input and tail current transistors. The cross-coupled transistors are placed at the top, reset devices and cross-coupled PMOS are placed in the same row. Each row has dummy transistors to fill in space and also help the matching. Space or boundary cell is put at two sides of the comparator core. It helps with alignment in each slice because the slice width is fixed by either capacitor DAC or comparator width.

4.4 SAR Logic

The SAR logic is shown in Figure 4.6. The comparator is triggered by enable signal from clock generation block inside each slice. After the comparator output signal is resolved, the NAND gate triggers the clock generation block. Inside the clock generation block, there are actually two paths of delay in parallel and one of them is selected to trigger clock edge generation by the a MUX gate. Another loop of the circuit is from the output of clock

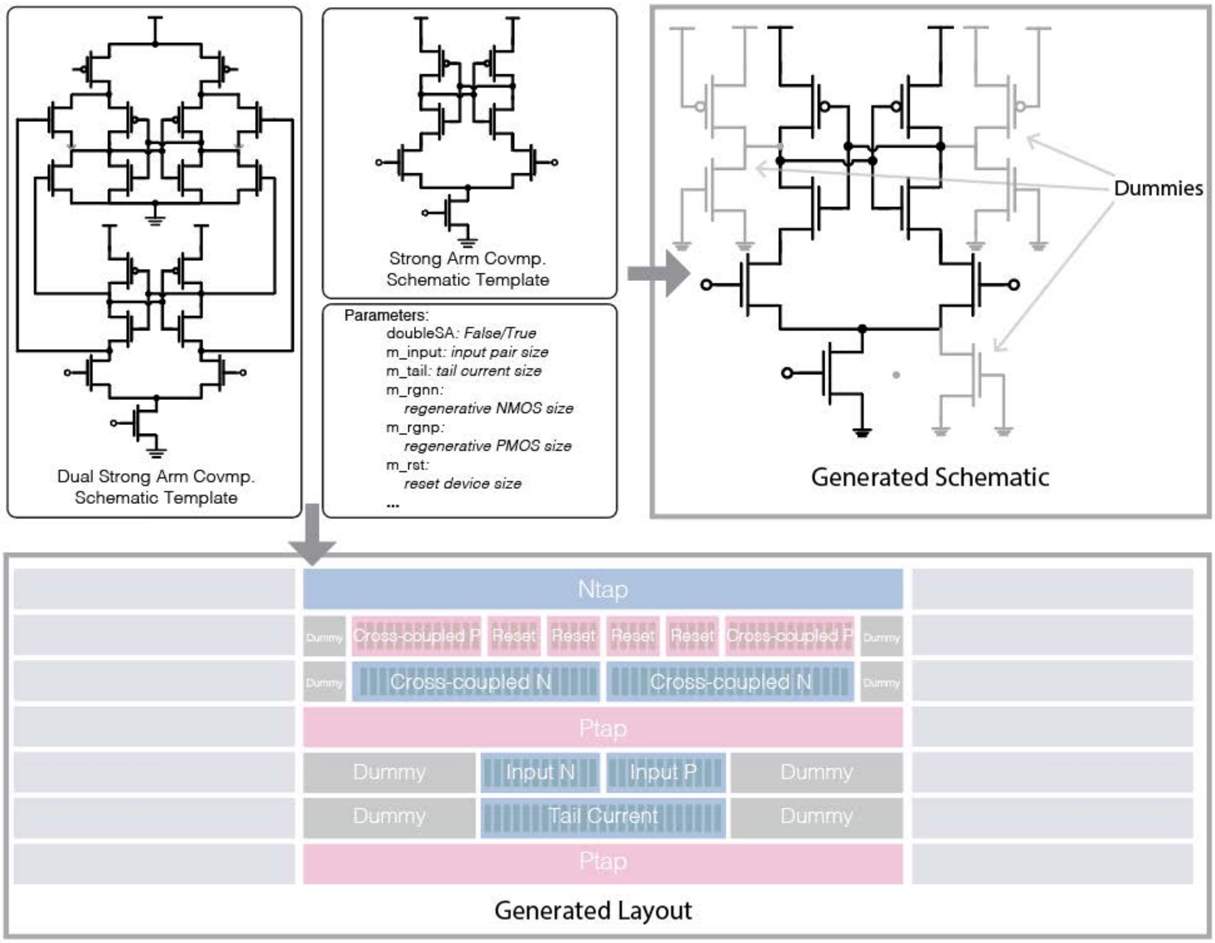

Figure 4.5: Strong arm comparator generator.

generation to shift register, shift register set the current bit that needs to be decided and change the connection of capacitor DAC. The comparator should only be triggered after the voltage on the capacitor DAC is settled. Therefore it is important to adjust the delay of two loops to match. The purpose of different delays in clock generation block is shown in Figure 4.7. The comparator output at the 3^{rd} iteration corresponds to the blue line at the right side. It takes T_1 to resolve the result. And the last iteration takes T_2 for the comparator to resolve and shown as the pink line shows. For a small difference at the input, it takes a longer time for the comparator to make decisions. Although later iterations take a shorter time for capacitor DAC to settle, the later iterations tend to take a longer time than the

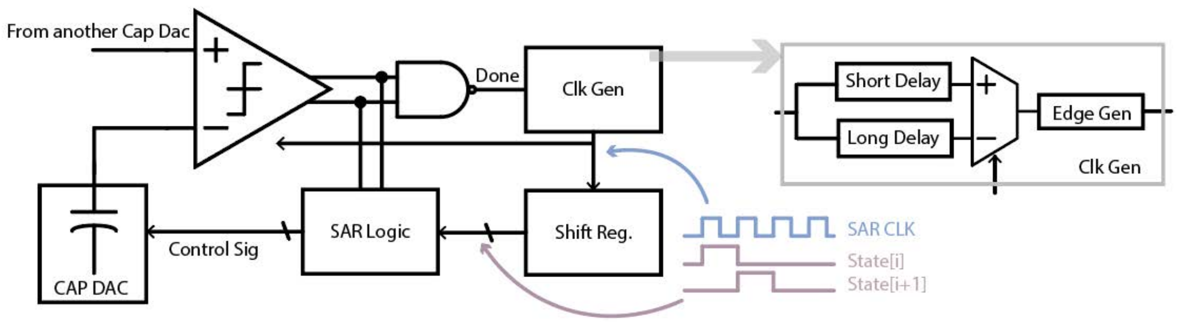

Figure 4.6: Diagram of SAR logic generator.

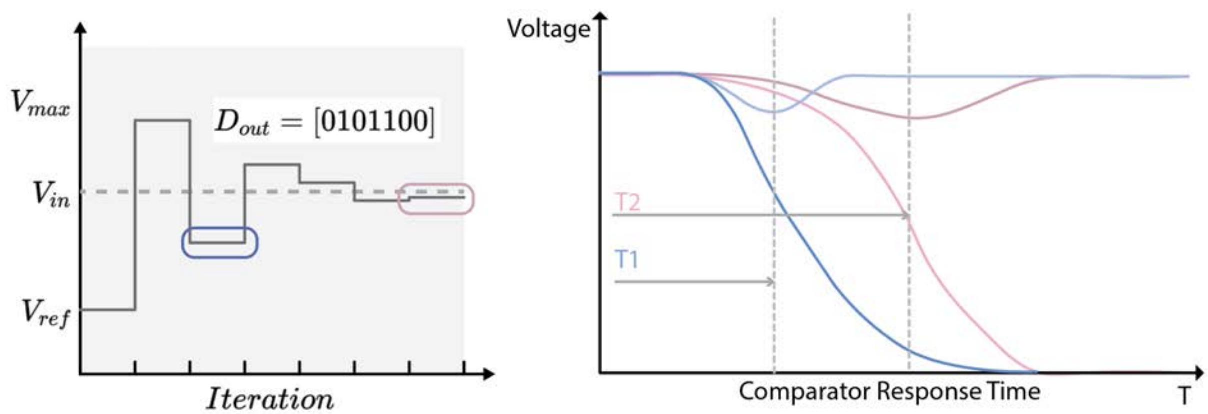

Figure 4.7: Illustration of Comparator Response Time at Different Bit

first several bits because the comparator needs much more time to generate output signal that trigger following gates. So the delay in clock generation block can be reduced in the first several bits. And it helps maximize the speed of asynchronous operation.

As for the layout generation, various types of logic gates are implemented first. Similar to digital circuit, the logic layout generators use these gates as standard cells to implement different types of logic circuit.

4.5 Top Level Generator and Implementation in Intel22 FFL

At the top level, the generator compares the dimension of sub-blocks, makes arrangement and connects different blocks. The example code below shows LAYGO generator that works with hierarchical design. The information about generated instances such as pin locations and block size are stored in a YAML file. When those instances are used in higher-level, the generator loads the YAML use those those data to perform necessary calculation. After that blocks are placed and connected.

Example code:

```
# Get size of block0 and place block0 at origin
block_size0 = laygen.get_tempalte(cellname0, libname0).size
block0 = laygen.relplace(instance_name0, template_name0, gridname,
         xy=(0, 0), template_lib0)
# Put block1 next to block0 use block_size0
block1 = laygen.relplace(instance_name0, template_name0, gridname,
         xy=(block_size0[0], 0), template_lib0)

# Get pins from instance
pina = laygen.get_inst_pin_xy(block0, pin0, gridname1)
pinb = laygen.get_inst_pin_xy(block1, pin1, gridname1)

# Get middle location of two pins
mid_y = (pina[1]+pinb[1])//2
# connect two pins
laygen.route_vhv(layerv0=layer1, layerh=layer2,  layerv1=layer1,
                 xy0=inp_xy_m5, xy1=inp_xy_m7, track_y=mid_y,
                 gridname0=gridname1, gridname1=gridname2)
```

The top-level generator of ADC assembles the ADC slices, sampling circuit, clock generation block as well as the retimer. One feature of this top-level generator is that the order of ADC slices is free to change for different applications. The time diagram of clock phases is shown in Figure 4.8. It is related to clock generation block. The clock generation is

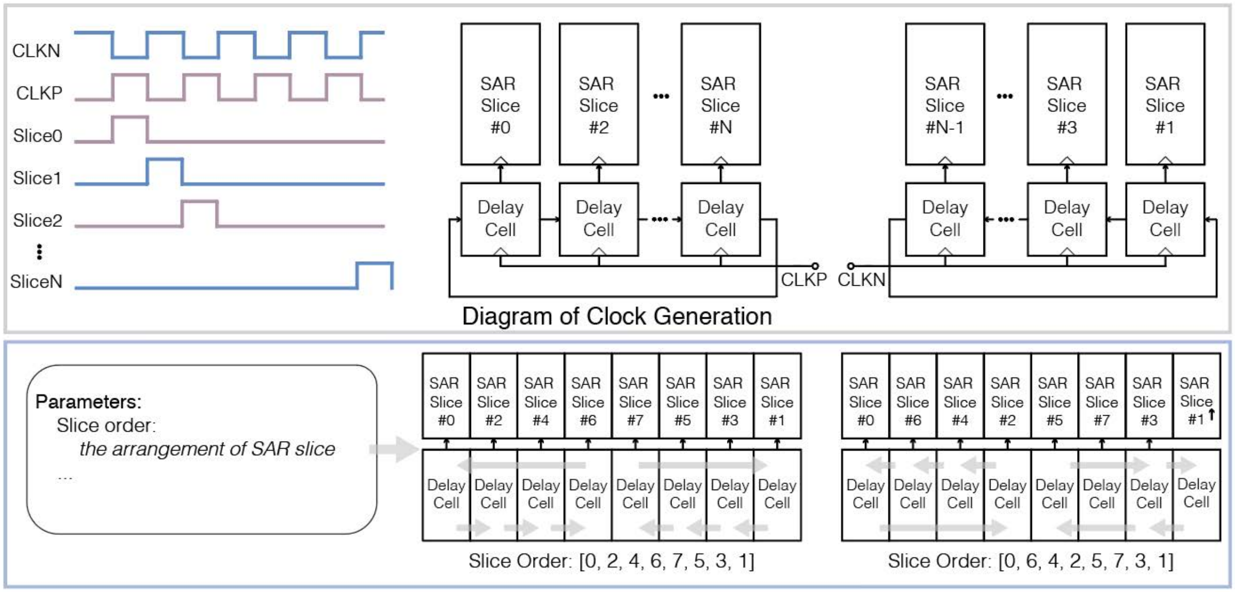

Figure 4.8: Slice order option in top-level generator.

two chains of delay elements driven by differential clock signals. The correct clock phase is guaranteed by the uniform delay from each delay cells. The delay of each cell is adjusted by capacitor loading at each delay cell's output. Therefore, when the slice order needs to be adjusted, as long as the order of even and odd slices are put at two sides. and connected by two different delay chains, the slice order can be easily changed. Two examples are shown in the bottom diagram in Figure 4.8.

This generator has been implemented in Intel 22nm FFL with the updated options that improve the sampling rate by decoupling the crosstalk between channels. The steps of implementing this chip are shown in Figure 4.9. First, a 9-bit 16-way time-interleaved SAR ADC core including decoupling capacitor and voltage references are fully generated by the generator. It works at sampling rate 10GS/s and simulated SNDR is 37.6 dB at Nyquist frequency. In order to set the configuration bits and read out quantized result. A digital block includes memory as well as scan chain is manually integrated at the top level with generated ADC core. Because generator makes it convenient to quickly generate different designs. Six sub-chips with different configurations are generated and integrated in the same way. They have different sampler sizes, different sampling strategies and different radixes for capacitor DAC.

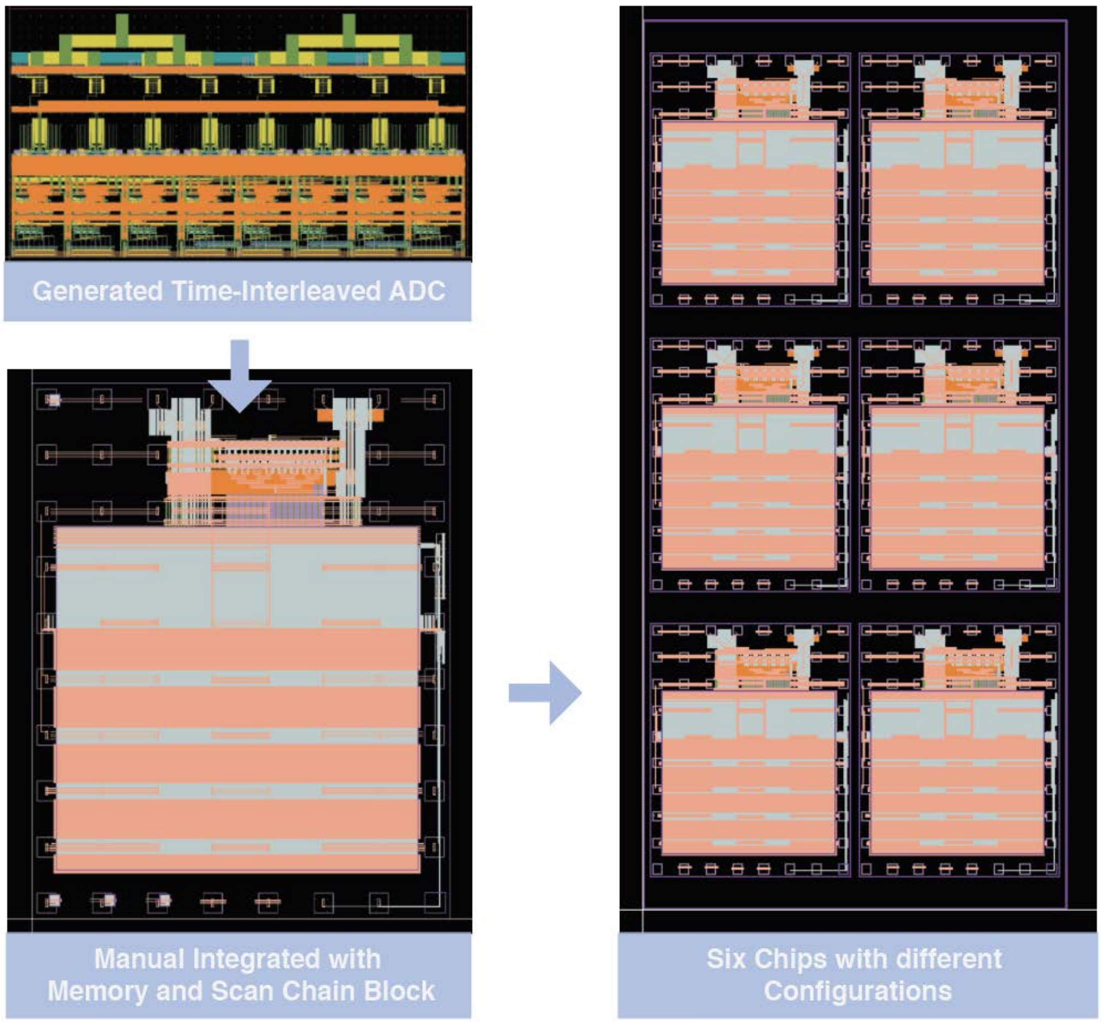

Figure 4.9: Chip integration flow.

Chapter 5

Conclusion and Future Work

The advantages of SAR ADC make it a popular ADC architecture in scaled CMOS process. Generator-based design methodology improves the efficiency of circuit design in advance technology nodes. The complex design rules are captured by generator and designers can quickly get feedback from post-layout effect. Circuit design using BAG makes it possible to reuse designs in different process technologies and generate designs with different specifications.

In this report, the working principle and design methodologies of high-speed time-interleaved SAR ADC are presented. The main goal is investigating design methodology and combined it with analog generators to enable design space exploration under different specifications. Also, the LAYGO layout generation engine is introduced. This report demonstrated the detail implementation of hand-made LAYGO primitives and the layout generation based on that. Example layouts and code are used to illustrate the usage of BAG. The prototype of time-interleaved SAR ADC generator is implemented in intel22 FFL with six different configurations. The testing boards have been fabricated and assembled. Figure 5.1 shows the testing board design for the chip. and the future work will be finish the measurement of the prototype chip . Also, there are several improvements worth to be considered in the future:

1. **Complete design script for SAR ADC**. While the layout and schematic generators have been implemented, main design procedure is still similar to traditional design methodology that relies on the designer's interpretation of the simulation result. A design script that captures key performances of circuits can be implemented to close the loop of

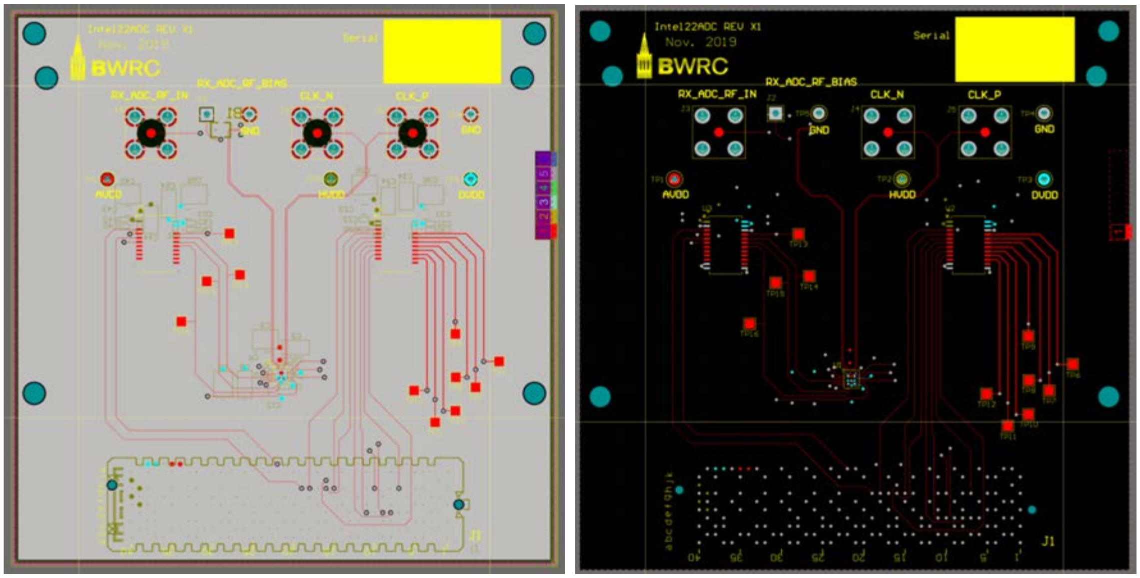

Figure 5.1: Testing board for prototype chip.

design automation.

2. **Improve the speed of SAR ADC generator.** Currently the single slice of ADC is optimized for a higher sampling rate. And multiple slices are necessary when the sampling rate goes beyond the capability of the single-channel design. However, the number of slices cannot go infinitely large because of non-idealities in channel mismatches. Also, increasing time-interleaving ratio brings the challenges to the sampling network. Therefore, techniques that can provide a higher sampling rate for this ADC generator are worth investigating.

3. **Improve the resolution of ADC generator.** The resolution of this SAR ADC generator is limited. The resolution of the ADC can also be improved by sampling circuit with higher accuracy, better element matching and new calibration method. Adding a second stage after SAR ADC will also be an effective way to improve resolution. Among different ADC architectures, ring-oscillator based voltage-controlled oscillator ADCs and ring amplifiers have demonstrated some highly competitive design metrics in scaled technologies. By combining and selectively enabling these two techniques, an adaptive-resolution architecture can be developed and will be made available for broader use.

References

[1] B. Murmann. *ADC Performance Survey 1997-2019,[Online].http://web.stanford.edu/ murmann/adcsurvey.html*. URL: `http://web.stanford.edu/~murmann/adcsurvey.html`.

[2] L. Kull et al. "A 24-to-72GS/s 8b time-interleaved SAR ADC with 2.0-to-3.3pJ/conversion and ¿30dB SNDR at nyquist in 14nm CMOS FinFET". In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, pp. 358–360. DOI: `10.1109/ISSCC.2018.8310332`.

[3] J. Cao et al. "29.2 A transmitter and receiver for 100Gb/s coherent networks with integrated 464GS/s 8b ADCs and DACs in 20nm CMOS". In: *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, pp. 484–485. DOI: `10.1109/ISSCC.2017.7870472`.

[4] B. Murmann. "The successive approximation register ADC: a versatile building block for ultra-low- power to ultra-high-speed applications". In: *IEEE Communications Magazine* 54.4 (2016), pp. 78–83. DOI: `10.1109/MCOM.2016.7452270`.

[5] P. Harpe et al. "21.2 A 3nW signal-acquisition IC integrating an amplifier with 2.1 NEF and a 1.5fJ/conv-step ADC". In: *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*. 2015, pp. 1–3. DOI: `10.1109/ISSCC.2015.7063086`.

[6] M. Konijnenburg et al. "22.1 A 769W Battery-Powered Single-Chip SoC With BLE for Multi-Modal Vital Sign Health Patches". In: *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2019, pp. 360–362. DOI: `10.1109/ISSCC.2019.8662520`.

[7] C. Liu et al. "A 10b 100MS/s 1.13mW SAR ADC with binary-scaled error compensation". In: *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2010, pp. 386–387. DOI: `10.1109/ISSCC.2010.5433970`.

[8] C. Liu. “27.4 A 0.35mW 12b 100MS/s SAR-assisted digital slope ADC in 28nm CMOS”. In: *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. 2016, pp. 462–463. DOI: 10.1109/ISSCC.2016.7418107.

[9] L. Kull et al. “22.1 A 90GS/s 8b 667mW 64 interleaved SAR ADC in 32nm digital SOI CMOS”. In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2014, pp. 378–379. DOI: 10.1109/ISSCC.2014.6757477.

[10] B. Razavi. “Design Considerations for Interleaved ADCs”. In: *IEEE Journal of Solid-State Circuits* 48.8 (2013), pp. 1806–1817. DOI: 10.1109/JSSC.2013.2258814.

[11] B. Razavi. “The StrongARM Latch [A Circuit for All Seasons]”. In: *IEEE Solid-State Circuits Magazine* 7.2 (2015), pp. 12–17. ISSN: 1943-0590. DOI: 10.1109/MSSC.2015.2418155.

[12] A. Yu et al. “Understanding Metastability in SAR ADCs: Part II: Asynchronous”. In: *IEEE Solid-State Circuits Magazine* 11.3 (2019), pp. 16–32. ISSN: 1943-0590. DOI: 10.1109/MSSC.2019.2922890.

[13] J. P. Keane et al. “16.5 An 8GS/s time-interleaved SAR ADC with unresolved decision detection achieving 58dBFS noise and 4GHz bandwidth in 28nm CMOS”. In: *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, pp. 284–285. DOI: 10.1109/ISSCC.2017.7870372.

[14] S. M. Chen and R. W. Brodersen. “A 6-bit 600-MS/s 5.3-mW Asynchronous ADC in 0.13-μm CMOS”. In: *IEEE Journal of Solid-State Circuits* 41.12 (2006), pp. 2669–2680. DOI: 10.1109/JSSC.2006.884231.

[15] Wei Yu, Subhajit Sen, and B. H. Leung. “Distortion analysis of MOS track-and-hold sampling mixers using time-varying Volterra series”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 46.2 (1999), pp. 101–113. DOI: 10.1109/82.752910.

[16] N. Kurosawa et al. “Explicit formula for channel mismatch effects in time-interleaved ADC systems”. In: *Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference [Cat. No. 00CH37066]*. Vol. 2. 2000, 763–768 vol.2. DOI: 10.1109/IMTC.2000.848838.

[17] E. Chang et al. “BAG2: A process-portable framework for generator-based AMS circuit design”. In: *2018 IEEE Custom Integrated Circuits Conference (CICC)*. 2018, pp. 1–8. DOI: 10.1109/CICC.2018.8357061.

[18] J. Han et al. “A Generated 7GS/s 8b Time-Interleaved SAR ADC with 38.2dB SNDR at Nyquist in 16nm CMOS FinFET”. In: *2019 IEEE Custom Integrated Circuits Conference (CICC)*. 2019, pp. 1–4. DOI: `10.1109/CICC.2019.8780169`.

[19] Wenbo Liu and Yun Chiu. “An equalization-based adaptive digital background calibration technique for successive approximation analog-to-digital converters”. In: *2007 7th International Conference on ASIC*. 2007, pp. 289–292. DOI: `10.1109/ICASIC.2007.4415624`.

[20] A. Papadopoulou, V. Milovanovic, and B. Nikolic. “A low-voltage low-offset dual strong-arm latch comparator”. In: *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. 2017, pp. 281–284. DOI: `10.1109/ASSCC.2017.8240271`.