Department of Physics and Astronomy
University of Heidelberg

Master Thesis in Physics
submitted by

Philipp Wolfgang Dauer

born in Forchheim (Germany)

2022

Analog-to-digital conversion for mixed-signal computing:

Components for a successive-approximation ADC

This Master Thesis has been carried out by Philipp Wolfgang Dauer at the
Kirchhoff-Institute for Physics,
Heidelberg University
under the supervision of
Dr. Johannes Schemmel and Prof. Dr. Wolfram Pernice

Abstract

BrainScaleS-2 (BSS-2), a neuromorphic platform developed at Heidelberg University, successfully demonstrates a hybrid in-memory computing architecture. At the interface between the analog and digital domains, a multichannel, parallel analog-to-digital converter (ADC) enables accessing the rich analog neuron and synapse dynamics from the digital surrounding. This ADC will prospectively be replaced by a new successive-approximation ADC (SAR ADC) operating at Nyquist-rate. For this purpose, the present work presents the design process of a mixed-signal simulation framework, the digital SAR logic and the required comparator module including an adjustable (5 bit) capacitive reference generator. The comparator, based on a double-tail sense amplifier, requires 298 ps and 80 fJ for each decision in post-layout simulations at an input voltage difference of 1 mV. In its fast mode (7 bit), the SAR ADC achieves a sampling rate of 125 MS/s in pre-layout simulations with a maximum differential non-linearity (DNL) of 0.21 LSB and an energy consumption of 4.01 pJ per conversion. In its precise mode (8 bit), the ADC still achieves a sampling rate of 43.7 MS/s with a maximum DNL of 0.50 LSB and energy consumption of 4.01 pJ per conversion. Post-layout simulations of the entire ADC suggest the finalization of the prototype as well.

Zusammenfassung

BrainScaleS-2 (BSS-2), eine an der Universität Heidelberg entwickelte Plattform, demonstriert erfolgreich eine hybride In-Memory Rechenarchitektur. An der Schnittstelle zwischen der analogen und der digitalen Domaine ermöglicht ein vielkanaliger, paralleler Analog-digital-Wandler (ADC) das Auslesen der vielfältigen Dynamiken in den analogen Neuronen und Synapsen durch ihre digitale Peripherie. Dieser wird perspektivisch durch einen neuen, auf sukzessiver Approximation basierenden Nyquist-ADC (SAR ADC) ersetzt. Zu diesem Zweck beschreibt die vorliegende Arbeit die Entwicklung einer Mixed-Signal Simulationsumgebung, der digitalen SAR-Logik, sowie des benötigten Komparator Moduls einschließlich eines einstellbaren (5 bit) kapazitativen Referenzgenerators. Der auf einem zweigliedrigen Leseverstärker basierende Komparator benötigt in layoutbasierten Simulationen bei einer Eingangsspannungsdifferenz von 1 mV 298 ps und 80 fJ für eine Entscheidung. Im schnellen Modus (7 bit) erreicht der SAR ADC in Simulationen auf Transistorebene eine Abtastrate von 125 Mhz bei einer maximalen differentiellen Nichtlinearität (DNL) von 0,21 LSB und einem Energiebedarf von 2,47 pJ pro Wandlung. In seinem präzisen Modus (8 bit) erreicht er unter gleichen Bedingungen immer noch eine Abtastrate von 43,7 Mhz bei einer maximalen DNL von 0,50 LSB und einer Energieaufnahme von 4,01 pJ pro Wandlung. Auch die layoutbasierten Simulationen des ADCs legen die Erstellung eines Prototyps nahe.

Contents

Acronyms

ADC analog-to-digital converter.

AdEx adaptive exponential integrate-and-fire.

ANN artifical-neural-network.

ASIC application-specific integrated circuit.

BPTT backpropagation through time.

BSS-2 BrainScaleS-2.

CADC column-parallel analog-to-digital converter (ADC).

DAC digital-to-analog converter.

DNL differential non-linearity.

DPI direct programming interface.

DTSA double-tail sense amplifier.

DUT design under test.

FF flip-flop.

FPGA field-programmable gate array.

FSM finite state machine.

GPU graphics processing unit.

HDL hardware description language.

HICANN DLS high-input-count-analog-neural-network digital-lerning-system.

INL integral non-linearity.

LIF leaky integrate-and-fire.

LSB least significant bit.

MADC membrane ADC.

MOM metal-oxid-metal.

MOS metal-oxide-semiconductor.

MOSFET metal-oxide-semiconductor field-effect transistor.

MSB most significant bit.

NMOS N-type MOS.

NN neural network.

PEX paracitic extraction.

PLL phase-locked loop.

PMOS P-type MOS.

PPU plasticity processing unit.

RTL register transfer level.

SA sense amplifier.

SAFF sense amplifier flip-flop.

SAR successive-approximation register.

SIMD single instruction, multiple data.

SNN spiking-neural-network.

SRAM static random-access memory.

STA static timing analysis.

STDP spike-time-dependent plasticity.

UMV universal verification methodology.

VLSI very large scale integration.

1 Introduction

Analog-to-digital converters (ADCs) have become indispensable in many devices today. By converting an electrical current or a voltage into a series of bits, they make physical quantities accessible to the rich prospects of digital data processing. In a way they are the "eyes and ears of a digital system" (Pelgrom, 2017). Without ADCs, no smartphone would send a call, no digital camera would take a picture, no computer controlled robot would operate in a factory, and physicists would not be able to trace the trajectories of high-energy particles in modern experiments.

Another scope of application is analog or hybrid computing, where analog data must be transferred back into the digital domain. Digital computing has undisputed advantages such as great flexibility or error correction. However, real-world applications are often based on differential equations and matrix multiplications, which are costly to solve in the digital domain. This is where analog or hybrid computing excels by solving these directly by the nature of its circuits. One modern hybrid computer is the accelerated neuromorphic BrainScaleS-2 (BSS-2) system (Schemmel et al., 2020; Pehle et al., 2022). Neuromorphic systems implement biologically inspired neurons and synapses and enable the formation of neural networks (NNs). Such networks are also suitable for breaking with yet another classic approach: the von Neumann architecture. Contrary to von Neumann computers, where memory and computing unit are physically separated (Von Neumann, 1993), both are united in the synapses of BSS-2 (Friedmann, 2013). This so-called "in-memory computing" approach thus largely circumvents the von Neumann bottleneck of sequential data processing from which classical computers suffer.

For the next generation of neuromorphic chips in BSS-2, an improved and accelerated column-parallel ADC (CADC) operating at Nyquist-rate is required. While the current implementation of the on-chip CADC reaches sampling frequencies of below 2 MS/s at 8 bit resolution (Schreiber, 2021), a redesigned version aims for 125 MS/s at 7 bit resolution. This improvement should allow increasing the learning efficiency and inference speed in NNs. To be competitive with the current implementation, the new ADC should also provide a precise mode with 8 bit resolution at a sampling frequency of at least 10 MS/s. For this purpose, the new ADC will implement a dual-mode successive-approximation architecture.

A successive-approximation converter quantifies the magnitude of an analog signal by approximating it by the output signal of a digital-to-analog converter (DAC). Hereby it makes use of a successive-approximation register (SAR), which implements a binary

search algorithm: It takes N cycles to yield a resolution of N bits. For this purpose, the DACs in SAR ADCs are often implemented by arrays of N binary scaled capacitors. Czierlinski (2022) implements an appropriate dual-mode capacitive DAC for the new SAR ADC and a transmission gate for all internal switches. Within this thesis, we develop, present and discuss the other components of this SAR ADC: a mixed-signal verification framework, a comparator unit, and the digital control logic.

Firstly, Section 3 describes the development of a mixed-signal verification framework. The verification framework makes use of the new custom python module `TeststandAMS` that enables to set up, execute and evaluate hardware simulations in this programming language. These simulations run in a mixed-signal-simulator, joining the event-based behavior simulation commonly used in the digital domain with a simulation that solves the circuit according to Kirchhoff's laws. Together they enable functional verification.

Secondly, Section 4 discusses the implementation of a dual-mode successive-approximation register. Its digital logic fetches comparator decisions and reconfigures the capacitive DAC accordingly. By storing each decision and thus the approximation path, it provides ADC's digital output code. From an abstract point of view, a successive-approximation register implements a large finite state machine (FSM). However, smart memory and signal path partitioning allow keeping the logic efficient and robust at the same time.

Then, Section 5 present the design process of a comparator unit. In a SAR ADC, a comparator is used to compare the output of the DAC with the input voltage. In fact, the actual implementation of the DAC modulates the input voltage, so the comparator compares to a reference voltage. This work also presents a corresponding reference voltage generator that is built from a 5 bit adjustable capacitive voltage divider. The comparator core itself implements a double-tail sense amplifier. A double-tail sense amplifier is a further development of the common sense amplifier flip-flop which is used, e.g., for SRAM readout and optimized for large input voltage ranges.

Lastly, Section 6 evaluates the performance of the entire ADC including the work of Czierlinski (2022). By combining all components, we simulate its characteristics with respect to the sampling frequency based pre-layout models. Furthermore, post-layout simulations allow a realistic performance estimation of the entire ADC.

The development of an ADC is a challenge in itself. Microelectronic engineers have spent decades developing and improving various design architectures. However, for us in the Electronic Vision(s) group, this project is the first design of a high-speed ADC. This thesis marks the first design iteration of an ongoing process.

2 Background

Before starting the development of a new component, it is generally useful to define target values and specifications for it. First, we explain and discuss some common definitions for ADC specifications. Different ADC architectures are also briefly mentioned, yet this topic can be found in more detail at Czierlinski (2022). Secondly, the BSS-2 system, for which the new ADC is developed, is introduced. Then, the targets and specification are defined based on the planned applications. Finally, the concept for the new ADC is presented.

2.1 Analog-to-digital converters

An analog-to-digital converter (ADC) converts a time- and value-continuous, electrical quantity into a quantized, digital representation of the same. For this purpose, it samples an electrical signal at some time t_s and returns a corresponding sequence of N bits. Thus, it grants access to data processing with discrete numbers, such as integers and floating points. Such a quantization always comes with a loss of information: This so-called quantization error is the fundamental accuracy limit of an ideal ADC. To gain reliable insights into the behavior of the electrical signal, the ADC needs to sample it sufficiently frequent and precise.

Furthermore, distortions and noise from different sources influence the conversion. Generally, suppressing them results in an increased power consumption and area.

It is common practice to distinguish two input signal configurations: single-ended and differential. A single-ended ADC compares an input signal to a global signal (e.g., ground) and a differential one determines the difference of two signals. For the remainder of the thesis we consider a voltage V as input signal and an ordinary binary encoded bit sequence as output code C:

$$C \in \left\{0, 1, 2, ..., 2^N - 1\right\} \quad \text{with} \quad N \in \mathbb{N}. \tag{2.1}$$

The smallest resolvable value is the value of the least significant bit (LSB).

Monotonicity. An ADC should provide a simple and use-case orientated transfer function from a physical quantity to a number. This transfer function describes the assignment of a sampled voltage $V(t_\mathrm{s})$ to a digital code C and itself should be time independent. Ideally, therefore, one and the same input voltage V should be always mapped to the same

output code C.

A natural assignment of a given input voltage V to a digital output code C is monotonic. This is due to the fact, that it directly comes with an easily invertible transfer function. Typically, one would expect a monotonously rising behavior:

$$C(V) \leq C(V + \Delta) \qquad \forall\ \Delta > 0,\ \forall\ V. \tag{2.2}$$

2.1.1 Linear relation

In addition, it is often claimed that the transfer function is approximately linear over the dynamic range of the ADC. The linearity is limited by the resolution limit of the digital code C. Counterexamples can be found, e.g., where more sophisticated transfer functions balance the non-linearity of the sensors (e.g., some CMOS cameras). However, a linear regression of output codes C with respect to input signal leads to a simple and sufficient mapping for many applications:

$$C = a \cdot V - b. \tag{2.3}$$

In an ideal ADC, the parameters gain a and offset b are given by:

$$\begin{aligned} a &= \frac{2^N}{V_{\max} - V_{\min}}, \\ b &= a \cdot V_{\min}, \end{aligned} \tag{2.4}$$

where $V_{\min}$ and $V_{\max}$ determine the allowed dynamic range V_{FS} of the ADC. Using Equation (2.3), the LSB becomes the weight of $V_{\mathrm{LSB}} = 1/\mathrm{gain}$.

Physical measurements of these characteristics are typically performed using a sin-wave stimulus, as for example a saw-tooth stimulus becomes hard to generate for high ADC sampling frequencies. In simulations, as presented in this thesis, the generation of the stimulus is no obstacle. Instead, the simulation execution time is the limiting factor. Therefore, we sweep the input voltage with a sufficient granularity.

Extraction of transitions. To simplify further computations and interpretations, one extracts the voltages that correspond to a code transition from the sweep. All further evaluation methods depend on these trip points. Inspired by a measurement procedure of

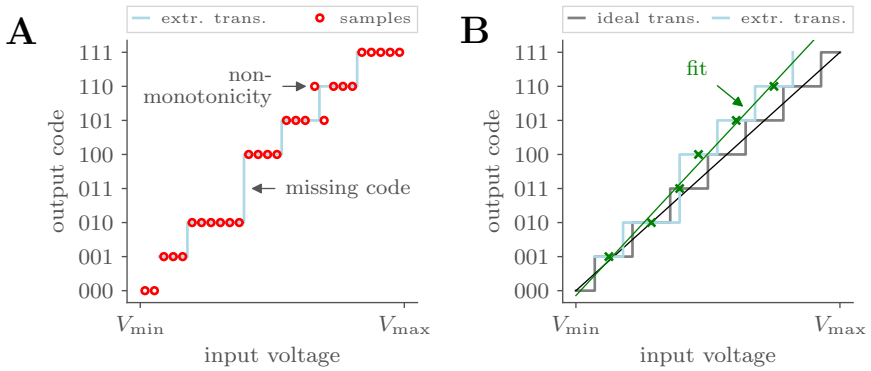


Figure 1: Extraction of a linear transfer function from sampled data. **A**: Extraction of the code transitions. **B**: A linear fit (green) through the mean voltage of each code determines gain a and offset b. Their deviation from an ideal ADC (gray) is called gain- and offset-error.

the company Applicos, we developed an evaluation method appropriate for our problem. Given an ordered set of strict monotonic and equidistant input voltages:

$$V_{\text{min}} \leq V_{\text{j}} < V_{\text{j}+1} \leq V_{\text{max}} \quad \text{for j} \ \in [0, M \in \mathbb{N}] \tag{2.5}$$

with corresponding output codes C_{j} and assuming $C(V)$ to be monotonic, we use a cumulative sum of a histogram with integer binning to find the positions of trip points $j_{\text{trans}}(\text{i})$ for output code $C = \text{i}$:

$$\text{j}_{\text{trans}}(\text{i}) = \sum_{\text{m}=0}^{i} \left(\text{histogram}_{\text{j}\in[0,M]}^{\text{bin: m}\in[0,2^N]}(C_{\text{j}}) \right)_{\text{m}} . \tag{2.6}$$

From this, we can read of the transition voltage $A_{\text{i}} = V_{\text{j(i)}}$. In fact, this also handles non-monotonic positions and missing codes.

The resulting trip points can be extrapolated to a set of mean voltages $V_{\text{i}} = (A_{\text{i}+1} + A_{\text{i}})/2$ that correspond to a certain output code C_{i}.

Gain- and offset-error. Using linear regression, one can extract the parameters gain a and offset b (c.f. Equation (2.4)) from the extracted input voltages and output codes. This linear regression can either be based on an ordinary linear fit over all possible codes or by only referring to the first and the last output code. Here, we use the former method, as this is more precise.

Given an ideal V_{min} and V_{max}, one can measure the deviation of a regressed gain to an ideal

one. A large gain error indicates either a reduced dynamic range or a reduced resolution inside the dynamic range. In the same manner, one can measure the deviation of the offset. The offset error indicates an additional shift in the dynamic range.

Beside the limitations of the dynamic range, an offset or gain error does not deteriorate the quality of a linear voltage-to-code mapping. However, where an absolute mapping matters, offset- and gain-errors directly lead to large absolute deviations.

2.1.2 Non-linearity

Non-linearity is the term used to describe the errors that remain after the offset and gain errors have been corrected. Usually, differential and integral non-linearity are considered.

The differential non-linearity (DNL) is a measure for the local step size deviation. Given real transitions A_i and the ideal step size $\Delta A_\mathrm{ideal} = V_\mathrm{LSB} = 1/\mathrm{gain}$, it returns the ratio of real to ideal step size for each step. To be centered around zero, it is often shifted by one.

$$\mathrm{DNL}(A_\mathrm{i}) = \frac{A_{\mathrm{i}+1} - A_\mathrm{i}}{\Delta A_\mathrm{ideal}} - 1 \qquad \forall i \in 0...(2^N - 1) \tag{2.7}$$

A DNL of ≤ -1 is equivalent to a missing code. For a good ADC performance, it is desirable to keep the DNL between $\pm 0.50\,\mathrm{LSB}$. In the case of gradient based optimization in analog computing (c.f. Sections 2.2.2 and 2.2.3), a consistently small DNL helps to find the correct gradients. This is the case because the numerical determination of gradients is based on the formation of local differences, on which the DNL has a large influence.

In the same manner, one can define the integral non-linearity (INL). Contrary to DNL, INL links the real transition position A_i to the ideal one.

$$\mathrm{INL}(A_\mathrm{i}) = \frac{(A_{\mathrm{i}+1} + A_\mathrm{i}) \cdot \mathrm{gain}}{2} + \mathrm{offset} - i \tag{2.8}$$

The INL of an ideal ADC is zero for all possible output codes. Furthermore, the mean of the INL only measures the quality of the fit used to determine gain and offset. Some literature consider INL not with respect to the output code, but only take the maximum of its absolute value.

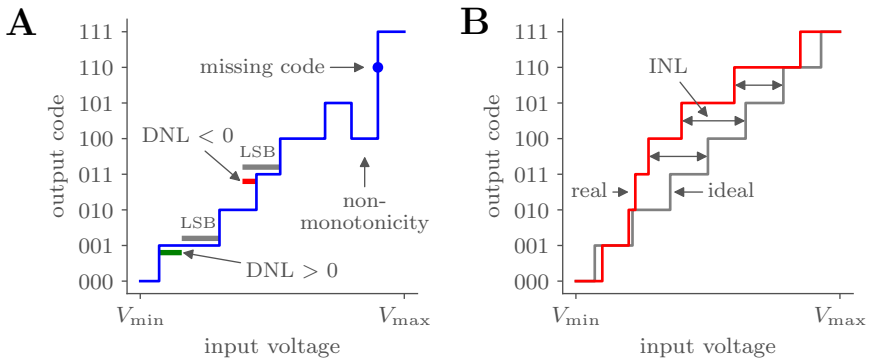


Figure 2: Common measures for non-linearity in Nyquist-ADCs. **A**: The DNL determines the variation in the individual step widths. **B**: The INL determines the absolute deviation of the real transition function from the ideal one. Illustrations inspired by Pelgrom (2017).

2.1.3 Introduction to ADC architectures

Due to the huge variety of application areas for ADCs, there are many implementations (Bashir et al., 2016). Each implementation is optimized for a special use case. However, one can find recurring architectures. The following section describes three architectures and one concept that are used or discussed in this thesis.

All discussed ADCs consist of at least one comparator, which compare one or more reference voltages with the (modified) input voltage. To be able to control the sampling time t_s, an ADC is typically equipped with an implicit or explicit sample-and-hold stage. Furthermore, some architectures are more likely to be adapted from a single-ended input to a differential one.

Single-slope ADC. The current implementation of the CADC on the current chip version (c.f. Section 2.2) is a single-slope ADC (Schreiber, 2021). A ramp generator generates a saw-tooth reference voltage that covers the desired dynamic range of the ADC. This reference voltage is then compared to an input voltage using a comparator. A counter starts when the ramp starts and stops when the reference voltage exceeds the input voltage. Hence, it is also called “counter-ramp-ADC”.

Since the ramp generator can be separated from comparator and counter, this architecture is easy to parallelize. The single-slope ADC also has advantages in terms of energy. However, due to the ramp and the counter, it takes 2^N steps to achieve a resolution of N bit. This limits the maximum sampling frequency and makes them unattractive for high resolutions. Concepts like double-slope ADCs try to face this fundamental speed limit,

but also stay in an exponential relation between precision and necessary steps (Bashir et al., 2016).

Flash ADC. A flash ADC utilizes 2^N comparators with individual reference voltages to achieve a resolution of N-bit. Hence, a conversion takes only one step, but is exponentially expensive in power and area. In some measure, it is the counterpart to the single-slope ADC. To reduce the number of comparators, some modifications can change their comparators reference voltages to reach their result only after multiple steps.

Successive-approximation ADC. A SAR ADC balances sampling frequency, energy consumption and area usage. Utilizing a DAC, the input or the reference voltage is modified in binary steps, to bring the two closer together, just like a binary search (Figure 3). Due to the binary steps, each step represent one bit of output code starting with the most significant bit (MSB). Hence, the number of conversion steps is N for an N-bit ADC (Pelgrom, 2017).

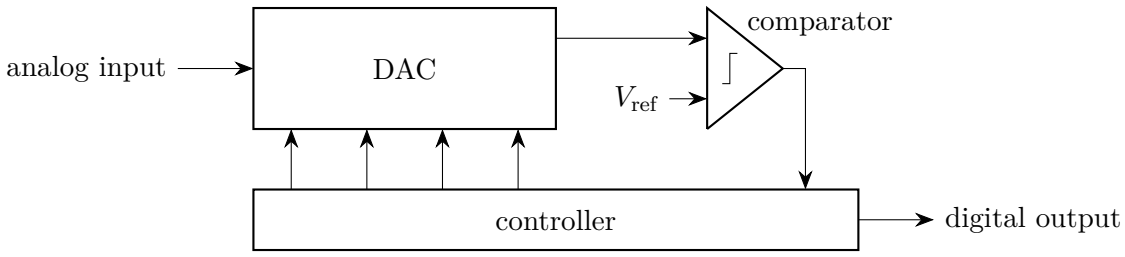


Figure 3: General concept of a typical successive-approximation ADC, utilizing a (usually) capacitive DAC, a comparator and control logic implementing the successive-approximation register.

In many cases, the DAC is realized by a switchable, binary weighted capacitive array. Here, one can distinguish between a *charge redistribution* (Kugelstadt, 2000; Harpe, 2018) and a *charge sharing* (Chen et al., 2018; Craninckx and Van der Plas, 2007) approach. Both can reuse their capacitive DACs as sample capacitors for their input stage. They differ in how the sampled charge is adopted: While the former reload the capacitors during the conversion using charge from an external power supply, the latter balances the charge on the capacitors by changing their polarity. Hence, in case of *charge sharing*, the final charge and voltage on the capacitor is path dependent. Therefore, the steps are not invertible. This becomes crucial when considering parasitic capacitance and hinders general analytical solutions. However, during conversion *charge sharing* is power supply

independent and also aims for a low power consumption. Using the more common *charge redistribution* approach, one finds a simple analytical solution for the ADC's algorithm: According to charge-division and due to the binary weight, each step m is able to add or subtract $V_{\mathrm{FS}}/2^{m+1}$ to the sampled voltage V.

The concept of a pipeline ADC is very similar to a capacitive SAR ADC. Instead of using one DAC and iterating its voltage in place, it disaggregates the steps to individual stages. This typically works in the current domain and allows reusing the stages after each step.

Interleaving ADC. Interleaving multiple ADCs is a common approach to increase the sampling frequency. Here, independent ADC input stages sample an input voltage at slightly different points in time. Now one or more independent ADC channels can evaluate the sampled voltages. Further samples can be taken in parallel from unused input stages. This can be used to take very few samples with very high frequency that are evaluated afterwards or accomplish an almost dead-time free ADC.

2.2 Applications in BrainScaleS-2

Von Neumann architecture has been dominated computing for decades. However, NNs suffer from the associated von Neumann bottleneck. For this reason, today many artifical-neural-networks (ANNs) utilize graphics processing units (GPUs), which follow the "stream process" architecture. However, the human brain, which serves as a template for neural network models, uses a completely different architecture: in-memory computing. Here, memory and data processing engine are nearby or even realized in the same physical structures. One approach of in-memory computing is the emulation of neural networks in analog circuits. BrainScaleS-2 (BSS-2) is a mixed-signal neuromorphic system, that flexibly implements analog neurons and synapses, but moreover benefits from the advantages of digital signal processing and communication (Schemmel, 2021). The analog observables in networks need to be accessed by its digital environment utilizing ADCs.

2.2.1 System overview

BSS-2 contains mixed-signal application-specific integrated circuits (ASICs) along with software- and communication-infrastructure. The utilized high-input-count-analog-neural-network digital-lerning-system (HICANN DLS) chip, designed in a 65 nm low-power CMOS node and fabricated by TSMC, is currently available in the third version of generation

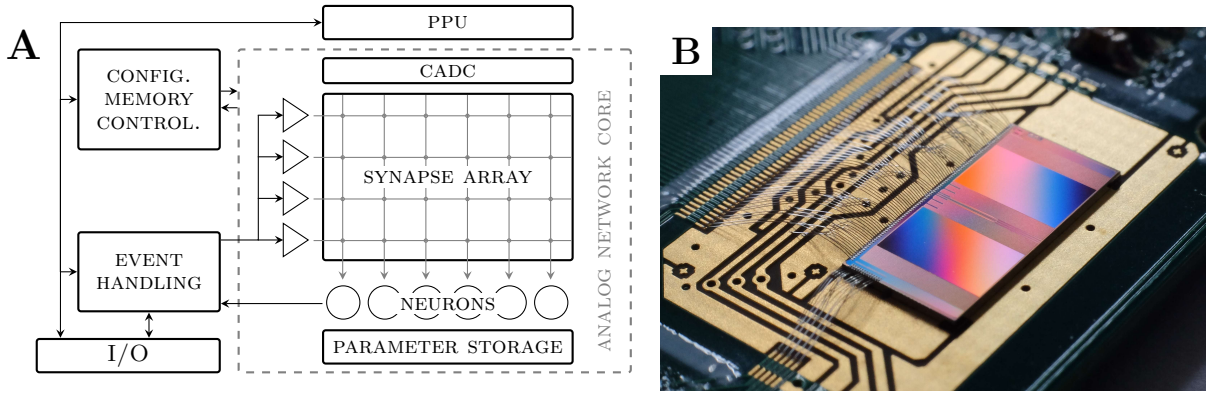


Figure 4: Visual introduction of HICANN DLS. **A**: Block-level diagram taken from Billaudelle et al. (2020). **B**: Photo of a HICANN DLS Xv2 chip, photograph provided by Eric Müller.

"X". It implements an analog neural network core, digital microcontrollers with vector extensions, ADCs and DACs, as well as, event handling and a high-speed link interface. Furthermore, the surrounding BSS-2 system provides experiment control with external memory and I/O based on field-programmable gate arrays (FPGAs). Current systems include one FPGA per HICANN DLS-chip, connected by a 1 Gbit high-speed link.

Analog neuromorphic core. As it emulates accelerated, scaled and biologically inspired neuron and synapse behavior, the analog neural network core on HICANN DLS is the main component of BSS-2. The 512 neuron circuits, distributed equally over four blocks and two hemispheres, offer a rich spectrum of accurate dynamics (Billaudelle et al., 2022). They implement differential equations of the adaptive exponential integrate-and-fire (AdEx) neuron model, which is an extension of the more common leaky integrate-and-fire (LIF) neuron model:

$$C_{\text{mem}} \cdot \frac{\mathrm{d}V_{\text{mem}}}{\mathrm{d}t} = -g_{\text{leak}}(V_{\text{mem}} - V_{\text{leak}}) + I(t). \tag{2.9}$$

Here, C_{mem} denotes the membrane capacitance, g_{leak} the leak conductance and V_{leak} the leak potential. Whenever the neuron membrane potential V_{mem} reach a threshold voltage ϑ, a spike is emitted and the membrane resets to V_{reset} (Lapicque, 1907; Gerstner and Kistler, 2002; Brette and Gerstner, 2005). All additional external currents, such as synapse interactions, are summarized in the time dependent offset current $I(t)$. By connecting multiple neurons, one can abandon these point neuron model in favor of a dendritic and spatially resolved one (Kaiser et al., 2022).

By default, the adjustable membrane capacitance C_{mem} is configured to its maximum value

of $2.46 \pm 0.01\,\mathrm{pF}$ (Billaudelle, 2022). Furthermore, the leak conductance g_{leak} reaches a maximum large-signal transconductance of $\approx 10\,\mu\mathrm{S}$. Such a configuration, which for example occurs in the implementation of the reset condition, results in a membrane time constant $\tau_{\mathrm{mem}} = C_{\mathrm{mem}}/g_{\mathrm{leak}}$ of $\approx 250\,\mathrm{ns}$.

Each neuron receives potential input from 256 individual synapses, arranged in an array. Therefore, synapse drivers transform digital 5 bit pre-synaptic events into a voltage based stimulus with length Δt and send them horizontally through the synapse array. The synapses themselves store a 6 bit weight to generate a respective current I and, if enabled by the correct provided address, dump a charge $Q = I \cdot \Delta t$ to a neuron's vertical synaptic input line. Here, this charge is integrated on a leaking capacitor, modeling an exponentially decaying synaptic interaction kernel and driving a current- or conductance-based synaptic input to the neuron membrane (Billaudelle et al., 2022). For spike-time-dependent plasticity (STDP), each synapse is equipped with correlation sensors. (Friedmann et al., 2016; Billaudelle et al., 2019)

On-chip ADCs. HICANN DLS is equipped with two types of ADCs: a fast and precise pseudo two-channel ADC, called membrane ADC (MADC), and a slow and low resolution CADC. Each type of ADC has a different area of applications. A quantitative comparison of their specifications can be found in Tables 1 to 4.

The CADC reads voltages from the synapse correlation sensors as well as from variables in the differential equation, such as the membrane potential V_{mem}. For this purpose, each synapse row corresponds to two designated CADC channels. Designed as single-slope-type ADCs (c.f. Section 2.1.3), each chip hemisphere shares a common saw-tooth signal, generated by integrating a current on a capacitor. Since the CADC lacks an explicit sampling stage, it cannot guarantee a precise sampling time. Providing a 8 bit resolution and a maximum individually triggered sampling rate of 1.85 MS/s, their performance is inferior to the fast MADC.

The latter is a single channel 10 bit Nyquist-rate SAR ADC with a maximum continuous sampling rate of 62.5 MS/s. By alternating between two inputs, the MADC provides a pseudo-two-channel mode at half the sampling frequency (LSM, 2015). Due to its area- and power-consumption, HICANN DLS provides only a single MADC.

Plasticity processing unit. The two on-chip general-purpose microprocessors (one for every hemisphere) are extended by special-purpose vector units implementing the single

instruction, multiple data (SIMD) principle. These vector units are designed to calculate and perform synapse weight updates from observables in the analog core unit. On the one hand, they have access to almost every configuration on the chip. On the other, they are directly linked to CADC's readout channels (Friedmann, 2013; Friedmann et al., 2016). Thanks to the *Power ISA 2.06* architecture of its general-purpose parts, programs for the plasticity processing units (PPUs) can be cross-compiled on a host computer (Heimbrecht, 2017).

2.2.2 Inference with spiking-neural-networks

While ANNs perform difficult tasks fast and with high accuracy, they suffer from high power consumption. Inspired by the human brain, spiking-neural-networks (SNNs) use binary spikes in time as well as continuously integrating neuron membranes to save energy in sparse networks. Modern learning methods, such as backpropagation through time (BPTT) allow training these networks efficiently. The lack of analytical gradients in the physical system can be bridged using e.g., *surrogate gradients*. Here, the learning algorithm performs the forward path on hardware storing spikes and membrane traces as well as in a simplified numerical SNN-simulator. Later, the backward path uses the gradients from the simulator, but inserts measured spikes and traces from the hardware to compute weight updates. Cramer et al. (2022) and Göltz et al. (2020) have proven the learning and inference capabilities of BSS-2 in a series of different application and data sets.

However, learning algorithms that utilize continuous state variables like membrane potentials are limited in temporal resolution by the current CADC's sampling frequency. Increasing the sampling frequency in a redesigned ADC as well as the data link speed would reduce the dependency on interpolations. Moreover, an improved sampling frequency would allow using the neurons to operate at shorter timescales $\tau_{\text{mem}} = C_{\text{mem}}/g_{\text{leak}}$ and therefore increase the inference speed. Currently, the implemented SNNs emulate a continuous sampling mode for the CADC by using the workaround of regular trigger signal from the PPU.

Another approach in SNNs is incorporating STDP (Billaudelle et al., 2019). These biologically inspired learning rules incorporate measured correlations between pre- and post-synaptic events for each neuron (Friedmann et al., 2016). Originally, the CADC was developed specifically to allow the PPU to access these correlation sensors. With STDP one can implement structural plasticity, which explicitly keeps a SNN sparse.

2.2.3 Analog vector-matrix multiplication

Vector-matrix multiplication accounts for a significant fraction of computing power used in the forward path of ANNs (Oh and Jung, 2004). HICANN DLS's architecture can also be used for analog vector-matrix multiplication as well as an analog inference accelerator for ANNs. Storing the matrix in the synapses' weights and pushing input vectors through the synapse drivers, each synapse affects the synaptic line by multiplying the two. After charge integration on the synaptic line, the respective voltage equals the sum over the multiplication results (Weis et al., 2020). In the absence of dedicated sampling capacitors in the CADC, the current pulses are integrated on capacitors of the leaky neuron membranes. Finally, the CADCs digitize the voltages to make the result of the analog calculation accessible by the digital environment.

The full application is seamlessly integrated with `pytorch` (Spilger et al., 2020). Due to the in-memory computing approach and the power efficiency of the system, this so-called *HAGEN* mode is promising to be competitive to other state-of-the-art implementations of ANN-accelerators (Stradmann et al., 2022). However, the current implementation does not reach the speed limit of the synapses, which dump their charge in a maximum of 4 ns. Integrating directly on the sample stage of a new and faster ADC as well as increasing the bandwidth of the high-speed FPGA-link can help to get closer to this limit.

2.2.4 Calibration of analog circuits

Crucial to the accuracy and richness of the dynamics in the circuits is that each of their innumerable parameters can be adjusted individually. Calibration is used to compensate device mismatch. Moreover, it enables to adapt the circuit to different models (Leibfried, 2021; Weis, 2020). Here, an almost orthogonal mapping of circuit- and biological-parameters supports efficient and robust calibration routines (Billaudelle et al., 2022). The calibration routines are mainly based on observations with relatively large time scales which allow the use of the current CADC.

2.3 Design targets for a new column-parallel ADC on BSS-2

The applications mentioned above result in requirements for a redesigned CADC. Due to the wide area of applications, two modes with different focus were suggested: a fast mode and a precise mode. The former should serve the needs of the analog vector-

matrix multiplication, while the latter mainly replaces the functionality of the current implementation. The new Nyquist-rate ADC is planned to operate parallel in hundreds of channels. All targets are subject to a balanced overall performance of the ADC.

Sampling. One of the main reasons to redesign the current CADC is its lack of high sampling frequencies (see Table 1). While the current design is well suited for reading from the correlation sensors in a plasticity experiment, it is one of the limiting factors for fast inference applications. For many applications one would also like to support continuous sampling. On the other hand, the individual triggering of samples should still be possible, and therefore a low latency startup is desirable.

The new ADC's fast mode is adapted to the maximum synapse operation speed. The synapses are able to perform one charge pulse in 8 ns using the maximum current pulse length of 4 ns. For future implementations of the *HAGEN* mode, one would like to dispense with integrating the charge in the membrane capacitance. Instead, one would then directly use the synaptic line as integration capacitance, which speeds up the entire calculation. This not only reduces the integration time constant but also enables the omission of preparatory measures such as resetting the membrane voltage. Therefore, the synapses current pulses should be aligned with the ADC's sampling windows. Including timing jitters and an integration time constants, this results in the required sampling window of 6 ns. To stay within the synapse operation speed of 125 MS/s, the maximum allowed dead-time is 2 ns.

A precise mode should be at least five times faster than the current implementation. Reaching a maximum sampling frequency in the same order of magnitude as the MADC is preferable as this leads to improvements as discussed in Section 2.2.2. The tracking period should be long enough to not decrease the ADC's precision. However, this mode is not designed to be dead-time free.

Unit	speed / MS/s	clock speed / MHz
CADC	1.85	500
MADC	62.5	750
New ADC (fast mode)	125	$\leq$1000
New ADC (precise mode)	≥ 10	$\leq$1000

Table 1: Target sampling frequency of the new ADC in comparison to the current on-chip ADCs (Schreiber, 2021; LSM, 2015).

Area. As it is planned to locate the new ADC in the same position as the current CADC, the area constraints of the previous design also apply to the new one. Therefore, Table 2 illustrates the dimensions of related current chip components. This is especially important for the column width that needs to fit the synapse array. However, the previous two channels per synapse column can be replaced by one. The two channels were necessary to allow parallel read out of the causal and acausal correlation sensors. An increased sampling frequency, which is below the typical time scales of these sensors, makes this limitation dispensable.

The channel height is limited by an efficient ratio of ADC area to synapse array area. Moreover, in the current HICANN DLS, it is also restricted by neighboring units such as the PPU.

Unit	width / µm	height / µm	area / µm^2
CADC (single channel)	5.88	45.70	265
CADC block w/ ramp generator	1592.15	45.70	72 761
Single synapse	11.76	8.00	94
Synapse block w/ driver	1541.50	2097.90	3 233 913
MADC	340.12	317.17	107 876
New ADC (single channel)	11.76	≈100	≈1160

Table 2: Aspect ratios and areas of on-chip ADCs and related VLSI circuits. Data taken from the Electronic Vision(s) internal layout database.

Energy. Analog in-memory computing comes with the claim of power efficiency and is characterized by this for edge applications. However, ADCs are often one of the most power-consuming parts in analog computing. It is planned that at full speed operation the total ADC power does not exceed 50 % of the total chip power. Nevertheless, our guideline is to stay at 10 % of the total chip power. Calculating with a total chip power of 1 W and 512 ADC channels, this leads to 196 µW per channel. Table 3 presents a classification of this target with respect to the current one-chip ADCs.

A reduced power also helps to reduce the effort for a sufficient power routing. Due to its significant effect on the total chip power, the new ADC should consume as little static power as possible. Furthermore, due to a total supply line inductance of about 10 nH, the current transients $\mathrm{d}I/\mathrm{d}t$ must also be sufficiently small. In addition, all active components should be detachable.

The used TSMC 65 nm low power process provides normal transistors for a 1.2 V power supply as well as thick-oxide transistors for voltages up to 2.5 V. Since the device's power

consumption grows quadratically with the supply voltage, we limit ourselves to the 1.2 V power supply.

Unit	max. power / µW	energy / $\frac{\text{pJ}}{\text{conv}}$
CADC (single channel)	13.4	7.20
MADC	2500	38.5
New ADC (single channel)	196	1.56

Table 3: Energy budget of the new ADC in comparison to the other on-chip ADCs (Schreiber, 2021; LSM, 2015).

Resolution. Different applications require different ADC resolutions. In order to be able to select a resolution, the associated dynamic input range is crucial. The 8 bit resolution distributed over the dynamic range from 0.0 V to 1.2 V of the current CADC (see Table 4) is sufficient for all current applications. Hence, the precise mode of the new ADC should aim for this resolution. At this point it should be noted that both currently implemented ADCs cannot be used reliably over their entire dynamic range. Due to effects occurring in the readout chain between observable and ADC, voltages below 150 mV cannot be resolved and voltages below 200 mV and above 1.15 V are distorted (Billaudelle, 2022).

In vector-matrix multiplication, a 5 bit synapse driver pulse is multiplied by a 6 bit synapse weight. The previously discussed future charge integration on the synaptic line leads to an adapted dynamic range. Synaptic interaction pulls down the synaptic line's potential by a few hundred millivolts. Since the synaptic line leaks to the supply voltage of 1.2 V, a reduced dynamic range of 0.6 V to 1.2 V seems to be sufficient for the fast mode. At the same time, one can reduce the number of bits, due to the low resolution of the input vector and the weights in the matrix.

The current CADCs operates with single-ended input signals. This design decision was motivated by the single-slope architecture. However, to reduce further modifications, the new ADC should also provide a single-ended input.

Unit	input range / V	resolution / bit
CADC	0.0 to 1.2	8
MADC	0.0 to 1.2	10
New ADC (fast mode)	0.6 to 1.2	6
New ADC (precise mode)	0.0 to 1.2	8

Table 4: Target resolution of the new ADC in comparison to the other on-chip ADCs.

Peripheral constraints. Beside all these ADC target specifications, multiple boundary conditions are given by technicalities of the HICANN DLS.

Firstly, the design provides a magnitude of parallel channels. Hence, a new ADC architecture must be robust against crosstalk.

Secondly, the number of references is limited. It is not possible to load any reference except power and ground. Moreover, any used static unloaded reference would be the same for all channels.

Lastly, the ADC is crossed by multiple control signals connecting the synapse matrix with the PPU. Since each ADC channel corresponds to one synapse column, these signals are 2×8 static random-access memory (SRAM) data signals (in/out) with 2 corresponding enable-signals as well as 2 lines for reading the causal and acausal correlation sensors.

2.4 Concept for a fast, small and flexible SAR ADC

This thesis presents components for a successive-approximation ADC, that features the two previously suggested operation modes in one physical implementation. This was preceded by an extensive search for a suitable ADC architecture. Details on the chosen architecture are presented in Czierlinski (2022).

2.4.1 Flexible capacitive DAC for a SAR ADC

Czierlinski (2022) present and discuss the concept for a *charge redistribution* SAR ADC illustrated in Figure 5. Furthermore, they developed a capacitive DAC optimized for the two required modes: It consists of two individual arrays, each implementing a DAC that is suitable for a 7 bit SAR ADC with a dynamic range of 0.0 V to 1.2 V. In operation, one array samples an input voltage on its capacitors, whereas the other adopts a previously sampled voltage in order to convert it into a digital code. By doing so, Czierlinski (2022) implement two interleaving ADCs. However, other physical components such as the comparator and the controller are shared between both ADCs to reduce the required area. Coupling these 7 bit-full-range DACs, one gets a combined DAC that fits the requirements of a 8 bit SAR ADC. The capacitors for the array are designed as metal-oxid-metal (MOM) capacitors using crossed, dense metal lines in multiple metal layers. Details on the switching scheme used to implement the two modes can be found in Section 4.2.

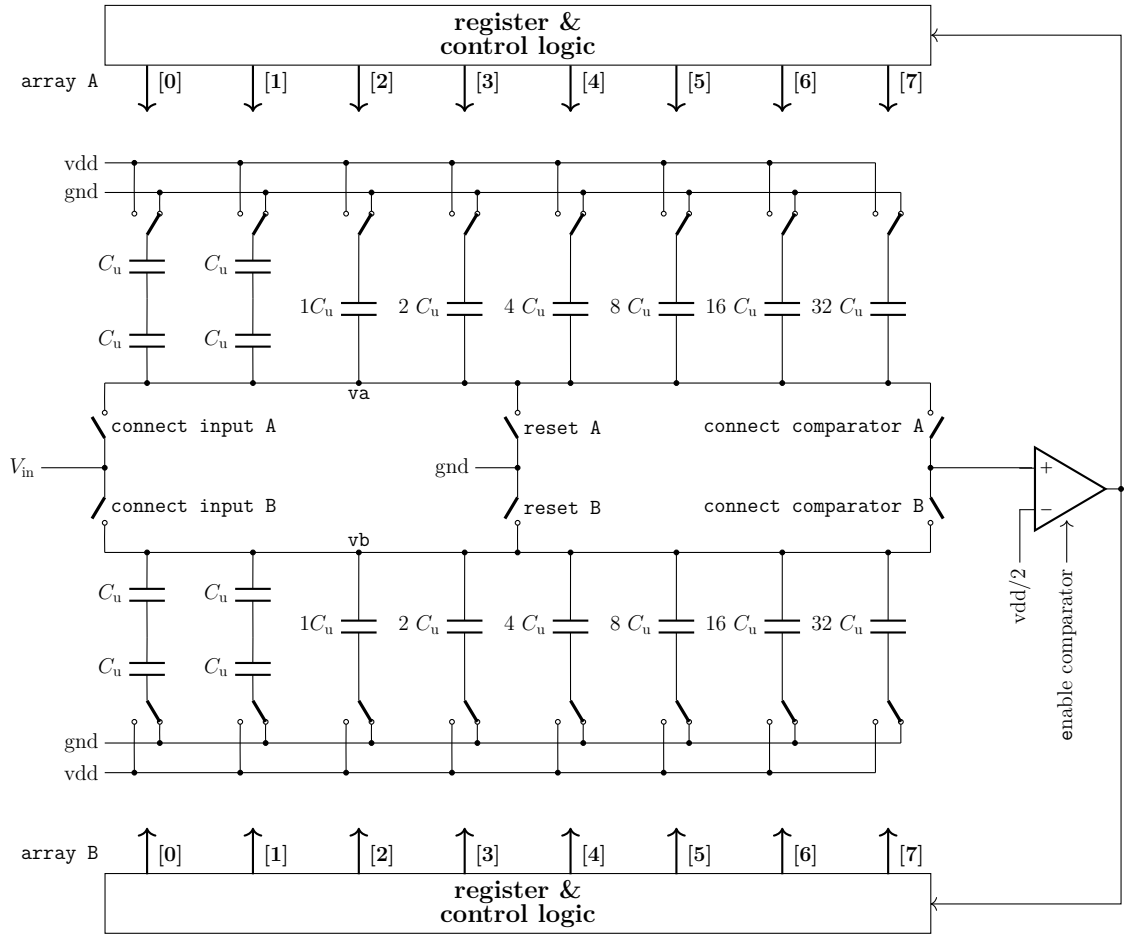


Figure 5: Overall schematic of the new charge redistribution ADC concept which special emphasis on the capacitive DAC implementing a fast and a precise mode. Taken and adapted from Czierlinski (2022).

Fast mode. The required fast mode is implemented using the two DACs operating interleaved. Due to the fact, that a 7 bit ADC with a rail-to-rail dynamic range equals the properties of an extended half-rail-to-rail 6 bit ADC, this configuration exceeds the given constraints without suffering any particular disadvantages.

While one DAC operates as sample-and-hold stage, the other one performs the conversion of its previous sampling period. The conversion uses a *try-and-reject*-principle. This means that the DAC first adds a certain amount of charge, which then is removed again if the new arranged voltage is greater than half the reference voltage (0.6 V). If the new voltage is smaller, the additional charge stays in the capacitive array. The charge is moved by switching the capacitors bottom plate either to ground or to the reference voltage. To be able to remove charge also in the first iteration, the input voltage V is sampled on one capacitive array with almost all capacitors are connected to ground; only the MSB-capacitor is connected to the supply voltage. By doing so, the binary weighted capacitors

refer the total charge equally to both rails.

Precise mode. In the precise mode, both capacitor arrays are connected. Both capacitor arrays sample the input voltage V, but one refers to ground and the other to the supply voltage. If the comparator output is high, the previously supply-referred MSB capacitor switches its reference to ground. Otherwise, the ground-referred MSB capacitor changes its reference. In the next step, the newly adopted voltage is compared again, and its result applied to the next capacitor accordingly. This continues until the LSB capacitor is reached. At this point, another last comparator decision is performed after the corresponding result was generated and applied. This process generates 8 bit with two times seven binary sized capacitors.

2.4.2 Floor plan

A first iteration of the new ADC consists of two capacitive 7 bit DACs, a comparator with a suitable reference generator and some transmission gates in its full-custom part as well as a synthesized digital controller. The floor plan of the full-custom part is dominated by the two capacitive digital-to-analog converter arrays. The utilized process node provides nine metal layers. Including their shielding, they occupy metal layers 4 to 8 of the nine metal layers provided in the TSMC 65 nm low power process. Metal layer 3 is used for the crossing synapse-PPU-signals, power distribution and some long range analog signals. Most of the control signal routing (e.g., for the transmission gates and the DAC) is done on the second metal layer. Aside from the transmission gates, the comparator and its reference generators are also planned to be placed below the capacitive DAC. Hence, these components need to be flat. Moreover, some local decoders and combinatorial logic are planned to join the full-custom parts to reduce the overall area consumption. Due to their required routing resources the digital controller is currently placed next to the capacitive array.

3 Mixed-signal verification framework

Verification is an important task in hardware and very large scale integration (VLSI) design. The verification of a design proofs its correct behavior. “Correct” means that a design meets the previously defined design specifications. Once manufactured, silicon hardware is generally immutable. Therefore, each module, unit or system needs to be verified beforehand, no matter if it operates in the analog or the digital domain.

Verification can be done in many ways. One distinguishes simulation based verification and formal verification. As the name suggests, simulation based verification uses simulations to check the behavior of a design. These checks can be carried out in different methodical ways. For digital designs, visual inspection of simulated waveforms is the simplest verification method, but it has only low test coverage. The test coverage can be increased by using systematic or randomized test patterns that are automatically evaluated. Formal verification refers to a design reduction down to mathematical expression (e.g., boolean equations), which then can be used to find an equivalence between two designs or a design and a specification. Verification of analog designs is driven by small- and large-signal simulations that carry out sweeps over different process corners, supply voltages and temperatures or make use of Monte-Carlo methods to indicate device variations.

Cadence® is one of the market leaders for hardware development environments. However, it lacks the ability to run, adapt and evaluate functional hardware tests from a widely used programming language such as `python`. In this chapter, we present and discuss the development of a `python` interface for *Cadence Spectre*® *AMS Designer* and *Cadence Xcelium*™ simulator and its application in a generic Nyquist-ADC verification framework.

Test bench for simulation based verification. In order to perform simulation based verification, a test bench provides and coordinates the necessary resources. One popular standardized and modularized test bench is provided by the universal verification methodology (UMV) (Accellera, 2015). Figure 6 illustrates the components and the signal flow in a classical design test bench using a nomenclature inspired by the UMV.

A test bench verifies a design under test (DUT). Therefore, the DUT is placed in a functional environment. Test signals that stimulate the DUT are generated by a module called driver. A monitor tracks the corresponding responses of the DUT. Finally, a scoreboard evaluates these with respect to the input signal. This can be done by comparing them with a model or the specification.

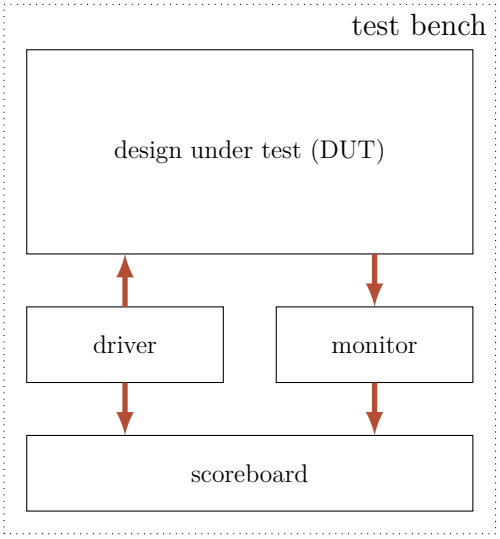


Figure 6: Classical test bench design for hardware verification. Inspired by Burkhardt (2012).

For generalization, the DUT is often coupled to the rest of the test bench using a design specific interface. This allows to keep driver, monitor and scoreboard free from design specific elements and reuse them in multiple designs.

3.1 Introduction to hardware simulation

The used circuit simulation technique depends on the required abstraction level. Here we would like to highlight four levels of abstraction: The functional behavioral description, the register transfer level (RTL), the transistor-level description (schematic), and the geometric description of the circuit in silicon (layout). Except for the register transfer level (RTL) which is a digital domain specific structural representation, these abstraction levels are general. In a top-down-approach, one starts conceptualizing a design entry at the highest abstraction level and end up with a silicon chip that contains geometrical structures (Gajski et al., 1994). Thereby, information gained at a more precise implementation phase, can be annotated in higher abstraction levels. This can be resistances and capacities that are extracted or estimated from a layout, which are either directly inserted as devices into a schematic or used to model correct delays on a RTL.

By choosing a suitable abstraction level, the simulation becomes efficient in of simulation speed and optimal adapted to a use case. In general, one can distinguish at least two simulation paradigms: On the one hand there are Spice-class simulators, which simulate schematics with all its components and under consideration of Kirchhoff's rule by numerically solving differential equations. On the other hand, there are event-driven simulators

that simulate behavioral descriptions and logic at a RTL by scheduling, evaluating and executing events of an event queue. A purely digital design flow (c.f. Section 4.1.2) can do without SPICE simulation, because the gain in knowledge is marginal compared to a back-annotated event-based simulation. However, event-based simulation are often too imprecise to model pure analog circuits. Mixed-signal designs contain circuits that belong to the digital domain and other circuits that belong to the analog domain. By combining the two simulation paradigms, simulation performance in the digital domain is increased without compromising simulation accuracy in the analog domain (Balasubramanian and Hardee, 2013).

From here on we use the terms electrical and logic for the simulation paradigm and the terms analog and digital for the intended operation domain. This nomenclature is inspired by the naming of E2L element presented in Section 3.1.3.

3.1.1 Simulations in the electrical domain

Simulations in the electrical domain allow simulating a circuit on a physical level. Here, we use *Spectre*, a Spice-class simulator developed and distributed by *Cadence*. *Spectre* is a commercial redesign of Spice, a "simulation program with integrated circuit emphasis" originally developed at the EECS department of the University of California by Nagel and Pederson (1973).

Spectre is equipped with various numerical algorithms to solve a circuit according to Kirchhoff's laws. Therefore, it provides different models for different circuit elements as for example resistors, capacitors, conductivities and voltage sources. But also more complex models like for diodes and metal-oxide-semiconductor field-effect transistors (MOSFETs) are supported. Chip manufactures provide multiple adopted models for their devices, including process parameters and process variations. This circuit solving capability can be used for different types of analysis, such as transient, DC, AC or noise analyses. A transient analysis evaluates the circuit's evolution over time. Therefore, it uses a DC analysis to find a properly converged set of initial values for every node to start with. Such DC analyses are also used to investigate the circuit's static behavior. *Spectre* also provides meta-analyses like Monte-Carlo-Analysis that draw circuit instances with parameters from a given distribution to model e.g., device mismatch.

In practice, in order to enable a simulation, the database behind the graphical circuit schematic needs to be translated into a netlist. This netlist contains parameterized instances of device models and electric primitives like resistors and capacitors. In a fur-

ther step, one adds the configuration of an analysis and sets different design parameters (Quarles et al., 1994). Together, this results in a source file, that is interpretable by the simulator.

The Spice simulator engine itself analyses the given circuit according to Kirchhoff's voltage and current law. To do so, it generates a mathematical expression of the circuit in matrix representation. It replaces all non-linear elements by linearized models to solve the expressions[1]. During a transient analysis, the time steps are typically adjusted to the gradients of the signals in the circuit.

Even Spice and *Spectre* are fully controllable by using a command line interface, software packages like *virtuoso ADE* provide a graphical user interface (GUI) for the simulators. At the latter, the full workflow is hidden and executed automatically. But also the recorded signals are originally only available in the GUI. Only the explicit usage of functions enables further external data handling.

3.1.2 Simulations in the logic domain

A digital signal can usually exhibit four symbols: *Zero*, *one*, *undefined* and a *high-impedance* state. The majority of digital designs use synchronous logic. This means that all signals are evaluated synchronously to a global timing signal, called clock. This enables a structural, RTL representation of a design.

Verilog[2] and its further development *SystemVerilog*[3] are hardware description languages (HDLs) that are capable of representing almost all abstraction levels. However, their most essential feature is, that they intrinsically provide an executable model of the design. In their definition they describe the functioning of an event-based simulator. Here we use the *Cadence Xcelium* simulator to simulate designs on a RTL. *Xcelium* is one of the big three commercial *Verilog* implementations.

The term "event" in event based simulator refers to an update or change of a signal or variable in the design. *Verilog* provides blocking (example: `A <= B;`) and non-blocking (example: `A = B;`) assignments of signals, which can occur in `always @()` (free-running process) or `initial` (single execution) blocks. Further, *Verilog* also provides continuous assignments, introduced by `assign` statements. These assignments generate update

[1] A typical Spice simulation implement the Newton algorithm by means of a sparse LU factorization. In order to be able to parallelize the simulations, one no longer requires global convergence today and decomposes the matrix into individual parts.

[2] Language reference: IEEE Std 1364-2005 (2006)

[3] Language reference: IEEE Std 1800-2017 (2018)

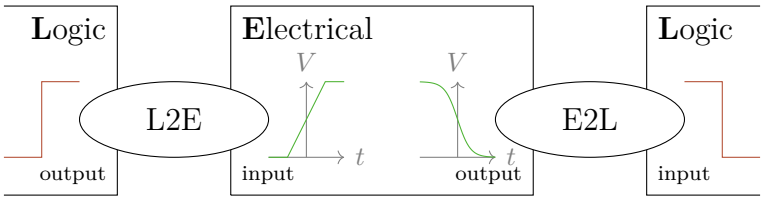


Figure 7: Connect rules define the simulator behavior of signals that crosses the domain boundaries.

events during the simulation, that are directly executed or scheduled for a later execution. After resolving all hierarchies and module instances in a design, an iterative scheduling, evaluating and executing process starts. A time wheel runs with a given precision and defines time slots. In each time slot, the simulator evaluates the design for planned changes in signals or variables (queuing events) and executes them by scheduling new events. Thereby *Verilog* defines the order, in which the different assignments are processed.

3.1.3 Simulation of mixed-signal designs

A mixed-signal simulator runs an electrical domain simulator and a logic domain simulator in parallel. To transfer signals from one simulator to the other, both needs to be synchronized. In our case, *Xcelium* acts as master and *Spectre AMS Designer* as slave that accesses *Xcelium*'s database.

Special elements, called connect-rules, are automatically inserted at the ports between modules that are simulated in the electrical domain and modules that are simulated in the logic domain (c.f. Figure 7). On the one hand, an E2L (electrical-to-logic) module converts electrical voltages to a logic symbol. Thereby, a voltage below a given first low-threshold becomes a "zero" and above a second high-threshold a "one". The regime in between is interpreted as "undefined". Furthermore, the E2L implements termination capacitors. On the other hand, an L2E (logic-to-electrical) module converts a logic symbol to an electrical signal. Here it typically uses a piece-wise linear voltage source, that is extended by termination resistors and capacitors to generate a more real source. The parameters of the connect-rules are specified in a configuration file (e.g., `ie_card.scs`), which is parsed to the compilation step of the simulator.

Developed by *Cadence*, *Verilog-A* is a HDL or netlist language for the electrical domain that is based on elements of *Verilog*'s structure. For electrical signals, *Verilog-A* introduces the discipline "electrical", which basically describes an electrical node. Beside the

instantiation of models, *VerilogA* also supports using different analog operators and different functions to connect the nodes. This includes linear relations, but also derivatives and integrals of signals in a design, which together allows abstract behavior modeling.

The feature set of *Verilog-A* enables *Verilog* to be extended to a mixed-signal HDL: *VerilogAMS*[4]. In this sense, a *VerilogAMS* does not fully belong to the electrical or logic domain, but provides a number of signals that belong either to former or to latter. Hence, it is not possible to directly connect a signal of the discipline electrical to one of the logic domain. But by using more flexible data types as integers and reals, one can describe the behavior response of an electrical signal to a logic and the other way around. By doing so, connect-rules can be described fully in *VerilogAMS*.

3.2 Interfacing Cadence with Python

In order to interface with the hardware and circuit simulators, there are often two main options: Either one uses a graphical user interface or one utilizes the command line interface of the underlying simulator engine. A graphical user interface enables an intuitive user experience: Every simulator option can be found and selected in menus and input masks. After setting up a simulation, a corresponding configuration file can be downloaded, which enables to repeat this configured simulation multiple times. The command line interface, however, allows executing such pre-configured simulation setups. Writing a configuration by hand provides the full power of the simulations, without searching for specific options in drop-down menus. Nevertheless, it requires extra effort to get familiar with all the configuration needed. Often, manual work is needed to adopt all configurations in complex simulation environments. Both options prevent fast iterating based on simple but powerful scripts.

The biggest disadvantage, however, is that simulations cannot be easily automated to depend on the results of other simulations. Here software lacks on simple automated post-simulation evaluation options like fitting functions to masked signal waveforms. This becomes crucial when simulating complex calibration routines e.g., to investigate, whether a circuit reaches its target specification (Kriener, 2017). Thereby one typically make use of Monte-Carlo methods and draw parameters from a distribution to simulate device variations.

The `python` module `Teststand` respectively `TeststandAMS` is an attempt to develop a

[4]Language reference: Verilog-AMS Language Reference Manual (Little et al., 2014)

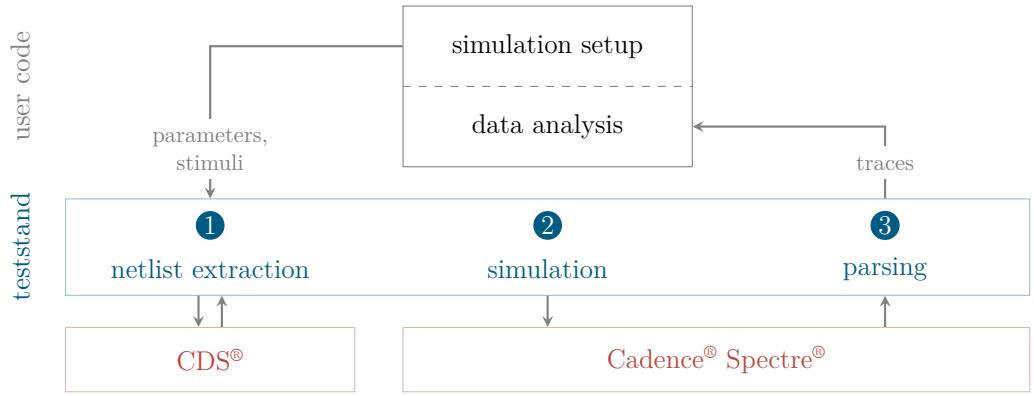


Figure 8: Structure of a teststand-based simulation including the interaction with the Cadence Design Suite, taken from Grübl et al. (2020).

simple accessible toolbox, that creates, adapts and executes hardware simulator configurations (Grübl et al., 2020). Therefore, it runs different *Cadence* tools and simulators and keeps track of the required and provided input and output files. In this sense, it does not implement a new simulator, but an interface to existing simulators. For waveform analysis, it exports all probed signals directly to python, which enables the rich functionality of a full programming language. `TeststandAMS` allows a scripted and automated simulations jointly with short simulation iteration cycles. While the original `Teststand` is a creation of Sebastian Billaudelle, this thesis fully redesigned it for the purpose of mixed-signal simulations.

3.2.1 Teststand

`Teststand` is a custom python module, that interfaces the *Spectre* simulator (Grübl et al., 2020). Therefore, the module reads, creates and modifies configuration files and runs *Spectre* simulator, as well as other tools from the *Candence Design Suite* (CDS) in subprocesses.

As shown in Figure 8, in a first step it directly extracts the netlist from a *Cadence* design database using a modified *ocean* script. In a further step, `Teststand` is able to adapt simulator options in the netlist file, such as temperature and design parameters. Moreover, it adds different analysis configurations, as well as probing statements. When enabled, `Teststand` adds more circuit elements such as voltage- and current sources to the netlist. Within this work, capacitors and resistors are added to the list of netlist extension devices. Together, this enables creating the test bench with the power of a

full programming language like python. Finally, `Teststand` passes the modified netlist to a *Spectre* sub-process. The separation of netlist generation and actual simulations makes it possible to use the same primary netlist with different adaptations in multiple simulations. This in particular applies to Monte-Carlo-analyses, which also bases upon one single primary netlist. Here, multiprocessing of the *Spectre* sub-processes enables fast end efficient simulation. Every simulation log is parsed to a log file so that a targeted investigation is possible in case of errors. In step three, all simulation results are parsed back to python utilizing `numpy` objects. Thereby, `Teststand` make use of the external and free python/C++ library `libpsf`[5]. `Teststand` is successfully integrated into the design flow of the Electronic Vision(s) group (Grübl et al., 2020; Schemmel et al., 2020).

3.2.2 TeststandAMS

Inspired by `Testand`, this thesis presents a python interface for a mixed signal simulator. Thereby, the API is a consequent derivative of the original `Teststand`. However, it newly separates the front-end, that sets up the simulation, from the file handlers and run engines. This allows a flexible adaptation and continuous development of `TeststandAMS`. To fit the requirements of mixed-signal simulations, the front-end classes and definitions are adapted and optimized. `TeststandAMS` uses tools from *Cadence AMS Designer*.

Mixed-signal simulations requires a netlist generator, that decodes electrical as well as logic circuit descriptions. Here, `TeststandAMS` enables the netlist generation for modules in the *Candence Design Suite* (CDS) library architecture. As known from `Teststand`, one first instantiates a custom CDS library, before instantiating `TeststandAMS`'s main class. Thereby, `TeststandAMS` triggers the super tool *Cadence runams* to create a *VerilogAMS* netlist (`netlist.vams`-file) down from one top-level module. This netlist holds the information for all schematics modules, while all other modules, that are already available in a HDL, are included externally into this netlist (`textInputs`-file).

Aside from the netlist, *runams* generates configuration files for a simulation setup. In the following step, `TeststandAMS` reads this files, adjust them according to the required simulation options and writes them to the simulation run-directory. In particular, `TeststandAMS` applies individual circuit parameters setting these up in the `cds_gloabls.vams` file. The analog simulator *Spectre* is configured in the `amsControlSpectre.scs` file. Here, `TeststandAMS` writes out the analyses for every simulation. Due to the character of mixed-signal simulation, these are typically limited to transient and Monte-Carlo anal-

[5]The source-code of the python module `libpsf` can be found on Github (`https://github.com/henjo/libpsf`, state: 29.11.2014).

```
# Define CDS library
self.cds_lib = CDSLib()
self.cds_lib.add_include("mydesign/cds.lib")
self.cds_lib.add_include("connect_lib/cds.lib")

# Perform net-list generation of the CDS-based design
testbench = TeststandAMS(cds_lib=cds_lib,
                         design=Design("mylib", "mycell", "my_view"),
                         model=Model("modelfile.scs", "section"),
                         netlist=True,
                         root=Path("netlist_rundir"))

# Define analysis
analysis = TransientAnalysis("voltage_trace", stop=10e-9)

# Set up simulation
simulation = SimulationAMS([analysis],
                           parameters={"transistor_width": 0.1e-6}
                           connrule=1.2,
                           save=[Net("my_toplevel.myinstance.mysignal")]
                           timescale=(1e-9, 10e-12),
                           add_hdl=["my_toplevel.sv", "hdl_module.v"],
                           set_toplevel="my_toplevel")
simulation.path = Path("sim_rundir")

# Run simulation
results = testbench.simulate(simulation, workers=10)
```

Figure 9: Code snippet of a `TeststandAMS` simulation.

yses. Here moreover, `TeststandAMS` sets up the circuit temperature and the simulation accuracy. The used Spice-models for the elements in the netlist are added in the `spiceModels.scs`-file. Connect-rules are specified in the `ie_card.scs`. `TeststandAMS` make use of general connect-rule definitions, that scales with the supply voltage. Signals, that should be stored by the simulators are denoted in the `probe.tcl`-file.

All these files are represented in different classes derived from a common file base class. These classes perform the described file manipulations; often using regular expressions and replacement. Unfortunately, the pure `runams` output files are not suitable to set up a successful simulation. Hence, some misconfigurations are also fixed in using these file modifiers. However, they are all hidden behind the front-end simulation setup API. Similar to `Teststand`, the user sets up a simulation containing all information about the upcoming file modifications. The actual modification happens jointly to the simulation run call.

The actual simulation is performed utilizing *Xcellium*, by running the super-tool *xrun*

with mixed-signal options. Again, this tool is executed in a parallelized and controlled sub-process engine, which enables simulating the same design with different options simultaneously. The various options are passed on to the simulator via the configurable tool call of *xrun*.

After the actual simulations, probed signals are exported to `python`. Due to a different and more difficult output data format (`sst2`-format), `TeststandAMS` needs a different waveform reader than `Teststand`. The proprietary file format requires the usage of another *Cadence* tool (*SimVisDButil*) to export the probed signals as `csv`. Finally, we read the `csv`-data to python and parse them to `numpy` data structures.

3.3 Application in an ADCs characterization framework

The development of a new fast and efficient parallel ADC is the first use case of `TeststandAMS`. Therefore, we developed a generic ADC verification framework. Here, test bench concepts, ADC characteristics and simulation methods meet.

In order to develop a verification framework, one needs to define its scope first. The given design specifications serve as basis to find observables and quantities, that describes the fit of a design to its specification. Therefore, the design's field of application influences, which aspects of the unit are covered by the test suite.

In case of the new ADC, introduced in Sections 2.3 and 2.4.1, the tests focus is on the dynamic range including (offset and gain error), the DNL and the INL. Due to is methods, the verification framework operates on a functional verification level. Furthermore, functional verification can not replace a rigorous verification of the underlying modules, but it does measure how all parts perform together as unit.

Having this said, in its core the verification framework uses a staircase shaped stimulus to sweep the ADC's input voltage through the target dynamic range. Using a sufficiently dense step width, one can extract all code transitions from this simulation. In this sense, the framework is rigorous as it covers systematically the full input range. However, the framework only covers "DC" input conditions, bandwidth limitations are not detectable.

Data flow. Figure 10 shows a block schematic of the framework's data flow, exemplary for the new SAR ADC. The process is mastered by a python script, which uses `TeststandAMS` and performs the output data's post-processing. As supposed by `TeststandAMS`, the framework uses a logic top-level test bench module, which is written in *SystemVerilog*.

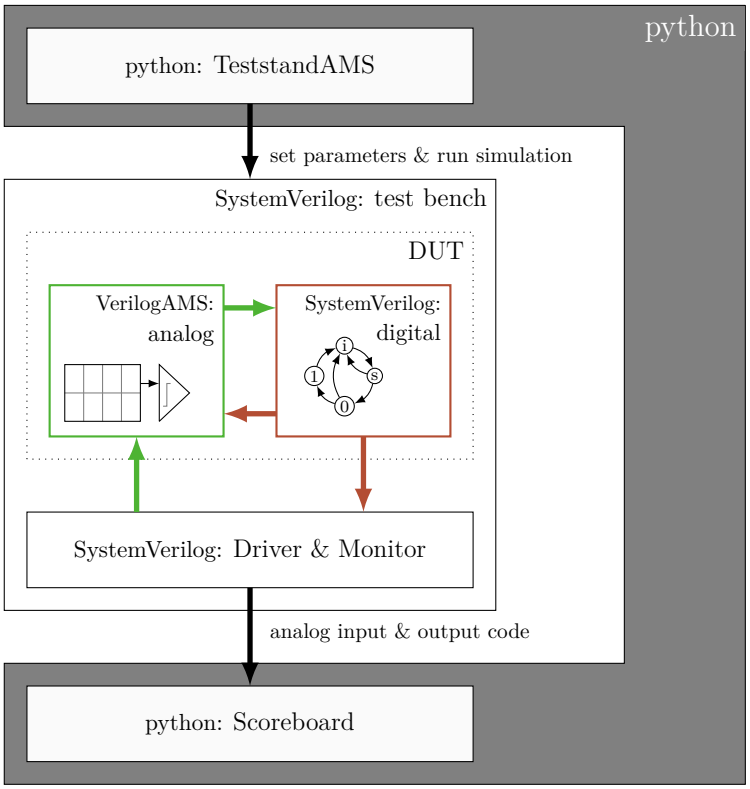


Figure 10: Suggested ADC verification strategy using `TeststandAMS`. The mixed-signal simulation is set up and started from python. The test bench itself is a *SystemVerilog* module containing a driver- and monitor-unit as well as the DUT, which can consist of modules from both simulation domains.

At this level, one also includes other units that serve as environment for the DUT, for example global signals. Moreover, it integrates on the one hand the DUT and on the other a driver-and-monitor unit. The provided driver-and-monitor unit, also written in *SystemVerilog*, generates the introduced staircase stimulus and writes the ADC output code, together with the input voltage, to a file. For this purpose the electrical signals are internally represented as real values. A custom *VerilogAMS* voltage source at the interface to the electrical domain, assigns the requested input voltage to the ADC's input node. Since only *SystemVerilog* allows real values in port lists, the intermediate real value representation of the input voltage is converted to a 64bit signal using *Verilog*'s build-in functions. The time continuity is realized by using a suitable timescale directive. As an ADC contains analog and digital elements, the DUT can contain subunits in the electrical, as well as in the logic domain. However, here we decided to use the *Cadence* design library for all elements in the ADC. So, the netlist is generated at this hierarchy level, enabling simulating the digital circuits in the logic as well as in the electrical domain. A scoreboard written in python evaluates the ADC's responses.

Scoreboard evaluation. The scoreboard implements the equations and methods from Section 2.1. Different helper functions implement different aspects of the evaluation. The respective results are stored in a summary. First, a checker investigates the monotonicity of the output code and ensures equidistant input voltage steps.

Now, the scoreboard can extract the output code's transition points as described in Section 2.1.1. All further evaluations base on these transition points. But due to the n_{steps} discrete input voltage steps, the precision of the transition points A_{i} is limited. Given all intervals between input voltage steps, that show a code transition and without further knowledge, we assume the distribution of actual transitions to be uniformly distributed inside these intervals. Consequentially, the transition points' standard deviation is determined to:

$$\Delta A_{\text{i}} = \frac{V_{\text{max}} - V_{\text{min}}}{\sqrt{12} \cdot \text{n}_{\text{steps}}} \quad \forall \text{i} \in \left[0, 2^N - 1\right]. \tag{3.1}$$

Here, n_{steps} is the number of equidistant steps in the input voltage sweep, V_{min} and V_{max} the limits of the sweep. The factor $1/\sqrt{12}$ comes from the standard deviation of a uniform distribution. In case of non-monotonic transition behavior, the performed trip point search returns an averaged code transition point. This further increases the transition points uncertainty. Generally, the transition point extraction bases on the assumption, that the ADC performance is time independent. Here, a visual inspection, that monitors ADC's voltages and currents, can ensure that this is the case. Furthermore, we do not include any systematic errors from the simulator, since a suitable simulator setting (numerical tolerances, ...) makes them negligible.

Gain and offset of the ADC are determined by a first-order polynomial fit. Hereby, the scoreboards uses the voltage at the centers between two transition points. The scoreboard calculates a gain and offset corrected DNL and INL with Equations (2.7) and (2.8). Using Gaussian error propagation one can estimate the DNL's standard deviation to:

$$\Delta\text{DNL}(A_{\text{i}}) = \sqrt{\frac{1}{6}\left(\frac{V_{\text{max}} - V_{\text{min}}}{\text{n}_{\text{steps}}} \cdot \text{gain}\right)^2 + \left(\Delta\text{gain} \cdot (A_{\text{i+1}} - A_{\text{i}})\right)^2} \quad \forall \text{i} \in \left[0, 2^N - 1\right]. \tag{3.2}$$

In the same way, we calculate the standard deviation of INL using Equation (2.8)

$$\Delta\text{INL}(A_{\text{i}}) = \sqrt{\frac{1}{24} \cdot \left(\frac{V_{\text{max}} - V_{\text{min}}}{\text{n}_{\text{steps}}} \cdot \text{gain}\right)^2 + \left(\Delta\text{gain} \cdot \frac{A_{\text{i+1}} + A_{\text{i}}}{2}\right)^2 + \Delta\text{offset}^2}. \tag{3.3}$$

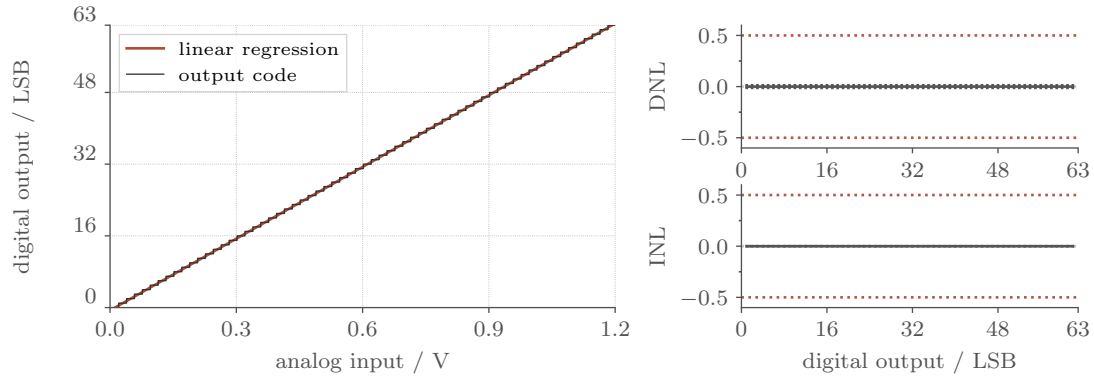


Figure 11: Results of the evaluation with the help of the presented verification framework using the example of an ideal 6 bit ADC from the *Cadence* library. The input voltage range was sampled in 1024 steps and the result was approximated with a linear fit after determining the code transitions. The DNL and INL, given in LSB, are calculated from the deviations from this fit.

Framework testing. According to their definition, the mean value of INL and DNL must always be zero. Apart from numerical deviations, a continuous monitoring of these values measures the quality of the fit.

To ensure that the generic ADC framework performs as expected, we implemented test cases at different levels. The test suit covers the different helper functions behavior. Moreover, functional tests keep track of the framework's performance. Here, the test suit contains prominent examples of artificial output codes. To also test its proper integration in `TeststandAMS`, the tests simulates the ideal ADC provided as library module by *Cadence*. Latter is shown in Figure 11. All functional tests are regularly and for every software change executed by a Jenkins[6] server. The framework as well as `TeststandAMS` are integrated in the Electronic Vision's containerized software development flow.

[6]The official webpage of Jenkins can be found at `https://www.jenkins.io/` (accessed: 26.12.2022).

4 Successive-approximation register

Each successive-approximation ADC requires a controller implementing a binary search algorithm. Basically, a successive-approximation register (SAR) ensures that a comparator's decision leads to a corresponding change in the configuration of the DAC. Moreover, it generates the digital output code from the sequence of comparator decisions. Hence, the controller is a key part of the SAR ADC and just as critical in terms of speed, area, and energy as the capacitive DAC or the comparator.

There are very different approaches to design a fast SAR logic (Xu and Xu, 2017; Gao et al., 2015; Chio et al., 2008). The control logic is usually strongly tailored to a specific ADC. As described in Section 2.4.1, the new ADC provides two modes and two switching schemes, that the controller needs to fit. However, due to this complexity in the new ADC, it is difficult to adopt many of these approaches.

The controller's main functionality repeats individually in every ADC channel. A target clock frequency of 1 GHz in mind, the ability to share the surrounding logic between channels is limited. In the following sections, we present a controller for the new dual mode SAR ADC.

4.1 Introduction to digital hardware design

The controller implements digital logic. In order to use common verification tools (c.f. Section 4.1.2), this logic operates synchronously to a globally distributed clock signal. Hereby, one resorts on a combination of sequential and combinatorial elements realized by characterized standard cells. These elements differ in the fact that sequential logic's output depends on an input at a specific, previous point in time (e.g., the latest positive clock edge), so it stores information, while the output of combinatorial logic ideally only depends on its current input. Sequential elements are for example master-slave-flip-flops (FFs) or latches, whereas the elements realizing an "and", a "or" or a "not" gate are combinatorial. Typically, different sequential nodes, called registers, are connected using combinatorial logic.

Real combinatorial logic has delays, because they are on the one hand build from silicon transistors (gate delay) and, on the other hand, connected by real wires that inadvertently implement RC elements (propagation delay). In deep sub-micron processes, the delay through the wires exceeds the gate delay (Sylvester and Keutzer, 2000). Signals on different paths through the combinatorial logic have different delays. Due to these signal

path-depended delays, a simultaneous change in the input signals, can lead to unexpected intermediate output states, before the actually output state is reached. This behavior is called “glitch”. Glitches are acceptable for synchronous logic, but in signals controlling transmission gates or the DAC, they can lead to undesired behavior.

The longest signal path must be shorter than one clock period minus the setup and clock-to-output time of the sequential logic. The setup time describes the minimum time that the input signal of a FF must be stable before are positive clock edge. The clock-to-output time describes how late after a positive clock edge the FF must have taken over the new value at its output. The hold time is the third characteristic time of a FF and specifies how long after a positive clock edge the input signal must still be stable so that the signal is also reliably accepted.

4.1.1 Finite state machines

A finite state machine contains a finite number of states $Z^t \in \{Z_0, \ldots, Z_i\}$ between which it alternates depending on its input vector X^t. Automata theory knows deterministic FSMs as a subset of non-deterministic ones (Hofmann and Lange, 2011). Given a current state Z^t and a specific input vector X^t, the deterministic one have exactly one possible transition to a new state X^{t+1}. Every FSM can be illustrated in a state diagram.

The mathematical method of FSMs can be used to describe sequences in synchronous digital hardware. Thus, they are often used in decision-making and control logic (Katz, 1994). From the state Z and the input signal X one can derive an output signal Y. There are two main dominant methodologies: The Moore machine and the Mealy machine (Chiuchisan et al., 2010). Both modeling techniques consist of a next-state logic $Z^{\mathrm{t}+1} := f\left(X^{\mathrm{t}}, Z^{\mathrm{t}}\right)$, a state register Z^{t} and an output logic Y^{t}. A transition, even if it is recurrent, is forced in every clock cycle. However, both techniques differ in how they generate their output signal.

In a Mealy machine, the output signal depends on the stored state and the input signal $Y^{\mathrm{t}} := g\left(Z^{\mathrm{t}}, X^{\mathrm{t}}\right)$. Often, this leads to a reduced required storage and increases the response time. However, possible glitches at the input stage are transmitted to the output.

In a Moore machine the output signal only depends on the current internal state $Y^{\mathrm{t}} := g\left(Z^{\mathrm{t}}\right)$. Hence, a change in the input signal is only applied at the output in the next clock cycle. This avoids propagating glitches through the FSM to its output. Yet, a Moore machine is not as fast a Mealy machine. In general, a Moore machine requires more

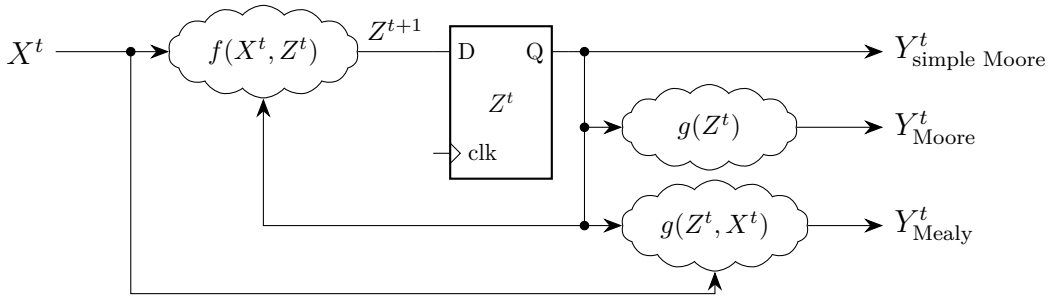


Figure 12: Differences and similarities between the most common implementations of finite state machines: Simple Moore machine, Moore machine and Mealy machine.

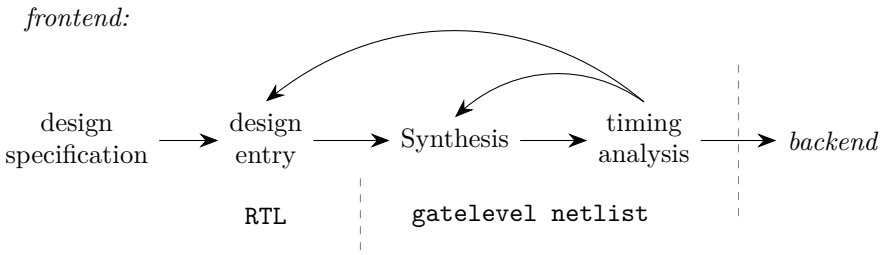


Figure 13: Digital frontend design flow and its representation level, inspired by Harter (2002). If the time constraints are not met after synthesis, the design process must be repeated.

internal states than a Mealy machine in order to produce the same output spectrum. A subclass of Moore machines are the simple Moore machines. These are simpler because the output signal Y^{t} is equal to the state representation $Y^{\mathrm{t}} := Z^{\mathrm{t}}$. This completely avoids glitches caused by runtime differences.

One can also build mixed forms of these automaton classes. The states Z are realized by sequential logic. To indicate the number of FFs used for the state representation, we utilize the common bit notation $Z[b:a]$. In this way, we also access individual FFs $\hat{Z}[a]$. Here, we denote partial state representations by a hat. To concatenate different FFs, we use the curly braces $Z = \{\hat{Z}[a], \hat{Z}[b]\}$.

4.1.2 Digital design flow

Generally, one divides the digital design flow into a generic frontend and an ASIC specific backend. The frontend design includes a circuit behavior description without any information about the physical chip. This generalizes an implementation and makes it easy to reuse a design in multiple chips or even in different process nodes. As illustrated by Harter (2002) in Figure 13, the frontend design covers the path from a design specification

to a first gate-level netlist. The gate-level netlist contains instances of standard cells that implement the building blocks of digital hardware, e.g., FFs and combinatorial gates. In this "trow-over-the-wall" approach, the subsequent backend design includes floor planning (including IO-ring), design placement and routing, a clock tree synthesis and power routing (Grübl, 2007). Here, we focus on the frontend design flow.

Frontend design flow. The frontend design flow is dominated by two blocks: The creation of a RTL design entry on the one side and the syntheses on the other. While former requires a lot of manual work, the latter is supported by, e.g., *Synopsis Design Compiler*® or *Cadence*® Genus™ tool kits. During the design entry creation, one generates specific registers and precise logic sequences from a design specification. In our case, the switching schemes description in Section 4.2 is the design specification. During this creative process, one can revert to principles like FSMs (c.f. Section 4.1.1). Keeping future design steps and the physical implementation in mind reduces the number required design cycles. This targets, for example, the implemented set of combinatorial logic between two sequential elements: As each combinatorial gate comes with some delay.

Synthesis. In a first step during synthesis, *Synopsis Design Compiler* software requires access to timing and cell information from the used process node (syn, 2018). Compiling the RTL source files, it generates an abstract module and hierarchy view on the design. The following elaborate step resolves the module behavior and therefore unifies the module instances. The actual synthesis creates a boolean and primitive representation from this. This representation is mapped to the given standard cells of a process node.

In order to guide this mapping process, one adds design constraints like the target clock frequency. A proper target set helps to the algorithm to find a good optimum, while over-constraining could entice the algorithm to stack in local optima. With this, it determines the choice and order of required standard cells.

Hereby, the correct sizing of the standard cells is important. The wide majority of cells are available in different driving strengths. If the driving strength is too low, it takes a long time for the wire and gate capacities to recharge and the cell's output signal becomes slow. A cell that is too large not only consumes too much space, but also typically has a larger input capacitance. Hence, more stages are needed for the same logic increasing the energy consumption and the total gate delay. As rule of thumb, fan-out of four fits the timing-energy relation best: Each transistor with gate width W and gate length L drives an equivalence of a total transistor gate area of $4\ W\ L$ (Hedenstierna and Jeppson, 1994).

To verify the timing closure, *Synopsis Design Compiler* software uses a static timing analysis (STA). Each signal path is annotated with its expected delays and compared to the setup and hold times of the terminating sequential logic. The path delays are determined from the predetermined cell delays and estimated wire delays. In a first step, a wire load model is used to estimate the wire delays statistically without any knowledge about the physical placement or routing. This estimate is correspondingly imprecise. If a cell mapping doesn't fit the constraints, the tool tries to optimize it by iterating this process with an adopted focus on the designs structure.

4.2 Switching scheme of a flexible SAR ADC

The new ADC's switching scheme of the transmission gates and DAC is fixed by its architecture choice. Due to the requested two modes, there are two architectures in one ADC: One implementing two individual and interleaved ADCs and another that uses these both parallel to increase their resolution. For a better understanding of the following section, the reader is advised to refer to Figure 5 from Section 2.4.1.

Independently of the operation mode, we introduce a synchronous global sample signal `g_sample`, that indicates the ADC's tracking period. The tracking period is the time interval during which a capacitive element in the ADC is connected to the input and thus charges to it.[7] Evaluated at the next positive clock edge, its high state indicates the ADC to start a new tracking period. If `g_sample` is low again, the ADC moves on with the conversion of the sampled input voltage. To ensure the synchronous character of the `g_sample` signal, it will be guided parallel to the clock. Using `g_sample`, the track period is adjustable to different use cases.

As illustrated by Figure 5, the new ADC implements two arrays, which we index with the characters "A" and "B". Each of these arrays provides seven binary weighted capacitors as well as one additional capacitor, that equals the smallest binary weighted capacitor. While one side of the capacitors is shorted to the common mode lines `va` or `vb`, the other is connected either to ground or to the supply voltage. In practice, the latter is implemented by eight individual inverters: If an inverter's input signal is high, the capacitor is connected to ground, if it is low, the capacitor is connected to the supply rail. These inverters' inputs are embraced to a bus `array A[7:0]`/`array B[7:0]`, where bit 7 indicates the largest capacitor. Furthermore, the common mode lines `va`/`vb` can be connected to the ADC's

[7] In general, a distinction is made between sample-and-hold and track-and-hold input stage circuits. This will not be discussed further here.

input utilizing transmission gates controlled by `connect input A`/`connect input B`. In the same way, the signals `connect comparator A`/`connect comparator B` control the transmission gates between `va`/`vb` and the comparator input. `va`/`vb` are connected to the electrical ground node when the signals `reset A`/`reset B` are high.

The signal `enable comparator` requests the comparator for a decision until the next clock cycle. In Figures 14 and 15, the actual comparator decision moment is marked by orange dots. Here it becomes clear, that the comparator is triggered using the negative clock edge. The impacts of this decision are discussed in detail in Section 5.5.2.

For this switching scheme it is sufficient if the output code is only valid for one clock cycle, since in the final implementation the output code is buffered by a synchronizer anyhow. This synchronizer allows the PPU or any subsequent data processing and storing logic to access the output code, even it is driven by a different clock.

In the following, we discuss the details of both operation modes with respect to the switching scheme to understand their influence on the requirements for the controller design. The waveforms in Figures 14 and 15 serve as visual guideline.

Fast mode. As already mentioned in Section 2.4, the fast mode interleaves two 7 bit ADCs. Hereby, each full track-and-convert cycle takes at least 16 clock cycles. With the help of the length of `global sample` the tracking period can be extended. However, Figure 14 shows the fastest operation using a 6 clock cycles tracking period.

Before the first conversion or after a time without conversions, both ADC arrays are in their "idle state". In doing so, `va` and `vb` are connected to ground enabling `reset A` and `reset B`. Furthermore, their transmission gates to the input and to the comparators are open. Also, the comparator is disabled and the content of the ADC's output register in not valid. At this time, the positions of the capacitors inverters `array A[7:0]` and `array B[7:0]` do not matter[8].

[8]It turns out that it is important to disconnect the comparator from the common mode line before setting `array A[7:0]` / `array B[7:0]` back to the sampling position. This is because if both operations happen simultaneously, the comparator input node is charged to a random potential which influences the subsequent sample.

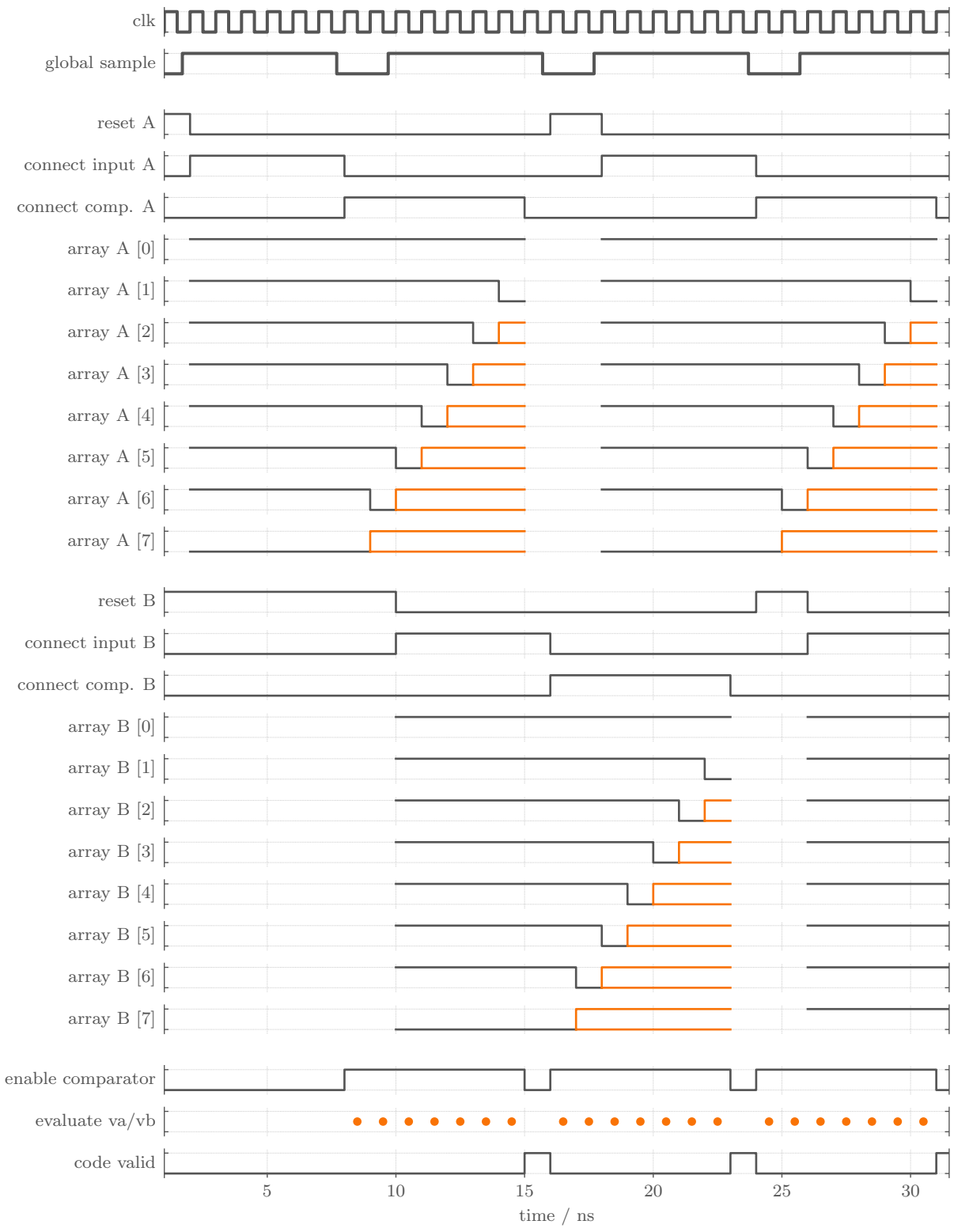


Figure 14: Switching scheme in the ADC's fast mode using a 1 GHz clock and the most dense sampling configuration. Orange traces mark applied comparator decisions. Gaps inside the traces illustrates points, where their actual level doesn't matter.

If after a positive clock edge the controller sees the `global sample` signal high, the capacitors inverters `array A[7:0]` should be already in the 7 bit sampling position. In the 7 bit sampling position, the array's largest capacitor is connected to the supply, which due to the inverter's character equals a low control signal of `array A [7].` The other capacitors in this array are connected to ground by applying a high signal. The `reset A` signal is released and `connect input A` enabled. `Array B` does not change.

Finishing the sampling period of array A, `global sample` is released again. At the following clock edge, instead of `connect input A` the signal `connect comparator A` becomes high. Simultaneously, the logic requests the comparator to evaluate `va` at the next falling clock edge and until the next rising one. At each first decision of a conversion, the switching scheme differs from the discussed try-and-reject approach, as the MSB capacitor is already pre-charged during the sampling period. This shifts the total common mode voltage to the half supply, which comes with relaxations of the comparator's input range characteristics. Moreover, the first comparator decision already determines the output code's MSB and the further position of the `array A [7]`. In Figure 14, this decision result is illustrated using the ambiguous orange trace. Concurrently at the next positive clock edge and due to the try-and-reject approach, the second significant bit (`array A [6]`) refers to the supply. Again, the comparator provides the corresponding decision, which is subsequently applied to `array A [6]` as well as to the output code at the next positive clock edge. Now, this scheme replicates until `array A [2]`, which are so far in total six decisions. The last and seventhsdecision for the output code's LSB applies to the "try" of `array A [1]`. For want of further decisions, here a consecutive "reject" after is not required. With this last decision, the output code becomes valid, and the comparator is disabled again. Array A enters its idle state again, waiting for its next sampling order.

During array A's conversion, also array B is active. At the earliest two clock cycles after `global sample` is low, it can be enabled again. Now, as array A is not in its idle state, this signal applies to array B which behaves in the same way as array A previously. The two clock cycles with `global sample` in a low state are required, to enable the arrays to reset after each conversion. Moreover, it enforces a period, where neither of both arrays are connected to the comparator. This prevents an unintentional discharging of the capacitors. For further illustration, additional conversion cycles are shown in Figure 14.

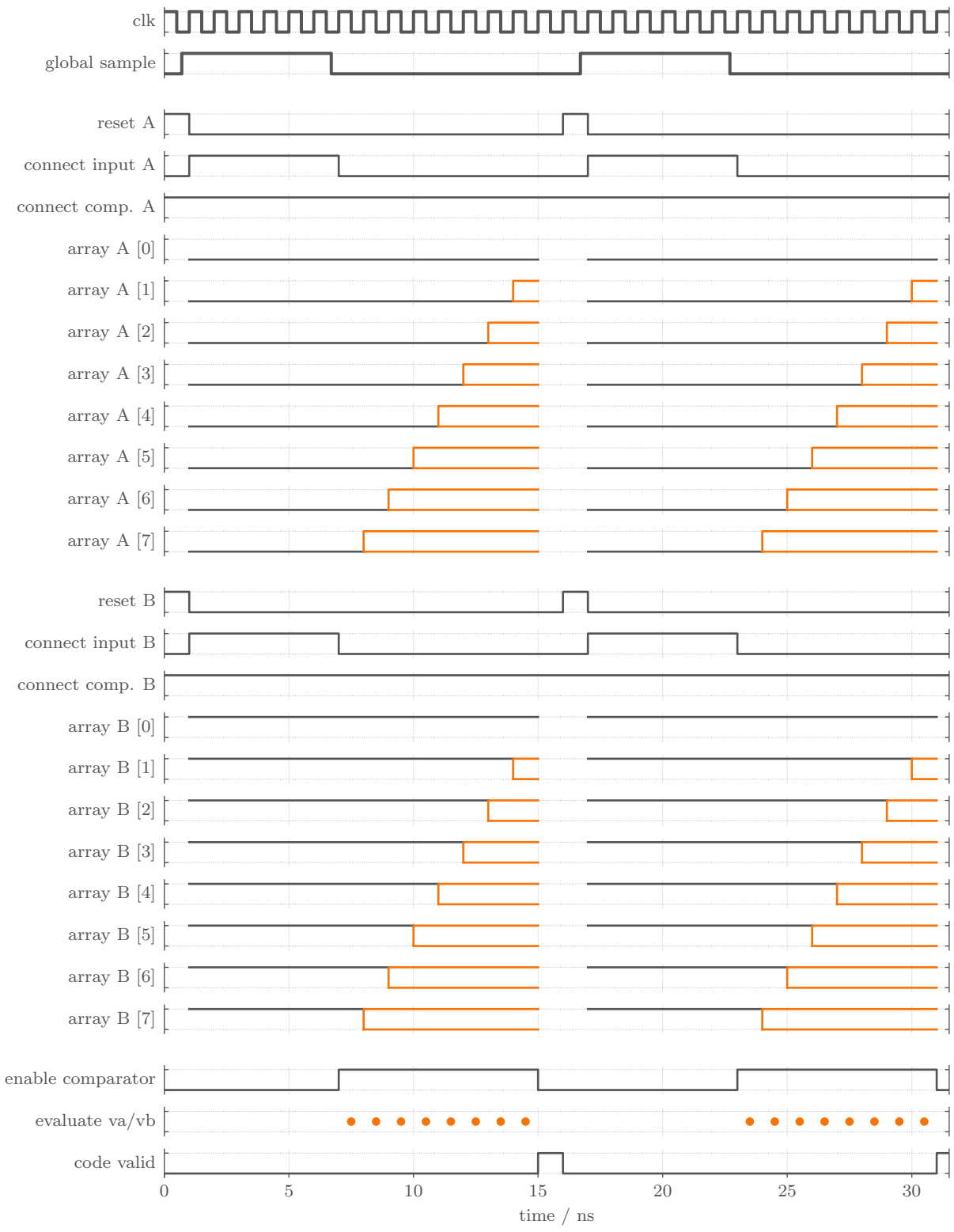


Figure 15: Switching scheme in the ADC's precise mode using a 1 GHz clock and the most dense sampling configuration. Orange traces mark applied comparator decisions. Gaps inside the traces illustrates points, where their actual level doesn't matter..

Precise mode. Similar to the fast mode, Figure 15 illustrates the switching scheme for the precise mode. Again, in its most dense sampling configuration we have 16 clock cycles per conversion cycle. The first difference to Figure 14 is, that both arrays switch in unison. Again, after a period of inactivity, both arrays are connected to ground by enabling `reset A` and `reset B`. Meanwhile, the transmission gates `connect input A` and `connect input B` are open. Since there is no reason to open them here, `connect comparator A` and `connect comparator B` are closed permanently. This prevents the capacitors and signals from additional charge injection caused by a switching. Not shown in the picture, enabling both comparator transmission gate signals together closes also an additional transmission gate between the arrays common mode lines `va` and `vb`. Hereby, if all transmission gates are sized equally the conductance between these quadruples.

Introduced again by `global sample`'s high state, the ADC starts tracking the input and in doing so charge the capacitors in the arrays. Due to the fact, that in the precise mode the ADC does not follow the "try-and-reject" principle, the arrays' sampling positions differ: While all capacitors in `array A` refer to the supply, the capacitors in `array B` refer to ground.

When `global sample` is low again, the actual conversion starts. For this purpose, inputs are disconnected again, and the comparator is enabled. The first decision holds for the output codes MSB as it registers whether the sampled voltage is greater than or less the half supply voltage. The same applies to the inverter setting: While before the two largest capacitors referred to different potentials, after the decision both are connected to the same potential at their bottoms. On the one side, if the input is greater the half supply, also the `array A [7]`-capacitor is connected to ground (high signal); the `array B [7]`-capacitor keeps referred to ground (high signal as well). On the other, if the input signal is less than the half supply, the `array B [7]`-capacitor adds charge, and hence increases the arrays common mode voltage, by switching to a low signal. Here, `array A [7]` would stay low.

Since the capacitors are binary scaled, repeating this scheme results in a successive approximation of the half supply by the common mode voltage. Along the way, the stored comparator decisions represent the primer sampled voltage in a digital code. The LSB decision happens after the signals `array A [1]` and `array B [1]` is adopted. After the LSB decision, which is not needed to be applied to the capacitor array, the output code is finally valid. At the same time, the arrays are reset by enabling `reset A` and `reset B` again. From here, the hole sequence can start again.

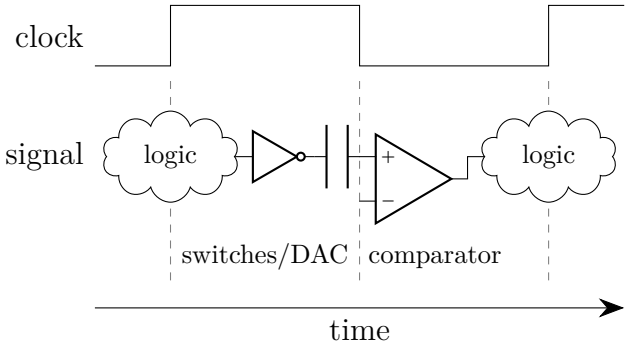


Figure 16: Time sequence in a clock cycle illustrating the temporal and causal dependencies in the ADC during conversion.

Time sequences in a clock cycle. The control logic is no less time critical than the comparator, switches and DAC. This becomes particularly clear when looking at Figure 16, which shows the division of a clock cycle during conversion. Here the signal path is matched in a stylized way with the time sequence. Starting at a positive clock edge, the arrival time of a signal at the output of the FSM Y^t determines when the switches and DAC can start to operate. This time is influenced by the clock-to-output time of the FFs and, in the general case, the processing time of function $g(Z^t, X^t)$. The DAC itself needs some additional time to settle down and apply the new configuration correctly to the input of the comparator. Here, an estimation by means of RC times, as done in Czierlinski (2022), is useful.

The comparator is triggered with the negative clock edge. This marks the beginning of the second half of the clock cycle. However, the trigger-tree (c.f. Section 5.5.2) must also be taken into account. The output signal of the comparator runs as new input X^{t+1} for the FSM through its next-state logic $f(X^{t+1}, Z^t)$ to be taken over by the subsequent FF while respecting the setup time. It is therefore important to keep the logic on this data path as lean as possible.

4.3 Implementation of a dual-mode SAR

The task of the controller is to implement the presented successive-approximation switching scheme. Generally, the controller can be considered as a large FSM covering both arrays and both modes. To ensure that the entire switching sequence has unique, deterministic state transitions, the entire state space is larger than the number of switches and output signals. Ultimately, the following consideration is about finding a practical, unambiguous representation of all states. Thereby the total space spanned by sequential

logic can be larger than the number of valid and accessible states.

Partitioning of the state space. To better reflect the nature of the new ADC, it is useful to divide the state space of this state machine into orthogonal sub-state spaces. In other words, we search for projections of the states that map these onto sequential logic. By using different projections, one can shift the ratio of combinatorial to sequential logic. Three axes are of particular interest here: first, the division into fast and slow modes; second, the individual consideration of the two arrays; and third, the separation of a data path from a control path.

The division of the state space into a fast and a slow mode is done by the external `fast_mode` signal. Thus, it does not describe an axis through the state space in the sense of the memory bits but implements the Mealy-machine principle. Nevertheless, it should be mentioned here. In fast mode, the two arrays operate as quasi-independent ADCs. To reduce even-odd effects between the arrays, the control for both arrays should be as identical as possible. If the states are represented so that the output signals for the two arrays can be generated orthogonally to each other, this simplifies the logic. Separating data and control paths means that signals that depend on a comparator decisions are treated differently than the rest. This is motivated by the fact that the potentially undefined comparator output should not cause the FSM to get stuck.

The mentioned subdivisions increases the number of flip-flops needed. On the other hand, they make the design more robust and easier to maintain. It should be noted that the output signals of the logic Y^t is not evaluated like synchronous logic, but continuous-time. The signals for the transmission gates and the inverters at the arrays must not have any glitches and must comply with the timing specifications from Section 4.2.

We present a sophisticated control logic that reduces the number of FFs as much as is practical. However, since we find that this comes with many drawbacks, this is followed by a discussion of a relatively naive but efficient control logic. To better understand the details of the implementations, we start with some general remarks about the state space and how parts of it can be represented. To do this, let's take a closer look at the switching sequence from Section 4.2.

Pattern in the switching scheme. In the fast as well as in the precise mode, we find almost similar pattern: As illustrated in Figure 17, both modes start in a "reset" situation (R) connecting the common mode lines `va`/`vb` to ground. The `global sample` enables

either both or only one array to move on to a tracking symbol (T) connecting the respective array(s) with the ADC input. This requires a mode specific sampling position at `array A[7:0]/array B[7:0]`. When `global sample` is low again, these arrays step forward into a conversion mode. Here, the labels *write register A/write register B* in Figure 17 notify, which capacitor array inverter is effected by the next comparator decision. However, due to the "try-and-reject" approach in the fast mode, before each decision the capacitor with the denoted index is connected to the supply. As after the LSB decision the inverters positions doesn't matter, we can use them anyhow to store the output codes LSB. Due to the reduced number of bits in the fast mode, *write register A/write register B* only goes down to one; in the precise mode it finally counts to zero. After seven or eight clock cycles, we recognize one clock cycle of a finish symbol (F), which sets the output code to valid. We can further observe, that if at least one array is in its conversion mode (C), the comparator is enabled. In the fast modes, these states of both arrays are interleaved while in the precise mode these sequences run in parallel. By this grouping or projection of the different states it becomes clear that there are memory elements needed which control `array A[7:0]/array B[7:0]`, others which control the transmission gates and some kind of counting logic.

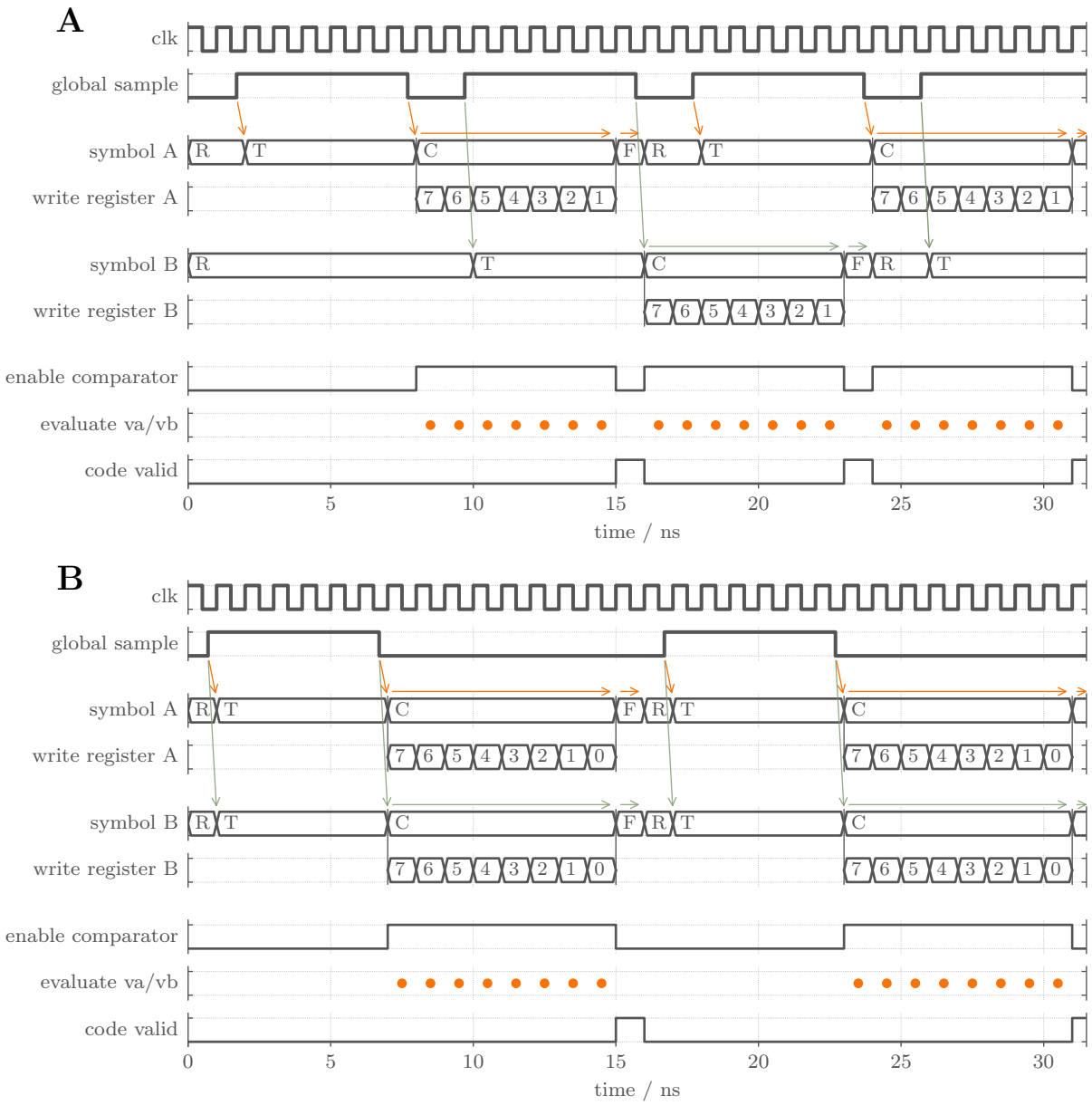


Figure 17: Wave forms illustrating pattern in the logic that controls the successive-approximation register. **A**: The fast mode is realized by interleaving the both arrays. **B**: In the precise mode, both arrays operate together. The symbols are abbreviated by (R) reset, (T) track, (C) conversion and (F) finish and corresponds to the positions of the transmission gates `reset`, `connect input` and `connect comparator`. The arrows mark causal relations.

4.3.1 Sophisticated, flip-flop reduced controller

By analyzing the possible states and its transitions, we find that the convert counter only applies the one array at a time or both simultaneously. Thus, also the output vectors `array A` and `array B` act dynamical subsequently or simultaneously. This in mind, we

end up with a state space spanned by:

$$Z\left[14:0\right] = \left\{\hat{Z}_{\text{cnt}}\left[2:0\right], \hat{Z}_{\text{ctr A}}\left[1:0\right], \hat{Z}_{\text{ctr B}}\left[1:0\right], \hat{Z}_{\text{dout}}\left[7:0\right]\right\} \tag{4.1}$$

Here $\hat{Z}_{\text{dout}}[7:0]$ holds the comparator decisions and therefore controls `array A[7:0]`/ `array B[7:0]`, while $\hat{Z}_{\text{ctr A/ctr B}}[1:0]$ is the transmission gate part of the state representation. $\hat{Z}_{\text{cnt}}$ is the memory of one common counting logic. For this implementation, the counting functionality as well as the control memory for the DACs are shared. The implementation is realized by a handcrafted Moore machine.

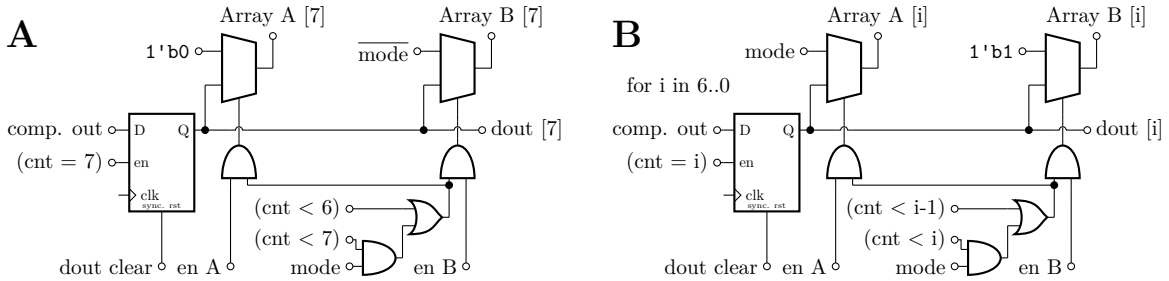


Figure 18: Data path in the flip-flop reduced controller for the inverter signals `Array A [7:0]` and `Array B [7:0]` using one shared register bank. The logic is driven by a counter "cnt", the ADC operation mode as well as "en A" or "en B". To accommodate both modes, the MSB capacitor in **A** has a different sampling position logic than the other capacitors $\mathrm{i} = 6 \ldots 0$ in **B**.

The limitations of this implementation can be shown best by the signal path between the comparator and `array A`/`array B` (c.f. Figure 18). As shown in Figure 17(A), in the fast mode, always only one array ever changes, while the other remains constant. In the precise mode, either both arrays change due to the comparator output or both outputs are constant. This leads to the idea of implementing only one FF per output code bit. However, to be able to have `array A[7:0]` and `array B[7:0]` in two independent positions (e.g., `array A[7:0]` takes its sampling position while `array B[7:0]` converts), we implement two demultiplexer banks in front of `array A[7:0]` and `array B[7:0]`. A demultiplexer has at least two input signals, one of which it selects as output. Here, each demultiplexer connects `array A[i]`/`array B[i]` either to its constant sampling position or to the dynamic $\hat{Z}_{\text{dout}}[i]$ FF. The sampling position can be considered as constant, because the ADC `mode` signal only changes when the ADC is switched off. On a strict interpretation, however, we would recognize a Mealy machine here. So in fast mode (`mode` is high) these are high except for those for the MSB capacitors. In the precise mode (`mode`

is low) `array A[7:0]` is low and `array B[7:0]` is high.

In each conversion step exactly one enable-FF ($\hat{Z}_{\text{dout}}[i]$) copies the comparator output. The corresponding FF is selected from the counter $\hat{Z}_{\text{cnt}}$.

The change between the input modes are driven by the counter's inverse temperature output. In other word, it is driven by a "smaller than"-logic. However, this is preceded by the fact that $\hat{Z}_{\text{ctr A/ctr B}}[1:0]$ of the respective array must be show the symbol "convert". Now, if an array is enabled, one bit after the other changes from its constant input to the output of the FF. This realizes the staircase behavior in the switching scheme as discussed in Section 4.2. In order to achieve the "try-and-reject" approach, the array takes the input from the register until `cnt` $< i$. So each inverter is already connected to the respective FF one clock cycle before this FF takes over the comparator output. Since the FFs are zeroed before each conversion, this ensures that the respective capacitor is first recharged against the supply. In the precise mode, the array takes the input from the register until `cnt` $< i - 1$. Here, the change from the static to the dynamic input happens after or during the FF copies the comparator output.

This implementation does pay attention to the signal path between the FF and the respective inverter, but the critical path is the one that is created by the selection logic. This starts at the counter and leads to the control port of the demultiplexer via the less-than logic. By the way, during the synthesis it is not at all guaranteed that the demultiplexer really becomes a demultiplexer, because the described functionality can also be generated differently. This attempt is yet promising, as without constraining, we achieve an area consumption of $\approx$350 µm^2. But in the end, the physical aware synthesis tries to keep the time constraints and uses large as well as power hungry standard cells. Using a 1 MHz target clock frequency and a wire-load model at a typical corner for the static timing analysis (STA) together with the load constraints of the parasitic extraction, this controller requires an area of $\approx$750 µm^2.

This implementation also has the disadvantage that it generates all outputs by combinatorial logic. As already discussed, this is dangerous, since glitches can arise through the synthesis. Although gray-like encoding is used to control the transmission gates, the same is not possible for `array A[7:0]`/`array B[7:0]`.

4.3.2 Naive but efficient controller

Given the switching scheme from Section 4.2 and the regarding the symmetry and the control and data signal splitting, one can naively implement the controller as (simple) Moore machine splitting the state space in two individual blocks, each containing the logic of one array. In order to separate data and control path, we perform the following split of the state space:

$$Z\left[23:0\right] = \left\{\hat{Z}_{\text{control, A}}\left[3:0\right], \hat{Z}_{\text{data, A}}\left[7:0\right], \hat{Z}_{\text{control, B}}\left[3:0\right], \hat{Z}_{\text{data, B}}\left[7:0\right]\right\} \tag{4.2}$$

It is directly obvious that the total number of memory bits is larger than in the example discussed above. The important point is, that no information flows from the data-part to the control-part of the state representation: It runs autonomously and does not require any comparator decision.

Similar to a simple Moore machine, the data dependent part $\hat{Z}_{\text{data, A}}[7:0]$ directly controls `array A[7:0]` and $\hat{Z}_{\text{data, B}}[7:0]$ controls `array B[7:0]`. By doing so, glitches are prohibited intrinsically, and the timing constraints are met easily. But the signal-parts symbols change with respect to the current value in the control part: They set them back to their sampling state or execute the "try"-part in the fast mode. To apply the successive approximation, accordingly to the current (convert counting) symbol in the control part, the individual bits of the signal-part copy the respective comparator output.

The counting logic and the transmission gates control are incorporated to $\hat{Z}_{\text{control, A/B}}$. Here, sequences from Figure 17 are decoded as shown in Table 5. To avoid glitches in the output vector Y, all critical transitions are implemented utilizing the ideal of a gray-counter: The order of the codes always remains the same, only in the fast mode, one jumps from C1 directly to FINISH. Their transitions are on the one hand driven by the array-specific "sample" signal, on the other they operate as rigid clock-driven sequential.

Implementation wise, these naive approach results in for example two `case` blocks per array: One for the control parts next-state logic and one to write the data-part. To fully describe the controller, some more combinatorial logic is required. In the fast mode, both arrays are externally interleaved by combinatorial logic. Therefore, the sample signal that triggers the ADC always shows up in "A", but in "B" only if the ADC is in the precise mode or if "A" is not in RESET or TRACK. Moreover, in the precise mode, the comparator is always connected to both arrays. However, the `enable comparator` signal is high, if one of $\hat{Z}_{\text{control, A}}[3]$ and $\hat{Z}_{\text{control, B}}[3]$ is high. The output code always shows the

Name	Code	R	T	C	F	cnt
RESET	4'b0000	1	0	0	0	
TRACK	4'b0001	0	1	0	0	
C7	4'b1001	0	0	1	0	7
C6	4'b1000	0	0	1	0	6
C5	4'b1100	0	0	1	0	5
C4	4'b1101	0	0	1	0	4
C3	4'b1111	0	0	1	0	3
C2	4'b1110	0	0	1	0	2
C1	4'b1010	0	0	1	0	1
C0	4'b1011	0	0	1	0	0
FINISH	4'b0011	0	0	0	1	

Table 5: Almost gray encoding of $\hat{Z}_{\text{control, A}}$ and $\hat{Z}_{\text{control, B}}$ together with its impact on the corresponding transmission gates `reset` (R), `connect input` (T), `connect comparator` (C) as well as an indication for a valid flag (F) and an associated counter (cnt). In a conversion, the sequence is executed from top to bottom. The red numbers annotate the changing places: Only in case of critical code transitions do several FFs change at once.

state of $\hat{Z}_{\text{data, A}}[7:0]$, only if $\hat{Z}_{\text{control, B}} = \text{FINISH}$ it selects $\hat{Z}_{\text{data, B}}[7:0]$. If $\hat{Z}_{\text{control, A}}$ or $\hat{Z}_{\text{control, B}}$ shows FINISH, the output code is valid. To prevent the state machine from getting stuck, $\hat{Z}_{\text{control, A/B}}$ automatically returns to RESET if a code is encountered that is not described in Table 5. This controller enables, to disconnect the comparator in the fast mode first, before resetting the common mode line. By doing so, at the start of a conversion the comparator's input node is always pre-charged to around 0.6 V.

4.3.3 Physical aware syntheses

For a realistic timing of the ADC, the digital logic needs to be cast to physical transistors. Here we used *Synopsis Design Compiler* to perform the synthesis. To account for the full custom part of the ADC, we apply the port capacitance determined in a paracitic extraction (PEX) of the layout as constraints to the synthesis. Using a 1 MHz target clock frequency, we estimate the maximum allowed time for the logic to generate an array-inverter or transmission gate signal to 300 ps. Consistently with the rest of the ADC, we use a wire-load model at a typical corner for the STA. No clock or reset tree is created in this synthesis.

Despite the increased number of flip-flops, the area required for the 158 standard cells is 482 µm^2. The leakage currents in the synthesized control logic have a power of 19.97 nW. In the fast mode, the digital controller requires 1.62 ± 0.02 pJ/conv, in the precise mode

it requires $3.00 \pm 0.02\,\mathrm{pJ/conv}$ (c.f. Section 6.2). The uncertainty here refers to its input voltage dependence. The increased energy consumption in the precise mode is due to the fact that both arrays work simultaneously and there is also one more switching operation.

The designs functional verification is performed with the verification framework from Section 3.3. As the test suite drives smoothly through the full input voltage range, each state transition in the comparator is tested. Since this incorporates the full ADC, it is presented in Section 6.2.

5 Double-tail sense amplifier

As its name suggests, a comparator compares the magnitudes of two analog quantities, e.g., two voltages, and returns the result as a binary signal. From an abstract point of view, it implements a discretization by introducing a decision boundary at the parity of the input signals. If $V_{i+} > V_{i-}$, the comparator yields a high signal, if it's the other way around, the output signal of the comparator is low. Every ADC architecture utilizes comparators (Pelgrom, 2017).

Conceptually, one distinguishes regenerative and time continuous comparators. The latter are ordinarily implemented as high-gain amplifiers most often centered around vanilla, continuously biased differential pairs. Due to its permanent power consumption, this approach would not fit into the new ADC design. Moreover, it requires a bias current whose provisioning of which requires further effort. Sense amplifier flip-flops (SAFFs) were initially designed for symbol identification in digital circuits, such as for example in on-chip transceivers (Schinkel et al., 2005; Zhang et al., 2000) or level detectors in SRAMs (Hsu and Ho, 2004; Chandankhede et al., 2014). However, they also proved to be good comparator circuits for different ADCs architectures (Xu and Xu, 2017; Gao et al., 2010; Naraghi et al., 2010; Reddy et al., 2012). Their clocked decision-making process and digital output characteristics make them the circuits of choice for fast, low-power ADCs.

5.1 Introduction to comparators

The new SAR ADC engage one comparator to evaluate the voltage on its capacitors. Its control logic fetches the comparator decision and computes the next step of the successive approximation from this. During a conversion, each comparator decision is responsible for one bit of the output code. This makes the comparator critical for a successful ADC implementation. In the following section, we introduce concepts, properties and characteristics that are required in comparator design.

5.1.1 Effects in the transfer curve

The transfer curve describes the output-level of a comparator with respect to an input voltage difference. Ideal characteristics (Figure 19A) exhibits an infinitely steep output-level transition at the zero crossing of the two input voltages. In reality, however, comparators can suffer from different types of distortions which we are going to discuss in the following

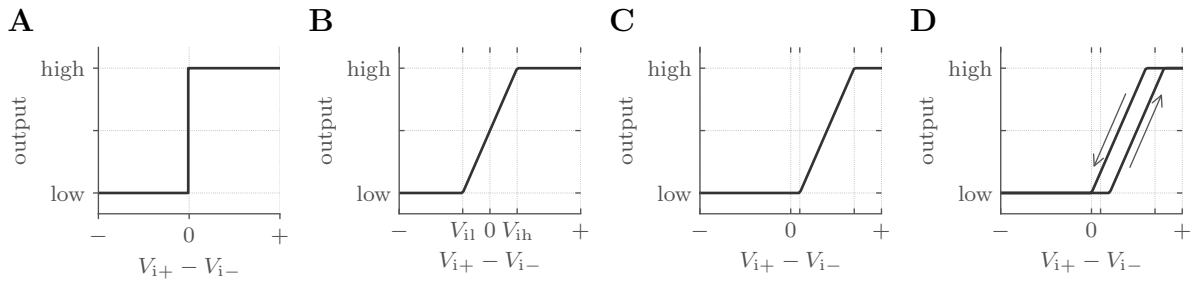


Figure 19: Comparator transfer curves for ideal comparators and distorting effects. **A**: Ideal comparator characteristics. **B**: Comparator with partially digitally undefined output levels. **C**: Effect of an input offset to the transfer curve. **D**: Comparator's transfer curve showing a path-dependency (hysteresis).

paragraphs.

In an ADC, the comparator's output-level is evaluated by subsequent digital logic. If during this evaluation the comparator's output signal is neither in the low-state nor in the high-state, the signal is digitally undefined which can cause metastability. Metastability describes the situation in which a flip-flop does not reach a stable equilibrium but persists in a meta-stable intermediate state. Figure 19B exhibits a comparator with a partially digitally undefined output state: Small input voltage differences are not sufficient to yield a full swing at the output.

Comparators can, furthermore, suffer from a static offset in the transfer curve. As illustrated in Figure 19C, a static offset shifts the decision boundary to non-zero input voltage differences. This can be caused, e.g., by to device or input impedance mismatch. There are multiple approaches to calibrate for static offset (Wong and Yang, 2004; Miyahara et al., 2008). However, dynamic offset errors, for example caused by noise, add uncertainty in the form of a washed-out tripping point in the transfer curve.

A comparator's decision can, in addition, become path dependent (Figure 19D). This means that the observed decision boundary depends on the direction from which you approximate it. This effect is called hysteresis.

5.1.2 Differential pair

A differential pair is a common circuit to amplify a voltage difference. Many comparators use this or modified version to amplify their input signal. As illustrated in Figure 20A, it is based on the concept that the sum of the currents through the left and the right branch equals the current trough a common bias source:

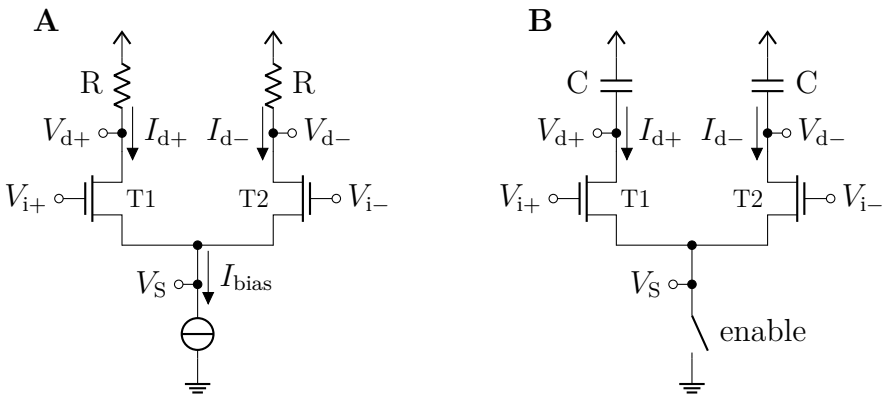


Figure 20: Illustration of the fundamental difference between the two basic types of differential pairs. **A**: A conventional, continuous-time differential pair uses a constant bias current. **B**: A regenerative differential pair driven by pre-charged capacitive elements does not require it.

$$I_{\text{bias}} = I_{\text{d+}} + I_{\text{d-}}. \tag{5.1}$$

Now, the left ($I_{\text{d+}}$) and the right ($I_{\text{d-}}$) branches are driven by the gates of the transistors T1 and T2 and hence by the input voltages $V_{\text{i+}}$ and $V_{\text{i-}}$. Since the two identical resistors R serve as loads, a difference between $I_{\text{d+}}$ and $I_{\text{d-}}$ cause a voltage difference between $V_{\text{d+}}$ and $V_{\text{d-}}$:

$$V_{\text{d+}} - V_{\text{d-}} = R \cdot \left(I_{\text{d+}}(V_{\text{i+}}, V_{\text{S}}) - I_{\text{d-}}(V_{\text{i-}}, V_{\text{S}})\right). \tag{5.2}$$

To handle the constant power consumption in a differential pair, one can reduce the cross current through T1 and T2 to the moment of the decision-making process. One way to do this is to use a regenerative pre-amplifier (Pelgrom, 2017). Here, we replace the resistors R by pre-charged capacitors C (c.f. Figure 20B). When enabling the common source current, both capacitors discharge with input voltage dependent slew-rates:

$$\text{slew-rate} = \frac{\mathrm{d}V_{\text{d}}}{\mathrm{d}t} = \frac{I_{\text{d}}(V_{\text{i}})}{C} \tag{5.3}$$

The voltages $V_{\text{d+}}(V_{\text{i+}}, t)$ and $V_{\text{d-}}(V_{\text{i-}}, t)$ become time dependent and only differ during the decision-making process. Following stages generate static signal from the slew-rates of the two branches. For example, a sense amplifier make use of two cross-coupled inverters. Before the decision, these are brought to a state of an unstable tipping point. During the decision, the input-depend slew-rates of the two branches than guide the cross coupled inverters to an equilibrium.

In each decision-making process, the transistors T1 and T2 pass through different operation modes: Before the differential pair is enabled, V_{d+} and V_{d-} are pre-charged to the supply voltage. Input voltage depended, T1 and T2 are usually not fully off but operate in their deep-sub-threshold, which slowly also pre-charges the node V_S. The differential pair is activated by closing the enable-switch. Thereby, the enable-switch forces V_S to ground. Now, T1 and T2 discharge V_{d+} and V_{d-}. As long as $V_{d+/-} > V_{i+/-} - V_{th}$, the channel of T1/T2 pinch-off (saturation regime) and the discharging current $I_{d+/-}$ is almost independent of the capacitor's discharging state. This is the phase in which the actual comparator decision is prepared. If $V_{d+/-} < V_{i+/-} - V_{th}$, e.g., because the capacitor is almost discharged, the transistor enter its linear regime. This means that the influence of the input voltage $V_{i+/-}$ on the discharge current $I_{d+/-}$ decreases. If the input voltage $V_{i+/-}$ has no sufficient magnitude, e.g., it is smaller than the threshold voltage V_{th}, the discharging current $I_{d+/-}$ is driven by the Boltzmann statistics. Here, the discharging current $I_{d+/-}$ is significantly reduced compared to the other regimes, and it takes longer to discharge the capacitors.

5.1.3 Error sources in comparators

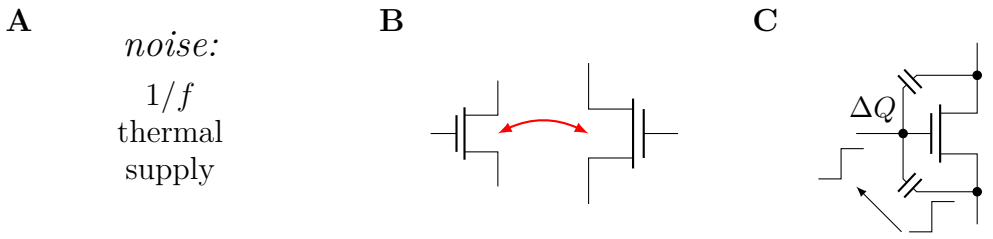


Figure 21: Exemplary sources of error in comparators. **A**: It is difficult to calibrate away the different types of noise, so the design itself should be robust against them. **B**: Device mismatch often lead to random static offset errors. **C**: Kickback in comparators is caused by charge injection in the differential pair.

Every circuit is subject to sources of error. First, we can identify different external or fundamental sources of noise: 1/f noise, thermal noise and supply noise. In addition, the manufacturing process causes device imperfections: Mismatch in transistor parameters. Finally, the design itself limits its accuracy: Capacitive coupling and the influence of the input impedance.

Noise. At temperatures above 0 K, charge carrier also move in equilibrium. This random movement is inherent to the statistic nature of thermal energy and leads to a slightly

fluctuating voltage in all electrical circuits. In addition, pink noise, also called 1/f noise, occurs in many electronic devices (Allen and Holberg, 1987). In a first approximation, if the input impedance R dominates the noise, the root-mean-square of this offset voltage is determined to be:

$$v_{\mathrm{RMS}} = \sqrt{4k_B T R \Delta f}. \tag{5.4}$$

Here k_B equals the Boltzmann constant, T the temperature and Δf the bandwidth (Pelgrom, 2017).

A comparator, moreover, can suffer from supply noise. Dynamic change in the supply voltage can lead to variations in its circuit's behavior. The noise components can act like a dynamic time-dependent offset, which can not be calibrated. In order to suppress this anyhow, the design itself needs to be robust against noise.

Device mismatch. The manufacturing process leads to imperfections that further limit the comparator's accuracy. With them in mind, one can optimize a design to become sufficiently robust. Unfortunately, typical comparators consist of multiple devices that should be as equal as possible.

Device mismatch occurs either due to geometrical differences or due to changes in process parameters like silicon doping concentrations. The former can be caused systematically by the approximate effect (Neighboring poly-silicon changes the geometry of implanted areas) or statistically by the limit of the process node. Generally, symmetry in design and layout helps to reduce these effects. Changes in silicon doping concentrations and variations of the gate-oxides thickness leads to threshold voltage mismatch. Pelgrom et al. (1989) measures the current factor mismatch in transistors as

$$\frac{\sigma^2(\beta)}{\beta^2} \approx \frac{A_\beta^2}{WL} + S_\beta^2 D^2. \tag{5.5}$$

Here, A_β and S_β are process parameters, while W and L are width and length of a rectangular transistor and D the distance to the neighboring devices.

With this the transistors properties like the transconductance also change. Using error propagation in a linearized transistor model, one can calculate the total effect on the comparator (Pelgrom, 2017, p. 296). Unfortunately, transistors in comparators with regenerative stages, as for example sense amplifiers (SAs), cross through their full dynamic range and hence do not fit a linear approximation. In first approximation, device mis-

match leads to a random static comparator offset, which is a second order effect in SAR ADCs.

Kickback. Kickback is an effect in comparators that comes with the intrinsic properties of transistors. As illustrated in Figure 21C, a transistor's gate does not only control the drain-source current but also implements a capacitive element with respect to the drain- and source-potentials. This capacitance is in first order linearly related to the transistor's gate area, but in second order also gate voltage dependent. While in a common differential pair the voltages $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ only change with the input signals $V_{\mathrm{i}+}$ and $V_{\mathrm{i}-}$, in a regenerative comparator $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ cross through their full dynamic range in every decision (c.f. Figure 20). This results in a significant charge transfer on the input lines $V_{\mathrm{i}+}$ and $V_{\mathrm{i}-}$. Pre-charging $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ stage again, approximately reverses this charge transfer. Thus, the comparator's input impedance matching influences the decision and leads to another random static offset. Since decreasing the gate area at constant transconductance properties increases the device mismatch, we need to find a sufficient trade-off between kickback and device mismatch.

5.2 Design constraints

To reach the goal of a flexible, fast and energy as well as area efficient ADC, all components must be coordinated with each other. Given the targets in Section 2.3, Czierlinski (2022) presented a suitable ADC concept (c.f. Section 2.4). Having said this, we can derive a fist set of specifications for a comparator.

The comparator should fit the ADC's fast mode as well as the precise mode. While the required accuracy differs, both modes operate in a single ended configuration comparing a full-swing signal (typically 0.0 V to 1.2 V) with half the supply voltage (typically 0.6 V). In the approximation algorithm, the latter serves as common mode voltage that is successively approximated by the DAC (Czierlinski, 2022). Hence, a change in this voltage is equal to a static input offset of the comparator and equivalent to an offset error in the ADC's output code (Pelgrom, 2017). Although this is a second order effect, we want to be able to calibrate for it by implementing an adjustable reference voltage generator. Moreover, a deliberate shift of the comparators decision boundary allows compensating for ADC gain and offset error by shifting the dynamic range. In this sense, the full comparator circuit should operate in a single-ended configuration.

An ideal 7 bit (8 bit) ADC with 1.2 V dynamic range has a LSB-step-size of 9.38 mV

(4.69 mV). Even if a static offset has a second order effect, effects that equal a time dependent offset or metastability should be suppressed. For flash ADCs, Pelgrom (2017, p. 295) suggests that the comparator's input offset should be five to ten times smaller than the LSB-step-size. Following this rule of thumb, we base our design on a minimum input voltage difference of 1.0 mV.

The given ADC architecture also determines the comparator's input impedance: Ultimately, the capacitive array implements an infinite impedance with a certain reservoir of charge. But as the comparator is separated from the capacitive arrays by transmission gates and the comparators input capacitance is small relative to the DAC's capacitance, we also need to consider them.

Delay time. As mentioned before, the comparator should base on a regenerative approach and thus requires a trigger signal to enable the differential pair. A sense amplifier holds the output signal in two cross-coupled inverters. In this sense, a sense amplifier provides two output signals (V_{o+} and V_{o-}), one from each inverter. The trigger as well as this output signal interface to the digital domain. To characterize a comparator design, we motivate a timing relation both signals. Inspired by the commonly used clock-to-output time in sequential logic like FFs, we define a trigger-to-output time: It starts at the half trigger edge (50 % of the supply voltage) and stops if one of the two output signals reaches 90 % of the supply voltage.

The ADC is driven by a 1 GHz clock, which should also serve as source for a comparator trigger signal. We decided to divide each clock cycle in two parts: At positive clock edges, the combinatorial logic generates array-inverter transitions (c.f. Section 4.2) that adapt the sampled voltage. At negative clock edges, the comparator is enabled makes a decision. So, we make use of both clock edges and need to account for a probably unsymmetrical clock duty cycle. In a typical case this would stop a comparator decision after ≈500 ps. Including the required FF hold time, we assume a maximum of ≈400 ps at 1.0 mV input voltage difference to be sufficient. Hence, the maximum trigger-to-output time should be less.

Power and area consumption. The comparator is one of the most power hungry circuits in the ADC. In order to stay within the required total power constraints, a first approximation of the core power of the comparator should stay below 100 fJ per decision. If the comparator ran constantly, this would equal a power of 100 µW, which is more than the half the ADC's total power consumption.

In order to fit to the area constraints, it was decided to accommodate the entire comparator unit, including the reference circuits, below the capacitive array. However, this influences the number of available metal routing layers: To save routing resources, the comparator should be as flat as possible.

5.3 Sense amplifier

Originally SA FFs are designed for high speed, low power applications in the digital domain. But especially voltage mode SAs are also interesting as comparators in ADCs. Today, there are many variations and implementations of sense amplifier (Zhu et al., 2013).

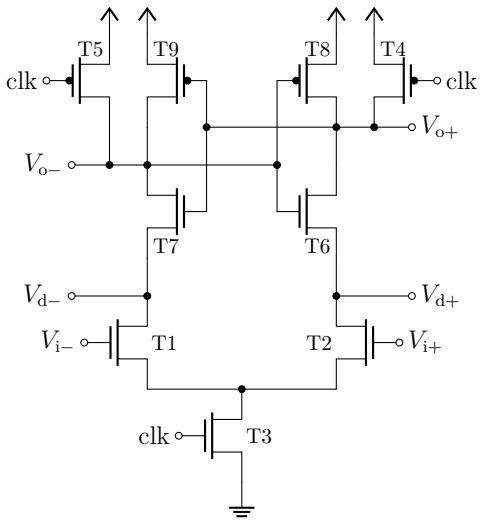


Figure 22: Circuit schematic of a classic single tail sense amplifier flip-flop.

Commonly, all SAFFs use a capacitive, regenerative differential input stage (c.f. Figure 20B) and some kind of cross coupled inverters as latch stage. Figure 22 illustrates a simple single-tail SA. Here, transistors T1 and T2 realize a differential pair, biased by the N-type MOS (NMOS) switch T3. The lines V_{d+} and V_{d-} serve as implicit capacitors. On top of this differential pair the transistors T6 to T9 implement two cross coupled inverters. The latch created in this way can be reset by transistors T4 and T5. Hereby, the reset transistors T4 and T5 become conducting (clock is low) and pulls V_{o+} and V_{o-} to the supply rail. This again pushes the NMOS transistors of the inverters, T6 and T7, to pre-charge V_{d+} as well as V_{d-}. Since the clock is still low, switch T3 is open as no cross current flows through the branches.

At a positive trigger edge, switch T3 closes and simultaneously the reset T4 and T5 is

released. The charges stored on V_{o+} and V_{o-} as well as on V_{d+} and V_{d-} drain via T1 and T2. Input voltage dependent, the drain currents $I_{d+}(V_{i+})$ and$I_{d-}(V_{i-})$ (c.f. Figure 20B) differ, which is amplified by the positive feedback of the cross-coupled inverters. Here, either T8 or T9 becomes conducing while either T7 or T6 opens. At the output V_{o+} and V_{o-} a full swing signal occurs.

In this configuration, only one common trigger signal is required. So both stages are intrinsically synchronized. However, due to the three stacked NMOS transistors, T1 and T2 suffer from a reduced headroom: The primer pre-charge of V_{d+} and V_{d-} is not the full supply voltage but limited by the conductance of T6 and T7. Hence, this design is not suitable for the required common mode voltage.

Several variations of this circuit attempt to address the problem of poor dynamic range. Some variations retain one tail, while others deviate from this approach and outsource the second stage. In the former, for example, V_{d+} and V_{d-} are explicitly pre-charged during reset by additional P-type MOS (PMOS) transistors. Further both branches can be connected using a small additional transistor to pre- and discharge both branches input-independently in every trigger cycle (Nikolic et al., 2000).

5.4 Implementation of a double-tail sense amplifier

Double-tail sense amplifier (DTSA), for the first time presented by Schinkel et al. (2007), deal with the limited headroom by moving the cross-coupled inverters to a second tail. Thus, both stages can operate with full swing voltages. As shown in Figure 23A, V_{d+} and V_{d-} are charged by two the PMOS transistors T4 and T5 replacing the cross-coupled inverters.

Today, there exists multiple variation of the original double-tail concept, which vary in the exact coupling of the latch stage. One approach is to use a conducting stage coupling (Van Elzakker et al., 2008): Here the corresponding NMOS transistors are inserted between T8 and T10 as well as T9 and T11 (c.f. Figure 53 in appendix). In order to remove the need for an inverted trigger signal, Miyahara et al. (2008) derive the latch-enable signal from V_{d+} and V_{d-} (c.f. Figure 54 in appendix). Furthermore, one can introduce another inverting stage between the differential pre-amplifier and the output latch (Jeon and Kim, 2010).

A first comparison of the individual double-tail variants was made based on simulations not yet including optimizations of transistor sizing: Schinkel et al. (2007), Miyahara et al.

(2008) and Jeon and Kim (2010) exhibit a similar level of performance with respect to their energy consumption and decision delay. The original DTSA, however, seems to be more robust against device mismatch. In addition, it is already successfully implemented in other SAR ADCs and in similar technologies, e.g. in Harpe (2018).

Going into future detail, in the original DTSA, both stages are coupled in a voltage mode fashion. The voltages $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ are directly applied to the gates of T13 and T12, inversely driving the branches of the cross-coupled inverters. As a side effect, T12 and T13 also serve as additional capacitive elements at $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$.

As long as `trigger` is low, T4 and T5 are closed and $V_{\mathrm{d}-}$ as well as $V_{\mathrm{d}+}$ are pulled towards the supply rail. This charges these implicit capacitors and forces $V_{\mathrm{o}-}$ and $V_{\mathrm{o}+}$ to ground. At this point, transistor T6 prevents the latch from producing large cross currents and switch T3 – acting as the differential pair's current sink – is open. When `trigger` transitions to high, $V_{\mathrm{d}-}$ and $V_{\mathrm{d}+}$ discharge similarly as in a single-tail SA (c.f. Section 5.3). T4 and T5 open and T6 closes, whereupon the latch is enabled to change its value. During the discharging process, either T12 or T13 – depending on the applied voltage – dominates the output stage and guide the cross-coupled inverter towards a stable equilibrium. With the respective positive feedback, the output stage latches and generates a full-swing signal.

5.4.1 Characterization methodology

To verify the performance of the comparator, we simulate it individually and in conjunctions with its periphery. For that purpose, we consider the following characteristics: delay time, energy per decision and input offset.

Figure 23(B) shows the basic evaluation technique. The upper plot illustrates the comparator's trigger signal. Similar to the clock, from which this signal is derived, the trigger signal as well as its inverse is driven with a frequency of 1 GHz. In our simulation setup, we generate the clock respectively the trigger signal by an ideal voltage source. If not explicitly mentioned, the clock transition from low (0.0 V) to high (1.2 V) and vice versa is configured to 30 ps. These values are based on experience and data sheets of the given process node.

As mentioned in Section 5.2, the delay is calculated from the transition of the comparator's output relative to the rising edge of the trigger signal. To be precise, we determine the time between the 50 %-mark of the rising edge and the point in time when one output

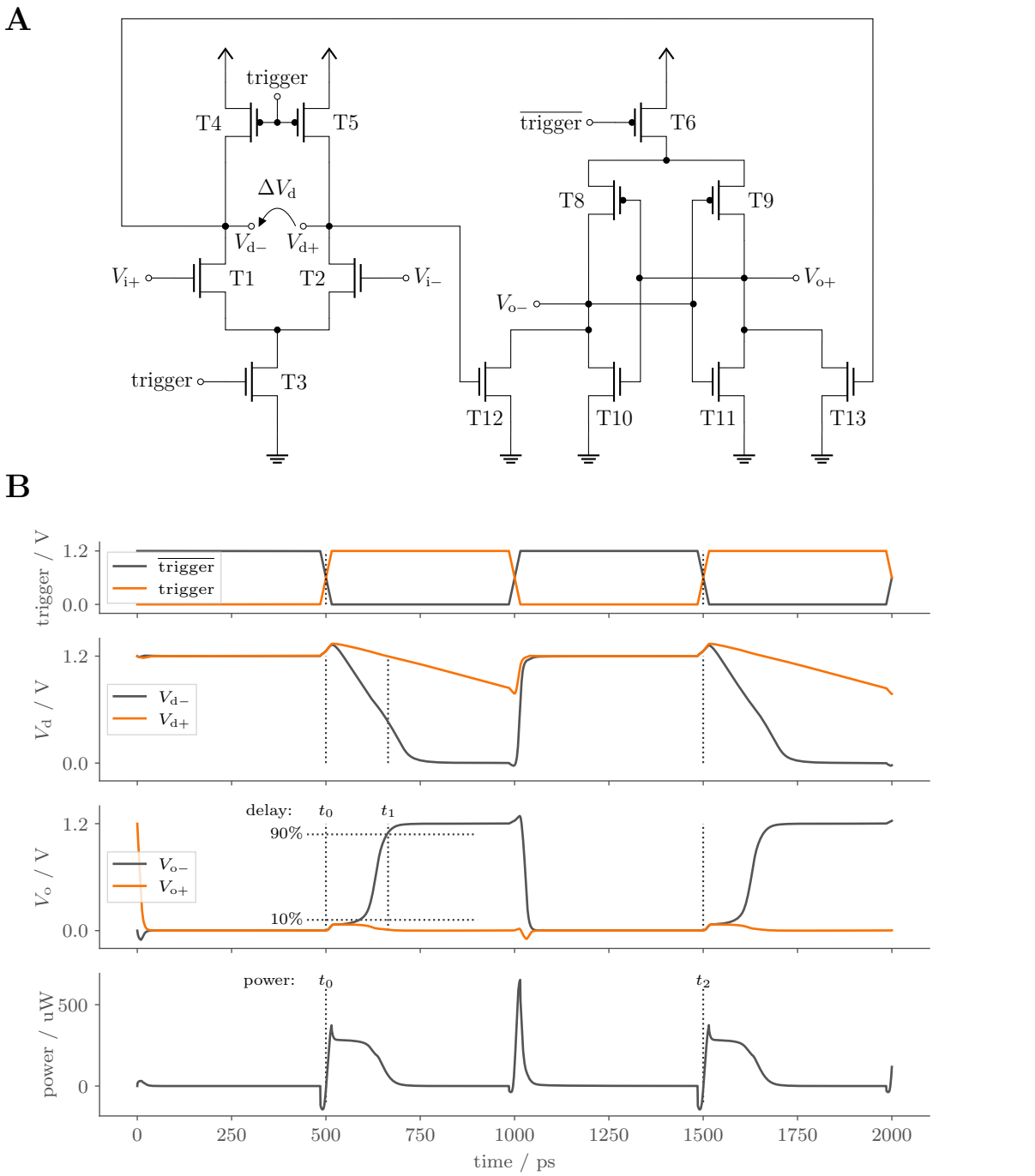


Figure 23: Double tail sense amplifier increases the transistor's headroom by reducing the number of stacked devices. **A**: Schematic of the implemented comparator core suggested by Schinkel et al. (2007) **B**: Internal as well as external signals extracted from transistor-level simulations. The dotted lines annotate points in times that are used to evaluate the comparator's delay and energy consumption.

branch reach 90 % of the supply voltage while the other output branch holds less than 10 %. Both times are annotated in Figure 23B as t_0 and t_1. When the comparator does not

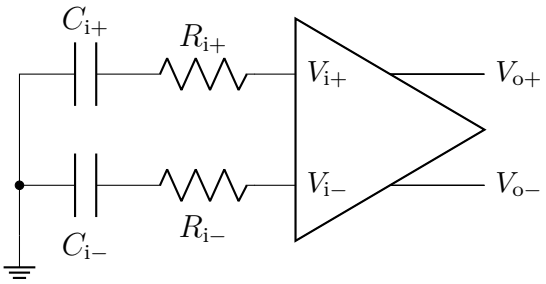


Figure 24: A series of a resistor followed by a pre-charged capacitor is used to investigate the kickback effect in the ADC. The resistors symbolize the transmission gates, while the capacitors represent the capacitive arrays and the capacitive reference DAC.

succeed to find a decision during the `trigger` high period, the output signal is interpreted as ambiguous.

The comparator's energy per decision is extracted from the currents and voltages at its supply input. We calculate the instantaneous power consumption as $P = U \cdot I$, as illustrated in the lower panel of Figure 23B. Furthermore, we numerically integrate $E = \int_{t_0}^{t_2} P$ over one period of the trigger signal to yield the energy per decision. Here we assume that every decision cycle behaves independently and identical. The assumption is based on the fact that due to the circuit's topology, all nodes are recharged in every cycle. In a first approximation, this means that the whole charge provided by the supply node is finally dumped into the ground node.

To determine the offset, a binary search approximates the comparators tripping point. Starting with initial boundaries of $\pm 100\,\mathrm{mV}$ and performing 10 iterations, we can estimate the input offset with a precision of less than $0.2\,\mathrm{mV}$.

Kickback causes significant transients and intermediate voltage jumps in the order of tens of millivolts at the input node of the comparator. Thus, they influence the discharging behavior of V_{d+} and V_{d-}. To account for the limited input impedance, we model the transmission gate with a resistor and the capacitive array as an ideal capacitor (c.f. Figure 24). Inspired by other transmission gates in the given process node (Billaudelle, 2022), we assume a resistance of $3\,\mathrm{k\Omega}$. Based on the proceeding development of the capacitive array, we estimate the capacitance to be $200\,\mathrm{fF}$. At the beginning of each simulation, these transistors are pre-charged to the desired input voltage. As long as the comparator is simulated individually, it operates without considering any load at its output.

5.4.2 Component dimensions

The traces shown in Figure 23B not only illustrates the measurement method, but also gives insights to the technical functionality of the individual stages. The sizing the components of the comparator defines its performance. Therefore, the physical understanding of the circuit is required, and the layout effects need to be considered beforehand.

Differential pair. During the decision process, the differential pair discharges the nodes $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$. The time required for a full-swing discharging of these nodes is determined by the capacitance $C_{\mathrm{d}+/-}$, input voltages $V_{\mathrm{i}+/-}$ and the geometrical aspect ratios W/L of T1 and T2. If this time is too large, the comparator becomes slow, if it is too small, the subsequent cross-coupled inverters have a too short response time and the comparator becomes less accurate. $C_{\mathrm{d}+/-}$ are recharged in every decision cycle and thus contributes directly to the energy consumption. They are dominated by the gate capacities of the strong-arm transistors T12 and T13 that couple the stages.

Since device mismatch between T1 and T2 directly leads to a comparator input offset, a good matching between them is required. Increasing the transistor size reduces the current factor mismatch at the cost of gate area, which in turn determines the kickback effect. Dividing T1 and T2 into multiple, parallel transistors ("fingers") and interleaving them properly reduces mismatch. To some extent, such a layout allows canceling the effect of the mismatch sources that occur with spatial gradients, e.g., variations in the gate-oxide thickness.

Transistors T3, T4 and T5 should operate as switches. In their closed configuration, their resistance should be correspondingly low. Especially T3 requires special care, as all charges from the nodes $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ passes through it. T4 and T5 are designed to fit to the magnitude of $C_{\mathrm{d}+/-}$ best. As their gates are recharged in each decision, their contribution to the total energy scales with their gate area. To reduce the leak current, all switches are dimensioned to a gate-length L of 80 nm.

Coupling to 2nd stage and cross-coupled inverters. Transistors T12 and T13 have an amplifying characteristic. Since they couple both stages in differential manner, they must be matched to not contribute significantly to the comparator's offset. Thus, they require transistor lengths L that are greater than the minimal length: $L = 120\,\mathrm{nm}$. But as described above, their gate areas corresponds to the capacitance $C_{\mathrm{d}+/-}$. The width of T12 and T13 ensure, that they can always overrule the latch's state.

	type	width / nm	length / nm	fingers
T1, T2	NMOS	700*	150*	4
T3	NMOS	300	80	4
T4, T5	PMOS	400	80	4
T6	PMOS	400	80	4
T8, T9	PMOS	400	120	4
T10, T11	NMOS	300	120	2
T12, T13	NMOS	300	120	4

Table 6: Transistor sizes in the comparator's first design iteration. *Parameters after post-layout optimization. All pre-layout simulations use for T1 & T2 a width of 500 ns and a length of 200 ns.

Transistors T8 to T11 implementing the feedback loop. For a balanced decision, the corresponding pairs of these transistors need to match as well. Nevertheless, increasing their gate area, decreases the feedback speed slightly, as the driven gate capacities increases as well. For reasons of symmetry, we decide to use the same transistor length L as in T12 and T13. The transistor widths are chosen with respect to balance NMOS and PMOS transistor speed.

To address all above aspects, we group the transistors according to their tasks. In the design process of a layout, several transistor fingers are usually bundled in wells by pairwise connecting their drain or source contacts. The transistor widths W of bundled transistors should match, to avoid symmetry brakes. Apart from the division into P-wells and N-wells, we find three classes of device accuracy requirements. The first class comprises T1 and T2, as these play the most important role for the comparator's performance. A second class includes all other transistors that require matching: T8 to T13. Last, all switches are optimized by minimizing their gate area in order to reduce the switching power while fitting the respectively required on resistances. All transistor dimensions are collocated in Table 6.

5.5 Transistor level simulations

Using the methods described in Section 5.4.1, this section evaluates the design proposed in Section 5.4.2 on a transistor level. Hence, the presented simulations incorporate transistor models without any knowledge about their physical placement. However, they allow a first approximation of the design's performance and require much less computation time enabling Monte-Carlo mismatch simulations. We run simulations with different initial states in the cross-coupled inverters with no observable change in behavior. Further,

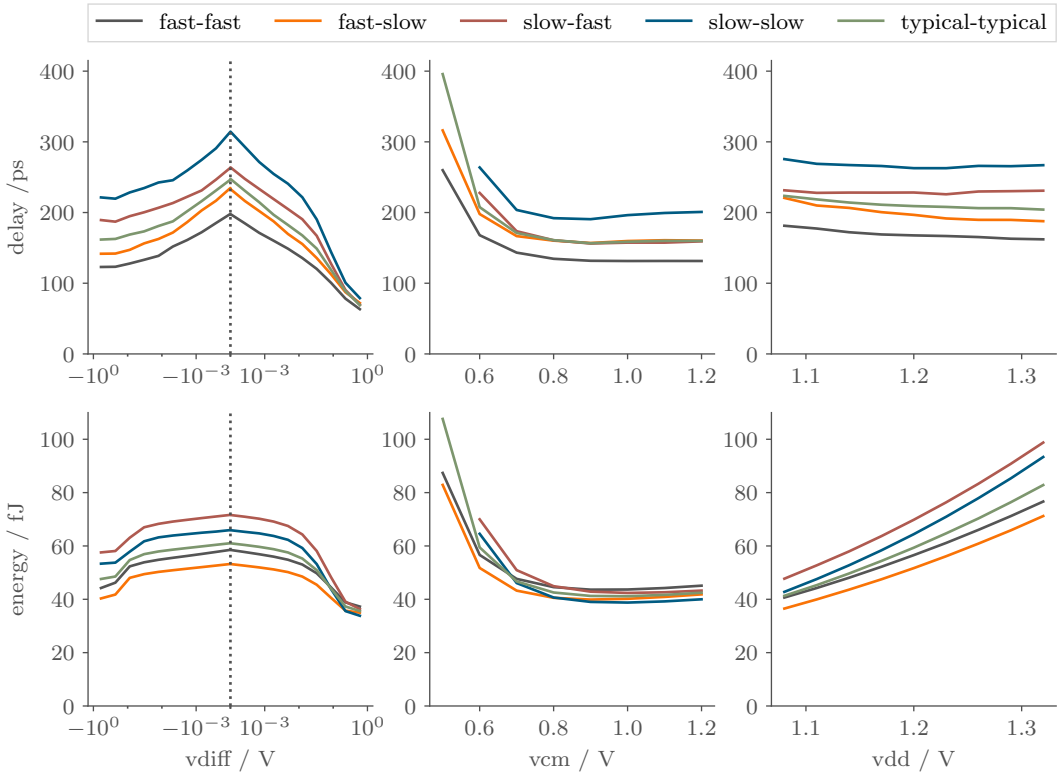


Figure 25: Delay time and energy consumption of the comparator on a transistor level with respect to the input voltage difference vdiff, the common mode voltage vcm and the supply voltage vdd for different process corners. The target supply voltage is 1.2 V, the target common mode voltage 0.6 V and the simulated input voltage difference 1 mV. The Y-axis, which represents the input voltage difference, is scaled by a symmetric logarithm.

it can be shown that the trigger edge quality in realistic ranges has a minor effect on the comparator's behavior. The same is true for a skew between the trigger signal and its inverse: As long as the active phases of both stages overlap in time sufficiently, the comparator remains functional. However, the energy consumption reduces if the inverted trigger signal, $\overline{\texttt{trigger}}$, is slightly after the non-inverted `trigger` signal. This is due to the fact that it reduces the time of the cross current through the second stage.

Process corner. Figure 25 shows the comparator delay as well as the energy with respect to the input voltage difference vdiff, the common mode voltage vcm and the supply voltage vdd in the different corners.

The delay with respect to the input voltage difference is plotted on a symmetrical, logarithmic X-axis. Thereby, $V_{\mathrm{i}+}$ (later: single-ended input) changes from decision to decision, while $V_{\mathrm{i}-}$ (later: reference voltage) stays at the ADC's nominal common mode voltage of

0.6 V. Below an input voltage difference of around 10 mV the comparator's decision time is dominated by the balance of power in the cross-coupled inverters. Here the decision is made after both V_{d+} and V_{d-} have already completed their equal-length discharge process. The decision itself is based on minimal differences in the discharging process of V_{d+} and V_{d-}, registered by V_{o+} and V_{o-} and amplified in the positive feedback loop to a full swing signal. The decision time grows exponentially decreased input voltage difference (Pelgrom, 2017).

For unambiguously negative input voltage difference, the decision time is limited by the time it takes to discharge V_{d-}. In the process, the discharging time of V_{d+} is much larger. On the other side, for unambiguously negative input voltage differences, V_{d+} is faster than V_{d-}, which speeds up the decision. The decision time depends on the process corner, whereby the NMOS corner is more influencing than the PMOS corner. This is among others due to the current factor in the transistors T1 and T2 of the differential pair.

The same trisection can be observed in the respective energy consumption. However, each section seems to belong to a characteristic energy consumption. In contrast to the delay time, the corners act differently on the energy consumption. Here, the mixed process corners are more influential, as they affect the effective cross currents through the comparator by the matching of the NMOS and PMOS threshold voltages.

To characterize the robustness against an ADC's common voltage change, we simulate different common mode voltages $V_{cm} = \frac{V_{i+}+V_{i-}}{2}$ at an input voltage difference of 1 mV. This input voltage difference was previously suggested and discussed (c.f. Section 5.2). In the delay time and the energy consumption, common mode voltages above 0.8 V perform similar. Below 0.8 V, one can observe some differences leading to ambiguous decisions as the trigger period continues. This is due to the reduced gain of the transistors T1 and T2 in the differential pair. A common mode voltage V_{cm} of 0.6 V is already at the tipping point to ambiguous decisions, especially when the NMOS transistors are slow. With increased input voltage differences or trigger periods, reduced common mode voltages V_{cm} are achievable as well. The energy consumption with respect to the common mode voltage V_{cm} is independent of the process corner. We continue with this design, as we have no better transistor parameters at this moment and the design targets are met in every process corner.

In the right panels of Figure 25, delay time and energy consumption is plotted with respect to the supply voltage. While the delay time is almost independent of supply voltage variations, the delay time act significantly on the energy consumption. The latter is expected, as in full swing digital circuits the first-order energy grows quadratically with

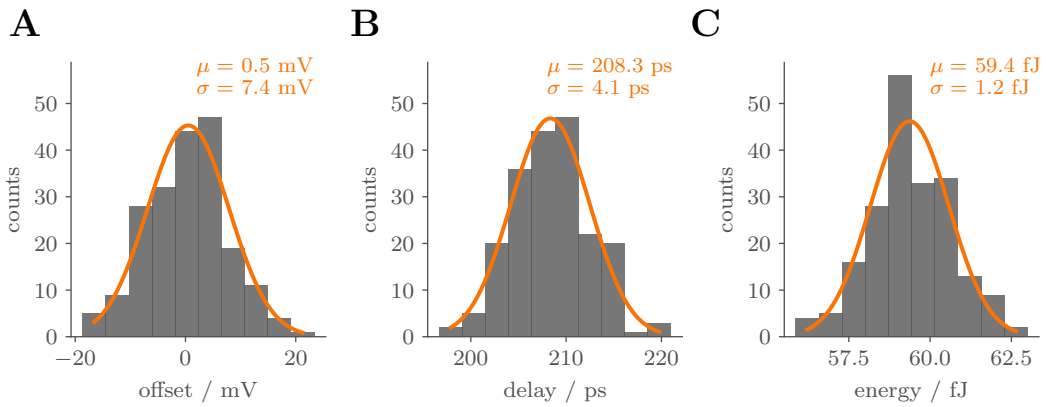


Figure 26: Mismatch effects extracted from simulations of 200 circuit instances drawn with Monte-Carlo methods from a device variation distribution. While the input offset error (**A**) shows a significant spread, the delay time (**B**) and the energy consumption (**C**) are almost not affected. Delay time and energy are extracted for an effective input voltage difference of $\Delta V_\mathrm{i} = 1\,\mathrm{mV}$ after correcting the offset error. The histograms are fitted by normal distributions.

the supply voltage (Chandrakasan et al., 1992).

In all illustrated measurements, the comparator decide correctly, but the missing data-points in centered panel indicated too late decisions.

Device mismatch. Monte Carlo methods are used to study device mismatch. First the comparators offset is determined for each sample and then in a second step delay time and energy consumption are simulated with corrected reference voltage $V_{\mathrm{i}-}$ and an effective input voltage difference of 1 mV.

The histogram in Figure 26A shows the offset error distribution determined by offset searches of of 200 circuit instances drawn with Monte Carlo methods. A Gaussian normal distribution (orange curve) is used to fit the data. For the comparator a device mismatch-specific input offset distribution with $\sigma(V_\mathrm{off}) = 7.4\,\mathrm{mV}$ is extracted. The mean of the offset error is $\overline{V_\mathrm{off}} = 0.5\,\mathrm{mV}$, which indicates that no significant systematical effects occur.

The extracted delay times and energy consumption also demonstrate a Gaussian behavior. Nevertheless, since the distributions are quite narrow and less distinct as for example variations through the process corner, the device mismatch is less relevant here.

Kickback effect. In every decision cycle, the transistors T1 and T2 pull and push charge into implicit capacitors created by their gates. This input voltage dependent effect leads to major transients on the input lines, as shown in Figure 27A. Therefore,

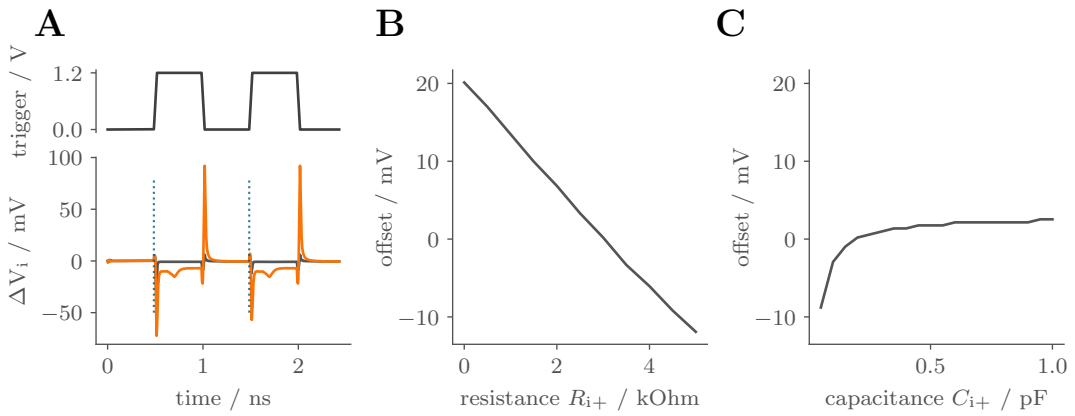


Figure 27: Simulated effect of input impedance mismatch at the comparator. **A**: Dependent on the input voltage, the kickback effect leads to transients on V_{i+} (gray) and V_{i-} (orange). **B**: By changing the resistance R_{i+}, we model an input impedance mismatch caused by the transmission gates. **C**: Above a certain threshold, a mismatch of the capacitive elements C_{i+} and C_{i-} have relatively small impact on the input offset.

a difference in the limited input impedance of V_{i+} and V_{i-} leads to an effective input offset as illustrated in Figure 27B+C. V_{i-} always provides the same input impedance model ($R_{i-} = 3\,\text{k}\Omega$, $C_{i-} = 0.2\,\text{fF}$), while the input impedance parameters of V_{i+} are adopted. Above around 0.2 fF, an increasing capacitance only has a slight impact on the offset. In practice, this means that using one capacitive array (fast mode) or both (precise mode) does not require much adjustment. However, if the input capacitance is further reduced, we observe an effect of up to −9 mV offset variation. Changing the resistivity parameter, which models the transmission gate, has a significantly different effect. An almost linear behavior between differences in the resistance and the resulting input offset can be observed. Hence, it is important to use the same transmission gates for all configurations.

Beside this transient effect, kickback can also have a not regenerative side. It is not guaranteed that all charge is fully recovered and preserved. This can be due to the voltage dependent gate capacities of the transistors T1 and T2. However, in the time scales of the ADC conversions, this is not relevant.

5.5.1 Layout creation

To be cast in silicon, the transistors of the proposed comparator are placed and routed. In such a small and sensitive circuit, the actual physical layout is critical to the performance of the design. Keeping symmetries is an effective way to avoid manufacturing effects.

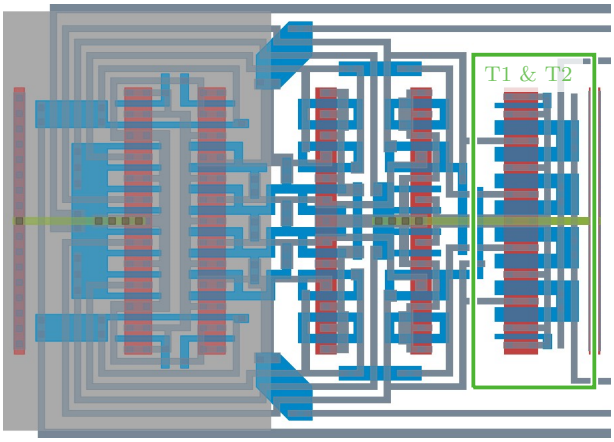


Figure 28: First layout attempt: In order to keep the layout "flat", we restricted our self to the layers up to metal 1. The color legend can be found in Figure 40.

Therefore, we bundle related transistors of the same size and use a common centroid approach with multi-finger transistors. Care must be taken to ensure that the flow directions of the fingers in each transistor each other out in total. At the outer borders of each transistor stack, dummy transistors are used to avoid approximate effects (c.f. Section 5.1.3). Guard rings need to be considered to separate the comparator from neighboring circuits and ensure a good bulk diode behavior.

"Metal 1 only"-version. In a first layout attempt (c.f. Figure 28), we created a layout only using the layers up to the first metal layer (gray). The aim of this approach is to keep routing capacities on the second metal layer free. In order to enable line crossings anyway, the poly-silicon layer (blue) – actually developed for transistor gates – is also used as an additional routing layer. Poly-silicon comes with the draw-back, that its sheet-resistance[9] is about 100 times lower than in metal layers. The large density of parallel metal 1 routing increases the wire capacitance. Especially, $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ catch parasitic capacities that increase the charge stored on them which are to be drained by T1 and T2. Due to large resistances, the RC-time constants of almost every wire grow. As presented in Figure 30 this has dramatic effects on the comparator's performance: The delay time as well as the energy consumption more than doubled. At a small input voltage difference, ambiguous outputs occur, causing the plateau at the respective delay time in Figure 30. Since this does not fit the design targets, the second metal layer is utilized in a second attempt as well.

[9] In CMOS processes, the resistance is given in ohm per square ($\Omega/\square$).

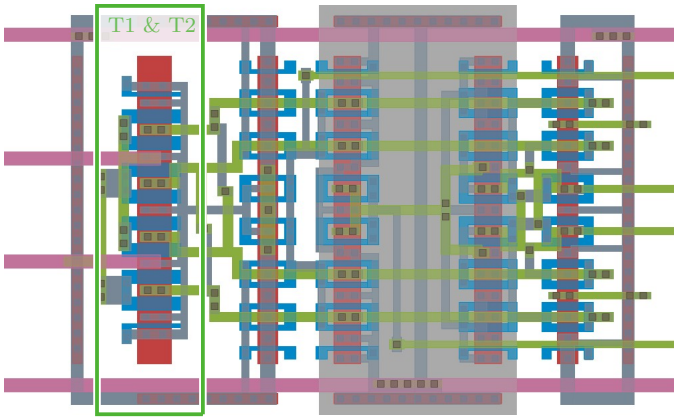


Figure 29: In the final layout, we also make use of the second metal layer decreasing the decision delay. The color legend can be found in Figure 40.

Speed and power optimized layout. Figure 29 shows a picture of the final comparator layout, utilizing also the second metal layer (yellow). Additionally, a part of the ADC's power routing on the third metal layer (pink) as well as the signal input on the same are shown. In difference to the layout above, the choice of the transistor bundled differ. Besides the restructured N-well area, the reduced usage of the first metal layer attracts attention. Moreover, this layout gets along without poly-silicon routing. These changes are reflected in quantitative improvements in Figure 30. We extract, in a so-called PEX, parasitic resistances and capacities from the layout and pass them to the simulator. For this purpose, we utilize *calibre*.

The layout still shows the lack of consideration of the additional conduction capacitance on $V_{\mathrm{d}+}$ and $V_{\mathrm{d}-}$ in the transistor-level simulations. To counteract this and further improve the layout, the transistor parameters are adjusted afterwards. The insights gained from the previous discussion are considered. Since increasing the total gate area is, due to the kickback effect, not an option, changing the aspect ratio of T1 and T2 increases their current factors at the costs of reduced device matching. While the previous total current factor was $\beta \propto W/L = 10$, the new one equals $\beta \propto W/L = 18.6$. The comparator core has an area of $67.6\,\mu\mathrm{m}^2$.

5.5.2 Digital interface

In order to operate the comparator in a silicon environment, it is connected to the environment via a trigger tree and an output latch. These to stages serve as interface elements

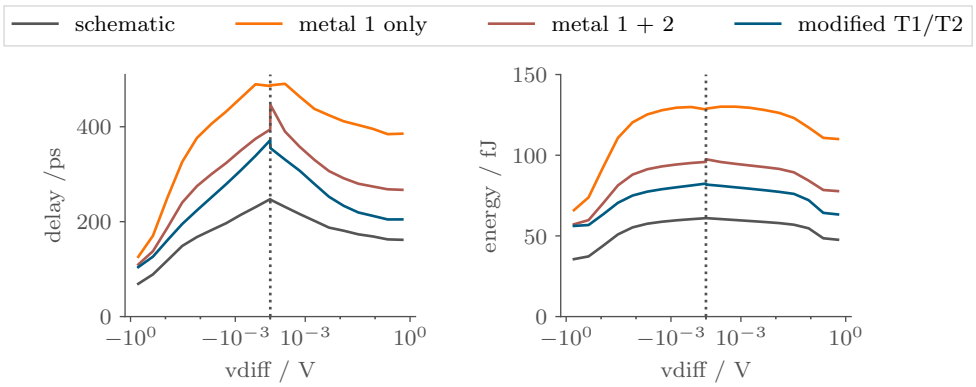


Figure 30: Comparison of the delay time and energy consumption between the transistor-only model and the simulations using the PEX of the different layouts. The Y-axis is scaled by a symmetric logarithm.

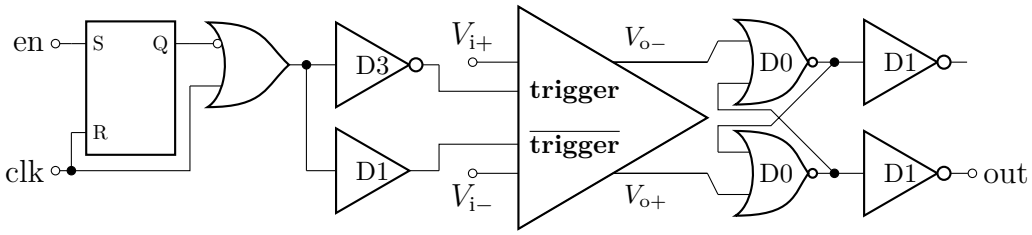


Figure 31: The comparator core (center) is embedded into buffer stages that form a digital interface. The trigger signal is derived from the clock signal and can be disabled by a clock-gating mechanism. To stabilize the output signal of the double-tail sense amplifier, it is buffered by a latch.

to the digital domain and reduce the required verification effort. Hence, they are build from standard cells provided by the manufacturer. Figure 31 shows a schematic of the utilized modules and how they are connected to the comparator. Energy simulations on these modules can be found in Section 6.2.

Trigger tree. The trigger tree derives the comparator's trigger signal from the digital clock and is illustrated on the left side of Figure 31. As already mentioned, it uses the negative clock edge for triggering the comparator and the positive one to pre-charge it again. Utilizing the negative clock edge lead to an unknown edge-timing, as typically only the positive clock edge of a clock source node (e.g., phase-locked loop (PLL)) is characterized in detail. Also clock jitter needs to be considered. To have full analog control about the skew between the trigger signal and its inverse, both have the same source node.

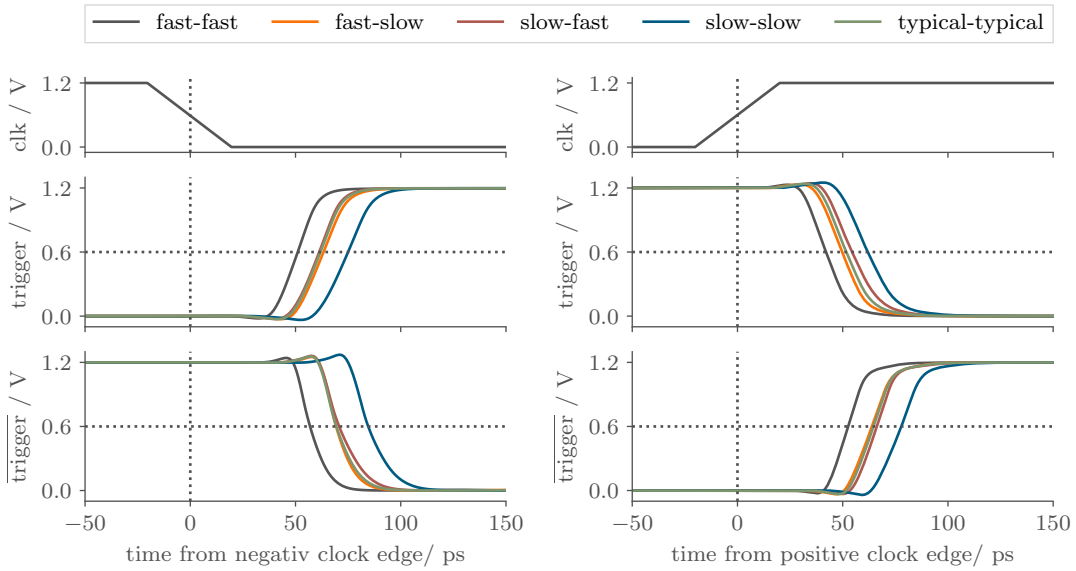


Figure 32: The trigger tree introduces an additional delay. The timing of the triggering (left) and pre-charging (right), illustrated for the different process corners, is normed to the half clock edge.

The trigger tree can be disabled via a clock-gate, which makes it possible to switch the comparator on and off safely. Therefore, a standard clock gate cell receives input from the `comparator enable` signal presented in Section 4.2. This enables using the provided timing characterization from the manufacturer to constrain the module port. One buffer and one inverter are tuned to the capacitive load of the switches in the comparator using the fan-out-4 rule of thumb (c.f. Section 4.1.2).

To verify the trigger tree's behavior, we perform a functional verification simulating it together with the comparator. Figure 32 shows both trigger signals relative to the timing of an ideal clock edge. Simulating different process corners gives some boundaries for the physical implementation. Due to the circuit's simplicity and a careful placement, we do not expect large variations between a transistor based simulation and a simulation that includes the wire parasitic. The transistor level simulation determines the expected (clock) gate delay in the typical corner to 62 ps (rising edge) and 52 ps (falling edge) for the `trigger` signal. The delay for its inverted signal is with 69 ps (rising edge) and 65 ps (falling edge) slightly larger, which reduces the required comparator power. The overall maximum observed delay is the rising edge of $\overline{\texttt{trigger}}$ (85 ps). These delays can be hidden through a useful clock skew, delaying the rest of the ADC behind the clock signal at this trigger source node.

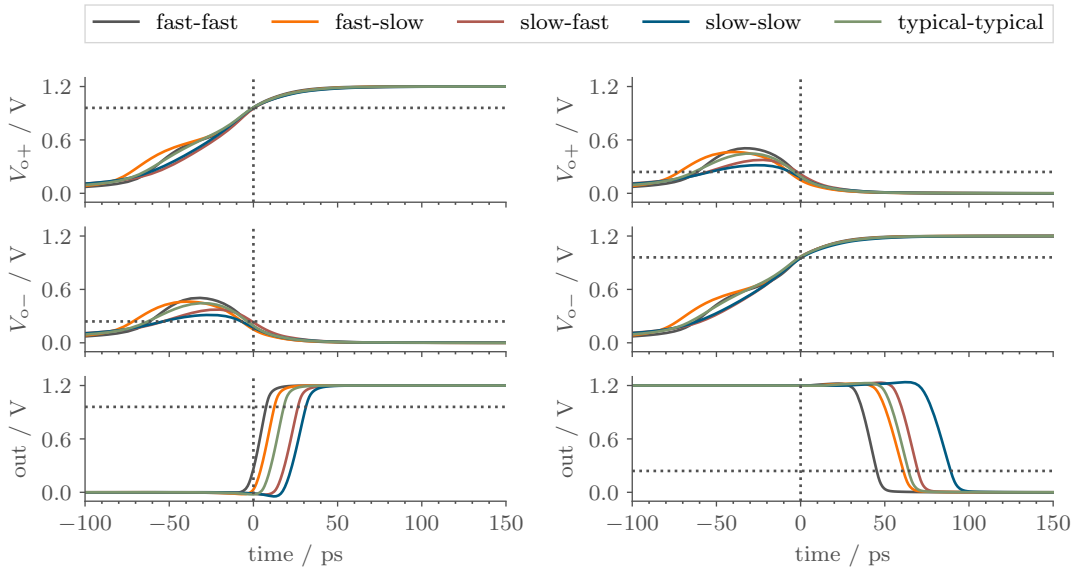


Figure 33: Behavior of the output latch at output level transitions in cooperation with the double-tail sense amplifier at an input voltage difference of $\Delta V_\mathrm{i} = \pm 1\,\mathrm{mV}$ for different process corners. The timing is normalized to the time when V_{o+} or V_{o-} reaches 80 % of the supply voltage.

Output latch. An active low latch is utilized to buffer the comparator's output for the digital domain (c.f. Figure 31). It reduces meta-stability in the following stages and balance the load at V_{o+} and V_{o-}. Every positive feedback loop potentially generates oscillations. In an active low latch, this is suppressed by a sufficient gain of the not-or gates. As additional load at V_{o+} and V_{o-} further increases the comparator's trigger-to-output time, these gates provide a driving strength D0, measured in units of the minimal standard cell inverter. To decouple the latch from the following components and to keep the fan-out rule of 4, one inverter gate each of driver strength D1 is implemented. Metastability in subsequent FFs is still possible if they sample at the output's time of transition, but is majorly suppressed by the output gain. Moreover, if the comparator fails to reach a decision in time, the latch keeps the previous value.

Since the transition delay through the output latch further increases the effective comparator delay, we determine the same in simulations with respect to the process corner. Figure 33 presents the comparators output stage V_{o+} and V_{o-} as well as the out signal illustrated on the very right of Figure 31. One comparator input (V_{i+}) alternates between 599 mV and 601 mV, while the other (V_{i-}) is fixed at 600 mV, causing output state transitions. For this simulation only, the comparator's inputs are driven by ideal voltage sources. The left panels show a transition from low to high, the right one a vice versa transition. All traces are centered to the related 20 %/80 % marks of V_{o+} and V_{o-}. We

can observe that a falling transition requires more delay time than a rising one, which is not unusual in FFs. In the typical corner, the transition time is 18 ps (rising edge) and 64 ps (falling edge) and in slow-slow corner 32 ps (rising edge) and 89 ps (falling edge). Again, we do not use a simulation with extracted parasitic.

5.6 Capacitive reference voltage generator

In order to generate required second input voltage for the comparator, the comparator unit should include a reference voltage generator. As the ADC architecture requires comparisons with the half supply voltage, this reference circuit should be optimized for this target. In addition, the reference generator should be adjustable to serve in other tasks: An adjustable reference voltage can calibrate for static comparator input offset and comparator input impedance mismatch. Finally, adjusting reference voltage also serves to calibrate for static internal reference mismatch, which could be caused by device mismatch.

There are multiple successful approaches of designing capacitive DACs and reference voltage generators (Pelgrom, 2017). Moreover, BSS-2 already provides a capacitive DAC (Hock, 2015). Nevertheless, we decide to take a new and perhaps unusual approach.

Here we present and discuss an adjustable capacitive voltage divider. This reflects on the capacitive nature of the array presented by Czierlinski (2022). In the comparator unit, this module is instantiated two times to alternately be in operation and recharged. To consider comparator's sensitivity to input impedance mismatch, these reference circuits are connected to the comparator negative input utilizing the same transmission gate as comparator's signal input. Further insights on this transmission gate can be found in Czierlinski (2022).

5.6.1 Capacitive voltage divider

A capacitive voltage divider is a very basic circuit and illustrates in Figure 34: Analogous to a resistive voltage divider, two capacitors C_1 and C_2 are connected in series between the supply rail vdd and the ground rail gnd. Due to charge retention and the fundamentals of electrostatics, a defined voltage out is established between them.

To control the shared charge, both capacitors are cleared in a reset state shown in Figure 34A: By connecting both plates of the capacitors to ground node or respectively the

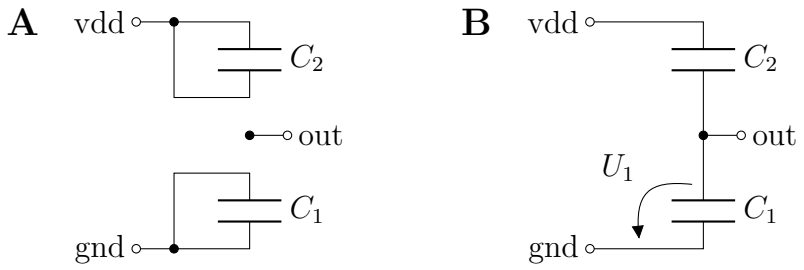


Figure 34: Capacitive voltage divider in its reset **A** and active **B** state.

supply voltage, the initial charge on the capacitors is known:

$$Q_{C_1}^{t_0} = Q_{C_2}^{t_0} = 0 \tag{5.6}$$

At this moment, the signal out is floating.

In an active state, both capacitors share one capacitor plate while the other remains connected to vdd and gnd, respectively. This pulls charge from the supply as well as from the ground to fulfill the Kirchhoff voltage law: The voltage between supply and ground must now drop across the capacitors. However, the total charge Q_{total} must be preserved:

$$Q_{\text{total}}^{t_1} = Q_{C_1}^{t_1} + Q_{C_2}^{t_1} = C_1 \cdot (\text{out} - \text{gnd}) + C_2 \cdot (\text{vdd} - \text{out}) \stackrel{!}{=} 0 \tag{5.7}$$

After some simple transformations we get an expression for the output voltage:

$$\text{out} = \frac{C_2 \cdot \text{vdd} + C_1 \cdot \text{gnd}}{C_1 + C_2} \tag{5.8}$$

So, two equally sized capacitors result in half the supply voltage, as required by the ADC. By changing either the voltages on the rails or the capacitance of the capacitors, one can adjust this voltage. While the former opens the doors to a minor power supply rejection, the latter enables us to build a DAC from this scheme. Since the capacitive array suggested by Czierlinski (2022) suffer from the same lack of power supply rejection, these effects cancel each other out to some extent: Half the supply voltage always results in half the maximum output code. It remains to be discussed whether this correlation is really desirable; at least it is deterministic and simple.

When refreshing, first the capacitive reference generator core fully discharges its capacitors. One half of the charge is released back to the supply rail and the other half to the ground rail. This charge is bound by the capacitors again, when they move on to

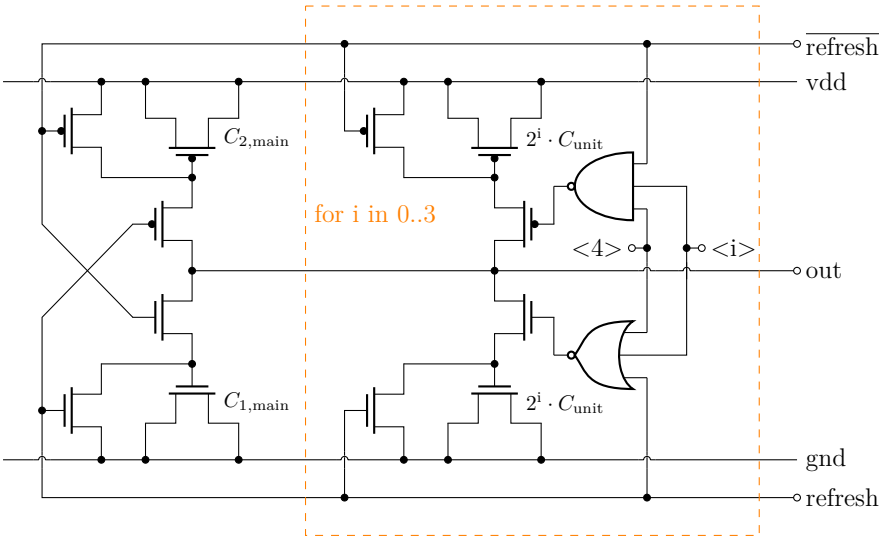


Figure 35: Schematic of an adjustable capacitive voltage divider as reference DAC for the comparator.

their active states. In this sense, there is no cross flow of charge and no effective power consumption in the voltage divider.

5.6.2 Implementation details

The adjustable capacitance of the reference DAC is created from multiple parallel capacitors each. C_1 as well as C_2 are each the sum of one larger and several binary weighted smaller capacitance. To select a specific output voltage, the latter are enabled respectively. Figure 35 shows a schematic of this scheme. The capacitors are build from MOSFETs with shorted source, drain and balk terminals, with the gate serving as capacitive element. On its left side the main capacitance $C_{1,\,\text{main}}$ and $C_{2,\,\text{main}}$ with the corresponding switches are visible, which are very reminiscent of the equivalent circuit diagram in Figure 34. The right side shows exemplary two of the selectable binary weighted additional capacitors that are multiples of the unit capacitor $C_{1/2,\,\text{unit}}$.

Capacitors. Capacitors utilizing transistors gates are called metal-oxide-semiconductor (MOS) capacitors. As their capacity is voltage dependent, they are typically not the capacitors of first choice (Hu, 2010, p.157f), but due to the fact that the reference DAC only provide a limited dynamic output range, this aspect is minor. On the other hand, MOS capacitors provides a relatively large capacitance per area, while only occupying the

layers up to poly-silicon. This enables placing the reference DAC below the capacitive array.

The capacitors connected to the supply rail are made from PMOS transistors while the ones connected to the ground rail are made from NMOS transistors. This provides a continuous depletion zone in the MOS capacitors and stabilizes their performance. To further reduce the impact of the threshold voltage variation, the MOS capacitors are implemented as low-threshold transistors.

Switches. Figure 35 also shows transistors that perform switching between the two states illustrated in Figure 34. Therefore, the `refresh` and its inverse are required. Since the switches on one MOS capacitor are of the same type and of the same size, the charge injection just cancels out when switching. To reduce leakage currents, they have a small width and a long length (c.f. Table 7). However, even made from unit cells, the number of parallel switch instances per MOS capacitor changes with respect to the capacitance. By doing so, we account for a sufficient RC time during refreshing the reference DAC. NMOS and PMOS transistors need to be balanced so that both have the same effective current factors.

The decoding of the digital value happens in a bank of triple NOR and NAND gates, build from TSMC standard cells with minimal driving strength. At a minimum digital value, C_1 is maximized by enabling all additional MOS capacitors while those of C_2 are disabled. In the center, all additional MOS capacitors are disabled. When applying the maximum digital value, C_2 is maximized. The refresh signal and its inverse is derived from a global refresh signal. It is gated by a global reference selection signal to always refresh only the non-active reference decoder. In addition, the refresh is signals buffered to deal with the large gate areas of the switches.

Number of required stages. When discussing the comparator core (c.f. Section 5.5), the expected input offset caused by device mismatch was determined to be $0.5 \pm 7.4\,\mathrm{mV}$. In addition, an ADC LSB in precise mode ideally counts around 4.7 mV. Due to the gain error in the ADC caused by non-ideal capacitors (see Czierlinski (2022)), this drops to less than 4 mV in the real application. For good calibration, a reference DAC LSB should be less than a LSB of the ADC. Hence, we decide for an C_{main}-to-C_{unit} ratio that allows for a resolution of 3.3 mV. To account for the process corners, we aim for a dynamic range of $\approx$100 mV. Together this results in a number of 5 bit, which at the same time represents the largest possible number of justifiable routing resources at this point. As already given

	type	width / nm	length / nm	fingers
$C_{1,\text{main}}$	lvt NMOS	2000	4120	2
$C_{1,\text{unit}}$	lvt NMOS	1200	280	
unit switch C_1	NMOS	200	200	
$C_{2,\text{main}}$	lvt PMOS	2000	4120	2
$C_{2,\text{unit}}$	lvt PMOS	1200	280	
unit switch C_2	PMOS	440	200	

Table 7: MOS device dimensions in the core of the capacitive reference DAC. The main capacitance $C_{1,\text{ main}}/C_{2,\text{ main}}$ are each connected to 13 unit switches per state; the binary weighted capacitance $C_i = C_{\text{unit}} \cdot 2^i$ utilizes one unit switch per state, only its MSB capacitors use two.

Figure 36: Corner simulations of the reference DAC. The digital value 0 is reached by the codes `5'b01111` and `5'b10000`.

by the symmetry of $C_{1,\text{ main}}$ and $C_{2,\text{ main}}$, the reference DAC's output should be centered around half the supply voltage.

In this discussion, calibration for impedance mismatch as well as adjustment of the dynamic input range of the ADC play a minor role. This is not due to their lack of relevance, but due to the missing design information of the corresponding components at this time. However, based on this, the component dimension of the reference DAC are listed in Table 7.

5.6.3 Simulations

The behavior of the reference DAC is investigated in simulations. An open-loop configuration of the reference generator reflects the high impedance input of the comparator. In this first simulation, the focus is on the capacitive voltage divider itself. Therefore, the digital code decoding stage is replaced by ideal voltage sources. We sweep the digital configuration performing one refresh cycle and evaluate the final voltage at its output.

Figure 36(A) shows this output voltages for the different process corners. While the typical, slow-slow and fast-fast are similar except for a small gain error, the mixed corners reveal offset errors. The latter come from the fact, that the capacitive voltage divider also incorporates a resistive voltage divider implemented by the switches' on resistance. Since the switches are build from different transistors types, any current factors mismatch will affect the output voltage. In the typical corner the DAC exhibit a gain of 3.3 mV/LSB with a y-axis intercept of around 598 mV. In the mixed corners the y-axis intercept can be determined to 576 mV (fast-slow) and 620 mV. Thus, in the typical corner, the DAC covers 6.9σ of the discussed mismatch-related input offset, but in the fast-slow corner only 4.0σ and in the slow-fast corner 4.4σ. This is based on the fact that the input offset of the comparator is process corner independent.

Analogous to the DNL/INL definitions (c.f. Equations (2.7) and (2.8)) for ADCs, one can calculate the same also for DACs. While their statements remain the same, only the voltages at the code transitions need to be replaced with the output voltages at a code. The calculated DNL and INL is shown in Figure 36B+C. Both are almost not influenced by on the process corner.

When simulating Monte Carlo mismatch, the models provided by the manufacturer do not provide statistical information about transistors gate capacity mismatch. However, it can be shown that device mismatch in the switches has only a minor effect on the discussed observable, while the characteristic peaks in the DNL still occur. This suggests that these characteristic peaks come from the MOS capacitors intrinsic properties. But since DNL as well asINL are in a sufficient range, this will not be investigated further here.

To estimate the total capacity of the reference DAC, we slightly discharge over a defined resistance while determining the RC time constant. It turns out, that this is a rather imprecise method and the fit-uncertainty of the time constant RC is significant. We determine, however, the capacitance to be in the range between 477 ± 25 fF and 546 ± 27 fF. Thus, the reference DAC has roughly the same capacitance as the ADC-array in its precise mode.

We observed, that initializing the reference DAC requires more than one refresh cycle. This is due to the charge and the related voltage, initially stored on the output node.

Simulation (c.f. Section 6.2) have shown, that a refresh cycle consumes 0.24 pJ.

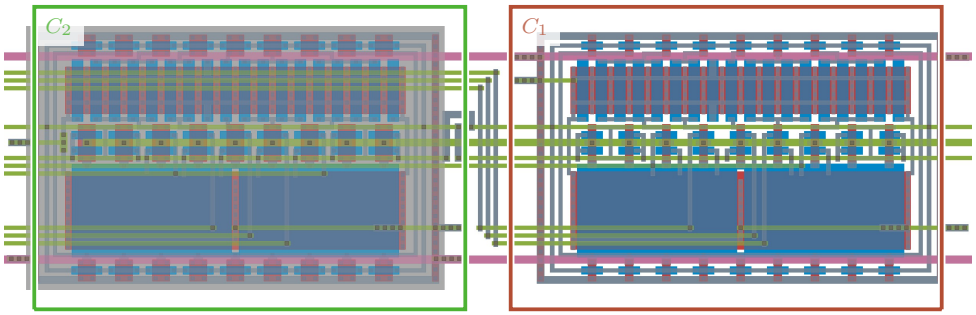


Figure 37: Layout of the capacitive reference DAC. The main-capacitors are located in the lower half, the selectable unit capacitors are placed in the upper half. The color legend can be found in Figure 40.

5.6.4 Layout creation

Figure 37 shows the layout of the reference DAC's capacitive core. It is divided in a part containing the NMOS and an N-well containing the PMOS devices. On the upper half, the main capacitance $C_{1,\,\mathrm{main}}$ and $C_{2,\,\mathrm{main}}$ are visible. Above, 18 unit MOS capacitors are placed, three of which are dummy MOS capacitors shorted to the respective rail. The switches are located in the center and at the bottom and top edges. The control signals are routed on the second metal layer (light green). Altogether, each capacitive reference generator core covers 6.76 µm × 25 µm silicon.

Since the decoder is build from TSMC standard cells, it is not drawn here. However, it covers an area of 6.76 µm × 7.5 µm.

Post-layout simulations of the comparator core determine the performance more realistically than pre-layout simulations. Figure 38 is generated by the same evaluation method as in Figure 36, but instead of sweeping the process corner, the typical layout version is shown. As orientation, also the typical corner is plotted. Although both show the same trends, they differ in details. Apparently, the parasitic capacities and resistances in the layout flat the edges in the DNL curve. Moreover, the gain of the DAC increases slightly, because the parasitic capacities affect the selectable capacitors more than the main capacitors.

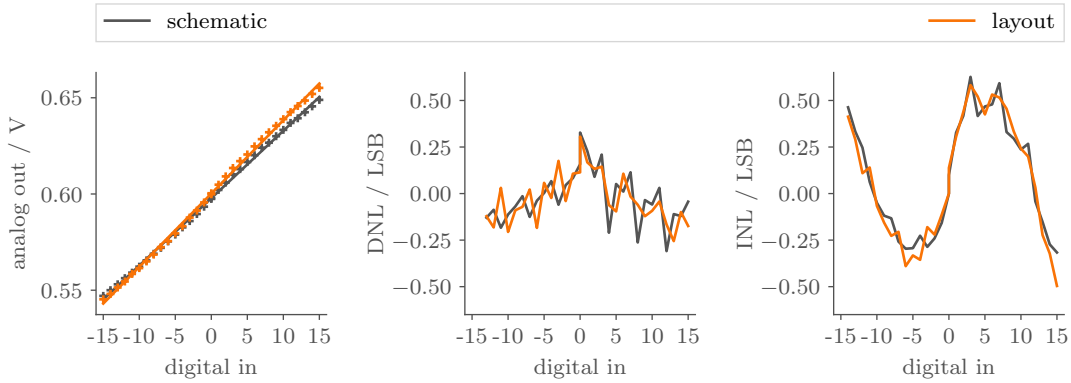


Figure 38: Reference DAC performance evaluation of a simulation utilizing layout data (PEX) in contrasts to a transistor-only model in the typical corner.

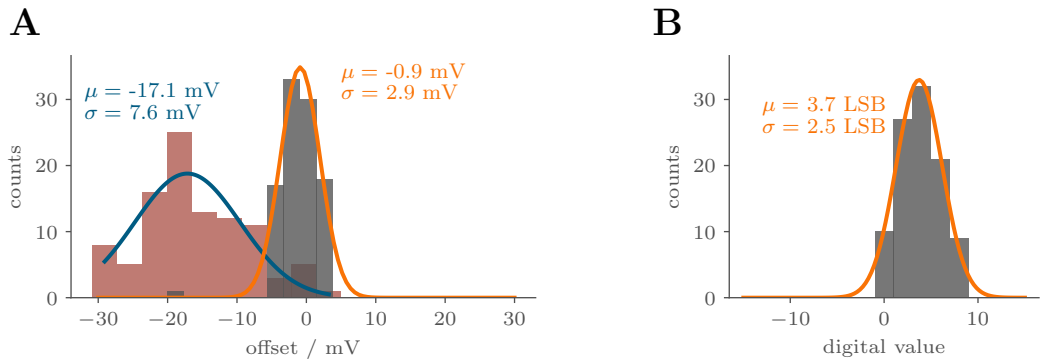


Figure 39: Histogram of the comparators offset compensation. The shift of the mean is caused by an input impedance mismatch and the use of small transmission gates. The solid curves fit Gaussian distributions to the data. **A**: Simulated comparator input offset distribution before (red) and after (gray) calibration. **B**: Distribution of configurations for offset compensation.

5.7 Summary

In this chapter, we present and discuss a fast and energy efficient comparator and its environment. Here, all components are tested altogether. This includes the comparator core, the clock tree, the buffering output latch and two reference generators. The reference generator is selected by the transmission gates designed by Czierlinski (2022).

Comparator input offset compensation. The capacitive reference generator is designed to, among other things, compensate the input offset of the comparator caused by device mismatch. To verify this, we simulate Monte Carlo samples of the full comparator unit and calibrate for the offset. The comparator's signal input is equipped with one transmission gate followed by an ideal voltage source. By doing so, we consider that the

input impedance mismatch in the final ADC may depend on the transmission gate as long as the upstream capacitance is sufficiently large.

The prior and remaining offset is determined by sweeping the comparator's input voltage while continuously triggering the comparator after an initialization sequence of the reference generator. To test the whole comparator for hysteresis at the same time, the sweep has a rising and a falling part. Here, the Monte Carlo samples of the comparator – even with the input transmission gates, its related feedback and a constantly changing input signal – show only minor hysteresis between the falling and the rising sweep of $0.44 \pm 0.14\,\mathrm{mV}$.

Physical quantities. The entire comparator unit is located below the area-dominating capacitive signal DAC. According to the floor plan, 2.5 µm are free on the longitudinal edges to place the inverters for the latter. Figure 40 shows the corresponding layout of the comparator up to the third metal layer. The comparator core is located at the bottom, directly above the trigger tree and the buffering output latch. In the center, two of the transmission gates can be found. The rest of the area is used by the two capacitive voltage dividers and their logic. Together, all comparator components utilize around $710\,\mu\mathrm{m}^2$.

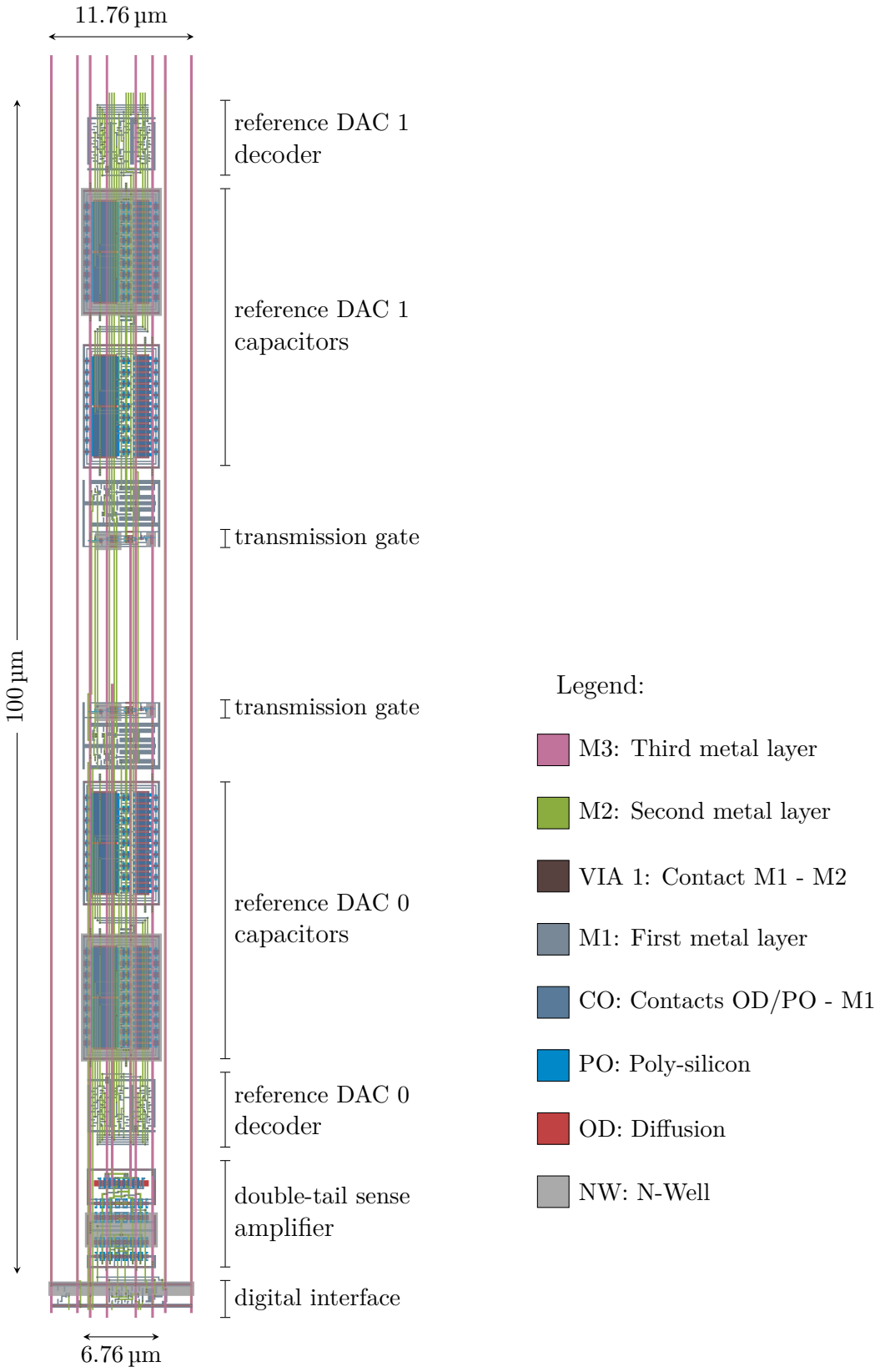


Figure 40: Layout of the entire comparator unit. Power and the input signals for the double-tail sense amplifier are routed on the third metal layer. Due to non-disclosure agreements, the digital standard cells provided by TSMC are hidden.

6 Silicon implementation

In order to meaningfully evaluate the new ADC design, we merge the parts discussed in this thesis with the results presented by Czierlinski (2022). For the following simulations, a joint effort can be assumed.

As mentioned in Section 2.4, the SAR ADC consists of a capacitive DAC, a comparator, and a control logic. These "local" parts are repeated in each ADC channel, and we consider the unit formed by them as the design under test (DUT). Their environment, i.e. global signals including the `sample` signal, is generated by an abstracted behavioral description. The simulation is performed using the framework introduced in Section 3.3.

Initial simulations show that the individual components can work together, but insufficient time buffers have been planned. Therefore, the most critical components are already optimized in this first iteration. The largest of these optimizations is adjusting the transistor sizes for the comparator layout and is described in Section 5.5.1. The existing trigger tree in the comparator also proves to be a problem as long as there is no synthesized clock tree for the digital part and both are fed from the same clock generation. To compensate for this simulation-related effect, we create two clock generators shifted by 100 ps. These clock generators are modeled by ideal voltage sources that generate pulses with a symmetrical duty cycle and a rise/fall time of 40 ps.

6.1 ADC offset error correction

First, we investigate the impact of the reference generator to the ADC output code. Figure 41 shows the ADC response to an analog input stimulus (0.6 V) for the fast as well as for the precise operation mode with respect to different reference DAC configurations. These results are based on a full transistor-level simulation with synthesized control logic in the typical process corner but do not include layout information as e.g. extracted resistances and capacitance. Since in this simulation environment there are variations between the two implemented reference DACs, only one reference DAC is considered here. To account for the different output code ranges in the two operation modes, the figure exhibits two scales. Both are adjusted in a way, that one LSB in the fast mode equals two LSB in the precise mode. When investigating both drawn lines we find an almost linear behavior with a similar negative slope but with different y-axis intercepts. The negative slope factor results from the fact that an increase in the reference voltage corresponds to a relative decrease in the input voltage. This can be illustrated by the

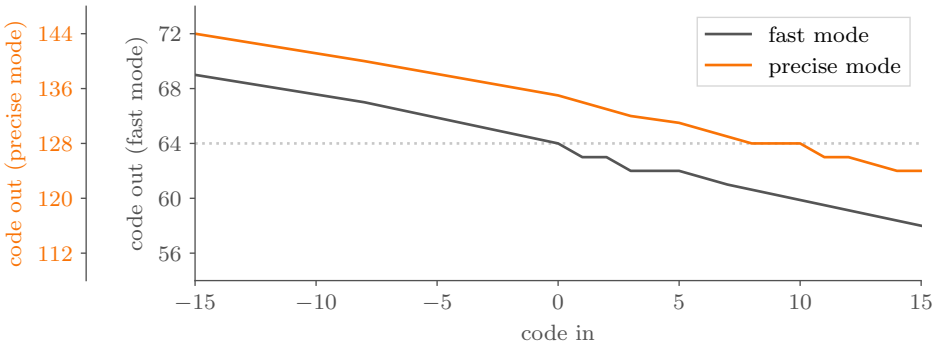


Figure 41: ADC response to an 0.6V input voltage with respect to the reference DAC setting.

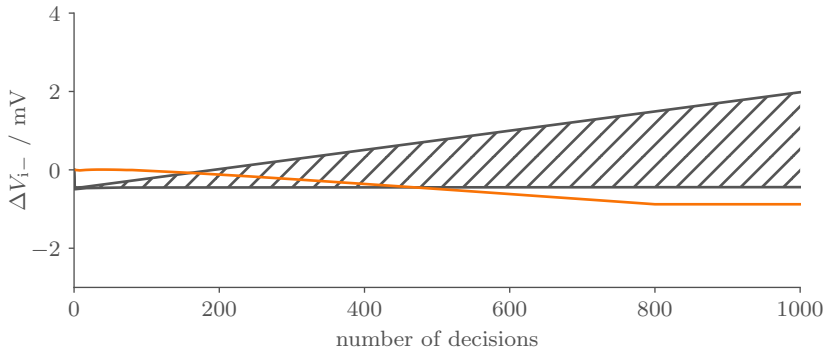


Figure 42: Shift in reference voltage due to comparator kickback. Gray hashed area: V_{m} in a comparator-core only simulation setup (c.f. Figure 24) for all possible V_{p}; simulated without reference DAC. Orange: Reference voltage in a full ADC simulation (8bit, vdd=1.2V, 1 GHz clock frequency).

MSB decision: An input voltage of 0.6 V leads to a high signal at a reference voltage of 0.59 V, while at 0.61V it sets the MSB to zero. Further, the different offsets are caused by the different input impedance in the different modes, leading to an effective offset error in the comparator decision. While in the fast mode only one array is connected to the comparator by one unit-transmission gate, in the precise mode both arrays are connected by two parallel unit-transmission gates. Thus, the charge injection from the kickback effect can be balanced faster compared to the fast mode and, above all, faster than at the reference input of the comparator. With the reference generator, the ADC gains a total ADC error offset compensation with a dynamic range of 11 LSB in the fast mode and 20 LSB in the precise mode.

Reference generator refresh cycles. The question remains how often the reference DAC has to be refreshed. Unfavorably, it turns out that this was only insufficiently

considered during the development of the same. However, there are different forces acting on the reference voltage.

Figure 42 extracts at least one of those: The kickback related effective charge injection. The gray, hatched area shows the voltage shift on the comparators input for a comparator-only setup as described in Figure 24. So far, there is no reference DAC involved; only a pre-charged capacitor and a resistor models the comparators input impedance. The spanned area indicates the simulations for the different voltages occurring at the signal input $V_{\mathrm{i}+}$ of the comparator during operation. For this purpose, we always extract the reference voltage $V_{\mathrm{i}-}$ of the comparator just before the comparator is triggered. Here, the comparator is triggered every 1 ns. This also explains the dip at the beginning: Between simulations, the resting state of the input voltage is not fully reached at the given input resistance. However, this can only be seen as a linear shift, which does not affect the following statements.

We can observe, that even in this ideal setting with optimized simulator configuration, the reference voltage $V_{\mathrm{i}-}$ changes as a function of repeated triggering of the comparator. It is indicated that this is also a function as a function of the input voltage $V_{\mathrm{i}+}$. This suggests that an effective charge injection occurs that is not recovered after a decision. This is probably due the voltage-dependent capacitance of the gates of transistors T1 and T2. Thereby the two differential branches can influence each other via the common source nodes. The phenomenon is also exciting because there is no DC path to and from the comparator input node except for the minimum conductance inserted by the simulator.

The orange line shows the evolution of the reference voltage from one single reference DAC in a simulated ADC while operating in the precise mode. We extracted the voltage at the reference input node of the comparator just before each conversion. Contrary to the expectation aroused by the above observation, this voltage trends to ground and not to supply. Since this simulation incorporates transistor models of the reference DAC a possible explanation would be the observation of lack currents in the switches of the reference DAC. This effect would be time dependent and would not depend on the number of decisions made by the comparator.

In summary, we can state that the prediction of the course of the reference voltage is complicated by many influencing factors. Therefore, it is not possible in this thesis to give an exact time or number of conversions or decisions until the reference DAC has to be refreshed. Unless otherwise specified, we refresh the reference DAC every 400 clocks in the following simulations. For this purpose, we alternate the two implemented reference DACs.

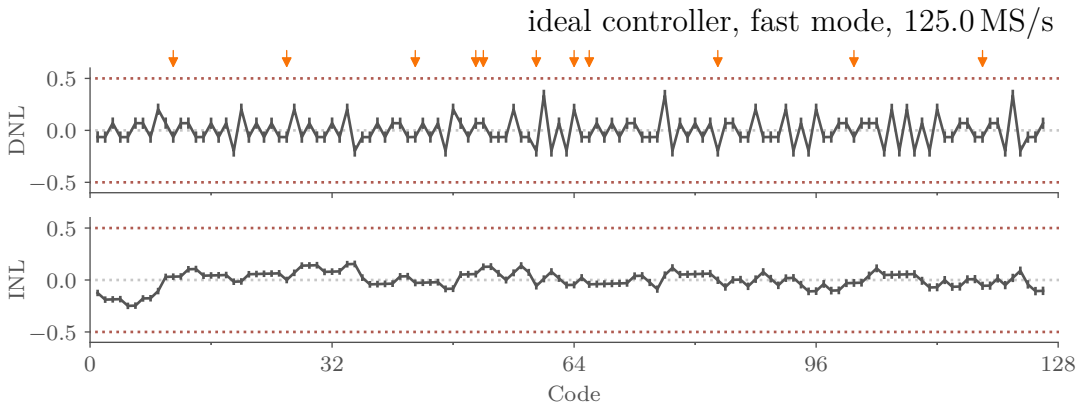


Figure 43: DNL and INL, extracted from a target oriented, optimistic transistor-level simulation of the ADC in its fast, interleaved mode (7 bit) at 125 MS/s sampling frequency (1 GHz clock frequency). The small orange arrows annotate non-monotonic code transitions. The control logic is modeled ideally.

6.2 Limits of the ADC

For an initial estimate of the accuracy of the ADC, it is simulated based on transistor models. In order to better investigate the influence of the individual components, we start with a simulation model that uses the RTL description of the control logic without considering any delays. In other words, we simulate an ideal, infinitely fast control logic. Thus, now we only see the effects caused by the components from Czierlinski (2022) and the comparator circuit. To further normalize the statements, we simulate in the typical process corner, at a temperature of 50 °C and without taking into account variations in manufacturing. The supply voltages are provided by ideal voltage sources at 1.2 V. Since we moreover do not consider layout effects, this is an optimistic estimate overall. As such, it serves as a reference to rank further simulation results.

We do not show sweeps over various supply voltages, temperatures and the different process corners here because they are discussed in detail by Czierlinski (2022). In summary, however, ADC characteristics deteriorate under worsened conditions. In some really harsh conditions (slow-slow process corner and reduced supply voltage), functional operation at full speed is no longer guaranteed. However, if one decreases the operation speed, the ADC becomes functional again and almost as precise as in normal conditions.

Target-oriented simulations. One of the targets for the redesigned ADC is to operate reliable at 125 MS/s in its fast mode. For the precise mode, there are less strict timing requirements. To achieve the 125 MS/s in the fast-mode, the ADC requires a 1 GHz clock.

Figure 43 shows the differential and integral non-linearity of the ADC in its fast, interleaving mode. Uncertainties are calculated from Equations (3.2) and (3.3). Non-monotonic code transitions are denoted by small arrows above the DNL plot. The ADC has gain of 114 LSB/V and an offset of −4.1 LSB. The maximum DNL is 0.34 LSB and the maximum absolute INL is −0.25 LSB. The gain-error (the ideal gain would be 106 LSB/V) is caused by additional capacitance which are caused by the shielding of the capacitive array. It decreases the dynamic range of the ADC. Thus, the offset error is not as dramatic as indicated by the offset value, because we want the center of the dynamic range to be located around 0.6 V. Given the gain error, we calculate the ideal offset to −4.8 LSB. Due to the results in Section 6.1, the reference DACs are both configured to +2. The disagreement of the ideal offset, and the simulated one can be explained by the evaluation method in Section 6.1: There we only take one single sample after the simulation start up, which differs from the following ones due to not pre-charged nodes at the comparator input.

With regard to the DNL and INL, the results are in line with our expectations. If the DNL were negligible, the ADC would be over-optimized for precision which probably would have been at the expense of area and energy consumption. However, if the DNL is greater than 0.5 LSB, accuracy errors in the ADC are more important than the quantization error. The same applies to the INL, which is generally less susceptible in SAR ADCs.

An important observation is that no systematic abnormalities are visible. However, the ADC already shows some of its weakest points here. At this level of simulation, we do not expect non-monotonic code transitions but observe three of them. It turns out, that the two interleaving ADCs individually achieve much better values for DNL and INL and also do not suffer from non-monotonic code transitions. Such an array-dependency must be caused by the influence of precharged nodes in the ADC generating some kind of memory effect.

To simulate the ADC in its precise mode, we start with the same clock frequency like in the fast mode, which leads to a sampling frequency of 62.5 MS/s. Considering Section 6.1, we set the reference DAC to +9. Figure 44A shows the corresponding DNL and INL results. The ADC has a gain of 223 LSB/V (ideally 213 LSB/V) and an offset of −13.2 LSB (ideally −6.0 LSB). This simulation shows no non-monotonic code transition. However, it exhibits two missing codes at the MSB decision, at the positions where the MSB decisions are applied (64 LSB / 192 LSB). At the same time, one LSB before the missing code, the code step is enlarged leading to an increased DNL. This is then also reflected by the INL. The same effect, but with less intensity, also occurs in the code transitions from the second-MSB decision. Here we note DNLs that are grater than 0.5 LSB. Also, in some

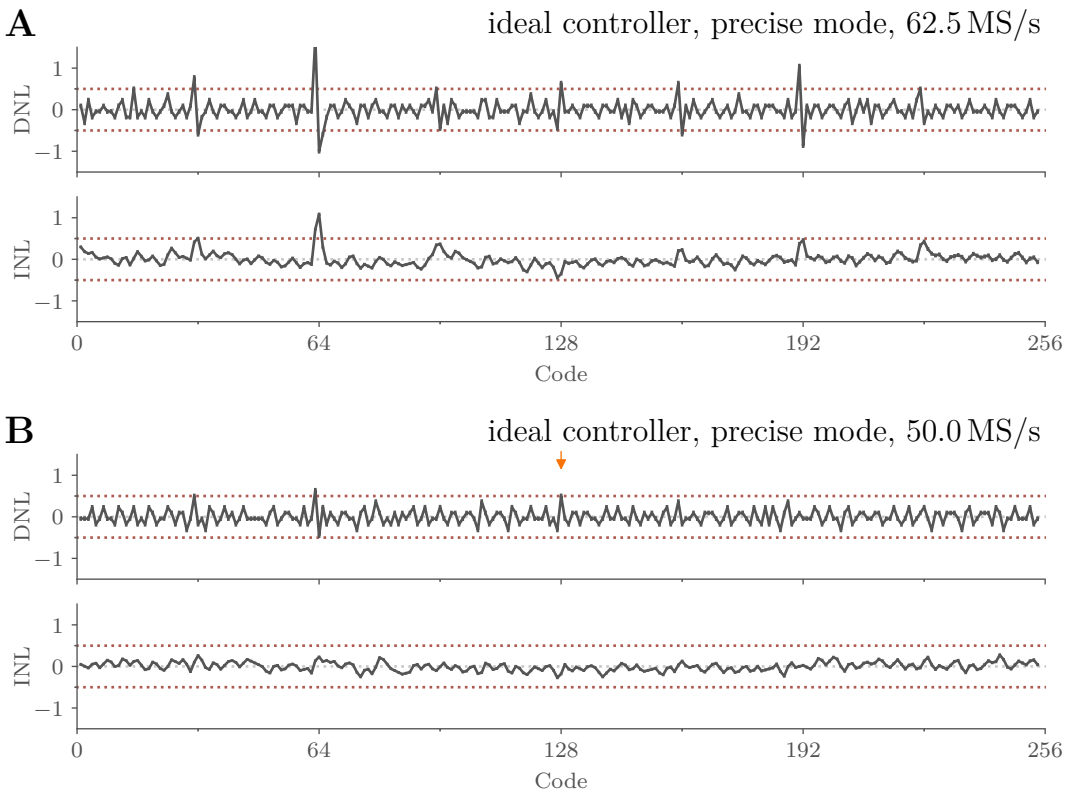


Figure 44: Reducing the clock frequency increases the accuracy. **A**: DNL and INL, extracted from a target oriented, optimistic transistor-level simulation of the ADC in precise mode (8 bit) at 62.5 MS/s sampling frequency (1 GHz clock frequency). The small arrows denote non-monotonic code transitions. The control logic is modeled ideally. **B**: Similar to A, but with reduced clock frequency (800 MHz) and thus a sampling frequency of only 50 MS/s.

other regular positions, this degeneration happens.

There is a common reason for all the effects mentioned: We are running the ADC too fast. Since all transmission gates and components are designed to fit the accuracy requirements in the fast operation mode, we can not act on the assumption, that this automatically fulfill the accuracy requirements of the precise mode at the same clock frequency. Each switch and each transmission gate reacts with a certain RC-time and the convergence to a new common mode voltage follows the same law. In precise mode, however, we have to converge to a new voltage twice as well compared to the fast mode, since we have one bit more of resolution here.

To account for this increased convergence requirements, we simulate the precise mode also with a reduced clock frequency (c.f. Figure 44B). With this change, the accuracy of the ADC increases. Even, if we do not observe any missing code anymore, now the ADC exhibits one non-monotonic code transition. As expected, the gain and the offset of the

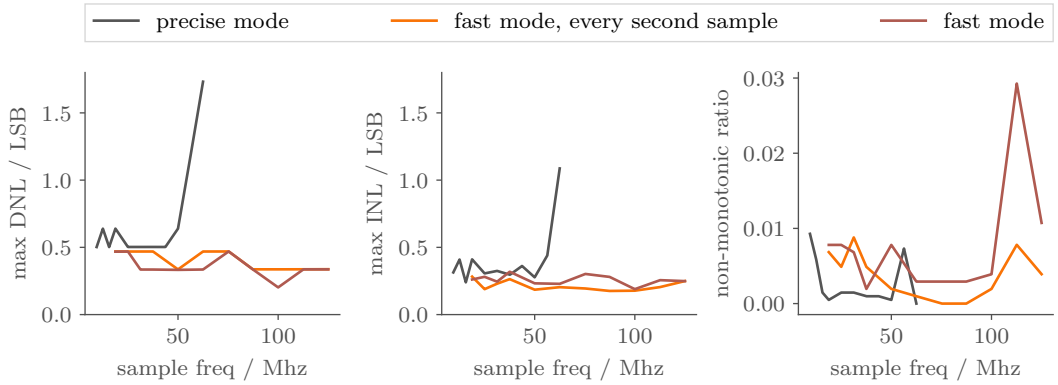


Figure 45: Simulated ADC accuracy based on a transistor level model and an ideal controller with respect to the sampling frequency. The sampling frequency is configured by the clock frequency. The precise mode (8 bit) requires 16 clock cycles per conversion and is thus at most only half as fast as the fast mode (7 bit). The non-monotonic ratio represent the ratio of non-monotonic code transitions with respect to the total number of samples.

ADC at reduced clock frequency are almost identical to the simulation at 1 GHz. The maximum DNL is 0.64 LSB and the maximum absolute INL is 0.28 LSB.

Frequency sweep. In a next step, the influence of the clock frequency and thus the sampling frequency on ADC performance will be investigated systematically. In Figure 45, the accuracy of the ADC is plotted against the sampling frequency, which is set by the clock frequency. The accuracy is represented by the maximum DNL and the maximum INL as well as by the relative proportion of non-monotonic sites in the samples taken. The simulations are performed in the fast mode (7 bit) and in the precise mode (8 bit), where for the fast mode the interleaved sub-ADCs are considered both together and individually.

Most impressively the precise mode shows a deteriorating trend in the DNL and INL for high sampling frequencies. Since all switches are designed for the required accuracy in the fast mode and thus at high clock frequencies the reload-times for the precise mode are not sufficient as expected. In such cases, the largest DNL shows up when applying the MSB decision, usually with an output code of 63 LSB. At lower frequencies the ADC reaches a maximum DNL close to 0.5 LSB.

The behavior of the maximum DNL and INL in 7 bit mode is completely different: Here the ADC seems to get worse at low frequencies. However, it always remains better than in the precise mode. A degradation of the non-monotonic code transitions is visible at a clock frequency of around 900 MHz. This is possibly caused by some phase response of the capacitive arrays in combination with the transmission gate. The comparator's

kickback regularly jolt this RC-circuit. Here, further investigations are required, also to exclude simulation-related effects.

In the right plot, the relative number of non-monotonic codes are shown for the same simulations. Non-monotonic code transitions are critical malfunctions in the ADC. Unfortunately, interleaved ADCs are predestined to exhibit so-called even-odd-effects, which causes a number of these non-monotonic code transitions. Since the simulations do not include component variations at this point and the non-monotonic behavior occurs even without the interleaving, other causes must be present. At all frequencies, the non-monotonic behavior occurs with the refresh cycle of the reference voltage generator. Especially for small clock frequencies, the absolute period of a refresh cycle increases. This goes along with a larger adjustment of the reference voltage and thus a significant jump in the reference voltage to compensate for lost charge caused by the leak current in the reference generator. Thereby, the number of non-monotonic code transitions increases for this smaller clock frequencies. In interleaved operation, the timing of the refresh cycle often corrupts one sample. However, for higher frequencies, the non-monotonic code transitions only occur close to expected code transitions. These are caused by the kickback of the comparator and subsequent effects as well as too small transmission gates and a too slow digital control logic. Already by connecting the comparator to the DAC in the precise mode leads to charge sharing which results in a voltage drop of almost 15 mV due to charge sharing between the comparator input node and the capacitive DAC. In some boundary decisions, the comparator does not provide its result in time to be fetched the digital logic.

In conclusion, we expect a maximum DNL of around 0.6 LSB in the precise and 0.3 LSB in the fast mode in the best case. Also, we do not expect for the non-monotonic code transitions to disappear completely.

Transistor level controller. After the previous section investigated the ADC neglecting the delays of the control logic, the following section includes a transistor based model for a synthesized controller. The synthesis of the successive approximation register itself is described in Section 4.3.3. At this point, we have finally simulated all components with some transistor properties, including an estimation of the timing and a transistor driving strength. The goal of the following simulations is to show that we can achieve equally good DNL and INL, even with the control delay as when we do not consider it. If we do not achieve this at full speed, reducing the frequency is a valid retrenchment of the design constraints.

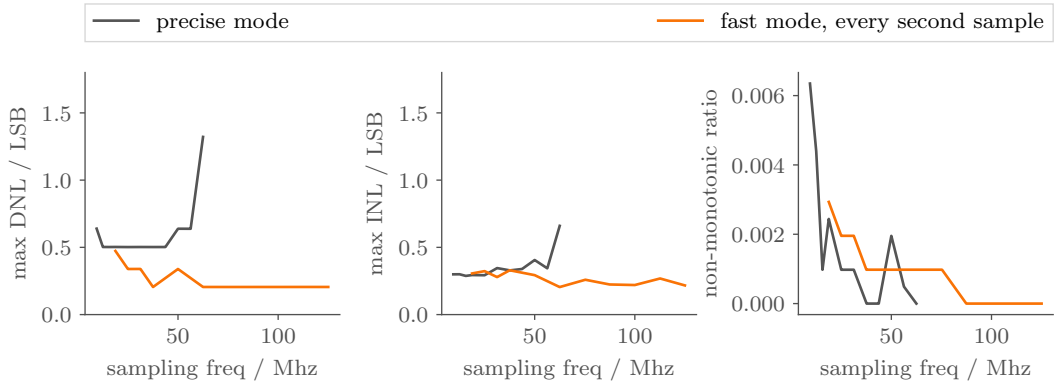


Figure 46: Maximum DNL and INL as well as ratio of samples exhibiting non-monotonic code tranitions for different sampling respectively clock frequencies for the precise (8 bit) and the fast mode (7 bit). Due to a non-symmetric syntheses result and resulting even-odd effects, in the fast mode we only observe one single array or every second sample. The precise mode (8 bit) requires 16 clock cycles per conversion and is thus at most only half as fast as the fast mode (7 bit).

In order to find a sampling and clock frequency that shows the desired behavior, we sweep the clock frequencies in both modes similar to what we have done before. Figure 46 shows the maximum absolute DNL and INL together with the ratio of non-monotonic code transitions. Since Figure 45 already indicates even-odd effects of the two interleaved operating ADCs and since due to the synthesizing process the signal paths of both ADCs differ, in the fast mode we only present the data of one single sub-ADC. The latter will be fixed in the future by instantiating two times the same physical module. So far, in practice, we track only every second sample, similar as in the results above.

Interestingly, we observe less non-monotonic code transitions than with the ideal controller. In terms of DNL and INL we see a slide shift of the accuracy towards lower frequencies in the precise mode. This can be explained by the fact, that the precise mode was already timing critical before and by introducing signal delays for the switches and the DAC control this is even extended.

As illustrated in Figure 47A the ADC reaches an almost similar performance with a synthesized controller compared to an ideal one. Nevertheless, the simulation method differs from Figure 43, since we again only utilize the samples of one array here. In this simulation, the ADC has gain of 114 LSB/V and an offset of −4.3 LSB. The gain does not change, since we have no changes in the parasitic capacitance. Due to the same configuration of the reference generators, same applies to the offset. The maximum DNL is 0.21 LSB and the maximum absolute INL is −0.22 LSB.

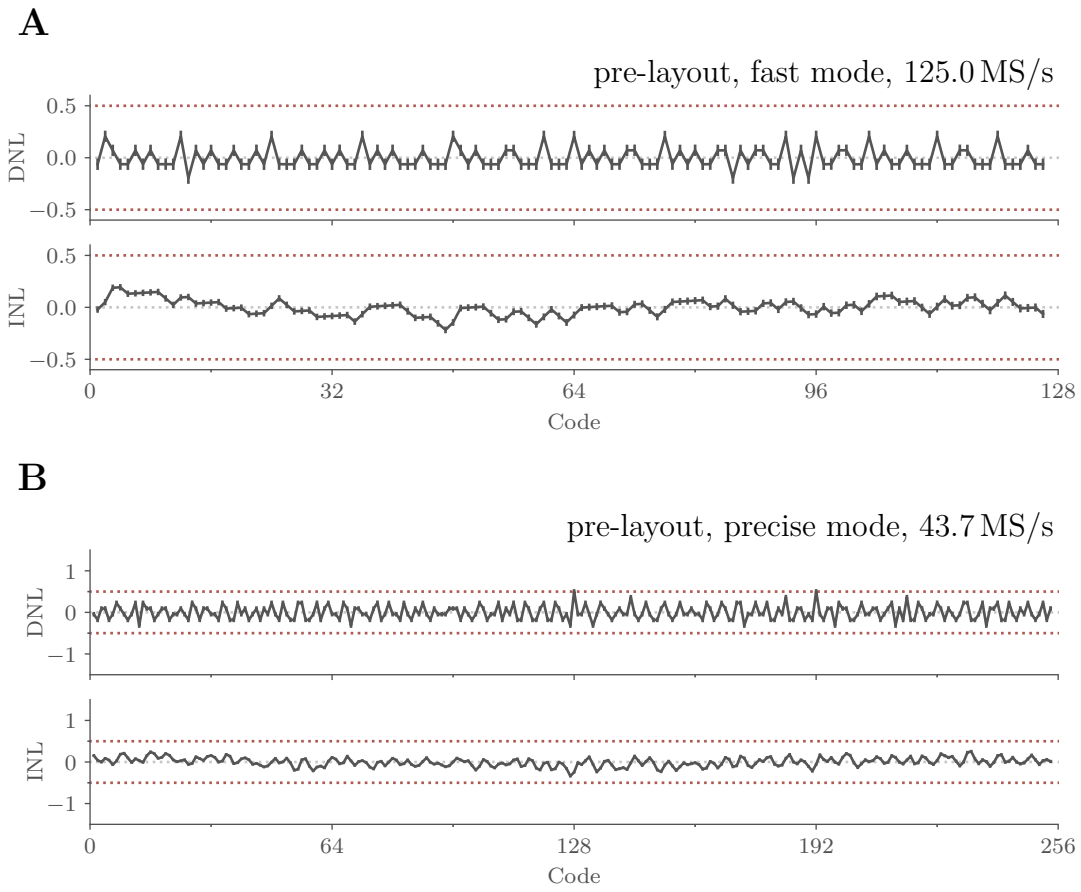


Figure 47: With a synthesized controller we get similar results like before. **A**: DNL and INL, extracted from a transistor-based simulation including the control unit of the ADC in its fast mode (7 bit) at 125 MS/s sampling frequency (1 GHz clock frequency). This plot incorporates results of only one array, so of only every second sample. **B**: Similar to A, but in the precise mode (8 bit) and with a sampling frequency of 43.7 MS/s (700 MHz clock frequency).

Based on the results of Figure 46, for the precise mode (Figure 47B) we select a clock frequency of 700 MHz. Here we do not observe any non-monotonic code transitions and the DNL and INL reaches their minimum. The simulated maximum DNL is 0.50 LSB and the maximum INL is −0.34 LSB. The corresponding ADC gain and offset is similar to the ones in Figure 44.

With these simulations, it is proven that the controller is functional. Except for the possible, not further investigated even-odd effect, there is no significant degradation of the performance due to the control logic visible. On the contrary, the ADC even seems to behave more benignly and is not as susceptible to the refresh cycles of the reference DAC.

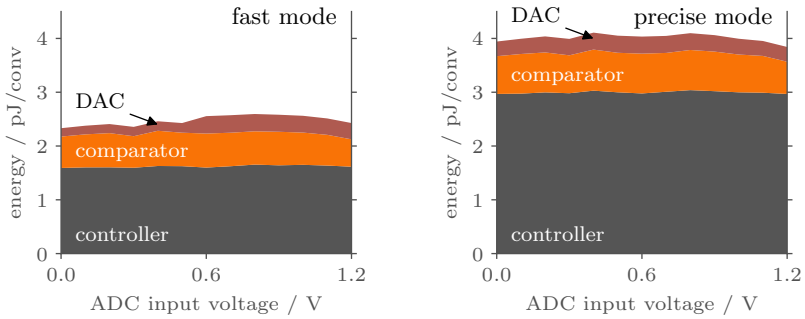


Figure 48: Energy consumption per conversion for the fast and the precise mode and different input voltages, determined in a transistor-level simulation including a synthesized digital logic.

Energy consumption. Before we go on with simulations that include layout information like parasitic capacitance and resistances, we take a closer look on the energy consumption of the ADC. We extract the currents at the power pins of individual components of the ADC. Similar to the energy extraction approach in Section 5.5, we calculate the energy as product of the supply voltage and the integral of the current over the time.

Figure 48 shows the extracted energy consumption of the ADC for different input voltages in the fast as well as in the precise mode. We can see that the energy consumption of the controller and the comparator are almost input voltage independent, while the DAC exhibits an increase of energy consumption for input voltages grater than around 0.6 V which is discussed in detail by Czierlinski (2022). Without considering the refresh cycle of the reference generator and only based on transistor models, the ADC requires 2.33 pJ to 2.60 pJ per conversion in its fast mode. At full speed this corresponds to around 0.31 mW per channel which is approximately 15 % of the total chip power when all ADC channels are enabled (c.f. Section 2.3). Hereby, the largest power consumer is the digital control logic with requires 1.62 ± 0.02 pJ per channel and conversion. The full comparator unit requires 0.61 ± 0.04 pJ per conversion. Hereby the double-tail sense amplifier itself consumes 0.35 ± 0.03 pJ per conversion which fit well with the results for one single decision in Figure 25. Further, the buffer unit has an energy requirement of 0.25 ± 0.02 pJ. The energy consumption of the reference voltage generator is neglectable except for the refresh cycles. Each refresh cycle independently of the mode requires around 0.24 fJ.

In the precise mode, the energy consumption of the control logic enlarges to 3.00 ± 0.02 pJ. This is the case, since now the logic of both arrays operate together converting one sample. Moreover, the comparator accounts for the additional decision and requires 0.72 ± 0.04 pJ in the precise mode per channel and conversion. We can divide this value

again between the double-tail sense amplifier ($0.41 \pm 0.03\,\mathrm{pJ/conv}$) and the buffer unit ($0.31 \pm 0.02\,\mathrm{pJ/conv}$), which scale almost linearly with the numbers of decisions. The simulated total energy consumption of ADC in the precise mode amounts from 2.33 pJ to 2.60 pJ. At a clock frequency of 700 MHz (43.7 MS/s sampling frequency), this leads to a power of 0.18 mW per channel.

More simulations showing the influence of the process corner can be found in Czierlinski (2022). There, also considerations about the maximum current changes can be found.

6.3 Layout based performance evaluation

Finally, we conclude are simulation series with simulations that include layout information. In this section we discuss a version, that contains both, a layout-based model for parts of the ADC that already provides layout information and a transistor-based model for the digital controller. Simulations, that include an extracted layout but an ideal control logic, can be found in Czierlinski (2022). Based on their results, we adjust the sampling frequency in the fast mode to 62.5 MS/s and in the precise mode to 12.5 MS/s.

Figure 49A shows the simulated DNL and INL for the ADC while it is operating in its fast mode. The simulation exhibits an ADC gain of 132 LSB/V and an offset of −13.7 LSB. Hereby the maximum DNL is 0.55 LSB and the maximum INL is −0.40 LSB. The orange arrows mark six non-monotonic code transitions. Figure 49B illustrates the same for the precise mode. Here, we observe a gain of 270 LSB/V and an offset of −35.4 LSB. Unfavorably, the simulations display a regular pattern of non-monotonic code transitions, missing codes and a maximum DNL of 1.53 LSB, which is also reflected in the INL (max. 0.73 LSB). The gain error is dominated by the parasitic capacities in the capacitive DAC. The offset error is not really compensated, because no adjustment of the reference DAC has been done before.

Before comparing these results, we want to point out, that due to the limited time, there is no possibility to repeat, adapt and improve this results during this thesis. The previous results show that the ADC is functional on a transistor-model level, including the control logic. Czierlinski (2022) shows that a layout-based ADC model can work. However, they have simulated the control unit on a register transfer level without modeling delays. Now, our simulation exhibits a significant performance decline in the fast mode and a corrupted precise mode.

It should be noted that our simulation is not directly comparable to the above. In order

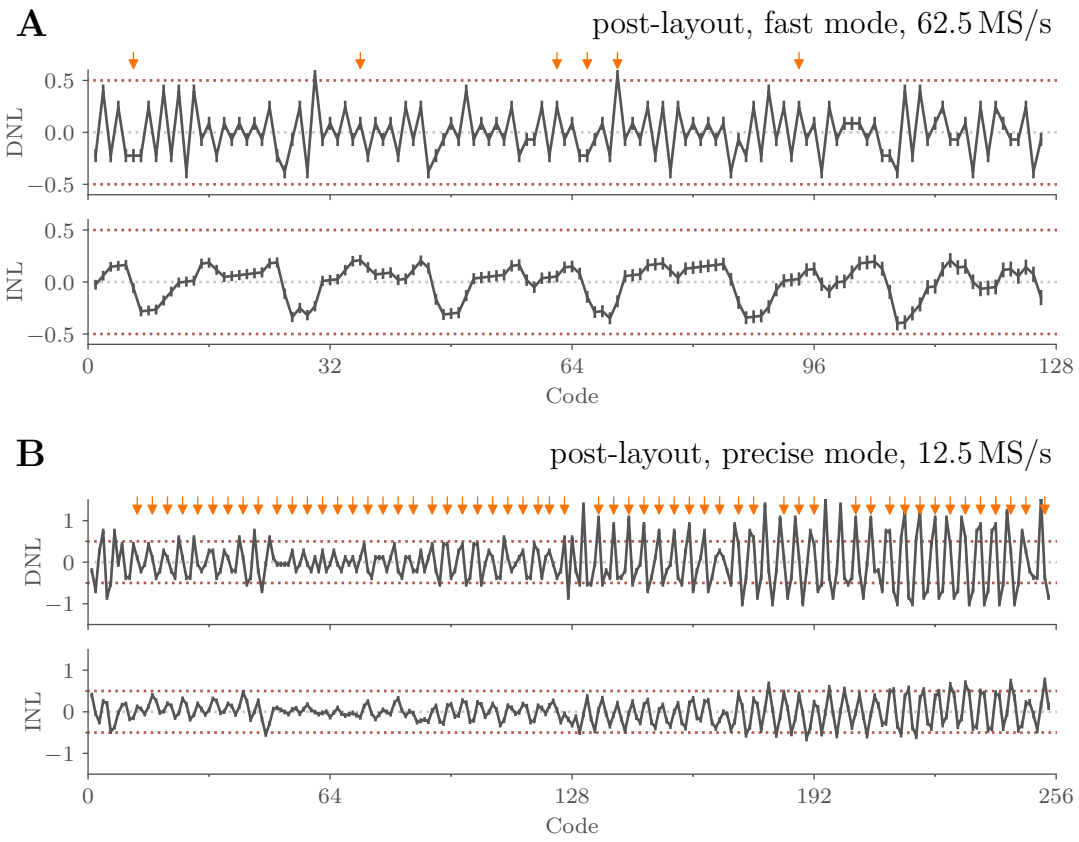


Figure 49: **A**: DNL and INL, extracted from a simulation that includes parasitic capacitances and resistances from the layout and a transistor-level control unit. The ADC operates in its fast mode (7 bit) at 62.5 MS/s sampling frequency (500 MHz clock frequency). This plot incorporates results of only one array, so of only every second sample. The small arrows denote non-monotonic code transitions. **B**: Similar to A, but in the precise mode (8 bit) and with a sampling frequency of 12.5 MS/s (200 MHz clock frequency).

to save simulation time, we split the sweep in eight equal slices and make use of parallel computing. We now join these eight pieces together, throwing away the first sample. It turns out that the first conversion is systematically too low, because at this point the node at the comparator input is not yet precharged to 0.6 V. When the comparator is connected the first time, we observe a charge sharing of this node and the capacitive array. However, in the INL of the fast mode, these eight pieces are clearly visible. In fact, we see only six and a half repetitions, because due to the gain error the first slice mostly contains the output code zero, and the last slice holds almost only output code 127. If there are such time dependencies or start-up effects in the ADC, they are hidden in the previous simulations due to the coherent sweep.

In contrast to the other simulations, in Figure 49B the refresh cycle of the reference voltage generator is reduced to 200 clock cycles. This is done to account for the clock

reduced frequency and constant leak current. Almost every non-monotonic code transition belongs to one of these refresh cycles. And interestingly, this is not only due to the refresh of the voltage itself but due to an even-odd effect between the two reference generators. This behavior is illustrated in Figure 56 (in appendix). Both reference generators are configured to the same value and except for very little routing effect, there should be no differences between the two. Such behavior was not observed in other simulations, also not in Figure 49A. Here, a further investigation is required.

But the precise mode exhibits also at least one more issue: For conversions where the MSB bit is high (digital out is greater 127), we see a structural deterioration in the DNL. Since such a behavior occurred neither in our previous simulations nor in the thesis of Czierlinski (2022), they must be caused by a change in load for the different arrays. Generally, it should be noticed, that all simulations by Czierlinski (2022) do not make use of the useful-skew, which is introduced to reduce the influence of the trigger-tree in the comparator. However, the clock frequency in the discussed simulations are that low, that such effects should not influence the ADC accuracy that much.

Even if the results shown here are not satisfactory on their own, some misbehavior can certainly be avoided by a different configuration. Since the PEX simulations presented by Czierlinski (2022) and the simulations including control logic are promising, it is not necessary to assume that the ADC does not work. Rather, this simulation draws attention to what has not yet received much attention.

7 Discussion

The aim of this work was the creation of a fast and parallel ADC for hybrid neuromorphic in-memory computing. Within this thesis, we developed several components and methods for a dual-mode successive-approximation ADC and tested them jointly with the work of Czierlinski (2022). Due to the complexity of this task, the presented first design iteration does not meet all targets discussed in Section 2.3. Even if individual parts need to be improved in further design cycles, the core concept of the ADC and its components are sustainable. We have with a functional design that is ready for the prototype phase.

This discussion starts with a short comment on the hardware design process: Hardware design is an iterative process. At the beginning of each design cycle, the overall requirement list is converted into constraints for the individual modules and interface definitions to connect them. However, during the design process we suffered from the effect that in small and time-critical circuits, the performance of design idea deteriorates by a factor of two in terms of energy and speed between pre-layout and post-layout simulations. A critical review of the target characteristics in each design step, as well as always designing for the worst cases, might have prevented this.

The following paragraphs comment on the current state of the design.

Evaluation method

All simulations on the full ADC are based on the concept presented in Section 3.3. Beside the artifacts caused by the simulator itself, the simulation approach to take equidistantly samples over entire input voltage range has inherent drawbacks. A Nyquist-ADC is based on the idea that each sample is fully independent of the previous ones. Assuming, this is true, performing a linear sweep will not cause inconsistencies. If this assumption is not true and the ADC has some kind of memory, we are blind to this kind of distortions. The input signal change only slowly and in one direction. Moreover, we can only detect effects that are also covered by the simulation model. In some cases, like in the evaluation of the variations in the capacitive DACs, the manufacturer does not provide the required models and statistics.

Due to the limited time for this study, it was not possible to analyze dynamic effects, for example noise or device distortions in the compounded ADC. As expected for SAR architectures, Czierlinski (2022) demonstrated that a shift in the supply voltage has a high impact on the output code. Therefore, it can be expected that the ADC suffers from

supply noise. To get an overview over all (jointly) performed simulations, the reader is invited to also consult Czierlinski (2022).

This thesis avoids introducing other well-known ADC properties such as the signal-to-noise ratio (SNR), signal-to-noise-and-distortion ratio (SNDR), spurious free dynamic range (SFDR) or the effective number of bits (ENOB). All these properties are intended to describe over-sampling-ADCs. They are recovered from the response of an ADC to a sinus-shaped input stimulus, which we do not mimic in simulations. If we were to mimic them, the simulation time would increase significantly due to their statistic nature. Hence, they are out of the scope for our simulation approach. Ideal Nyquist-ADCs are fully characterized by DNL and INL. One reason using these over-sampling-ADCs characteristics also for Nyquist-ADCs is that real chip measurements are often utilize sinus-stimuli due to a lack of fast and precise tooth-stimuli generators. One can show that the SNR is linked to the DNL and the SFDR to the INL (Khorramabadi, 2010). Nevertheless, we do not dare to attempt to convert the DNL into an ENOB at this point, as this involves many strong mathematical simplifications and inaccuracies.

Comparator

In this thesis, we designed a fast and energy efficient comparator based on a double-tail sense amplifier core. Beside this core module it incorporates a trigger tree, a buffering output latch, two adjustable capacitive reference voltage generators and the transmission gates to connect the latter with the core module.

Double-tail sense Amplifier. The most critical part of the comparator unit is its core unit build from a double-tail sense amplifier (c.f. Section 5.5). During development, we had a special focus on its delay time, input voltage offset and energy consumption per decision. We investigated the influence of different process corners, device variations, input impedance mismatch and static supply voltage changes using a transistor-based simulation model. After three layout attempts, we finally come up with a version, that met our design targets even with included extracted parasitic capacities and resistances. With calibrated input offset, our implemented double-tail sense amplifier achieves to reliably distinguish a 1 mV voltage difference in 298 ps by consuming only 80 fJ. We proved that input voltage offset can be calibrated using the reference DAC yielding a narrow post-calibration offset voltage distribution (2.9 mV).

In Table 8 we compare our design with other sense amplifiers that are designed for SAR

ADCs. The work from Schinkel et al. (2007) is the reference for all double-tail sense amplifiers and the model of the implemented comparator. Even it is manufactured in a larger process node, it is spatially slightly smaller than in the presented comparator. Since we focus more on energy consumption than delay time, our delay time ends up being more than twice as large. The second work in our comparison (Miyahara et al., 2008) is a popular extension. It does not only provide self-calibration but also rely on only one single clock. Except for the area, this work outrange the implemented design in terms of delay time and energy consumption. However, in a first simulation attempt on our part (c.f. Figure 54) did not show these expressive performance gain. Our work fits its design targets, which is good for a first attempt. Nevertheless, there is still room for improvements.

	A: Schinkel	**B**: Miyahara	This work
Process node	90 nm	90 nm	65 nm
Frequency / GHz	2.00	0.25	1.00
Supply / V	1.1	1.0	1.2
V_{cm} / V	0.7	0.6	0.6
Energy / fJ	113†	20	80
Delay / ps	125	122	298
Area / µm^2	82.5	280	100
Architecture	DTSA*	DTSA*	DTSA*

Table 8: Comparison of our simulated comparator with **A** Schinkel et al. (2007) and **B** Miyahara et al. (2008). Delay time and energy consumption are given for an input voltage difference of 1 mV. *Double-tail sense amplifier, †Measured at 50 mV input voltage difference.

To further increase the decision speed of the comparator, one must resort to other transistor configurations. Since the simulated dispersion of the offset is probably larger anyway due to subsequent transistor size adjustment, one can also fall back on the designs which have been discarded for this reason so far. The kickback and its effect on the comparator offset can be reduced by increasing the input transmission gates. As this component is designed by Czierlinski (2022), the transmission gate was not optimized in this work.

Capacitive reference DAC. The reference voltage generator, discussed in Section 5.6, is the second critical element in the comparator unit. We implemented an adjustable capacitive voltage divider, which is to some extent unorthodox for this task. After evaluating the entire ADC, we must note that this approach is not as successful as it initially promised. In particular, we are troubled by the charge loss due to comparator kickback and leakage. The fact that the reference DAC must be refreshed regularly also induces

that two of them have to be implemented. Already in simulations not including statistical variations, we observe even-odd effects. One advantage of our approach is that the comparator does not require any static power. But during every refresh cycle the transient power can be significant. This is caused by the transitions in the switches and their buffer-trees, as well as by the recharging of the array itself. Effectively, the latter does not require energy, but its effect on the transient charge flow must be added to the calculation for the decoupling capacities used to reduce the effect of (bond) wire induction.

The current encoding of the digital value in the reference DAC shows the largest capacity at the boundaries of its dynamic range, while in the middle all additional capacities to increase C_1 and C_2 are decoupled. In an ideal ADC, we expect the best configuration of the reference DAC around the center of its dynamic range: at 0.6 V. Thus, it might be a good idea to have the maximum capacity around this value. This can be achieved by simply changing the decoder which will inevitably become more complex. In Figure 55, such an encoding was successfully tested for an ideal decoder. However, we expect that this increases the influence of device mismatch on the output voltage for medium codes.

It is a balancing act to find a compromise between accuracy and dynamic range with a limited number of memory bits. We presented a compromise, but a final opinion can only come with the results of an in-silicon measurement. Especially since the comparator input offset only affects the ADC offset, it is more important to design a reference generator that is stable over time than one to balance all eventualities.

Digital interface. The trigger tree as well as the buffering output latch work as required and expected. Unfortunately, they require about one-third of the energy budget of the entire comparator unit. This was disregarded in the original energy planning, although these elements must switch in any decision. The simulation in Section 6.2 reveals a power consumption of less than 36 fJ per decision. However, apart from using transistors with a high threshold voltage, there is no (simple) way to reduce the required energy. High-threshold transistor exhibits are reduced leakage current. However, such a measure could increase the gate delays. For the trigger tree, the useful skew relaxing the comparator timing has proven itself worth and should continue to be used.

Controller

During this course, we developed two digital controllers implementing the SAR logic: One sophisticated, flip-flop reduced version and one naive but efficient controller. The flip-flop

reduced version (c.f. Section 4.3.1) seemed promising, but after its synthesizes and the assembling of the entire ADC that we noticed its shortcomings. Since the controller is critical in time, energy, and area, a further look on the final version's implementation is still required. For example, it is important that both arrays are driven by identical gate-level controller logic, which is currently not.

Not surprising the digital controller is the biggest power consumer in the ADC. Most papers, which are not sophisticated or experimental, mention the control logic only in a very limited way. Therefore, a direct comparison with literature is difficult. However, we can see from pictures of produced chips that it is not uncommon for the digital logic to consume a significant amount of space (Van Elzakker et al., 2008; Xu and Ytterdal, 2014).

In this thesis, we arrived at a gate-level netlist of the SAR logic. We performed simulations the on a RTL as well as on a transistor-level. As discussed in Section 4.1, the wire delay exceeds the gate delay in deep-sub-micron processes. So, we expect a significant performance reduction for the ADC when modeling also the wire-delays. However, the functionality of the controller itself should not break since the syntheses applied already a model for wire delays. The in-silicon version of the controller might be different from the simulated one. In-place optimizations and creation of a physical clock and reset tree will change the netlist. Thus, all simulations on the controller are just best guess estimations.

ADC performance

The new ADC should accomplish objectives in the classes of speed, accuracy, area, and energy. However, these are weighted by the application situation. Not all goals are achieved immediately in the first approach. For hybrid in-memory computing, the sampling rate is in the foreground. As shown in Section 6, some initial cutbacks have to be made at this point as well. However, this is not only because of the ADC itself, but also because we lack the time for a systematic layout-based speed analysis.

For now, we can proclaim that the ADC reaches the fast mode's target sampling frequency of 125 Ms/s in pre-layout simulations. In this configuration, the power consumption (2.47 pJ/conv) only slightly exceeds our personal, ambitious guideline, but we stay safely below the limit of 50 % of the total chip power. Post-layout simulations yield a sampling frequency of at least 62.5 MS/s. The fast mode exhibits a resolution of 7 bit which is also usable due to the DNL (max. 0.55 LSB) and INL (max. −0.40 LSB). As discussed in Section 6.3, the relative accuracy in the precise mode is less. We also did not succeed to cover a dynamic input voltage range from 0 V to 1.2 V (Section 6.2). The

observed non-monotonic code transitions are not discussed further here as there various reasons have already been evaluated in Section 6 and by Czierlinski (2022). Incidentally, the majority of the non-monotonic code transitions would have only a minor effect in real measurements. Presumably, they will be absorbed into the other noise that has not yet been simulated. As usual for SAR ADCs, the area is dominated by the capacitive DAC. This DAC already occupies the full area budget. As a result, if the digital logic is not placed below it, we do not match the spatial constraints. However, the overshoot is kept within limits.

	A: Xu	**B**: Van der Plas	**C**: Harpe	This work	
Process node	65 nm	90 nm	65 nm	65 nm	
Architecture	async. SAR	hybrid SAR	SAR	interl. SAR	SAR
Sampling mode	Nyquist	Nyquist	Nyquist	Nyquist	Nyquist
Resolution / bit	7	7	10	7	8
Sampling rate / MS/s	40	150	30	62.5	12.5
max. DNL / LSB	1.08	$\leq$1.00	0.55	0.55*	1.53*
max. INL / LSB	-1.20	$\leq$0.52	0.39	0.40*	0.73*
Energy / pJ/conv	7.46	0.89	2.39	2.47†	4.01†
thereof analog	1.72	-	1.67†	0.85†	1.02†
thereof digital	5.75	-	0.50†	1.62†	3.00†
Supply / V	1.0	1.0	1.0	1.2	
Area / µm^2	17,000	50,000	1,000	$\geq$1,400	

Table 9: Comparison of our simulated ADC with **A** Xu and Ytterdal (2014), **B** Van der Plas and Verbruggen (2008) and **C** Harpe (2018). *Post-layout simulation, †Pre-layout simulation.

Table 9 compares our ADC design with three other projects. These were chosen because, they were implemented in similar process nodes and architectures, and they have speeds of several dozen mega-samples per second. Nevertheless, none of them is designed for a highly parallel application, which is partly reflected in the area consumption. Van der Plas and Verbruggen (2008) also present an ADC that supports two different speed modes (6 bit and 7 bit). Their hybrid ADC consists of a 1 bit stage and a 6 bit stage which are coupled to yield 7 bit. Harpe (2018) demonstrates a very compact ADC using differential capacities. Build for 10 bit resolution, their accuracy is substantial. They utilize the same comparator circuit as we do. The work of Xu and Ytterdal (2014) focus on asynchronous logic and a double reference technique.

For now, the performance of our ADC is in the midfield compared to these other ADCs. However, we would like to emphasize that our ADC has a small spatial footprint. With a prototype, which is advisable in view of the results shown, the design can also be

examined outside the simulator. It would allow analyzing effects that are hard to cover in simulation. This regards in particular effects, which are difficult to estimate in post-layout simulations, such as the impact of noise and capacitor mismatch on the accuracy of the ADC. In future design iterations, the ADC might finally reach the all gathered performance objectives and design targets from Section 2.3. By doing so, the ADC will surpass the performance of the other ADCs in Table 9 in almost all metrics.

One final comment on the architecture: In general, it should be possible to reach significantly higher sampling frequency within the presented ADC architecture. By increasing the energy-budget and transferring the design to a faster process node the most important weak points should be solvable: The architecture always requires eight respectively 16 clock cycles for one conversion. If a clock frequency of a process node works for normal digital circuits, it should be possible to run our ADC architecture at this speed as well.

8 Outlook

This thesis represents possibly a milestone for different applications in BSS-2, but more important it indicates further steps for improvement. It lights up a promising way to a design well suited for the requirements in analog neuromorphic hardware. Thus, work does not stop with the completion of this thesis, but rather calls for driving the new ADC to its true potential. Apparent improvements as discussed in the last section, need to be implemented and tested. For this purpose, a silicon prototype is essential. There are different suggestions in Section 7, that need to be evaluated and prepared for further ADC versions. Especially a redesign of the reference voltage generator in favor of a more classical, smaller, and less susceptible design is mentioned. Afterwards, a placement of the local digital logic below the MOM capacitors of the capacitive array could decrease the total area.

Continuous development of a universal mixed-signal verification framework. Not only the ADC itself shows potential for further developments. The mixed-signal verification tooling evolved jointly with the ADC can be integrated to the usual co-simulation verification flow in the Electronic Vision(s) group. By executing software in contact with simulated hardware, one can verify the hardware jointly with the embedded low-level software. At this point, templates for a proper integration of a direct programming interface (DPI) needs to be included in `TeststandAMS`. `TeststandAMS` could serve as a provider of simulators and enable unit-based mixed signal simulations for parts of the hardware.

So far, `TeststandAMS` uses *Spectre* as analog simulation engine. With some effort, other simulators like *Cadence UltraSim* full-chip simulator could be provided as well. The *UltraSim* simulator comes with the benefit to partition a design into different parts which it then simulates with different precision according to the respective need. By doing so, one could increase the speed for transistor-based digital hardware models.

Moreover, the capability of parsing and modifying netlists should be added to `TeststandAMS`. `TeststandAMS` should be united with the original `Teststand`.

Finishing of the digital logic. In order to bring the ADC to silicon, a placement and routing of the digital logic unit is required. At this point, also the global part of the controller, generating the sampling signal needs to be transformed from behavioral code to a synthesized code. Moreover, the local controller unit can be tested for further improvements in area, energy, and efficiency. A full-custom implementation of the same

might also be considered.

The required two times 5 bit configuration storage per ADC channel for the reference calibration of the reference generators should be implemented. To save area, these might be realized as local SRAM, which are already available from the old CADC. However, latter need to be extended to fit the required number of bits.

In the future, the simple interface between the local and the global logic offers capabilities of a flexible grouping of multiple ADC channels. The trigger period of the ADC is controlled by `g_sample` (c.f. Section 4.2). If we implement multiple, independent instances of the `g_sample`-signal, an ADC-channel could log on one of these. The signal-selection could be controlled by data-bits from the local SRAM. Thus, one can configure mutable groups of channels.

Improvements in the analog part of the ADC. First, the transmission gates should be modified as they are currently not sufficiently wide. This would decrease the kickback effect as transients at the comparator's input can be balanced faster. Vice verse, it would increase the adaption speed of the comparator's inputs on changes in the capacitive array. Moreover, Sections 6.1 and 6.3 show impressively that the reverence voltage generator is not as stable and reliable, as expected. Implementing switches based on high-threshold transistors will reduce the potential leak currents.

Alternatives like band gap voltage reference circuits require constantly energy and are not adjustable. Thus, they are not suited for the new ADC. Probably we find suitable (capacitive) alternatives for the reference generator without a required "hard" refresh. For example, we could buffer the reference voltage on a second capacitive stage.

Silicon prototype. A silicon prototype is the final prove that a design works as intended. By measuring a produced chip, the quality of our simulations can be investigated. As the current design already functional, it is not necessarily required incorporating all the suggested improvements beforehand.

A prototype chip should give access to all important internal states. Therefore, we would implement only a very few ADC channels - just enough to detect crosstalk effects and collect channel statistics. Simultaneously, one should avoid possible additional error sources, such as a PLL. Hence, the clock signal will be generated externally, the configuration will be done by a simple joint test action group (JTAG) controller and the digital codes will be stored in a sufficiently large FF-FIFO.

Future of the ADC in BSS-2. The new ADC is planned to become an integral module in the next generation of BSS-2 ASICs. Here, it will replace the current CADC and improve the quality of analog data readout both in spiking and especially in non-spiking mode. But it might also shift the bottleneck in the signal chain. While up to now the analog-to-digital conversion has limited the operation speed, the PPU and the current implementation of the data transmission might be quickly utilized to capacity at the contemplated sampling frequencies. From now on, handling the digital data that comes in and needs to be processed could become an interesting topic. The synchronization of the ADC's digital output code with the PPU must be clarified. Currently, the PPU operates at a clock frequency of 125 MHz, which would be just enough to fetch one new sample per clock cycle.

A potential future application of a BSS-2-derived system could be symbol identification in optical communication (Arnold et al., 2022). The signals that are transported via fiber optics suffer from dispersion. Therefore, equalizers are used in transceivers to decode an incoming signal. SNNs on neuromorohic devices are one promising realization of these equalizers. However, no matter whether the neuromorphic devices are operating in the electrical domain like BSS-2 or directly in the optic domain, the signals need to be digitized at some point. Designed for neuromophic in-memory computing applications, the presented new ADC architecture might be well suited for this application.

Within this thesis, we have not reinvented the ADC, but achieved developing functional components for a fast dual-mode SAR ADC. In particular, the new ADC operates at significantly increased sampling frequencies compared to the current CADC and provides a defined sampling period. Thereby, the new ADC not only enables a parallel calibration of fast neuron dynamics (e.g., the exponential term in the AdEx circuits), but more importantly facilitates learning and inference applications with NNs. SNNs using the surrogate gradient method become faster, more precise, and thus more efficient. ADC's fast mode empowers rapid compute iterations for vector-matrix multiplications due to its sampling frequency and the fact that the neuron membranes are not needed anymore for charge integration. In summary, we have presented a specialized ADC for neuromorphic mixed-signal in-memory computing that we are now able to manufacture in a prototype series.

Bibliography

IEEE Standard for Verilog Hardware Description Language, IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001), pp. 1–590, doi:10.1109/IEEESTD.2006.99495, 2006.

Universal Verification Methodology (UVM): User's Guide, Accellera Systems Initiative Inc., 8698 Elk Grove Bldv Suite 1, 114, Elk Grove, CA 95624, USA, `https://www.accellera.org/images/downloads/standards/uvm/uvm_users_guide_1.2.pdf`, 2015.

DC Ultra: Concurrent Timing, Area and Test Optimization, Synopsis, `https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/dc-ultra-ds.pdf`, 2018.

IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language, IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012), pp. 1–1315, doi:10.1109/IEEESTD.2018.8299595, 2018.

Allen, P. E. and Holberg, D. R.: CMOS analog circuit design, Oxford University Press, 1987.

Applicos: ATX7006: Transition points in ADCs, `http://www.atx7006.com/articles/static_analysis/adc_transition_points`, accessed: 2022-10-22.

Arnold, E., Böcherer, G., Müller, E., Spilger, P., Schemmel, J., Calabrò, S., and Kuschnerov, M.: Spiking Neural Network Equalization on Neuromorphic Hardware for IM/DD Optical Communication, arXiv preprint arXiv:2206.00401, 2022.

Balasubramanian, S. and Hardee, P.: Solutions for mixed-signal soc verification using real number models, Cadence Design Systems, pp. 1–4, 2013.

Bashir, S., Ali, S., Ahmed, S., and Kakkar, V.: Analog-to-digital converters: A comparative study and performance analysis, in: 2016 International Conference on Computing, Communication and Automation (ICCCA), pp. 999–1001, IEEE, 2016.

Billaudelle, S.: From transistors to learning systems: Circuits and algorithms for brain-inspired computing, Ph.D. thesis, Universität Heidelberg, 2022.

Billaudelle, S., Cramer, B., Petrovici, M. A., Schreiber, K., Kappel, D., Schemmel, J., and Meier, K.: Structural plasticity on an accelerated analog neuromorphic hardware system, arXiv preprint arXiv:1912.12047, 2019.

Billaudelle, S., Stradmann, Y., Schreiber, K., Cramer, B., Baumbach, A., Dold, D., Göltz, J., Kungl, A. F., Wunderlich, T. C., Hartel, A., et al.: Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, in: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5, IEEE, 2020.

Billaudelle, S., Weis, J., Dauer, P., and Schemmel, J.: An accurate and flexible analog emulation of AdEx neuron dynamics in silicon, arXiv preprint arXiv:2209.09280, 2022.

Brette, R. and Gerstner, W.: Adaptive exponential integrate-and-fire model as an effective description of neuronal activity, Journal of neurophysiology, 94, 3637–3642, 2005.

Burkhardt, N.: Introduction to Functional Verification, 2012.

Chandankhede, R. D., Acharya, D. P., and Patra, P. K.: Design of high speed Sense Amplifier for SRAM, in: 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, pp. 340–343, IEEE, 2014.

Chandrakasan, A. P., Sheng, S., and Brodersen, R. W.: Low-power CMOS digital design, IEICE Transactions on Electronics, 75, 371–382, 1992.

Chen, B., Maddox, M., Coln, M. C., Lu, Y., and Fernando, L. D.: Precision passive-charge-sharing SAR ADC: Analysis, design, and measurement results, IEEE Journal of Solid-State Circuits, 53, 1481–1492, 2018.

Chio, U.-F., Wei, H.-G., Zhu, Y., Sin, S.-W., Seng-Pan, U., and Martins, R.: A self-timing switch-driving register by precharge-evaluate logic for high-speed SAR ADCs, in: APCCAS 2008-2008 IEEE Asia Pacific Conference on Circuits and Systems, pp. 1164–1167, IEEE, 2008.

Chiuchisan, I., Potorac, A. D., and Graur, A.: Finite state machine design and VHDL coding techniques, Development and Application Systems, p. 75, 2010.

Cramer, B., Billaudelle, S., Kanya, S., Leibfried, A., Grübl, A., Karasenko, V., Pehle, C., Schreiber, K., Stradmann, Y., Weis, J., et al.: Surrogate gradients for analog neuromorphic computing, Proceedings of the National Academy of Sciences, 119, e2109194 119, 2022.

Craninckx, J. and Van der Plas, G.: A 65fJ/conversion-step 0-to-50MS/s 0-to-0.7 mW 9b charge-sharing SAR ADC in 90nm digital CMOS, in: 2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, pp. 246–600, IEEE, 2007.

Czierlinski, M.: The implementation of a dual-mode DAC for an SAR ADC, Master's thesis, Universität Heidelberg, 2022.

Friedmann, S.: A new approach to learning in neuromorphic hardware, Ph.D. thesis, Heidelberg, Univ., Diss., 2013, 2013.

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., and Meier, K.: Demonstrating hybrid learning in a flexible neuromorphic hardware system, IEEE transactions on biomedical circuits and systems, 11, 128–142, 2016.

Gajski, D. D., Vahid, F., Narayan, S., and Gong, J.: Specification and design of embedded systems, Prentice-Hall, Inc., 1994.

Gao, H., Baltus, P., and Meng, Q.: Low voltage comparator for high speed ADC, in: 2010 International Symposium on Signals, Systems and Electronics, vol. 1, pp. 1–4, IEEE, 2010.

Gao, J., Li, G., Huang, L., and Li, Q.: An amplifier-free pipeline-SAR ADC architecture with enhanced speed and energy efficiency, IEEE Transactions on Circuits and Systems II: Express Briefs, 63, 341–345, 2015.

Gerstner, W. and Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity, Cambridge university press, 2002.

Göltz, J., Baumbach, A., Billaudelle, S., Kungl, A., Breitwieser, O., Meier, K., Schemmel, J., Kriener, L., and Petrovici, M. A.: Fast and deep neuromorphic learning with first-spike coding, in: Proceedings of the neuro-inspired computational elements workshop, pp. 1–3, 2020.

Grübl, A.: VLSI implementation of a spiking neural network, Ph.D. thesis, Universität Heidelberg, 2007.

Grübl, A., Billaudelle, S., Cramer, B., Karasenko, V., and Schemmel, J.: Verification and design methods for the brainscales neuromorphic hardware system, Journal of Signal Processing Systems, 92, 1277–1292, 2020.

Harpe, P.: A compact 10-b SAR ADC with unit-length capacitors and a passive FIR filter, IEEE Journal of Solid-State Circuits, 54, 636–645, 2018.

Harter, M.: Analysis of Back-End Tools in a Cell-Based Design Flow of a High Performance Multi-Million Gate ASIC, Diploma thesis, LSRA, 2002.

Hedenstierna, N. and Jeppson, K. O.: Comments on the optimum CMOS tapered buffer problem, IEEE journal of solid-state circuits, 29, 155–158, 1994.

Heimbrecht, A.: Compiler Support for the BrainScaleS Plasticity Processor, Bachelorarbeit, Universität Heidelberg, 2017.

Hock, M.: Capacitive Memory Specication, URL `https://brainscales-r.kip.uni-heidelberg.de/projects/cm-spec`, unpublished, accessable via Electronic Vision(s) OpenProject server., 2015.

Hofmann, M. and Lange, M.: Automaten auf endlichen Bäumen, in: Automatentheorie und Logik, pp. 163–171, Springer, 2011.

Hsu, C.-L. and Ho, M.-H.: High-speed sense amplifier for SRAM applications, in: The 2004 IEEE Asia-Pacific Conference on Circuits and Systems, 2004. Proceedings., vol. 1, pp. 577–580, IEEE, 2004.

Hu, C.: Modern semiconductor devices for integrated circuits, vol. 2, Prentice Hall Upper Saddle River, NJ, 2010.

Jeon, H. and Kim, Y.-B.: A CMOS low-power low-offset and high-speed fully dynamic latched comparator, in: 23rd IEEE International SOC Conference, pp. 285–288, IEEE, 2010.

Kaiser, J., Billaudelle, S., Müller, E., Tetzlaff, C., Schemmel, J., and Schmitt, S.: Emulating dendritic computing paradigms on analog neuromorphic hardware, Neuroscience, 489, 290–300, 2022.

Katz, R.: Finite State Machine Design, in: Contemporary Logic Design, The Benjamin/Cummings Publishing Company, 1994.

Khorramabadi, H.: Analog-Digital Interface Integrated Circuits, `https://inst.eecs.berkeley.edu/~ee247/fa10/files07/lectures/L12_2_f10.pdf`, 2010.

Kriener, L.: Characterization of Single-Neuron Dynamics in the Development of Neuromorphic Hardware, Master's thesis, Heidelberg University, 2017.

Kugelstadt, T.: The operation of the SAR-ADC based on charge redistribution, Analog Application Journal, 2000.

Lapicque, L.: Recherches quantitatives sur l'excitation électrique des nerfs traitée commeune polarisation, J. Physiol. Pathol. Gen, 9, 620–635, 1907.

Leibfried, A.: On-chip calibration and closed-loop experiments on analog neuromorphic hardware, Masterarbeit, Universität Heidelberg, 2021.

Little, S. et al.: Verilog-AMS Language Reference Manual, Accellera Systems Initiative Inc., 1370 Trancas Street 163, Napa, CA 94558, USA, 2.4.0 edn., https://www.accellera.org/images/downloads/standards/v-ams/VAMS-LRM-2-4.pdf, 2014.

LSM: MADC, URL https://gerrit.bioai.eu:9443/gitweb?p=hicann-dls-private.git;a=blob_plain;f=hicann-dls/units/madc/doc/MADC.pdf;hb=HEAD;js=1, microelectronic systems laboratory at EPFL (Lausanne), version 2, 2015.

Miyahara, M., Asada, Y., Paik, D., and Matsuzawa, A.: A low-noise self-calibrating dynamic comparator for high-speed ADCs, in: 2008 IEEE Asian Solid-State Circuits Conference, pp. 269–272, IEEE, 2008.

Nagel, L. W. and Pederson, D.: SPICE (Simulation Program with Integrated Circuit Emphasis), Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley, URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html, 1973.

Naraghi, S., Courcy, M., and Flynn, M. P.: A 9-bit, 14 μW and 0.06 mm^2 Pulse Position Modulation ADC in 90 nm Digital CMOS, IEEE Journal of Solid-State Circuits, 45, 1870–1880, 2010.

Nikolic, B., Oklobdzija, V. G., Stojanovic, V., Jia, W., Chiu, J. K.-S., and Leung, M. M.-T.: Improved sense-amplifier-based flip-flop: Design and measurements, IEEE Journal of Solid-State Circuits, 35, 876–884, 2000.

Oh, K.-S. and Jung, K.: GPU implementation of neural networks, Pattern Recognition, 37, 1311–1314, 2004.

Pehle, C., Billaudelle, S., Cramer, B., Kaiser, J., Schreiber, K., Stradmann, Y., Weis, J., Leibfried, A., Müller, E., and Schemmel, J.: The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity, Frontiers in Neuroscience, 16, 2022.

Pelgrom, M.: Analog-to-Digital Conversion, Springer, 2017.

Pelgrom, M. J., Duinmaijer, A. C., and Welbers, A. P.: Matching properties of MOS transistors, IEEE Journal of solid-state circuits, 24, 1433–1439, 1989.

Quarles, T., Newton, A., Pederson, D., and Sangiovanni-Vincentelli, A.: SPICE 3 Version 3F5 User's Manual, 1994.

Reddy, K., Rao, S., Inti, R., Young, B., Elshazly, A., Talegaonkar, M., and Hanumolu, P. K.: A 16-mW 78-dB SNDR 10-MHz BW CT $\Delta\Sigma$ ADC Using Residue-Cancelling VCO-Based Quantizer, IEEE journal of solid-state circuits, 47, 2916–2927, 2012.

Schemmel, J.: The BrainScaleS accelerated analogue neuromorphic architecture, Brain-inspired Computing, p. 14, 2021.

Schemmel, J., Billaudelle, S., Dauer, P., and Weis, J.: Accelerated Analog Neuromorphic Computing, arXiv preprint arXiv:2003.11996, 2020.

Schinkel, D., Mensink, E., Klumperink, E. A., van Tuijl, E., and Nauta, B.: A 3-Gb/s/ch transceiver for 10-mm uninterrupted RC-limited global on-chip interconnects, IEEE Journal of Solid-State Circuits, 41, 297–306, 2005.

Schinkel, D., Mensink, E., Klumperink, E., Van Tuijl, E., and Nauta, B.: A double-tail latch-type voltage sense amplifier with 18ps setup+ hold time, in: 2007 IEEE international solid-state circuits conference. Digest of technical papers, pp. 314–605, IEEE, 2007.

Schreiber, K.: Accelerated neuromorphic cybernetics, Ph.D. thesis, Universität Heidelberg, 2021.

Spilger, P., Müller, E., Emmel, A., Leibfried, A., Mauch, C., Pehle, C., Weis, J., Breitwieser, O., Billaudelle, S., Schmitt, S., et al.: hxtorch: PyTorch for BrainScaleS-2, in: IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning, pp. 189–200, Springer, 2020.

Stradmann, Y., Billaudelle, S., Breitwieser, O., Ebert, F. L., Emmel, A., Husmann, D., Ilmberger, J., Müller, E., Spilger, P., Weis, J., et al.: Demonstrating analog inference on the BrainScaleS-2 mobile system, IEEE Open Journal of Circuits and Systems, 2022.

Sylvester, D. and Keutzer, K.: A global wiring paradigm for deep submicron design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19, 242–252, 2000.

Van der Plas, G. and Verbruggen, B.: A 150 MS/s 133 μW 7 bit ADC in 90 nm Digital CMOS, IEEE Journal of Solid-State Circuits, 43, 2631–2640, 2008.

Van Elzakker, M., Van Tuijl, E., Geraedts, P., Schinkel, D., Klumperink, E., and Nauta, B.: A 1.9 μW 4.4 fJ/conversion-step 10b 1MS/s charge-redistribution ADC, in: 2008 IEEE International Solid-State Circuits Conference-Digest of Technical Papers, pp. 244–610, IEEE, 2008.

Von Neumann, J.: First Draft of a Report on the EDVAC, IEEE Annals of the History of Computing, 15, 27–75, 1993.

Weis, J.: Inference with Artificial Neural Networks on Neuromorphic Hardware, Master's thesis, Universität Heidelberg, 2020.

Weis, J., Spilger, P., Billaudelle, S., Stradmann, Y., Emmel, A., Müller, E., Breitwieser, O., Grübl, A., Ilmberger, J., Karasenko, V., et al.: Inference with artificial neural networks on analog neuromorphic hardware, in: IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning, pp. 201–212, Springer, 2020.

Wong, K.-L. and Yang, C.-K.: Offset compensation in comparators with minimum input-referred supply noise, IEEE Journal of Solid-State Circuits, 39, 837–840, 2004.

Xu, D. and Xu, S.: High-speed and low-power logic for SAR ADC, in: 2017 IEEE 17th International Conference on Communication Technology (ICCT), pp. 1166–1170, IEEE, 2017.

Xu, Y. and Ytterdal, T.: A 7-bit 40 MS/s single-ended asynchronous SAR ADC in 65 nm CMOS, Analog Integrated Circuits and Signal Processing, 80, 349–357, 2014.

Zhang, H., George, V., and Rabaey, J. M.: Low-swing on-chip signaling techniques: Effectiveness and robustness, IEEE Transactions on very large scale integration (VLSI) systems, 8, 264–272, 2000.

Zhu, J., Bai, N., and Wu, J.: A review of sense amplifiers for static random access memory, IETE Technical Review, 30, 72–81, 2013.

List of figures

List of tables

Appendix

RTL-level HDL code describing the SAR

The following three figures include the RTL code used to represent the digital logic of the ADC. The breakdown of the module `controller_local_top` into three parts (A, B and C) is only for presentation reasons.

```
1  module controller_local_top (
2      // left edge
3      input wire logic clk,
4      input wire logic res_b,
5      input wire logic g_sample,
6      input wire logic g_fast_mode,
7      input wire logic g_calibrate,
8      // top edge
9      output var logic [7:0] digital_out,
10     output var logic digital_out_valid,
11     // bootom edge
12     output var logic comparator_enable,
13     input logic comparator_out,
14     output var logic [7:0] A_array,
15     output var logic A_connect_input,
16     output var logic A_reset,
17     output var logic A_connect_comparator,
18     output var logic [7:0] B_array,
19     output var logic B_connect_input,
20     output var logic B_reset,
21     output var logic B_connect_comparator);
22
23 typedef enum logic [3:0]  {
24   RESET  = 4'b0000,
25   TRACK  = 4'b0001,
26   C7     = 4'b1001,
27   C6     = 4'b1000,
28   C5     = 4'b1100,
29   C4     = 4'b1101,
30   C3     = 4'b1111,
31   C2     = 4'b1110,
32   C1     = 4'b1010,
33   C0     = 4'b1011,
34   FINISH = 4'b0011  } sub_state_e;
35
36 sub_state_e state_array_A;
37 sub_state_e state_array_B;
38 logic A_sample;
39 logic B_sample;
40
41 // output logic
42 always_comb begin
43   digital_out_valid = (state_array_A == FINISH) | (state_array_B == FINISH);
44   digital_out[7:0] = ~(state_array_A == FINISH) & (state_array_B == FINISH) ?
45     B_array [7:0] : A_array[7:0];
46   comparator_enable = state_array_A[3] | state_array_B[3];
47 end
```

Figure 50: System verilog HDL code implementing of the SAR. Part A.

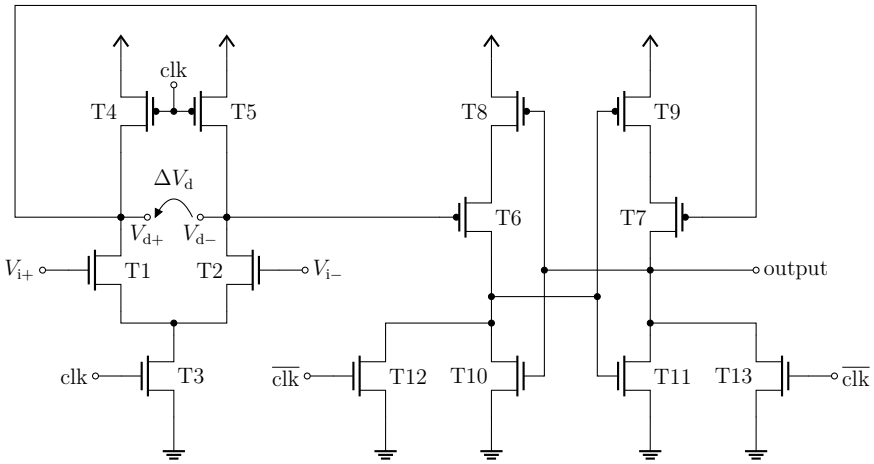
```
48  // ------------
49  // FSM Array A
50  // ------------
51  always_ff @(posedge clk or negedge res_b) begin
52    if (!res_b) begin
53      state_array_A <= RESET;
54    end else begin
55      case (state_array_A)
56        RESET:   state_array_A <= A_sample ? TRACK : RESET;
57        TRACK:   state_array_A <= !A_sample ? C7 : TRACK;
58        C7:      state_array_A <= C6;
59        C6:      state_array_A <= C5;
60        C5:      state_array_A <= C4;
61        C4:      state_array_A <= C3;
62        C3:      state_array_A <= C2;
63        C2:      state_array_A <= C1;
64        C1:      state_array_A <= g_fast_mode ? FINISH : C0;
65        C0:      state_array_A <= FINISH;
66        FINISH:  state_array_A <= RESET;
67        default: state_array_A <= RESET;
68      endcase
69    end
70  end
71  always_comb begin
72    A_sample             = g_sample;
73    A_connect_input      = (state_array_A == TRACK);
74    A_reset              = (state_array_A == RESET);
75    A_connect_comparator = (state_array_A[3] | !g_fast_mode);
76  end
77  always_ff @(posedge clk) begin
78    casex ({state_array_A, g_fast_mode})
79      {FINISH, 1'b1}: A_array <= 8'b01111111;
80      {FINISH, 1'b0}: A_array <= 8'b11111111;
81      {RESET, 1'b1}:  A_array <= 8'b01111111;
82      {RESET, 1'b0}:  A_array <= 8'b11111111;
83      {C7, 1'b1}: begin
84        A_array[7] <= comparator_out;
85        A_array[6] <= 1'b0; end
86      {C7, 1'b0}:   A_array[7] <= comparator_out;
87      {C6, 1'b1}: begin
88        A_array[6] <= comparator_out;
89        A_array[5] <= 1'b0; end
90      {C6, 1'b0}:   A_array[6] <= comparator_out;
91      {C5, 1'b1}: begin
92        A_array[5] <= comparator_out;
93        A_array[4] <= 1'b0; end
94      {C5, 1'b0}:   A_array[5] <= comparator_out;
95      {C4, 1'b1}: begin
96        A_array[4] <= comparator_out;
97        A_array[3] <= 1'b0; end
98      {C4, 1'b0}:   A_array[4] <= comparator_out;
99      {C3, 1'b1}: begin
100       A_array[3] <= comparator_out;
101       A_array[2] <= 1'b0; end
102     {C3, 1'b0}:   A_array[3] <= comparator_out;
103     {C2, 1'b1}: begin
104       A_array[2] <= comparator_out;
105       A_array[1] <= 1'b0; end
106     {C2, 1'b0}:   A_array[2] <= comparator_out;
107     {C1, 1'b1}: begin
108       A_array[1] <= comparator_out;
109       A_array[0] <= 1'b0; end
110     {C1, 1'b0}:   A_array[1] <= comparator_out;
111     {C0, 1'b0}:   A_array[0] <= comparator_out;
112   endcase
113 end
```

Figure 51: System verilog HDL code implementing of the SAR. Part B.

```
114 // ------------
115 // FSM Array B
116 // ------------
117 always_ff @(posedge clk or negedge res_b) begin
118   if (!res_b) begin
119     state_array_B <= RESET;
120   end else begin
121     case (state_array_B)
122       RESET:   state_array_B <= B_sample ? TRACK : RESET;
123       TRACK:   state_array_B <= !B_sample ? C7 : TRACK;
124       C7:      state_array_B <= C6;
125       C6:      state_array_B <= C5;
126       C5:      state_array_B <= C4;
127       C4:      state_array_B <= C3;
128       C3:      state_array_B <= C2;
129       C2:      state_array_B <= C1;
130       C1:      state_array_B <= g_fast_mode ? FINISH : C0;
131       C0:      state_array_B <= FINISH;
132       FINISH:  state_array_B <= RESET;
133       default: state_array_B <= RESET;
134     endcase
135   end
136 end
137 always_comb begin
138   B_sample             = g_sample & !(g_fast_mode & (A_reset | A_connect_input));
139   B_connect_input      = (state_array_B == TRACK);
140   B_reset              = (state_array_B == RESET);
141   B_connect_comparator = (state_array_B[3] | !g_fast_mode);
142 end
143 always_ff @(posedge clk) begin
144   casex ({state_array_B, g_fast_mode})
145     {RESET, 1'b1}:    B_array <= 8'b01111111;
146     {RESET, 1'b0}:      B_array <= 8'b00000000;
147     {FINISH, 1'b1}:     B_array <= 8'b01111111;
148     {FINISH, 1'b0}:   B_array <= 8'b00000000;
149     {C7, 1'b1}: begin
150       B_array[7] <= comparator_out;
151       B_array[6] <= 1'b0; end
152     {C7, 1'b0}:         B_array[7] <= comparator_out;
153     {C6, 1'b1}: begin
154       B_array[6] <= comparator_out;
155       B_array[5] <= 1'b0; end
156     {C6, 1'b0}:         B_array[6] <= comparator_out;
157     {C5, 1'b1}: begin
158       B_array[5] <= comparator_out;
159       B_array[4] <= 1'b0; end
160     {C5, 1'b0}:         B_array[5] <= comparator_out;
161     {C4, 1'b1}: begin
162       B_array[4] <= comparator_out;
163       B_array[3] <= 1'b0; end
164     {C4, 1'b0}:         B_array[4] <= comparator_out;
165     {C3, 1'b1}: begin
166       B_array[3] <= comparator_out;
167       B_array[2] <= 1'b0; end
168     {C3, 1'b0}:         B_array[3] <= comparator_out;
169     {C2, 1'b1}: begin
170       B_array[2] <= comparator_out;
171       B_array[1] <= 1'b0; end
172     {C2, 1'b0}:        B_array[2] <= comparator_out;
173     {C1, 1'b1}: begin
174       B_array[1] <= comparator_out;
175       B_array[0] <= 1'b0; end
176     {C1, 1'b0}:        B_array[1] <= comparator_out;
177     {C0, 1'b0}:        B_array[0] <= comparator_out;
178   endcase
179 end
180 endmodule
```

Figure 52: System verilog HDL code implementing of the SAR. Part C.

Alternative developments of the double-tail sense amplifier



	width / nm	length / nm	fingers
T1, T2	1000	200	4
T3	500	80	4
T4, T5	500	80	4
T6, T7	500	120	4
T8, T9	500	120	4
T10, T11	500	120	2
T12, T13	500	80	4

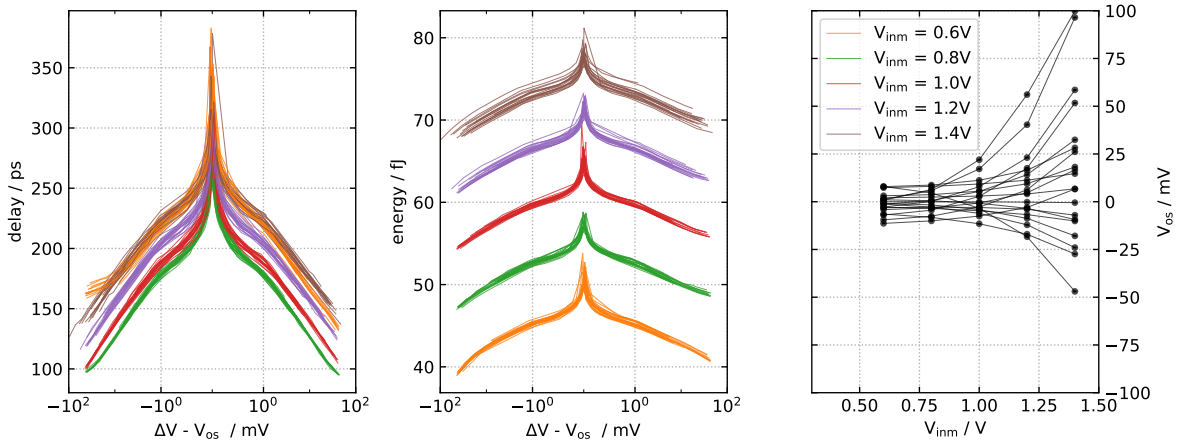
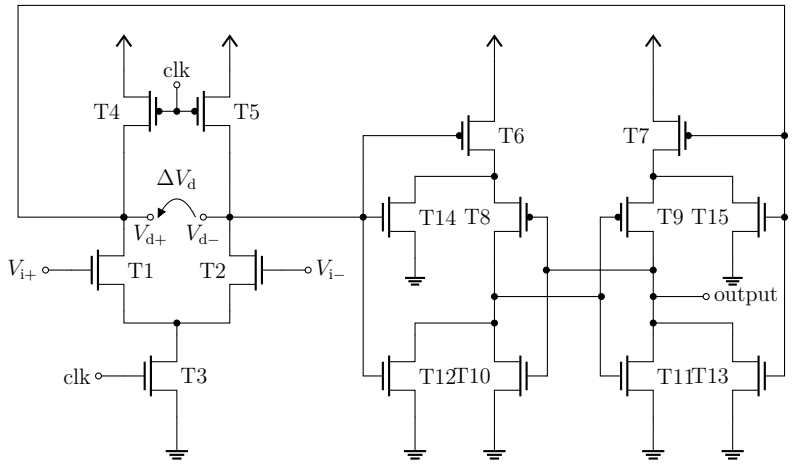


Figure 53: Improvement of the double-tail sense amplifier suggested by Van Elzakker et al. (2008): The pre-amplification stage has conductance based coupling to latch stage. The simulation results show 20 Monte-Carlo samples drawn from the transistor parameters above.



	width / nm	length / nm	fingers
T1, T2	1000	200	4
T3	500	80	4
T4, T5	500	80	4
T6, T7	500	80	2
T8, T9	500	120	4
T10, T11	500	120	2
T12, T13	500	120	4
T14, T15	500	80	1

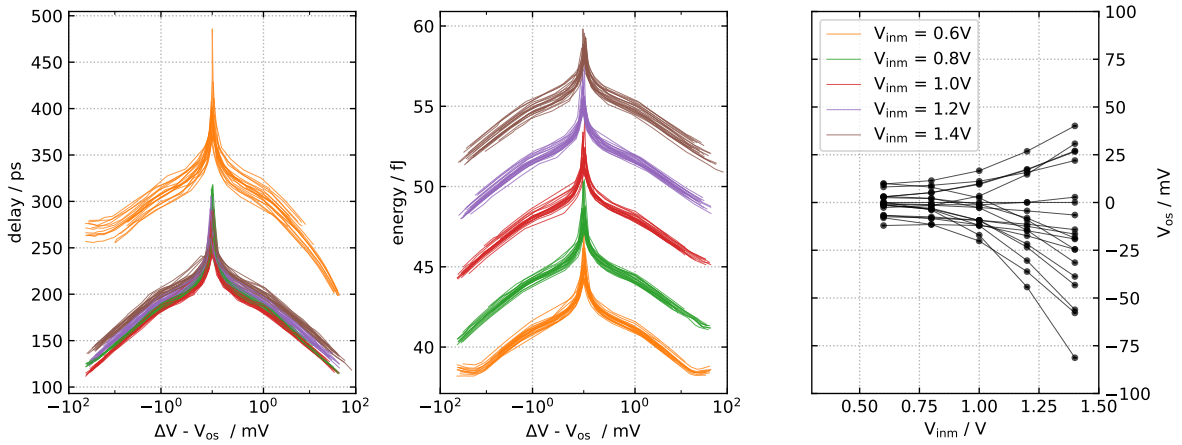


Figure 54: Improvement of the double-tail sense amplifier suggested by Miyahara et al. (2008): Instead of using the inverse trigger, the latch is fully controlled by the first stage. The simulation results show 20 Monte-Carlo samples drawn from the transistor parameters above.

Different encoding for the reference generator

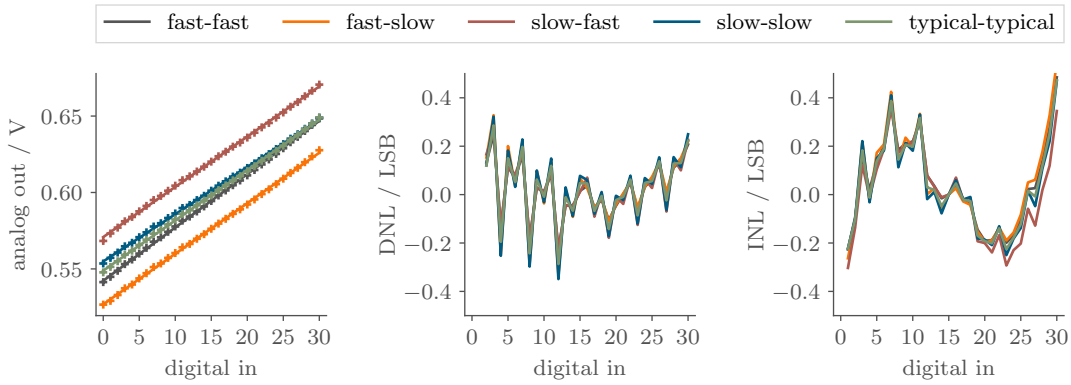


Figure 55: Capacitive reference DAC core with different decoding (maximum capacity in the center).

Even-odd effect in the reference generators

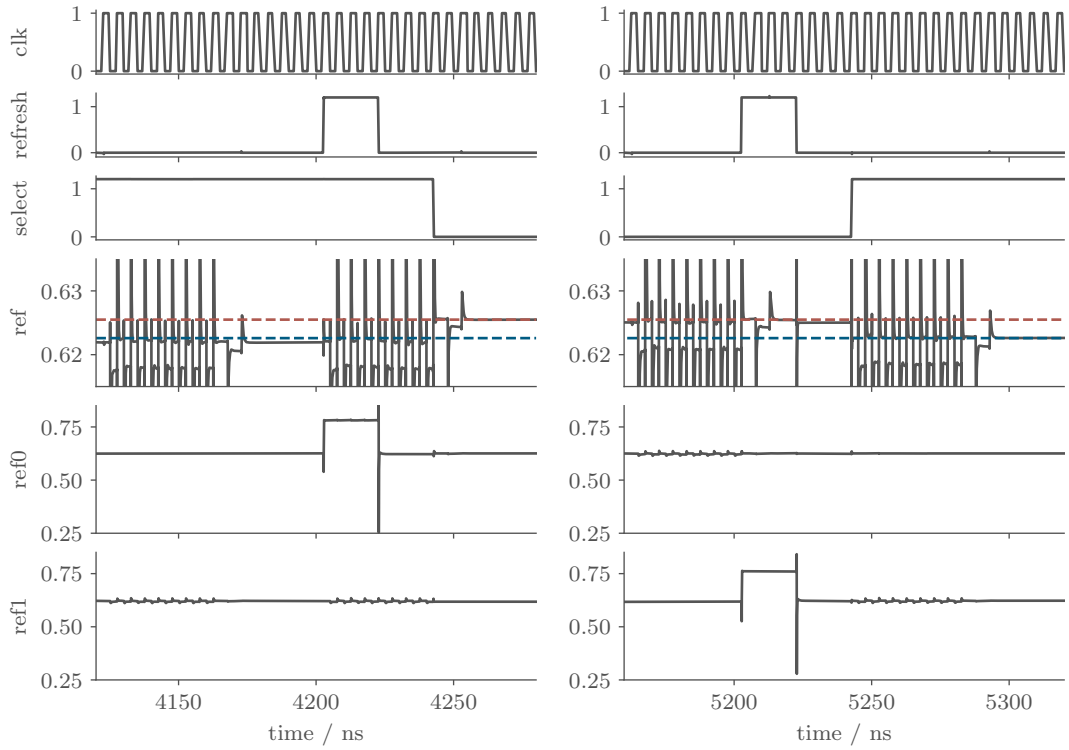


Figure 56: Illustration of the even-odd effect in the adjustable capacitive reference voltage generators, simulated based on layout-extractions. Both reference generators are configured to the same code. The red respectively blue dashed line mark the new refreshed voltages. It is clearly visible that the discharging effect plays a minor role in this situation.

Software used in this thesis

All circuit information and the simulation framework is available in the group internal software database. The simulations run in a singularity[10] image[11] on machines of the ASIC Labor of the Kirchhoff-Institut für Physik, Heidelberg.

Repository	Commit ID
`chip-teststand`	`1b1261fdd0843c1d310271f0a6f2b2af570d85d0`
`chip-sar-adc`	`N3298e89c7bb0b4a49b435878483d6c809a3641c4`

Table 10: Commit IDs for the related group internal repositories.

During the design process also proprietary software was used.

Product	Provider	Version
Design Compiler®	Synopsis™	R-2020.09
Cadence Design Suite	Cadence®	6.1.8 64bit
Spectre®	Cadence®	19.1.0 64bit
Xcelium®	Cadence®	19.03-s013 64bit

Table 11: Proprietary software used during this course of studies.

[10]The source code of SingularityCE can be found on github (`https://github.com/sylabs/singularity`, state: 14.12.2022)

[11]Location of the used image on the ASIC-machines: `/containers/stable/asic_2022-01-25_1.img`

Acknowledgment

I would like to thank

- Dr. Johannes Schemmel for all his support, ideas, technical discussions, and the opportunity to carry out this master thesis.
- Prof. Wolfram Pernice for being my second adviser.
- Milena Czierlinski for a good, successful, and trustful cooperation and a great friendship, both now during the development of the ADC and already during the whole master study.
- Sebastian Billaudelle for good suggestions and support in the analog domain.
- Andreas Grübel and Joscha Ilmberger for the technical support and good suggestions in the digital domain.
- Markus Dorn for providing the ASIC Labor infrastructure.
- Jakob Kaiser and Raphael Stock for the wonderful time we spent together in our office.
- Lennart Uecker, Sebastian Billaudelle, Moritz Sindram, Maike Clausen, Jakob Kaiser, Sarah Görlitz, Yannik Stradmann, Wolfgang Dauer, Ingo Stephensons, Anna-Theresa Arnold, Rouven Seibert, Lisa Maria Arnold, and Joscha Ilmberger for helping me with prove reading this thesis.
- the entire Electronic Vision(s) group.
- my family as well as my friends who supported me during the course of my studies. I would especially like to mention my parents Bettina and Wolfgang Dauer, as well as Anna-Theresa Arnold and Benedikt Kneißl.

The work carried out in this Master Thesis will be incorporated in future BrainScaleS systems. The BrainScaleS system received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 720270, 785907 and 945539 (Human Brain Project, HBP).

Statement of Originality (Erklärung)

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, December 19th, 2022