TU WIEN Informatics

Approximating Analog Waveforms by Adding Arbitrary Functions

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Technische Informatik

eingereicht von

Josef Salzmann
Matrikelnummer 11777724

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Ulrich Schmid
Mitwirkung: Univ.Ass. Dipl.-Ing. Dipl.-Ing. Jürgen Maier

Wien, 12. April 2021

Josef Salzmann	Ulrich Schmid

TU WIEN Informatics

Approximating Analog Waveforms by Adding Arbitrary Functions

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Computer Engineering

by

Josef Salzmann
Registration Number 11777724

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Ulrich Schmid
Assistance: Univ.Ass. Dipl.-Ing. Dipl.-Ing. Jürgen Maier

Vienna, 12th April, 2021

Josef Salzmann | Ulrich Schmid

Erklärung zur Verfassung der Arbeit

Josef Salzmann

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. April 2021

Josef Salzmann

Abstract

Determining the analog waveforms in electronic circuits is a computationally very expensive task. The reasons are manifold: On one hand the utilized models are very elaborate and thus hard to evaluate. On the other hand deriving results on a continuous time scale is inherently expensive.

In this thesis we thus aim to derive analog waveforms in a simplistic approximate fashion which reduces the required effort while keeping accuracy at a reasonable level. In detail pulses are used to fit arbitrary functions on in- and output. Based on the gathered information a mapping from in- to output parameters is developed which allows an analog simulation by passing on a very limited number of parameters. Results show very good agreement even for traces with multiple transitions where the error is in the range of a few percent.

Contents

CHAPTER 1

Introduction

In the process of developing digital circuits the analysis of their temporal behaviour is a crucial part. While these examinations are meant to ensure the correct behaviour of the resulting circuit, and therefore have to be performed thoroughly, they also should not consume too much time. Since time to market ought to be low, it is mandatory that the time spent on checking the temporal behavior is as low as possible, while maintaining reasonable accuracy.

Contemporary tools (e.g. SPICE [1], Mentor, etc.) for performing such analyses mainly use two approaches. The first method is to use highly sophisticated analog models such as BSIM [2, 3] to simulate circuits. This approach is rather time consuming and may yield results far beyond the required accuracy. The second method is to enhance digital simulators with characteristic delay values of the used components [4, 5]. Even though this approach uses far less resources regarding computational power and time, the obtained results are zero time transitions which provide no information about the shape of the signal. Yet other approaches include approximating the I-V Curve of a MOSFET [6] or approximating the analog behaviour by using simplified equivalent circuits [7, 8].

In this thesis we will focus on CMOS inverters [9]. The reason we will not consider other technologies is that virtually all semiconductor integrated circuits today are fabricated in CMOS technology [10]. The inverter implements the fundamental logic operation of negation (i.e. converting a binary input A to $\bar{A}$) and is a basic building block in the design of digital circuits [9].

Interpreting the voltage levels 0V as logic zero and V_{dd} as a logic one, we can see how the inverter works. If the input A is connected to 0V the transistor M2 pulls the output $\bar{A}$ to V_{dd}. M1 acts as high impedance and contributes nothing to the output (similar to an open switch). Connecting A to V_{dd} reverses the roles of both transistors and connects the output to 0V.

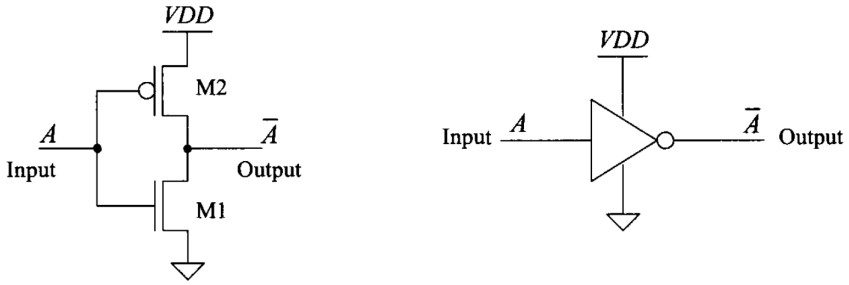

Figure 1.1: Schematics and Symbol of an Inverter [9]

Although the two extremes of V_{dd} and 0V are well defined, we did not consider the transition between those two extremes. Needless to say, that in a physical implementation we cannot expect the transitions to be infinitely fast. This is due to the fact that gates of the transistors, and to a lesser extent the wires connecting the logic gates, act as capacitors, which would need an infinite amount of current to be charged in zero time. These transient effects substantially affect the behaviour of a digital circuit and are responsible for the so called "delay" that a signal crossing a logic gate is exposed to. Therefore, in order to ensure the correct behaviour of a digital circuit, we have to take the region where both transistors conduct current and contribute to the output into account.

Analog simulators use highly sophisticated MOSFET models to investigate this behaviour. But these MOSFET models are very elaborate and hard to handle since they offer many parameters.

The aim of this thesis is to combine the low computational complexity of digital approaches with the high accuracy of analog methods. This will be done by approximating given analog simulations with an arbitrary function governed by a set of parameters, also called fitting-function. The obtained parameters are then used to generate a transfer function that establishes a connection between the parameters of input and output traces for the inverter. Finally this connection is used to predict the output behaviour based on a given input trace. Clearly, the choice of the fitting-function has a significant impact on the achievable accuracy. Therefore we investigate different possibilities and evaluate their applicability.

Chapter 2 gives an overview on curve fitting in general and how it is used in this thesis. Chapter 3 explains our approach of using transfer functions to predict the behaviour of an inverter, while also explaining the process of generating them. In Chapter 4 we will discuss the achieved results and some pitfalls we experienced. Finally Chapter 5 addresses some possible extensions.

CHAPTER 2

Curve Fitting

This chapter discuss how curve fitting is employed in this thesis. We want to approximate both input and output of an inverter by arbitrary functions governed by a set of parameters. It is therefore necessary to use a reliable method of determining values of these parameters such that optimal approximation is guaranteed. In the sequel we are going to introduce our base fitting function and explain the fitting process with an exemplary trace.

2.1 Curve Fitting Algorithms

Fitting a model to a data set is a fundamental problem in science. The goal is to minimize the deviation between prediction and reality, i.e., the minimize the fitting-error. There exists a variety of error metrics (e.g. absolute value, squared error, error in specified area) whereat least squares is a very common one [11]. Given a model function $F(t, \boldsymbol{x})$ depending on time t and a vector of n parameters $\boldsymbol{x}$ and a data set of m points (y_i, t_i) the metric results to

$$\sum_{i=1}^{m} \left[\frac{y(t_i) - F(t_i, \boldsymbol{x})}{\sigma_i} \right]^2 \tag{2.1}$$

which is aimed to be minimzed. The weighting vector σ can be used to give individual data points a different weight. In our case we will not use this possibility and set $\sigma_i = 1$.

2.2 Levenberg-Marquardt Algorithm

Different Algorithms exist for solving the problem stated above. In our case we need a fitting algorithm that fits an arbitrary function governed by an arbitrary number of parameters. In this thesis we will use the Levenberg-Marquardt algorithm, which was developed to solve nonlinear least-squares curve fitting problems, to fit our functions to the given data sets[12] [13]. The Levenberg-Marquardt algorithm is an established

method for solving nonlinear least-squares curve fitting problems [14]. It combines two minimization methods. The gradient descent method iteratively updates the parameters in the steepest-descent direction and therefore approaches the closest minimum. The Gauss-Newton method assumes the least squares function to be locally quadratic and finds the minimum. The Levenberg-Marquardt algorithm uses the first method when the parameters are far off the minimum and the second method for parameters that are close to the optimum. If several local optima exist, the algorithm will converge to the closest minimum and will possibly not find the global optimum [13].

2.3 Fitting Data

The reference data is obtained from SPICE-Simulations (see Fig. 2.1 for example). The waveforms are generated by investigating the input and output voltage of the fifth inverter of an inverter-chain, whereat the input of the inverter-chain is a series of very steep ramps, i.e. almost zero-time transitions.

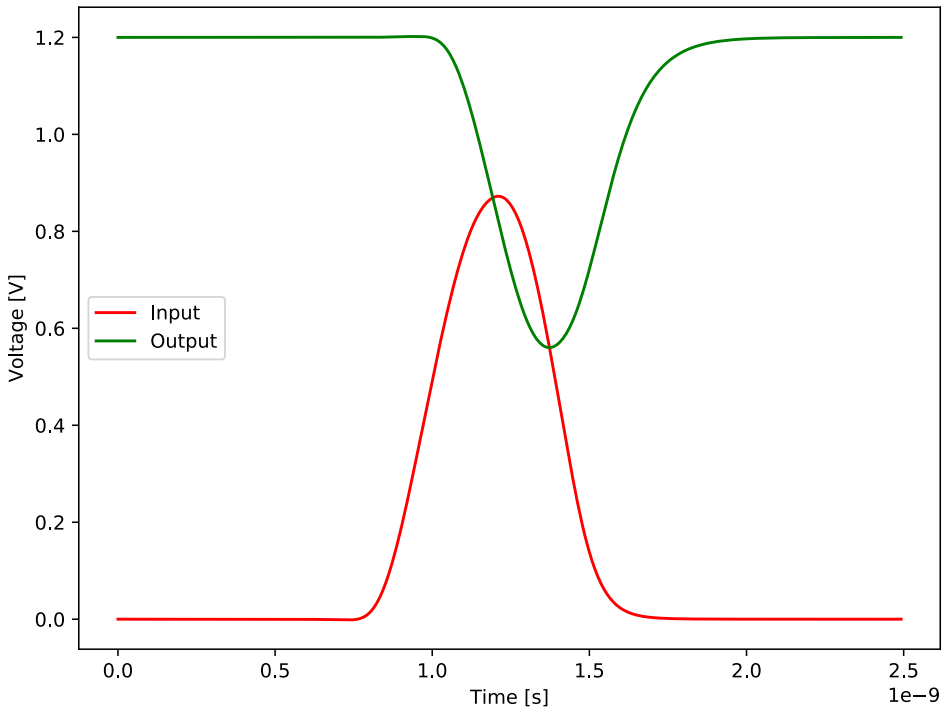

Figure 2.1: Example Waveforms of an Inverter

2.4 Fitting Function

One of the most important parts of the fitting process is the fitting function, also known as model function, and its parameters. This function is directly responsible for the fitting quality and therefore should be chosen carefully. In our case any function that is monotonically rising between 0 and 1 can be used as a model function. These requirements are the same as in the case of a cumulative distribution function. Concrete examples will be discussed in section 5.1. The function must provide at least two parameters, one for horizontal placement called "shift"-parameter and one that influences the speed of transition called "steepness"-parameter. A given switching waveform $F(t)$ can simply be enriched with the minimum required parameters by using the following scheme: $F(a(t-b))$, with a governing the steepness and b governing the shift. Further parameters may be supplied. Chapter 5.1 provides an example of how other characteristics of a function can be utilized to decrease the fitting-error.

2.5 Using tanh(x) as Fitting Function

Throughout this thesis we will use tanh(x), whose fitting process will be described in the sequel, as a fitting function unless specified otherwise. As a first step we will fit simple pulses. Fitting traces with an arbitrary amount of transitions will be explained in 5.2.

2.5.1 Function Definition

This function, also known as logistic function [15], is defined by:

$$F(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

and is a special case of a sigmoid function [16]. Before tanh(x) can be used as fitting function, we have to enrich it with the two minimum required parameters as described above. This leads us to:

$$F(t, a, b) = \frac{1}{1 + e^{-a(t-b)}} \tag{2.3}$$

For the sake of readability we will use t instead of x as parameter to indicate the time dependency.

Since we will deal with waveforms that have transition times in the range of nanoseconds we would end up with shift parameters in the range of 10^{-9} and steepness parameters in the range of 10^{9}. This is rather inconvenient for visualization and processing. In order to avoid this discrepancy, we will additionally add a multiplication factor of 10^{10} to t, which ensures that both parameters are in the same range for small traces:

$$F_s(t, a, b) = \frac{1}{1 + e^{-a(t \cdot 10^{10} - b)}} \tag{2.4}$$

Setting $a = 1$ and $b = 1$ now means that a transition with steepness of $\frac{0.1V_{dd}}{\text{ns}}$ is located at 0.1ns.

2.5.2 Fitting Process

Fitting a single transition with tanh(x) would not require a sophisticated algorithm, since the parameters could be calculated relatively easy by hand. The point of time where the transition passes $\frac{1}{2}V_{dd}$ would determine the shift parameters and the tangent at this point would allow us to calculate the steepness parameter.

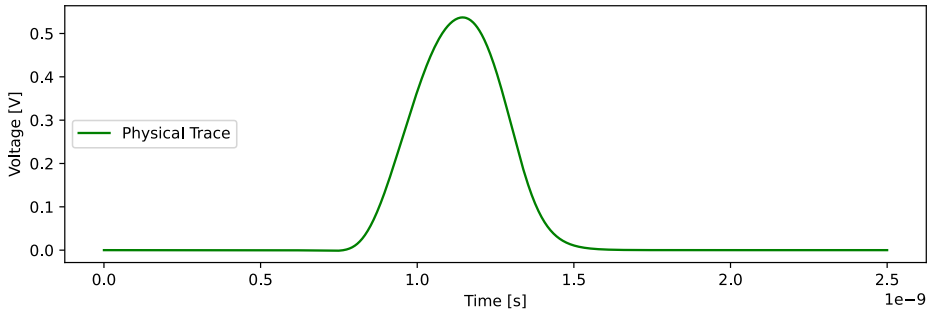

Figure 2.2: Example of a pulse in a circuit

Therefore we will demonstrate the fitting process with a pulse, which can be thought of as two transitions interfering with each other. The fitting function for a single pulse(see Fig. 2.2) must therefore have the following form:

$$F_{Pulse}(t, a_1, b_1, a_2, b_2) = V_{dd}(F_s(t, a_1, b_1) + F_s(t, a_2, b_2) - 1) \tag{2.5}$$

where:

a_1 = first transition steepness
b_1 = first transition shift
a_2 = second transition steepness
b_2 = second transition shift

Note that in this case we have to subtract 1 from the two transitions to ensure that the resulting curve is in between 0 and V_{dd}. (2.5) demonstrates one of the core ideas of the approach taken in this thesis. Approximating pulses, and traces with an arbitrary amount of transitions, by a sum of base-functions. We can think of a pulse as two transitions which happen to be very close together.

In addition to the fitting function we also have to supply (preferably tight) parameters bounds and initial guesses. These bounds can be determined by looking at the given pulse. The trajectory in Fig. 2.2 is located between $t = 0$ns and $t = 2.5$ns we can state the following conditions:

$$0 \leq b_1 \leq 25$$
$$0 \leq b_2 \leq 25$$

Regarding the steepness parameter bounds we know that a_1 must be positive since the pulse starts with a rising edge, it follows that a_2 must be negative. Estimating that the tangent at $\frac{1}{2}V_{dd}$ is far less than 10, we set up the following rather generous bounds:

$$0 \leq a_1 \leq 10$$
$$-10 \leq a_2 \leq 0$$

As initial guesses we can use the midpoint of the specified bounds. Supplying the fitting algorithm with this function, parameter bounds and initial guesses returns the values listed in table 2.1. The two single tanh(x)-curves and the resulting sum are depicted in Fig. 2.3.

Parameter	Fitting Result
a_1	1.177
b_1	10.595
a_2	-1.288
b_2	12.258

Table 2.1: Fitting Parameter Results

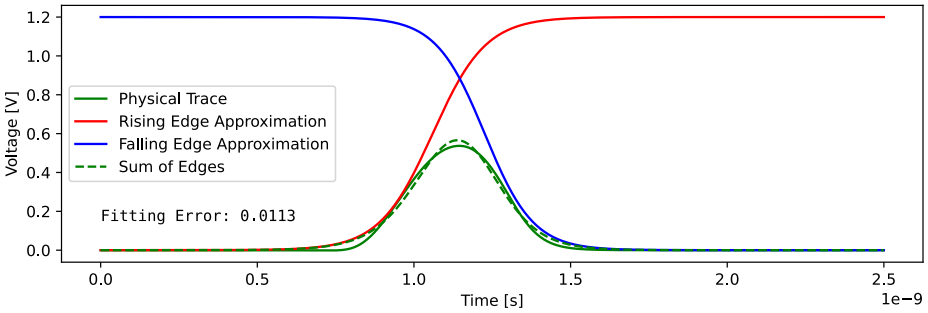

Figure 2.3: Pulse Fitting Results

a_1 and b_1 govern the rising transition while a_2 and b_2 govern the falling transition, the sum of both minus the compensation term of 1 approximates the pulse of fig. 2.2. The RMS-Error of the fitting is 1,13%.

Fig. 2.3 reveals that the steepness of a single transition does not necessarily resemble the maximum tangent of the physically occurring waveform, which is caused by an overlap of both transitions. In this case the falling transition (blue) already starts before the rising edge (red) is finished. This inference causes both transition to be steeper than the physical pulse.

CHAPTER 3

Transfer Functions

This chapter discusses a core part of this thesis. A set of transfer functions $F_T(\boldsymbol{x})$ is used to predict the parameters of a transition by using the parameters $\boldsymbol{x}$ of the current input transition and the last output transition. Since rising and falling transitions should be predicted, we need two sets of transfer functions $F_{T\uparrow}(\boldsymbol{x})$ and $F_{T\downarrow}(\boldsymbol{x})$. In this chapter we will show how to generate and use these transfer functions by the example of predicting a falling transition.

3.1 Input Output Transfer

We will first introduce the basic functionality of transferring a single transition from input to output in our model. This is executed by calculating the parameters for the output switching waveform based on the ones of the current input and the last output transition. This corresponds to the method used in [4] to predict the signal. It is crucial to take the last output transition into consideration when predicting the current output transition, since it contains a lot of information about how the singal will behave. Without this information the prediction quality, of for instance short pulses, would be significantly worse.

The minimum number of input parameters is depicted in Fig. 3.1. so_{n-1} represents the steepness of the previous output transition, si_n to the steepness of the current input transition and T to the difference in time between those two transitions. D resembles the delay between the current input transition and the corresponding output transition, so_n is the output transitions steepness. The maping between input and output parameters is established using $F_T(\boldsymbol{x})$:

$$(D, so_n) = F_T(si_n, so_{n-1}, T) \tag{3.1}$$

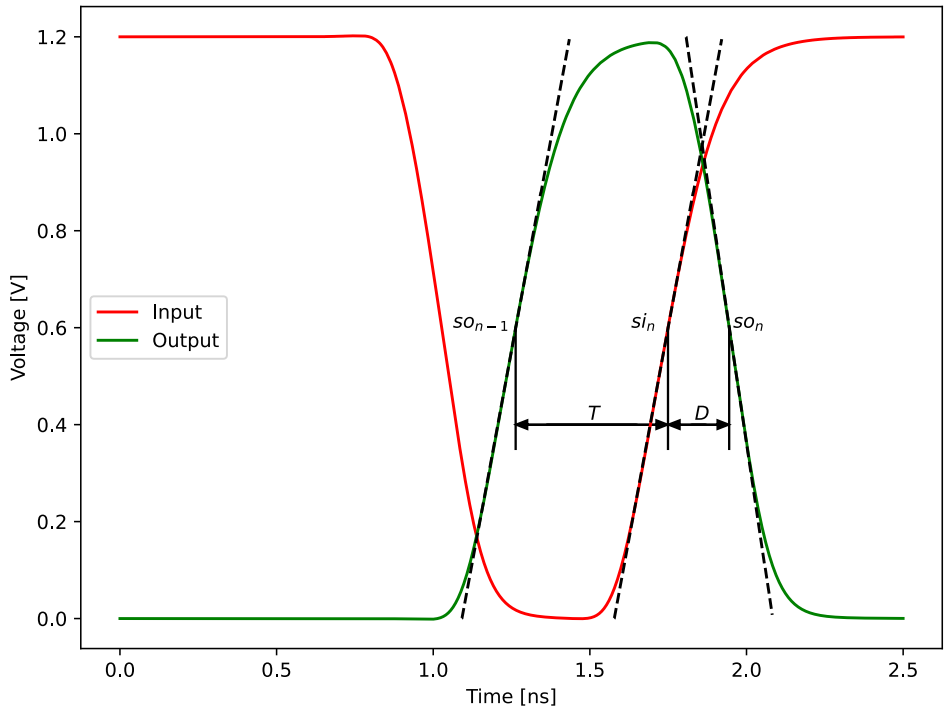

Figure 3.1: Input and Output Parameters Visualization

Transferring a whole trace of transitions means applying these calculations once for every transition. Algorithm 3.1 shows how a given list of input parameters is used to calculate the corresponding list of output parameters. Note that rising and falling switching waveforms have to be treated differently, hence the two different transfer functions "TransferRising" and "TransferFalling". Given that a single switching waveform is governed by n parameters, these functions take $2n-1$ input parameters and yield n output parameters. $2n-1$ because only the difference of the two shift parameters (T) is of interest, while all other parameters are used directly.

It is also worth noting that in the given implementation we take for granted that the parameters of the first output transition are known (see line 2). This is necessary since the transfer function must know T and so_{n-1} to calculate the next switching waveform. But for a real world application we cannot know the output parameters, since this is what should be calculated. To solve this problem we can simply assume the initial output transition at time $t = -\infty$ leading to $T = \infty$ and a common value for so_{n-1}, since high values of T mean that so_{n-1} does not affect the output transition.

Algorithm 3.1: Transfer-Trace

Input: List of input parameter tuples
Output: List of output parameter tuples

add (to_1, so_1) to Output;
Prev $\leftarrow (to_1, so_1)$;
for $(ti_n, si_n) \in$ *Input ascending in time* **do**
  $(to_{n-1}, so_{n-1}) \leftarrow$ Prev;
  T $\leftarrow ti_n - to_{n-1}$;
  if $si_n > 0$ **then**
    $(to_n, so_n) \leftarrow$ TransferRising(T, so_{n-1}, si_n);
  else
    $(to_n, so_n) \leftarrow$ TransferFalling(T, so_{n-1}, si_n);
  end
  $to_n \leftarrow to_n + ti_n$;
  Prev $\leftarrow (to_n, so_n)$;
  add (to_n, so_n) to Output;
end
return *Output;*

3.2 Transfer Function Structure

We will now formulate the generic structure of a transfer function. A transfer function for one output parameter will be defined as a polynomial of dimension $2n-1$, where n is the number of parameters of one switching waveform. The degree of this polynomial is variable and can be chosen by the user depending on the circumstances. High degree polynomials may be appropriate if the transfer functions were generated using big data sets, otherwise polynomials of small degeree should be used to avoid high interpolation errors. For example: given the minimum switching waveform implementation (i.e. 2 parameters: steepness and shift) and a degree of 2, the transfer function for the shift parameter would look like this:

$$\begin{aligned} F_{shift}(si_n, so_{n-1}, T) = a_0 + a_{1,1}si_n + a_{1,2}si_n^2 \\ + a_{2,1}so_{n-1} + a_{2,2}so_{n-1}^2 \\ + a_{3,1}T + a_{3,2}T^2 \end{aligned} \tag{3.2}$$

where:

a_0 = offset coefficient
$a_{i,j}$ = coefficient of jth power of ith parameter

Every transfer function has the same structure, they only differ by the coefficients $a_{i,j}$. A transition with n parameters will need n different transfer functions for each direction.

We can write (3.2) in a more generic way:

$$F(x_1, ..., x_{2n-1}) = a_0 + \sum_{i=1}^{2n-1} \sum_{j=1}^{d} a_{i,j} x_i^j \tag{3.3}$$

where x_i correspond to si_n, so_{n-1}, T. d is the degree of the polynomials.

3.3 Generating Transfer Functions

Generating a transfer function is equivalent to determining the $a_{i,j}$ for every parameter, which is once again achieved by using the Levenberg-Marquardt algorithm [12]. To generate the transfer functions we need a rich data set of traces covering every possible parameter combination occuring in a waveform that can be handled by this model. In order to achieve the desired variety of parameters we have to use waveforms as inputs that influence every parameter.

Since we use $n = 2$ parameters per transition, our transfer functions take $2n - 1 = 3$ parameters as input. We therefore need three differents ways to influence the input waveform of the inverter chain in such a way that every fitting parameter can be varied. In our approach we used three time intervals to influence all three fitting parameters, see Fig. 3.2.

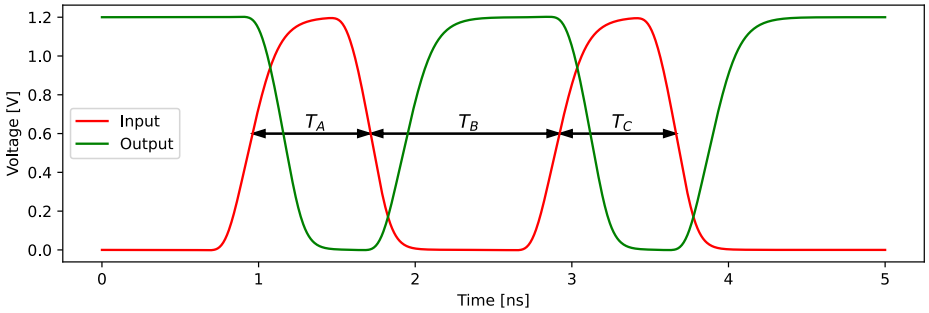

Figure 3.2: Transfer function data generation

The three time intervals T_A, T_B and T_C each govern one parameter of the fitting. We are only interested in the third input and output transition and the second output transition. The parameters of the second output transition and the parameters of the third input transition are used as input. The parameters of the third output transition are the output of the resulting transfer function. This means that T_B directly influences the time between the last output transition and the current input transition T. T_A indirectly influences the steepness of the second output transition and T_C indirectly influences the steepness of the third output transition.

Fig. 3.3 shows how T_A influences the steepness of the second output transition indirectly. Since the first and the second transition are closer together than in Fig. 3.2, we can expect a different steepness.

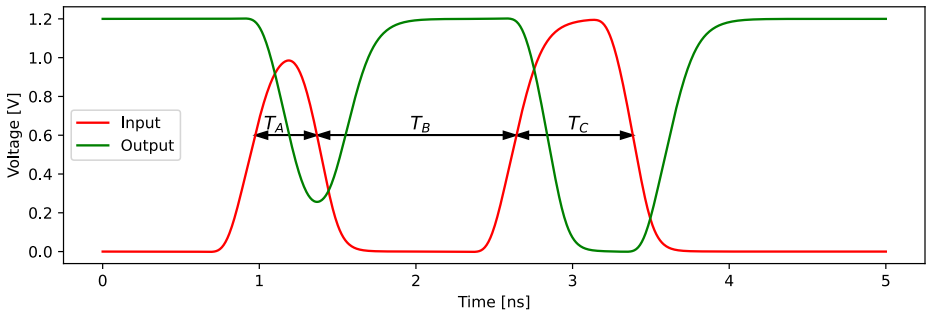

Figure 3.3: Input trace with small T_A

Assigning all three time intervals 10 different values 1000 traces are generated, which can then be used to construct the transfer function. In this thesis this setup was chosen.

Determining the minimum and maximum values of T_A, T_B and T_C such that the full potential of the model is utilized is a difficult task and is currently done by estimation. Especially the minimum value of T_B is an important parameter, since too small values will result in no output transition at all.

Determining the maximum values of T_A, T_B and T_C is of importance, since too big values will generate unnecessary test traces which might blow up the data set and may thus affect the quality of the resulting transfer function.

3.4 Tool Usage

The tool to generate the transfer functions and use them to predict the inverter's output behaviour is written in python and split up into several programs. This section will give an overview.

3.4.1 Fitting Function

The tool requires a definition of our fitting function, which is supplied to the tool by a python file with the following content:

1. Sigmoid Function: A function named "sigmoid" that takes the fitting functions parameters and time as input and returns the value of the fitting function. Care has to be taken regarding the scaling of the function, see 2.5. It is required that steepness is the first and shift the second parameter.

2. Parameter Names: The names of the single parameters as an array in the order of appearance in the sigmoid function as array. Starting with ”steepness” and ”shift”.
3. Rising Initial Guesses and Bounds: Initial Guesses and Bounds for the specified parameters in case the steepness is positive.
4. Falling Initial Guesses and Bounds: Initial Guesses and Bounds for the specified parameters in case the steepness is negative.

Technically the initial guesses of the shift parameter can be set to arbitrary numbers, since the tool will generate its own guesses (see 5.2 for further details).

3.4.2 fit_trace.py

This program is responsible for fitting a single SPICE-File. Its inputs are:

- V_{dd}
- Path to SPICE-File that should be fitted
- Path to fitting function
- Option for generating a picture of the fitting and the original trace

The output of the program are the fitting parameters of the trace in csv format. Since we usually want to fit a big number of traces there exists a program called fit_traces.py which takes a folder path instead of a dat-file path and calls fit_trace.py for each dat-file in the specified folder. It will also write every fitting result into a single file called fitting_parameters.txt, which will be used to generate the transfer functions. The runtime of these programs consists in large portions in evaluating the Levenberg-Marquardt Algorithm and therefore is highly dependent on the quality of the initial guesses and bounds provided by the user.

3.4.3 generate_transfer_functions.py

After all SPICE-Files have been fitted by fit_trace.py we can generated the transfer functions. generate_transfer_functions.py requires the following inputs:

- V_{dd}
- Path to folder of fitting_parameters.txt
- Path to fitting function
- Degree of polynomials

Once again the Levenberg-Marquardt Algorithm is used to generate the transfer functions based on the fitting values in fitting_parameters.txt. The output of this program is a transferFunctions.txt file containing the generated transfer functions.

3.4.4 transfer_trace.py

Finally this program can be used to predict an output trace based on the created transfer functions. The inputs are:

- V_{dd}
- Path to csv-file of the trace that should be predicted
- Path to fitting function
- Path to transfer function with a rising transition as input
- Path to transfer function with a falling transition as input
- Optionally providing the path to the dat-file of the trace to predict will generate a picture of the prediction and the original trace

The program implements algorithm 3.1 and runs in $\mathcal{O}(nap)$.
where:

n = Number of transitions of trace
a = Degree of polynomials
p = Number of parameters of fitting function

Since a and p are always low constants the algorithm runs in linear time with respects to the length of the input trace.

CHAPTER 4

Evaluation

This chapter discusses the results of the generated functions and their shortcomings. Additionally the behaviour of poor transfer function is showcased.

4.1 Reasonable Transfer Functions

We will examine the results of transfer functions that were generated using a sufficiently rich data set. Fig. 4.1 shows the trace generated by the transfer functions and Table 4.1 the corresponding trace parameters. Each cell of the table contains the two parameters governing the corresponding transition. The first value is the steepness, which must alternate with each transition. The second parameter is the shift relative to the starting point, which increases as the rows descend.

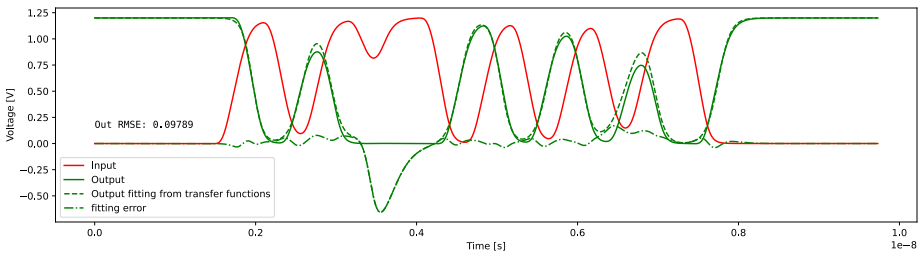

Figure 4.1: Example trace and its prediction by transfer functions

Fig. 4.1 illustrates a shortcoming of the model. We can observe that the fourth and fifth transitions of the input trace cause no corresponding transitions at the output. Therefore the second column contains blank spaces in these two rows. But the trace generated using the transfer functions contains values in these rows. It is already apparent from Fig.

Input Fitting	Output fitting	Transfer function results
1.311, 17.673	-1.659, 19.582	-1.611, 19.601
-1.447, 23.277	1.156, 25.921	1.183, 25.861
1.217, 27.884	-1.302, 29.171	-1.330, 29.299
-1.289, 33.940		0.676, 36.385
1.131, 35.063		-1.799, 34.269
-1.651, 42.980	1.291, 45.431	1.287, 45.362
1.286, 48.709	-1.525, 50.487	-1.540, 50.444
-1.478, 53.811	1.213, 56.344	1.228, 56.318
1.254, 58.912	-1.392, 60.516	-1.432, 60.503
-1.312, 63.798	1.113, 66.541	0.827, 66.151
1.182, 68.095	-1.283, 69.154	-1.430, 69.539
-1.632, 75.033	1.299, 77.465	1.294, 77.477

Table 4.1: Fitting Parameters and Transfer Function Results

4.1 that these two transitions cannot be correct, since the trace undershoots to less than -0.5V. Inspecting the generated transitions explains this behaviour. The shift parameter of the fifth transition is smaller than the shift parameter of the fourth transition. This means that the third transition, which is a falling transition, is followed by yet another falling transition, which violates the principle that a transition is followed by transition of alternating direction. This cancellation behaviour cannot be captured by the transfer function and must be implemented separately as done in [17].

In a similar way we have to take care of too high values of T. Since the T value essentially tells us how much influence the last transition has on the current transition, we can treat values above a certain threshold T_{max} all the same.

$$\forall T > T_{max} : T = T_{max} \tag{4.1}$$

This is because at a certain threshold the effect of the last transition on the current transition vanishes. In this thesis we used an estimated value for T_{max}. Future work may investigate exact threshold values.

4.2 Poor Transfer Functions

This section discusses the effects of transfer functions which were generated using data sets being either too small or not covering the whole range of parameter combinations.

A transfer function is a function mapping a subset of $\mathbb{R}^{2n-1}$ to $\mathbb{R}$. If the function encounters a parameter combination outside the subset of $\mathbb{R}^{2n-1}$ it will generate results with a non negligible extrapolation error. This error propagates to the next transition, since the steepness of the calculated output transition is used as an input parameter when calculating the parameters for the next transition. From transition to transition

the error accumulates and eventually diverges. This divergence is visually illustrated in Fig. 4.2.

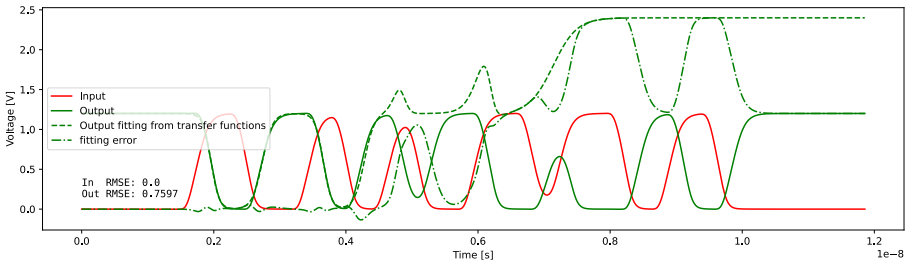

Figure 4.2: Waveform generated by poor transfer function

Output fitting	Reasonable transfer function	Poor transfer function
-1.650, 19.580	-1.650, 19.580	-1.650, 19.580
1.296, 27.370	1.295, 27.400	1.298, 27.355
-1.651, 36.573	-1.656, 36.576	-1.624, 36.570
1.301, 42.503	1.288, 42.467	1.305, 42.786
-1.406, 48.729	-1.395, 48.728	-1.821, 48.614
1.164, 52.993	1.174, 53.082	1.451, 48.025
-1.645, 61.672	-1.638, 61.698	-2.007, 61.868
1.144, 71.198	1.121, 71.205	0.882, 60.368
-1.267, 73.389	-1.268, 73.499	0.559, 71.373
1.303, 84.562	1.302, 84.544	11.487, 463.791
-1.622, 91.105	-1.604, 91.125	9.881e+04, -8.550e+04
1.296, 98.727	1.295, 98.754	1.328e+15, 4.652e+16

Table 4.2: Parameters obtained from different transfer functions.

Table 4.2 lists the transition parameters of the output waveform. The first value of each cell is the steepness of the respective transition, while the second value is the transition shift. The first column are the parameters directly obtained from fitting the waveform with the Levenberg-Marquardt algorithm. The second column is filled with parameters generated using a reasonable transfer function. Finally the third column lists the parameters obtained from a poor transfer function.

We can observe that the parameters of the first few transitions are nearly identical in all columns. But in the fifth row it can be seen that the steepness parameter generated by the poor transfer function is approximately -1.8 while it is approximately -1.4 in the other columns. Consequently the achieved analog waveforms deviates more and more until, eventually, forbidden traces are predicted, since multiple rising edges in succession occur.

This illustrates that, once the values used for fittings are exceeded, the parameters diverge. In contrast to the poor transfer function with diverging parameters due to extrapolation errors, the reasonable fitting does not show any signs of divergence at all. The errors observed in the second column are in the range of a few percent and are effectively interpolation errors.

Similar problems may arise when the degree of the generated transfer functions is too high. High order polynomials may introduce such high interpolation errors that they face the same problems as transfer functions generated with a poor data set. This may be counteracted with a larger data set or lowering the degree. In our thesis we mainly used polynomials of degree three and data sets with a few thousand samples. Future work is devote to find more optimal constellations of data set size and transfer function degree.

CHAPTER 5

Extension

This chapter discusses the shortcomings of the used fitting function and tries to mitigate the problems by introducing a new parameter. The introduction of this new parameter will be demonstrated and its implications discussed. We will show how this significantly lowers the fitting error, but introduces other problems. The second part will present how traces of arbitrary length are currently fitted and what problems this solution encounters.

5.1 Shortcomings of tanh(x) as Fitting Function

Fig. 5.1 shows the main sources of error for a fitting that uses tanh(x). The shape of the curve cannot be influenced in these two points if only steepness and shift parameters are used, since the inherent shape of tanh(x) is not affected by them. If a parameter would be available to change the shape of the fitting function in these regions we could significantly lower the fitting error. This parameter will effectively control the curvature of the fitting function.

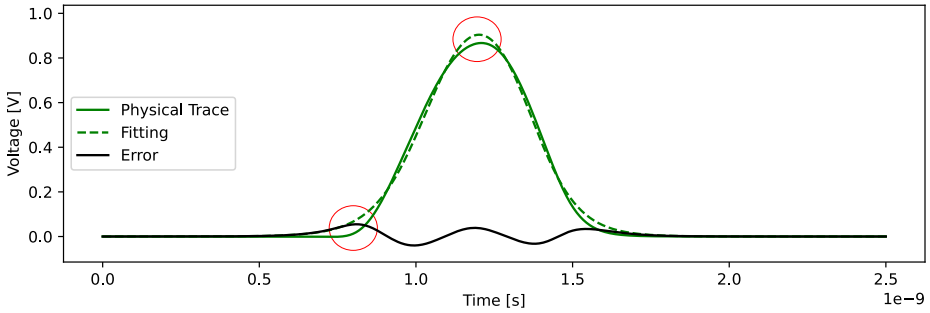

Figure 5.1: Main sources of error of tanh(x) fitting function

5.1.1 Notion of curvature

Mathematically speaking curvature denotes the second derivative and can be intuitively thought of as the amount by which a curve deviates from being a straight line.

The parameter c we are going to introduce determines the concentration of curvature along the trace. With $c = 0$ the fitting fuction is not affected at all while for $c \to \infty$ it approaches $S(x)$ with

$$S(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{5.1}$$

S(x) basically consists of three straight segments. The second derivative of S(x) is zero at all points except $x = -1$ and $x = 1$. This can be interpreted as concentrating the curvature at these locations. Fig. 5.2 illustrates different mathematical functions and S(x).

Since the concentration of curvature at $c = 0$ depends on the shape of the given function, we will use a function that has a low inherent curvature concentration. At first glance $\frac{x}{1+|x|}$ would be a promising candidate since it converges to its minimum and maximum values the slowest, but since its higher order derivatives cannot be evaluated at $x = 0$ we cannot use it as base function. $\arctan(x)$ is better suited and will be extended by the parameter c in the sequel.

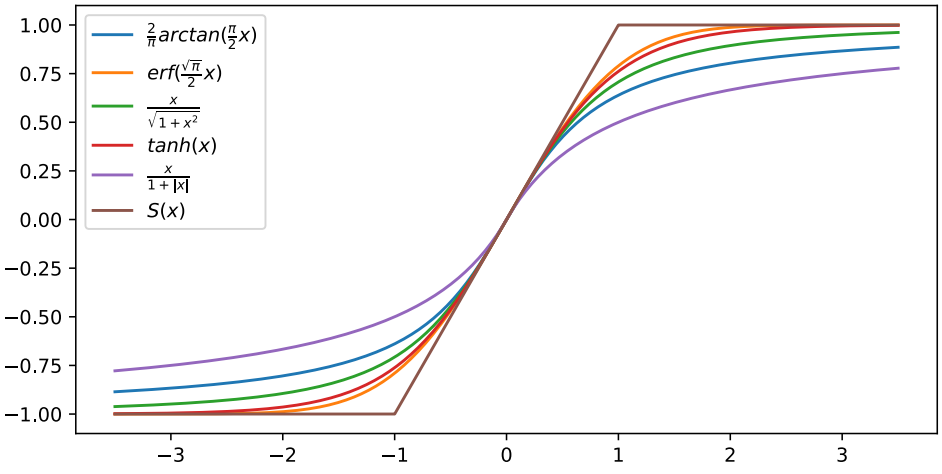

Figure 5.2: Different Sigmoids

5.1.2 Defining an inner function for non negative integers

In this section we will search for a function $f(x, c)$ such that $\arctan(f(x, c))$ implements our notion of curvature given by the previous section for $c \in \mathbb{N}$. We will later show how we can define the inner function such that the fitting algorithm achieves good results. Since we chose $\arctan(x)$ as our base function f has to satisfy:

$$\arctan(f(x, 0)) = \arctan(x) \tag{5.2}$$

$$\arctan(f(x, \infty)) = s(x) = \frac{\pi}{2} S\left(\frac{2}{\pi} x\right) \tag{5.3}$$

For $c = 0$ it follows that:

$$f(x, 0) = x \tag{5.4}$$

For $c = \infty$ we can use the following idea: Every bijective function $f(x)$ satisfies the condition $x = f(f^{-1}(x))$. Consequently setting $f(x, \infty)$ to $\arctan(x)^{-1}$, i.e. $\tan(x)$ (5.5), $\arctan(f(x, \infty))$ will evaluate to x between -1 and 1. Therefore:

$$f(x, \infty) = \tan(x) \tag{5.5}$$

A way of satisfying both equations 5.4 and 5.5 is to approximate $\tan(x)$ with a Taylor series at $x = 0$ and using c as degree of approximation. Therefore $f_I(x, c)$ can be expressed as

$$f_I(x, c) = \sum_{n=0}^{c+1} \frac{\tan^{(n)}(0)}{n!} x^n \tag{5.6}$$

Since $\tan(x)$ is an odd function, no even parts will be part of the Taylor series. Therefore $f_I(x, c)$ only needs to contain the odd derivatives:

$$f_I(x, c) = \sum_{n=0}^{c+1} a_n x^{2n+1} \tag{5.7}$$

with a_n being defined as

$$a_n = \frac{\tan^{(2n+1)}(0)}{(2n+1)!} \tag{5.8}$$

This uniquely determines $f_I(x, c)$ for every non negative integer. Fig. 5.3 illustrates a few instances of $\arctan(f_I(x, c))$ for different values of c. It can be seen that for increasing values of c the curvature gets concentrated to $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Fig. 5.4 illustrates this behaviour. With increasing c, the second derivative approaches the two delta functions $\delta(\frac{-\pi}{2})$ and $-\delta(\frac{\pi}{2})$. This can also be thought of as compressing $\arctan(x)$ such that it approaches $S(x)$.

Note that $\arctan(\tan(x))$ would result in a periodic function ($y = x$ for $\frac{-\pi}{2} < x < \frac{\pi}{2}$, $y = x - \pi$ for $\frac{\pi}{2} < x < \frac{3\pi}{2}$,...) because of the poles of $\tan(x)$ at $(2k + 1)\frac{\pi}{2}, k \in \mathbb{Z}$. This is not the case for $\arctan(f_I(x, c))$, since $f_I(x, c)$ is a polynomial without such periodic poles no matter of c.

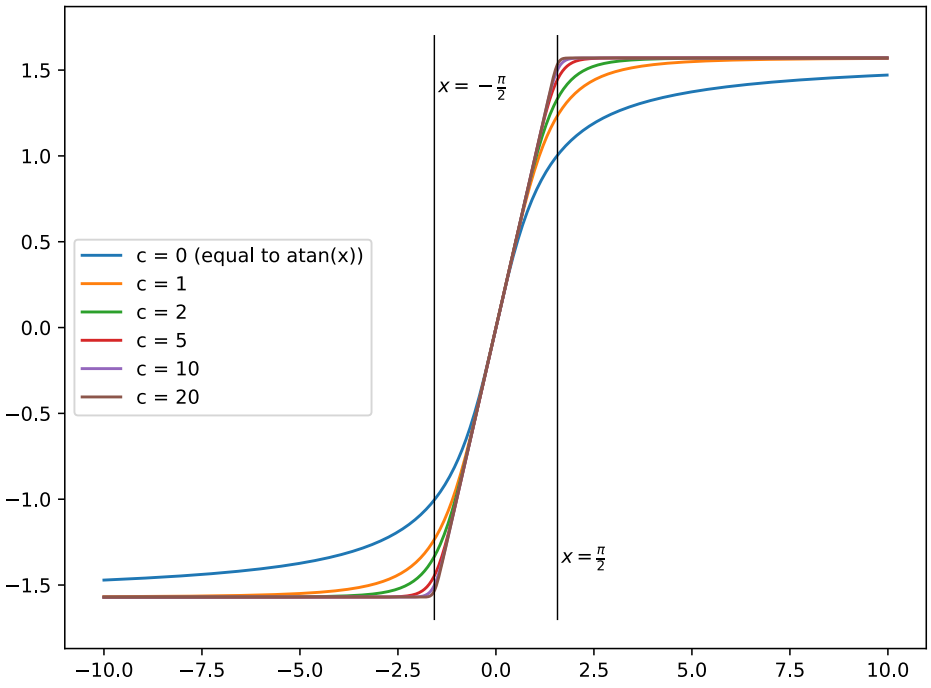

Figure 5.3: $\arctan(f_I(x, c))$ with c being a non negative integer

5.1.3 Defining an inner function for non negative real numbers

We already know how our fitting function behaves for $c \in \mathbb{N}$. To be able to supply the defined curvature parameter to a fitting algorithm we need to define the behaviour of our function for c in an interval that is a subset of the real numbers.

For this purpose we can use the following idea:

For every real number c we get: $\lfloor c \rfloor \leq c \leq \lceil c \rceil$. We can now identify two cases: If $c = \lfloor c \rfloor$ we can use the function for integers presented in the previous section. If $c > \lfloor c \rfloor$ we will use the following definition of $f(x, c)$:

$$f_R(x, c) = f_I(x, \lfloor c \rfloor) + (c - \lfloor c \rfloor) \cdot a_{\lceil c \rceil + 1} \cdot x^{\lceil c \rceil + 1} \tag{5.9}$$

which is equal to:

$$f_R(x, c) = \sum_{n=0}^{\infty} a_n \cdot x^n \cdot m(c + 1 - n) \tag{5.10}$$

where $m(x)$ ensures that the first c elements are scaled with one, the $c + 1$th element is

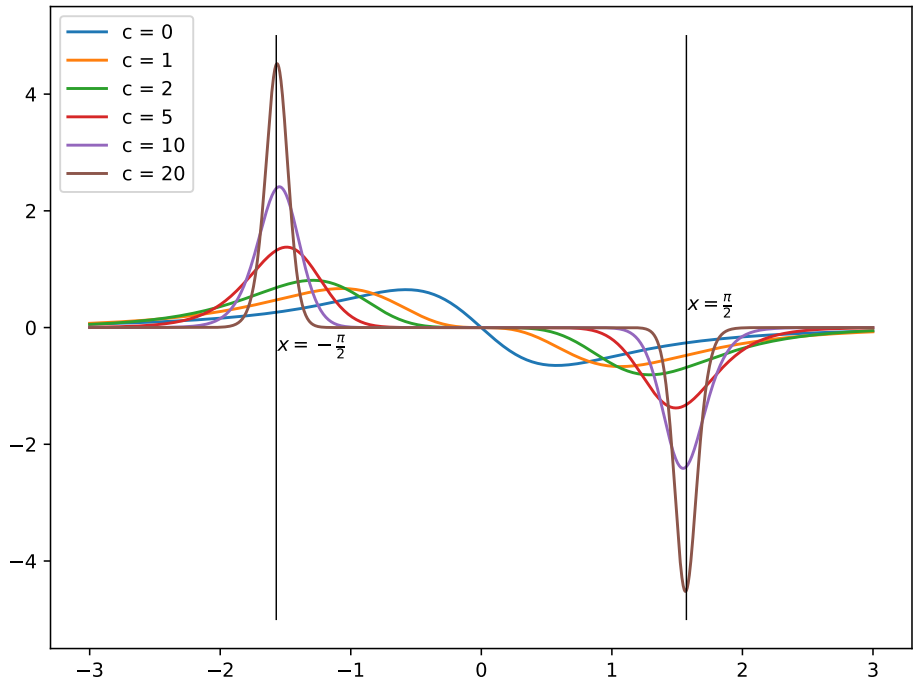

Figure 5.4: second derivative of $\arctan(f_I(x, c))$

scaled with $(c - \lfloor c \rfloor)$ and all elements above this index are multiplied with zero.

$$m(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{5.11}$$

Fig. 5.5 depcits the resulting waveform for c between 0 and 1. Since the fitting function is defined for real numbers, it can be used to approximate real trajectories. Nevertheless, at the moment c influences the curvature at the beginning and end of the transition simultaneously. Another degree of freedom would be possible by allowing separate values, which is presented in the sequel.

5.1.4 Separate Parameters

In order to supply the fitting algorithm one curvature parameter for each side, we will use two different functions and add them up such that the resulting function is governed by two curvature parameters. One function will define the curvature for $x > 0$ and the

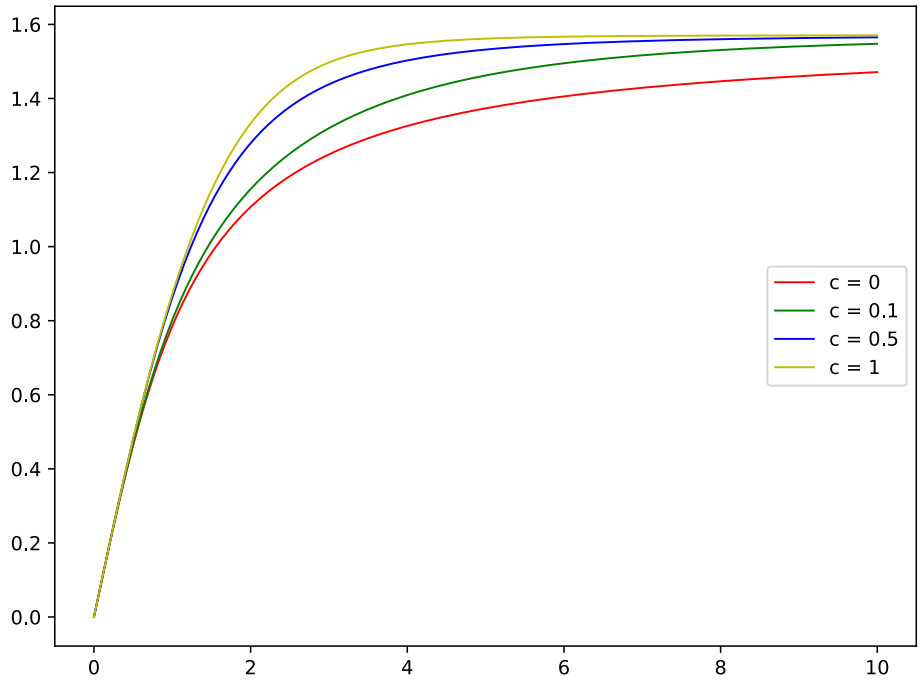

Figure 5.5: $\arctan(f(x, c))$ with c being a real number

other one for $x < 0$.

$$f_{\text{Fitting}}(x, c, d) = \frac{\arctan(f_R(x, c))}{1 + e^{1000x}} + \frac{\arctan(f_R(x, d))}{1 + e^{-1000x}} \tag{5.12}$$

The terms $\frac{1}{1+e^{1000x}}$ and $\frac{1}{1+e^{-1000x}}$ can be considered "smooth" Heaviside-functions. Their purpose is to introduce a continuous handover. Fig. 5.6 illustrates that in this case the curvatures of both sides can be influenced independently.

5.1.5 Using $f_{Fitting}$ as fitting function

After equipping $f_{Fitting}$ with a steepness and shift parameter, we can finally use it to fit traces. Fig. 5.7 shows an example trace fitted by $f_{Fitting}$. Compared to $\tanh(x)$ a decrease in RMS-Error by a factor of more than five is achieved.

Unfortunately $f_{Fitting}$ is no viable option to $\tanh(x)$ when it comes to generating transfer functions. The two curvature parameters on each side of $f_{Fitting}$ lead to "overdetermination". Due to the fact that the fitting algorithm has to consider 4 dimensions per switching waveform, and the curvature parameters of adjacent switching waveforms may

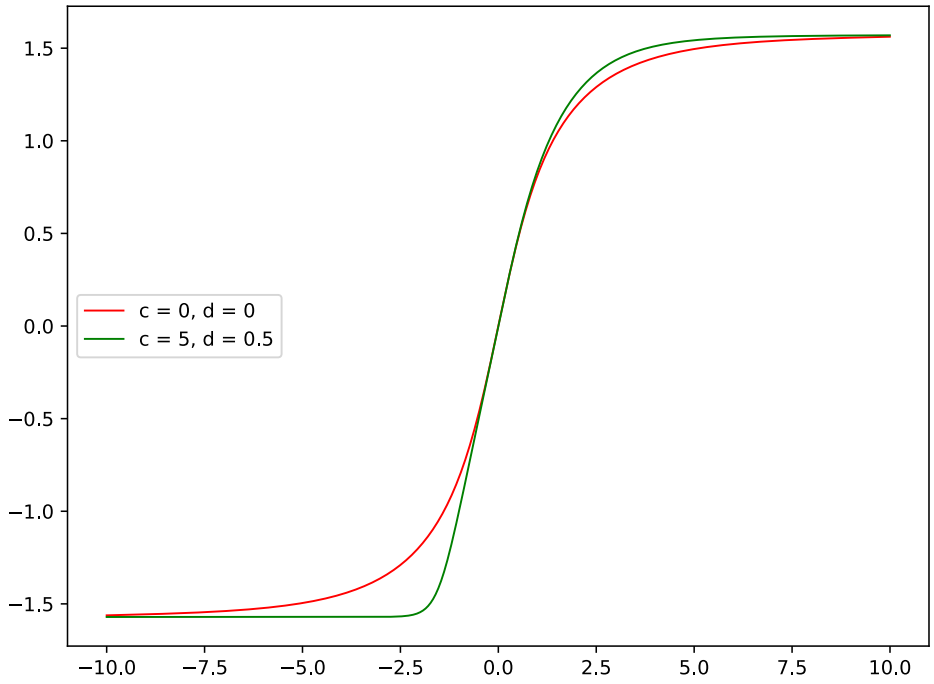

Figure 5.6: Resulting sigmoid with its two independent curvature parameters

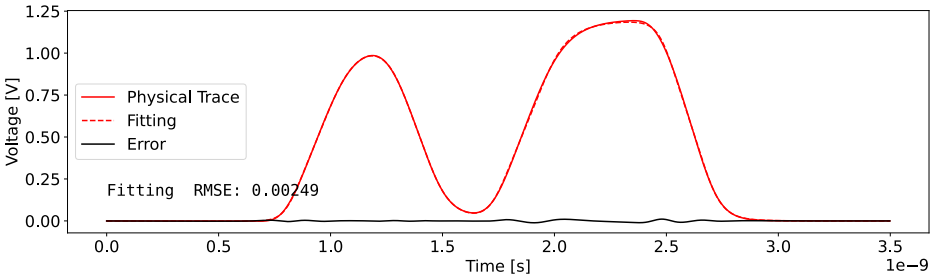

Figure 5.7: Fitting with curvature parameters

affect each other, several local minima are created. This leads to results that may not be the global optimum, which results in erratic behaviour of the fitting parameters.

Fig. 5.8 illustrates the left curvature parameter and the steepness parameter versus T_B (see Fig. for reference 3.2) of the third output transition. T_A and T_C are set to fixed values. We can observe the spikes in the curvature trajectory. The big spike between 0.8ns and 0.9ns even affects the steepness parameter. The presence of several local

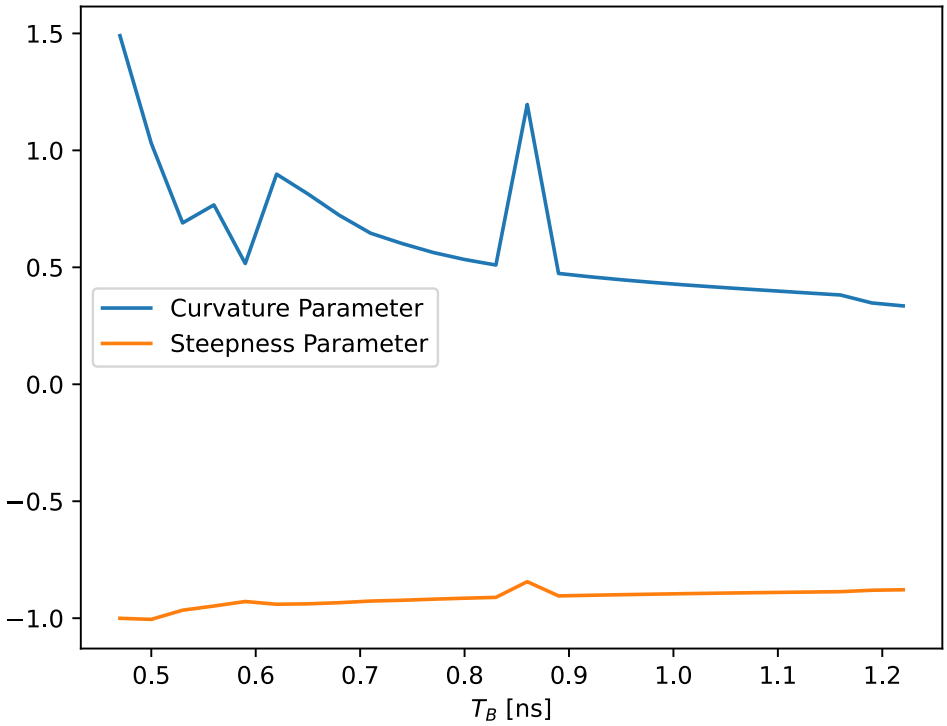

Figure 5.8: Erratic behaviour of fitting parameters

optima leads to unpredictable fitting parameters, which would lead to transfer functions generating unpredictable results. This means that curvature parameters implemented as above are not suitable to be used as input for transfer functions. Although the fitting error is five times less compared to fittings without curvature parameters, it cannot be used to predict the behaviour of the inverter. Future work may investigate different approaches of integrating curvature parameters into the work flow.

5.2 Fitting Traces of Arbitrary Length

In Chapter 2 we showed how to fit a single pulse consisting of two transitions. We had to manually come up with parameter bounds and initial guesses. But this is not feasible when dealing with traces of arbitrary length. Fig. 5.9 gives an example of such a trace. If we want to automate the fitting process for traces of arbitrary length, the tool has to determine how many switching waveforms are needed to fit the trace and provide reasonable initial guesses.

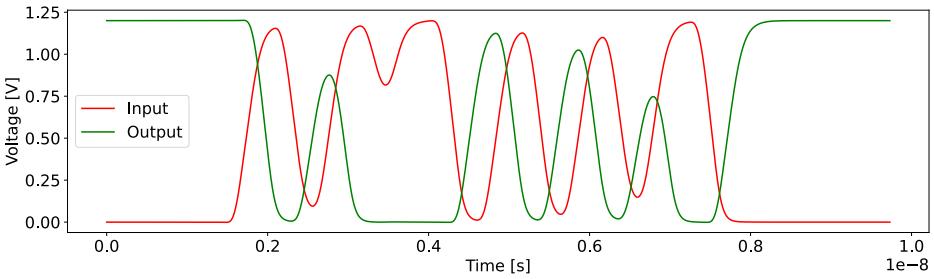

Figure 5.9: Example Trace

5.2.1 Representing arbitrary long traces as functions

In the case of a pulse we know how many transitions in which order are needed and how many compensation terms are necessary. The fitting function for a trace with m transitions is given by the following equation:

$$F_{\text{Trace}}(t, x_1, ..., x_{n \cdot m}) = V_{dd} \left(\sum_{i=0}^{m-1} F_s(t, x_{n \cdot i+1}, x_{n \cdot i+2}, ..., x_{n \cdot i+n}) \; - F_{Comp}(x_1, m) \right) \tag{5.13}$$

The parameters $x_1, ..., x_n$ describe the first switching waveform, the parameters $x_{n+1}, ..., x_{2n}$ describe the second, and so on. F_{Comp} ensures that the resulting trace is in between 0V and V_{dd} and is given by:

$$F_{Comp}(x_1, m) = \left\lfloor \frac{m}{2} \right\rfloor - C(x_1, m) \tag{5.14}$$

where

$$C(x_1, m) = \begin{cases} 1 & \text{if } \; x_1 < 0 \text{ and } m\%2 = 0 \\ 0 & \text{otherwise} \end{cases}$$

x_1 is the steepness parameter of the first transition, which tells us if the trace starts with a rising or falling transition. This is an important detail when computing the number of compensation terms.

5.2.2 Determining initial guesses and bounds

It is possible to supply the program with initial guesses and bounds for every parameter except the shift parameter. This is why we have to implement a method to determine reasonable initial guesses and bounds for the shift parameter.

We will do this by identifying the turning points (i.e. the second derivative equals zero) of the given trace. This can be done by calculating the first derivative and searching

for sections in which the first derivative is constant and non zero. A constant positive section implies a rising transition at this point, while a constant negative section implies a falling transition. We also know that after a rising transition we need to search for a falling transition and vice versa. The algorithm we came up with can be parameterized with three different values: section size, amount of allowed change within the section and minimum absolute value. We choose values so that the turning points of our given traces are found relatively reliably, the three parameters would have to be adjusted accordingly. Turning points of traces with different speed or voltage may not be found reliably. An example is displayed in Fig. 5.10, whereat the vertical black lines indicate the turning points.

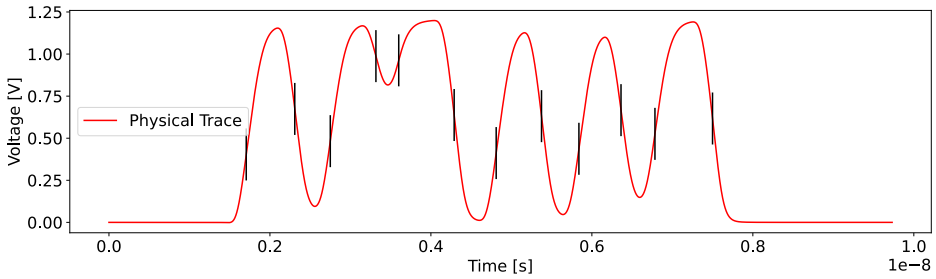

Figure 5.10: Found turning points of trace

The turning points can be used as initial guesses for the shift parameters, whereat bounds can be determined by the mean of two adjacent turning points. As already mentioned, bounds and initial guesses of the steepness parameters have to be specified manually. It is only necessary to supply them once for a rising transition and a falling one. If a transition would be governed by more parameters, their initial guesses and bounds would also have to be specified for both edges.

Transition	Initial Guess	Lower Bound	Upper Bound
falling	1	0	10
rising	-1	-10	0

Table 5.1: Steepness Parameter Initial Guesses and Bounds

This leads us to the initial guesses and parameter bounds listed in Table 5.2.

Parameters with an odd index are steepness parameters, while even indices are shift parameters. Bounds and initial guesses of steepness parameters in Table 5.2 are taken accordingly from Table 5.1 while those for the shift parameters are provided by the turning points. Fig. 5.11 depicts the initial guess as trace compared with the trace to be fitted. Finally the bounds and guesses can be handed over to the fitting algorithm.

Table 5.3 compares the initial guesses to the fitting results of all parameters. We can observe that the shift guesses are very close to the obtained fitting values. Fig. 5.12

Parameter	Initial Guess	Lower Bound	Upper Bound
x_1	1	0	10
x_3	-1	-10	0
x_5	1	0	10
x_7	-1	-10	0
x_9	1	0	10
x_{11}	-1	-10	0
x_{13}	1	0	10
x_{15}	-1	-10	0
x_{17}	1	0	10
x_{19}	-1	-10	0
x_{21}	1	0	10
x_{23}	-1	-10	0
x_2	17.057	0	20.072
x_4	23.087	20.072	25.293
x_6	27.500	25.293	30.335
x_8	33.169	30.335	34.586
x_{10}	36.003	34.586	39.447
x_{12}	42.891	39.447	45.501
x_{14}	48.112	45.501	50.928
x_{16}	53.743	50.928	56.066
x_{18}	58.389	56.066	61.005
x_{20}	63.620	61.005	65.728
x_{22}	67.837	65.728	71.414
x_{24}	74.991	71.414	100

Table 5.2: Trace Parameter Initial Guesses and Bounds

depicts the fitted curve. As in the case of a single pulse the error is in the range of less than 2%.

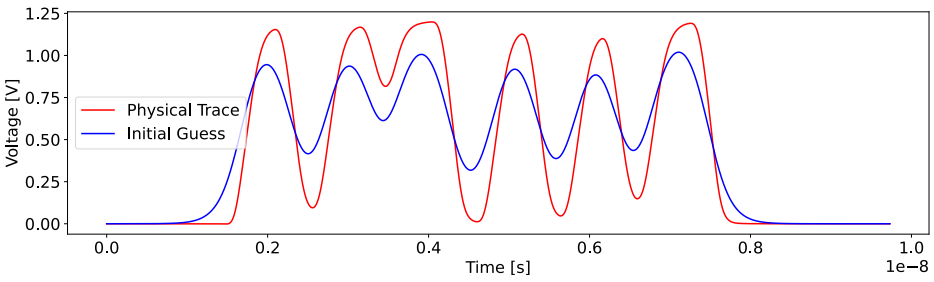

Figure 5.11: Initial Guess of Trace Fitting

Parameter	Initial Guess	Fitting Result
x_1	1	1.311
x_3	-1	-1.447
x_5	1	1.216
x_7	-1	-1.289
x_9	1	1.130
x_{11}	-1	-1.651
x_{13}	1	1.285
x_{15}	-1	-1.478
x_{17}	1	1.254
x_{19}	-1	-1.312
x_{21}	1	1.182
x_{23}	-1	-1.632
x_2	17.057	17.673
x_4	23.087	23.276
x_6	27.500	27.884
x_8	33.169	33.939
x_{10}	36.003	35.062
x_{12}	42.891	42.980
x_{14}	48.112	48.709
x_{16}	53.743	53.811
x_{18}	58.389	58.912
x_{20}	63.620	63.798
x_{22}	67.837	68.094
x_{24}	74.991	75.032

Table 5.3: Trace Parameter Initial Guesses and Fitting Results

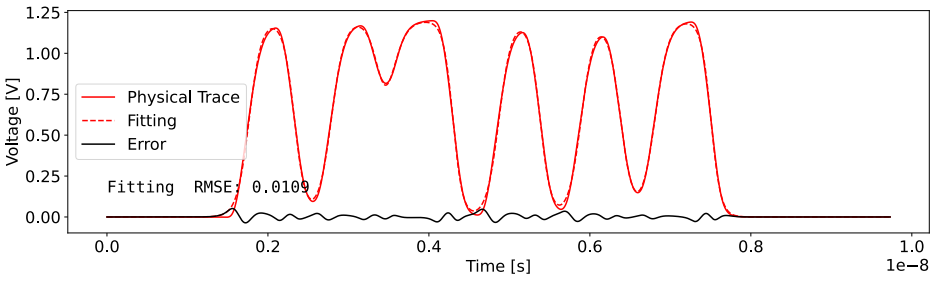

Figure 5.12: Fitted Trace

CHAPTER 6

Conclusion

In this thesis we introduced a novel method to predict the analog behavior of an electronic circuit using very few parameters that are propagated through the logic. We especially investigatet the inverter by fitting both input and output signals by arbitrary functions and maping input to output parameters using a set of transfer functions. Although this method is limited to input waveforms with certain properties, the behaviour of digital circuits is mostly covered. Results show that the prediction error lies in the range of a few percent and is highly dependent on the choosen fitting function. To further minimize the prediction error we introduced additional parameters, which significantly reduced the fitting error but introduced new problems. Future work will be denoted to refining this approach in order to achieve optimal prediction quality.

List of Figures

List of Tables

Bibliography

[1] Laurence Nagel and Donald O Pederson. Spice (simulation program with integrated circuit emphasis). 1973.

[2] B. J. Sheu, D. L. Scharfetter, P. . Ko, and M. . Jeng. Bsim: Berkeley short-channel igfet model for mos transistors. *IEEE Journal of Solid-State Circuits*, 22(4):558–566, Aug 1987.

[3] Y. S. Chauhan, S. Venugopalan, N. Paydavosi, P. Kushwaha, S. Jandhyala, J. P. Duarte, S. Agnihotri, C. Yadav, H. Agarwal, A. Niknejad, and C. C. Hu. Bsim compact mosfet models for spice simulation. In *Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES 2013*, pages 23–28, 2013.

[4] M. J. Bellido-Diaz, J. Juan-Chico, A. J. Acosta, M. Valencia, and J. L. Huertas. Logical modelling of delay degradation effect in static cmos gates. *IEE Proceedings - Circuits, Devices and Systems*, 147(2):107–117, April 2000.

[5] S. H. Unger. *Asynchronous sequential switching circuits*. Wiley-Interscience, 1969.

[6] T. Sakurai and A. R. Newton. Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, April 1990.

[7] S. Nazarian, H. Fatemi, and M. Pedram. Accurate timing and noise analysis of combinational and sequential logic cells using current source modeling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(1):92–103, Jan 2011.

[8] K. Chen and Y. H. Kim. Current source model of combinational logic gates for accurate gate-level circuit analysis and timing analysis. In *VLSI Design, Automation and Test(VLSI-DAT)*, pages 1–4, April 2015.

[9] R. J. Baker. *The Inverter*, pages 331–352. IEEE, 2010.

[10] Sorin Voinigescu. *High-Frequency Integrated Circuits*. The Cambridge RF and Microwave Engineering Series. Cambridge University Press, 2013.

[11] Ake Bjorck. *Numerical methods for least squares problems*, volume 51. Siam, 1996.

[12] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[13] Henri Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering, Duke University*, pages 1–15, 2011.

[14] Manolis Lourakis. A brief description of the levenberg-marquardt algorithm implemened by levmar. *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by Levmar*, 4, 01 2005.

[15] Lowell Jacob Reed and Joseph Berkson. The application of the logistic function to experimental data. *The Journal of Physical Chemistry*, 33(5):760–779, 2002.

[16] Nikolay Kyurkchiev and Svetoslav Markov. Sigmoidal functions: some computational and modelling aspects. *Biomath Communications*, 1(2), 2015.

[17] D. Öhlinger, J. Maier, M. Függer, and U. Schmid. The involution tool for accurate digital timingand power analysis. In *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8, 2019.