1 Introduction

1.1 Motivation

Since the 1960s, the square-law model for complementary metal-oxide-semiconductor (CMOS) transistors has been used extensively to analyze and design analog and digital integrated circuits. An advantage of the square-law equations is that they are easy to derive from basic solid-state physics, algebraically simple and yet useful for gaining insight into basic CMOS circuit behavior. As a result, the square-law model remains useful as a "warm-up tool" for students in circuit design, and it is featured in all popular analog integrated circuit textbooks (examples include [1], [2]).

On the other hand, it is well known that the square-law MOS model is plagued by several limitations, especially when it comes to short-channel transistors:

- Modern MOSFETs are impaired by numerous mobility degradation effects, related to their short channel length, thin gate oxide and their generally more complex structure and doping profiles. In strong inversion, with gate overdrive voltages ($V_{GS} - V_T$) of several hundred millivolts, the error in the transconductance predicted by square-law models with constant parameters is of the order of 20–60%.
- In moderate inversion, with gate overdrive voltages below 150 mV, the square-law model breaks down altogether and it may be in error by a factor of two or even more. This deficiency applies to all MOSFETs, regardless of channel length. However, the issue has become more pronounced with short channel devices, since moderate inversion represents a design "sweet spot" for a variety of circuits in these technologies [3]–[5].
- In weak inversion (subthreshold operation), the current flows by diffusion (like in a BJT) and the square-law model must be replaced with an exponential I–V relationship.

The above-stated issues are clearly visible in Figure 1.1, which shows the current density plot of a realistic 65-nm transistor, together with exponential and square-law approximations. The exponential provides a reasonably good fit for very low V_{GS} (weak inversion) and the quadratic approximation begins to make sense a few hundred millivolts above the device's threshold voltage (vertical dashed line). The transition from weak to strong inversion should ideally be smooth and continuous,

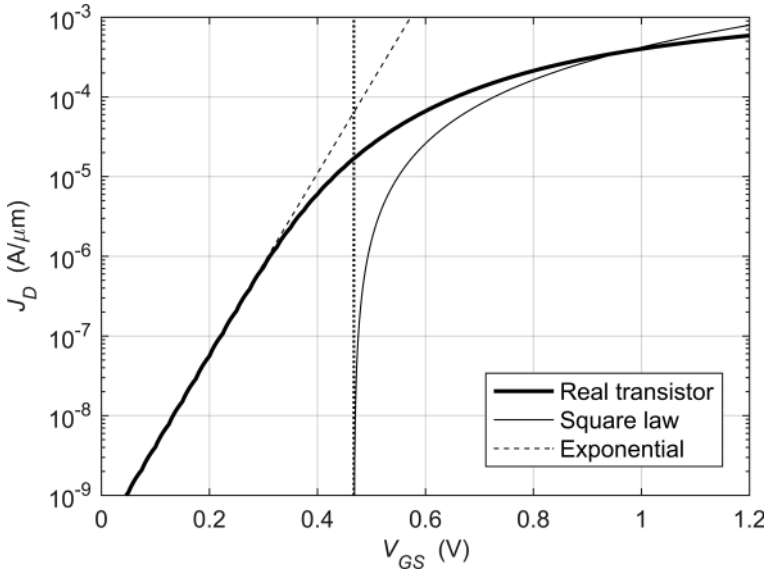

Figure 1.1 Current density of a minimum-length n-channel device in 65-nm CMOS technology versus V_{GS}. The dotted vertical line corresponds to the device's threshold voltage (see Chapter 2 for further details).

but finding a physical relationship that bridges the exponential and square-law approximations turns out to be non-trivial. In addition, at very large V_{GS}, the current density of the real device and the quadratic model diverge again due to the mentioned mobility degradation effects.

The above-stated modeling limitations are a great nuisance when it comes to design, since the square-law hand calculations described in textbooks typically won't match simulations for a classical flow (see Figure 1.2). Modern circuit simulation relies on complex device models such as BSIM6 [6] or PSP[1] [7], which are carefully crafted to reflect the "real" device characteristic in Figure 1.1. The result is a significant disconnect between hand analysis and simulation results, and consequently, designers tend to shy away from hand-calculations and resort to a design style built on iterative and time-consuming SPICE-based "tweaking."

There are several issues with the iterative simulation-based design of analog circuits. The problem is that the designer loses insight about the tradeoffs as well as the ability to sanity-check the results. While an equation-based design can reveal fundamental issues with a topology and help the designer advance his or her circuit architecture, it is difficult to gain knowledge about the fundamental limits of a circuit via repetitive sizing and simulation. What used to be design now resembles reverse engineering, which is highly undesirable for anyone who is interested in leading-edge innovation.

[1] The PSP compact MOSFET was developed by Philips Semiconductors and Penn State University.

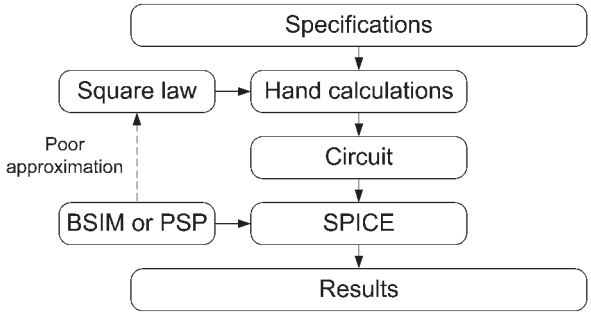

Figure 1.2 Typical analog circuit design flow based on square-law hand calculations SPICE simulation using advanced models.

The second issue is that highly iterative design based on SPICE "tweaking" is typically incompatible with the time-to-market pressure seen in today's IC developments. As a response to this problem, universities and EDA vendors have created solutions that look to automate the iterative process, leveraging the vast amount of computing power available today. The work of [8] provides an extensive reference list of such programs and categorizes them as full design automation (FDA) tools. While an FDA approach can help overcome the design time issue, it comes with the same problem as manual tweaking: It is even more difficult for the designer to gain analytical insight and intuition, which is an important ingredient for topology selection and innovation.

Taking a step back, we note that the key problem is not the equation set that describes the circuit, which tends to be either amenable to manual derivation or available in standard textbooks and publications. The main issue lies in linking the device sizing parameters (geometries and bias currents) to the transistor's representation within the circuit, typically in form of a small-signal model. Therefore, while using FDA can be appropriate and justified in some cases, it goes one step further than required, providing full automation at the expense of analytical insight.

The design approach described in this book falls under the category of full design handcrafting (FDH) [8]. It builds on classical hand analysis methods and eliminates the gap between hand analysis and complex transistor behavior using SPICE-generated lookup tables (see Figure 1.3). The tables contain the transistor's equivalent small-signal parameters (g_m, g_{ds}, etc.) across a multi-dimensional sweep of the MOSFET's terminal voltages. Since using the lookup table data closely captures the behavior of the SPICE model, the approximation issues of Figure 1.2 are eliminated and it is possible to achieve close agreement between the desired specs and the simulated performance without iterative tweaking. Though in some cases the calculations can literally be done by hand, it is usually more efficient to implement the design flow through a computer script. In this book, we chose the popular Matlab® environment for designing such scripts.

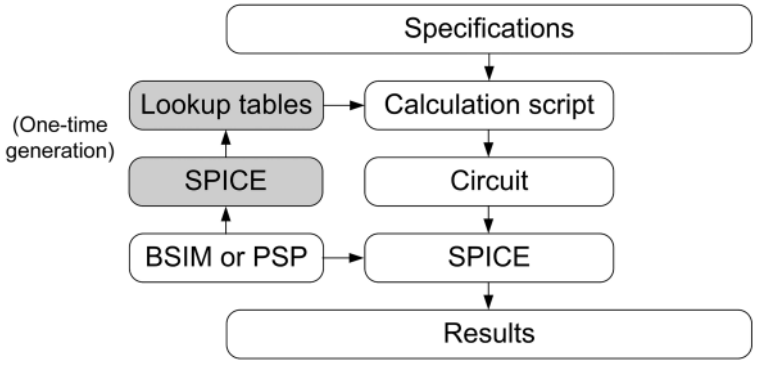

Figure 1.3 Analog circuit design flow used in this book.

It is worth noting that the outlined approach does not resemble the "SPICE in the loop" approach [9], [10] advocated in the 1980s. The main differences are: (1) The lookup tables are created once and stored permanently; they do not get updated with each circuit simulation run. (2) The design scripts tend to use abstract and simplified circuit models. This often means that the designer does not need to worry about auxiliary circuits that may be required to get a SPICE simulation to work. For example, it is possible to create a design script that evaluates the small-signal performance of an amplifier under the assumption that the bias point is perfectly set. How exactly that bias point is established can be determined later, after studying the first-order performance tradeoffs.

To implement the design flow of Figure 1.3, we need the following ingredients:

- A convenient way to generate and access the lookup table data. The generation of the proposed lookup table format is described in Appendix 2. Examples on how to access and use the stored data are given throughout this book (including an introductory example in Section 1.2.2).
- A suitable way to translate the design problem into a script that helps us study the key tradeoffs and ultimately computes the final device sizes. Most of this book is dedicated to this part of the flow. By means of examples, we study design problems of varying complexity and the derived scripts can form the basis for future design problems that the reader will encounter.

A key aspect of the proposed methodology is that we interpret and organize the lookup table data based on the transistor's inversion level, employing the transconductance efficiency g_m/I_D as a proxy, and key parameter for design. This metric captures a device's efficiency in translating bias current to transconductance and spans nearly the same range in all modern CMOS processes (~3…30 S/A). When combined with other figures of merit (g_m/C_{gg}, g_m/g_{ds}, etc.), thinking in terms of g_m/I_D allows us to study the tradeoffs between bandwidth, noise, distortion and power dissipation in a normalized space. The final bias currents and device sizes

follow from a straightforward de-normalization step using the current density (I_D/W). We will take a first look at this normalized design approach in Section 1.2.2.

The idea of g_m/I_D-based design was first articulated by Silveira, Jespers et al. in 1996 [11]. Since then, the approach has been continuously refined through academic research (see e.g. [12]–[17]) and is being taught at various universities. Several companies known to the authors have integrated lookup table based design into their design environments. These efforts were driven by the first set of graduates being exposed to the methodology in school. Despite this growing popularity, much needs to be done to make the approach accessible to a broader community and specifically those engineers who have not acquired the material at the university. The goal of this book is to provide a comprehensive resource that will accomplish this.

It is important to note that several other authors have made contributions toward a design methodology that follows the spirit of full design handcrafting with bridges between hand analysis and simulation. Among them are the inversion coefficient (IC) based flows by Binkley [18], Enz [19], and Sansen [20] as well as the 2010 g_m/I_D-centric book by Jespers [21]. The main difference between these works and the present book is that they are still based on analytical device models. Instead of working with purely numerical lookup table data, these methodologies assume that the transistor characteristics can be fit to model equations (typically EKV [22]) that are more complex than the square-law model, but not too complex to be included in a design script environment. This approach is certainly workable for today's mainstream technologies, but we decided to go with a sizing approach that is agnostic to the increasingly complex physical behavior of nanoscale transistors. Despite this goal, we still make use of the EKV model to build intuition, but won't use it to compute the ultimate device sizes. This approach is made transparent in Chapters 2–4.

1.2 The Analog Circuit Sizing Problem and the Proposed Approach

Before outlining the remainder of this book, we feel that it is important to provide a short (and simplified) walk-through of the proposed design methodology. For this purpose, we assume that the reader is familiar with CMOS square-law design and we use the shortcomings of the square law to motivate our approach.

Generally, the types of analog circuits that we consider in this book fall into the class-A category, which means that they are operated with constant bias current. A basic example is the differential pair shown in Figure 1.4, which is usually part of a larger circuit (not shown for simplicity). Sizing the circuit in Figure 1.4 means that the designer must find suitable values for

- the bias current I_D;
- the device width W;
- the channel length L.

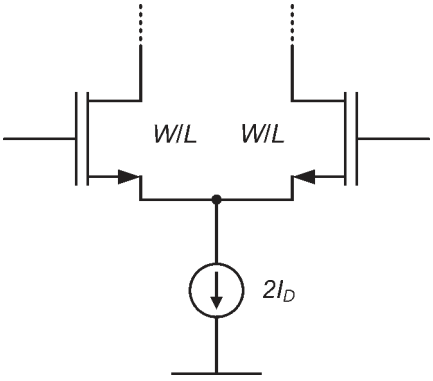

Figure 1.4 Differential pair.

For this introduction, we will assume that through some design process, we determined that the differential pair should implement a certain transconductance (g_m). How does this requirement translate into values for I_D, W and L? We will initially approach this question using simple square-law expressions and then refine our treatment to arrive at the proposed methodology.

1.2.1 Square-Law Perspective

Ideally, we would like to have an equation that relates all relevant parameters of the above example with one another. The square-law model used in standard textbooks provides such an expression [1]:

$$g_m = \sqrt{2\mu C_{ox}\frac{W}{L}I_D}\,. \tag{1.1}$$

Even though this formula is inaccurate for modern devices, it clarifies a basic, and generally important, point: there are an infinite number of choices for W, L and I_D that all lead to the design goal of realizing a certain value of g_m. To continue, we need a feel for the tradeoff that we are making by picking one of these many solutions.

To make progress, let us articulate what we would ideally like to achieve: We want to meet the design goal using the lowest possible current and the smallest possible transistor size. In absence of any other constraints (to be considered in later chapters), this immediately implies that we should use the smallest available channel length L (for example, L_{min} = 60 nm for the technology used in this book).

The remaining question is whether we should minimize the current or the device width. Note that achieving both simultaneously is not possible, since the product $W{\times}I_D$ is fixed. To think about this tradeoff systematically, we introduce two figures of merit that relate the design objective (g_m) to the variables that we want to minimize:

$$\frac{g_m}{I_D} \quad and \quad \frac{g_m}{W}. \tag{1.2}$$

Using the standard textbook square-law expressions [1], we can write these figures of merit as:

$$\frac{g_m}{I_D} = \frac{2}{V_{OV}}. \tag{1.3}$$

$$\frac{g_m}{W} = \frac{\mu C_{ox}}{L} V_{OV}. \tag{1.4}$$

Here, $V_{OV} = V_{GS} - V_T$ is the quiescent point gate overdrive voltage of the transistor. Physically, large V_{OV} means that the channel is more strongly inverted, i.e. more inversion charge is present underneath the gate.

The key observation from the above equations is that the gate overdrive controls how efficiently we employ current (I_D) or device width (W) to generate the desired transconductance. The designer can pick a large V_{OV} to arrive at a small device width or a small V_{OV} to minimize the current. Thus, the gate overdrive voltage can be viewed as a "knob" (see Figure 1.5) that fully defines the sizing tradeoff. Also, note that once V_{OV} has been chosen, the required device current (for a given g_m) follows directly from (1.3); no technology-specific parameters are needed (assuming the square law holds).

In addition, the choice of V_{OV} sets the minimum V_{DS} for which the transistor remains saturated ($V_{Dsat} = V_{OV}$ for a square-law device) and it also determines the

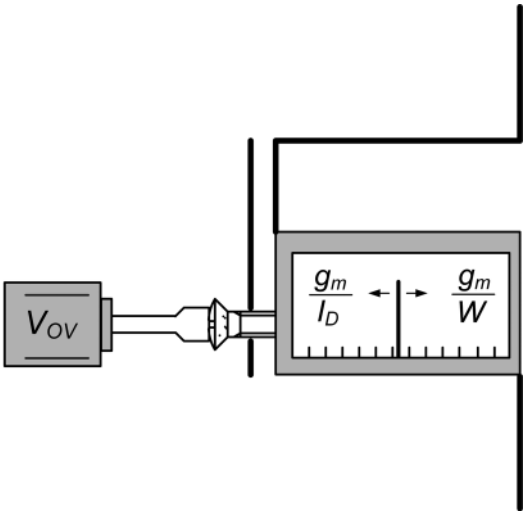

Figure 1.5 The gate overdrive voltage V_{OV} is a "knob" that lets us control the tradeoff between g_m/I_D and g_m/W.

circuit's linearity [23]. It is therefore not surprising that the gate overdrive is among the most important parameters used in square-law centric circuit optimization. For example, the seminal work by Shaeffer and Lee [24] studied the relationship between the gate overdrive voltage and the bandwidth, noise and linearity of a low-noise amplifier (LNA). It was found that the tradeoff between these performance metrics is fixed once a certain V_{OV} is chosen.

Unfortunately, and as already discussed in Section 1.1, the square-law model has become obsolete for design with modern MOS transistors. To see this, consider Figure 1.6, which plots the figures of merit of (1.2) for a minimum-length n-channel device in 65-nm CMOS. For reasons discussed in Chapter 2, the square-law expressions fit reasonably well only for a narrow range of gate overdrives in strong inversion (say V_{OV} = 0.2…0.4 V). Thus, (1.3) and (1.4) do not accurately link V_{OV} with g_m/I_D and g_m/W and the expressions are consequently unsuitable for design in the given 65-nm technology.

One way to solve this problem is to develop a more sophisticated equation set that can capture the physics of a modern device more accurately. However, as already explained, we want to eliminate the undesired tradeoff between algebraic model complexity and adequacy for design using a numerical approach.

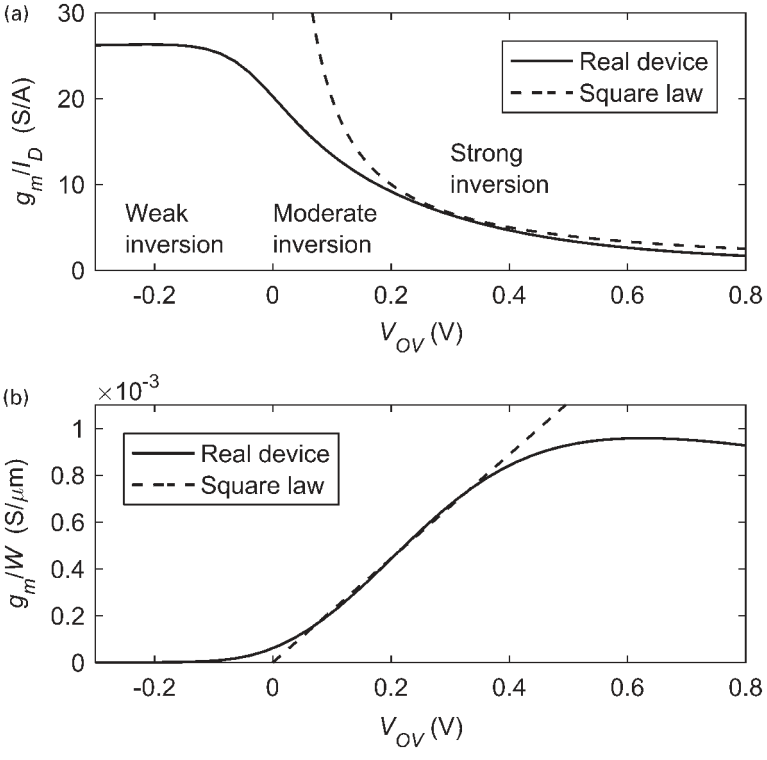

Figure 1.6 g_m/I_D and g_m/W for a minimum-length (L = 60 nm) n-channel in 65 nm CMOS. V_{DS} = 1 V and V_{SB} = 0 V.

1.2.2 Capturing the Tradeoffs Using Lookup Tables

This book advocates lookup tables to quantify the tradeoff between the relevant device figures of merit in each design (including, but not limited to g_m/I_D and g_m/W). The lookup tables can be generated (once) using a SPICE-like circuit simulator and the data can be stored in a file for future use (see Appendix 2). This is illustrated in Figure 1.7. Starting with a "well-calibrated" model file from the silicon foundry, we perform DC sweeps (and noise simulations) in four dimensions (L, V_{GS}, V_{DS} and V_{SB}) and tabulate all relevant device parameters along these sweeps for a fixed device width W. While one could in principle include the device width as a fifth sweep variable, this is not necessary since the parameters scale (approximately) linearly with W across the typical range encountered in analog design. We validate this important assumption of width independence for parameters like g_m/I_D, g_m/W, etc., in Appendix 3.

The sweeps can be repeated for all devices offered by a given foundry, leading to one lookup table per transistor type. With this flow, the quality of the lookup table data is of course directly linked to the quality of the foundry models. If the foundry models are poor, it will be challenging to produce a working circuit altogether, independent of which tool set and sizing methodology is used. Model quality assurance is therefore outside the scope of this book. However, proper inspection of the lookup table data (leveraging the physical intuition conveyed in Chapter 2) can sometimes expose modeling issues. For example, the shape of the g_m/I_D curve in Figure 1.6 may reveal discontinuities, improper location of the weak inversion plateau, etc.

To make the lookup table data easily accessible in Matlab, we have created a function (called "lookup") that allows us to read all transistor model parameters as a function of the applied bias voltages (with interpolation capabilities). Further details on this function are found in Appendix 2. To give the reader a basic feel for its usage, we provide two simple examples from the Matlab command line[2]:

```
>> ID = lookup(nch, 'ID', 'VGS', 0.7, 'VDS', 0.5, 'VSB', 0,
'L', 0.06)
ID =
9.3127e-04
>> Cgs = lookup(nch, 'CGS', 'VGS', 0.7, 'VDS', 0.5, 'VSB', 0,
'L', 0.06)
Cgs =
7.0461e-15
```

To see the device width (in microns) for which this data was tabulated, one can type:

```
>> nch.W
ans =
  10
```

[2] In the given example, the gate length L is specified in microns. The argument "nch" specifies the device type and points to a Matlab structure containing the data. W is 10 μm, which is the width that was used to create the lookup table data; see Appendix 3 for more information on how this reference width was selected.

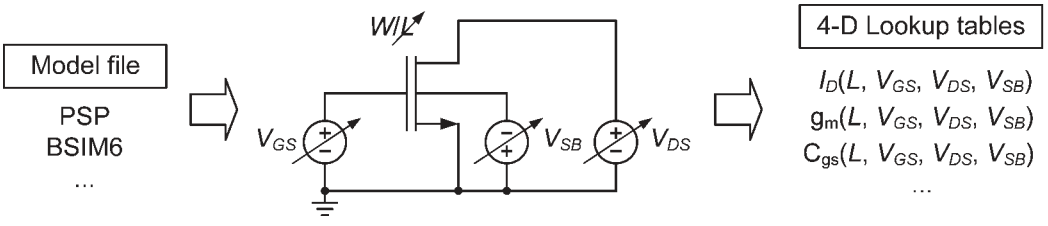

Figure 1.7 Lookup table generation using a four-dimensional SPICE sweep. The width W is set to 10 μm (5 fingers, 2 μm each) for the lookup tables used in this book.

Once these lookup tables have been generated, the remaining question is: how should the designer organize and use the data to gain insight into circuit sizing tradeoffs? To answer this question, we return to the differential pair example, which established a fundamental tradeoff between g_m/I_D and g_m/W.

One direction we could pursue with the lookup table data is to produce the "real device" curves of Figure 1.6, and pick a value for V_{OV} that dials in a suitable tradeoff between our figures of merit (g_m/I_D and g_m/W). However, since we are not dealing with a device that behaves per the square law, V_{OV} is just a remnant of an obsolete model. Hence, a key concept advocated in this book is to eliminate V_{OV} as a design variable altogether and instead link all design tradeoffs to the choice of g_m/I_D, which (like V_{OV}) can be viewed as a proxy for the device's inversion level.

As explained further in Chapter 2, g_m/I_D ranges from about 3…30 S/A in all modern CMOS technologies, where the lower end corresponds to strong inversion, the mid-range (around 12…18 S/A) amounts to moderate inversion, and the peak value is linked to weak inversion. In addition to indicating the inversion level, g_m/I_D is a useful figure of merit for another reason, as we have already discovered in our differential pair example. It directly quantifies the transconductance per unit of current invested in the device. Therefore, we advocate the unit of S/A (instead of 1/V).

With V_{OV} eliminated, the sizing tradeoff for our differential pair example is elegantly captured in a single plot, shown in Figure 1.8. Picking a small g_m/I_D means that we end up with a large g_m/W, implying a small device for a desired value of g_m. The opposite is true when we opt for a large value of g_m/I_D, where the device will be wider, at the benefit of reduced current. Note that the quantities plotted in Figure 1.8 can be extracted from the lookup table data introduced above. Each point of the tradeoff curve corresponds to a different V_{GS} value in the sweep. However, the exact value of V_{GS} for the desired tradeoff point (the chosen g_m/I_D) tends to be secondary from an optimization perspective. In our example, we only care about how much current and area we are investing to realize a certain value of g_m. More generally, we will in fact see throughout this book that width-independent parameters play a fundamental role in systematic tradeoff studies and circuit sizing.

To simplify the lookup of parameter ratios, the Matlab function introduced above has another usage mode that lets us directly fetch one ratio as a function of another. Below is a simple example that visualizes this functionality: