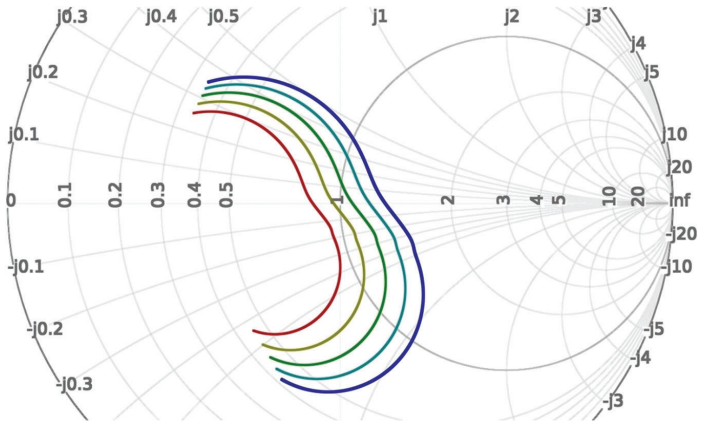
2

Manual Analog-Centric Design Style(s)



We collect now the most important best practices for efficient design and verification, the typical analog *flow*. We introduce briefly the concepts of specification margins, corner simulations, worst-case and Monte-Carlo. Corner and MC simulations are two standard approaches to discover the design behaviors, and both are simple in essence, because you just run a certain *fix* scenario, a *fix* "design" of experiments (DOE). Both methods act like a simple signal chain, without feedback; you as user have to inspect the results, and you have to decide on further simulations, if needed. To succeed in analog design a lot of experience and anticipation is required, but also the more systematical you work, the higher the chance for being in time and making a successful tape-out.

In this chapter, we also introduce the concept of worst-case corners (WCC), and discuss how to find them. Basically this is a simple task: Just run the simulations and look which ones are most critical. However, actually here we could apply also a more "mathematical" approach, like trying to find how the corner variables influence the corner results. This is (multivariate) modeling in terms of (performance) functions. Note, as in this chapter the focus is more

on manual best-practices, we discuss here on the basic math, and later when multivariate techniques are required for more difficult statistical techniques we pick up the topic in more details (Chapter 5).

We end with a little "benchmark" on "men versus machine"! Sometimes men wins, sometimes it is really good to have clever adaptive methods, because worst-case search is a highly nonlinear problem; and with brute-force methods this task would be often very time-consuming.

So this chapter is also important as a starting point, and because we discover some ideas to improve the flow and to compose a more assisted overall flow. An important outcome for a making a design is *learning* about design, e.g., you should understand why a design fails; in fact, even the yield is not the only thing you need to care about, e.g., you may sell non-perfect devices for reduced temperature range or reduced clock frequency. To showcase the impact of variability we present at the end of the chapter a small CMOS RF PA circuit, this also opens a series which we have named "Design with Pictures"—math, pictorial design techniques, and circuits, going slightly beyond purely introductionary examples.

One can learn a lot from looking to other fields of engineering, math, or science in general. The problem is usually that project time is limited, and designers have to focus too much on everyday problems and cannot care so much about flow problems, unfortunately. Actually, in the old days, designers typically focussed and spent probably more time on circuit functionality, for analysis and deep understanding, besides pure verification. During the design phase, designers collect a lot of know-how and invent new circuits and system solutions. The collection is step by step, design, testbenches, and verification coverage grow in parallel, and trust in your design comes also step by step, not in a final sign-off verification (this is maybe true only for LVS).

Let us pick up again the op-amp design example and similar ones:
A first step is usually picking a meaningful circuit topology (often with partially ideal sources and elements—resistor, capacitors, etc.) and making an initial testbench (e.g., for DC behavior). The circuit selection is an essential step, but it usually does not stop early (only in case of hard IP reuse). Often many refinements are needed, like outputs have to be extended, cascodes have to be added for high PSRR, and maybe you need protection circuits. Sometimes also little concept changes are needed because the full requirements have been available too late, and one concept might be if you need only one block (like a bias block), but if you need multiple ones, another implementation might be preferable. Often we can decide early how trustable and robust a design is, like

a CMOS Schmitt trigger is usually much more technology dependent (e.g., thresholds differ significantly for SF vs. FS corner, the 1st letter stand for the NMOS corner, the 2nd one for the PMOS, so SF means the combination of slow NMOS and fast PMOS) than one based on a differential pair (voltage accuracy depends almost only on mismatch and of course reference voltage accuracy). However, exactly how many differences exist depends on many things like spec ranges, technology, temperature, and supply range—and on additional circuit tricks like calibration or replica parts. In this circuit "finding" phase, a lot of interactive work and many simulations are required, so tool speed and usability often matter much more than automation. If done effectively, this "SPICE monkeying" phase is not so bad and you can learn a lot. Typically, if something goes wrong, you even learn more, e.g., you may be able to exclude bad solutions or improve existing ones for your specific application and for effects that might be not important in an older application of the predecessor block.

For Further Reading:
There are many good books available on analog design. As mentioned, we focus an analog in a wide sense, so we do not cover in much detail digital or mixed signal aspects, layout topics or modeling aspects, but in the following list some top references also on these topics are included. Have fun reading them!

- Willy, M. C. Sansen, *Analog Design Essentials*, Springer US, 2007.
- R. A. Pease, *Troubleshooting Analog Circuits*, Butterworth-Heinemann, 1991. R. A. Hastings, *The Art of Analog Layout*, Pearson Prentice Hall, 2006.
- J. Chen, M. Henrie, M. F. Mar, M. Nizic, *Mixed-Signal Methodology Guide*, Cadence Design Systems, 2012.
- W. H. Press, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.
- H. E. Graeb, *Analog Design Centering and Sizing*, Springer Netherlands, 2007.
- H. Graeb, *ITRS 2011 Analog EDA Challenges and Approaches*, in Design, Automation Test in Europe Conference Exhibition (DATE'2012), Dresden, Mar 2012, pp. 1150–1155.
- W. K. Chen, *Computer Aided Design and Design Automation*, CRC Press, 2009.
- R. Spence, R. S. Soin, *Tolerance Design of Electronic Circuits*, Imperial College Press, 1997.

2.1 Biasing and Transistor Sizing

Unfortunately, not all variations can be minimized easily, e.g., a single transistor amplifier stage will have a certain specific temperature variation for gain, output power, noise, and distortion. For obtaining a constant transconductance g_m and gain, usually proportional-to-absolute-temperature (PTAT) biasing is well suited (Figure 2.1), but (for sure) this lowers the intermodulation point IP3 and the slew rate at low temperatures. The latter can be stabilized with a constant bias current instead of PTAT, but this can e.g., lead to severe phase margin problems at low temperatures in feedback circuits—although feedback has certainly many benefits. For instance, negative feedback can make performances generally more stable (usually at the cost of larger noise and lower gain). Another method is using class-AB circuits (which are unfortunately more complex and often have limited common-mode voltage ranges); these can offer more constant g_m over the input voltage, and more current drive capabilities.

Besides the bias concept, also the transistor intrinsic behavior is a key factor, and not only one measure like transconductance g_m matters. The first decision is usually almost trivial and is on the operating region like off-state, ohmic region, or saturation region. In the later, for a given transistor current I_D (or

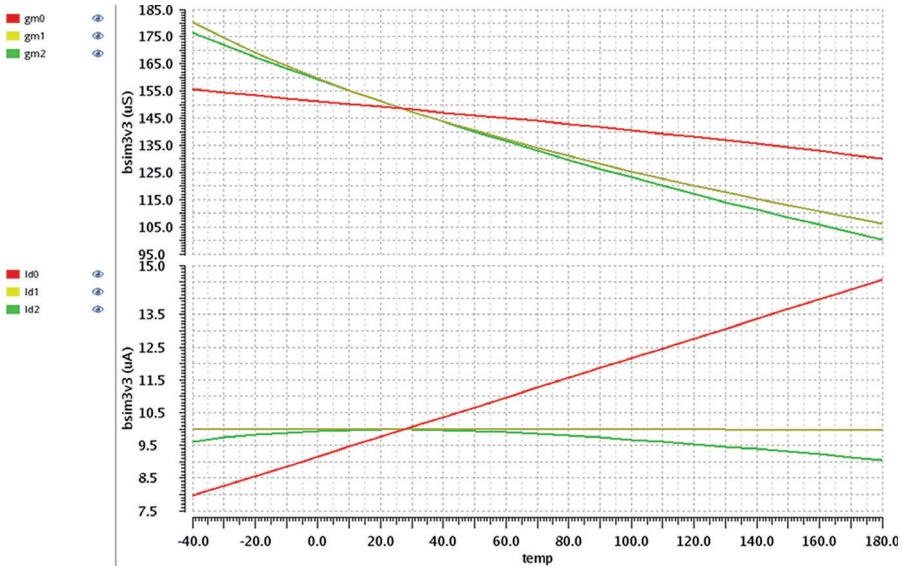


Figure 2.1 Typical MOS g_m and I_D behavior for PTAT, constant current, and constant–voltage biasing.

I_{C})—e.g., set according to experience, total current budget, noise level, slew rate, and output power—we can influence many kinds of measures in different ways, just by *sizing* the transistor width W and length L differently [Binkley]:

1. g_{m}-based sizing makes sense for the MOS saturation region and requires mainly a certain W/L ratio. Maximum g_{m} requires minimum length L, but that could be non-optimum regarding matching, flicker noise, or output conductance g_{DS}.
2. To make a circuit functional, maybe the V_{Dsat} is most important. Again, the W/L ratio matters mainly.
3. White-noise-based sizing requires a large transconductance and thus a certain minimum current and large W/L for voltage-amplifying stages, whereas for current sources you should not use too short transistors due to their bad matching and larger current noise.
4. Flicker-noise-based sizing requires usually larger gate areas than pure white-noise-based sizing. Also it could be that PMOS transistors are significantly better than NMOS in some technologies.
5. C_{in}-based sizing matters if you use MOS transistors as capacitors, but also in charge amplifiers, the optimum noise performance is reached if the transistor input capacitance is roughly equal to the generator capacitance.
6. Matching-driven sizing might be needed for low-offset amplifiers and requires a certain minimum area $A = W \cdot L$.
7. I_{Dsat}-based sizing makes sense in logic circuits and also in bandgap start-up transistors. W/L matters most. When looking to the maximum transistor current not only the silicon part transistor performance matters, for high-current applications also reliability, metallization and vias needs to be checked carefully.
8. R_{on}-based sizing is useful for switches and depends also on W/L. For lowest R_{on}, we need minimum L. Larger L makes sense if a certain matching accuracy is needed and is mainly useful for non-switching applications (like variable-gain amplifiers).
9. f_{T}-based sizing makes sense for many high-speed circuits. Typically, this requires a certain minimum current density, low gate capacitance, and short transistors. Unfortunately, it comes with bad DC characteristics like low voltage gain per stage and large mismatch. Of course in an op-amp, the stages should have a f_{T} beyond the GBWP—though you typically do not exactly know how much, because this depends also on layout parasitics.
10. TC-based sizing is often used in discrete designs or when temperature stability is of high priority. The idea is that in a MOSFET, there is a

certain V_{GS} in which the TC of I_{D} is zero, because the effects of V_{TO} and mobility cancel. Also a stable on-resistance can be obtained. The only pity is that this point is corner dependent and it does not lead to constant g_{m}; in addition, the area might be quite large, which matters most in power circuits.

11. In rare cases, you might be able to even tweak on technology parameters (like epitaxi thickness or substrate doping), but there will be still many compromises. For instance, high speed comes for sure with lower breakdown voltage for given semiconductor material (like silicon or GaAs). Also a BJT with high current gain β will have low early voltage V_{EA}.

Overall (and this list is not complete), no single method alone fits! For uncritical transistors, you may only need few seconds to select I_{D}, *W*, and *L*, but for critical ones, you need a mix of methods and many hours plus a full inspection of the full block performance (often even including Monte Carlo and corners). So overall, you have to check almost all the listed characteristics. For true RF circuits, also other electrical parameters (such as $\mathrm{f_{max}}$ and k-factor) and the layout matters, like number of fingers and metallization. Only really unimportant "near-digital" transistors can be regarded as uncritical; and we can use any small $\mathrm{L} \geq L_{\mathrm{min}}$ and any meaningful width. For something in between RF and simple digital, designers often follow a little flowchart that takes step by step at least the major measures like I_{D}, g_{m}, and f_{T} into account (see, e.g., [Sansen2]).

Besides the pure *sizing*, also a careful *type* selection is needed for all components like resistors (like poly versus well) and capacitors (like MIM caps versus MOS caps, with big differences according to substrate parasitics, quality factor Q, linearity, and breakdown voltage) and of course transistors (low V_{TO} versus high-V_{TO}, deep N-well, thick gate versus thin gate according to maximum terminal voltages, etc.). Even if we would follow all the mentioned rules, it may still happen that also other rules like on electromigration dictate us a change, e.g., increasing the transistor width to a certain minimum width like 100 um, although for electrical performance maybe 50 um would be better.

Note: For many circuits, figure of merits (FOM) are available, which describe the power-performance trade-off, like for ADCs, PAs, or VCOs. Check how close your circuit is to the best-designed circuits in similar technology to get a feeling how difficult your design and transistor sizing will be. Of course often such FOMs are only related to the core circuit and exclude bias generators, voltage reference generation, additional buffers, etc. In addition, you need some margins for production tolerances, temperature variations, etc.

Knowing about the circuit details and dependencies is of course still the key competence, and luckily with the support of computers, it is bit easier to obtain all the performances than with breadboarding. So when a designer wants to understand e.g., the region of stability for an op-amp or how the operating point for a transistor is defined by the bias circuit, he/she can still make it based on equations and datasheet plots, etc. And of course also doing parameter sweeps and minimizing TC, improving PSRR, etc. are important steps to make a design robust. Many problems have to be solved, and often graphical methods are very helpful (e.g., the load-line method or the Smith chart). They can help a lot to understand circuits, but later we will also see that the same is true for many advanced numerical methods. In this context and in our book, "manual" design should not mean "without computer" but using techniques already available in older EDA environments, i.e., design by the use of a schematic entry and a simulator, being able to do a corner analysis and Monte Carlo—but not "more."

There is no single most efficient best flow applicable to all kind of blocks. And one consequence of applying a mix of techniques and dealing with difficult problems is that almost always some iteration is required.

Good judgement is the result of experience. Experience is the result of bad judgement! By experience and working in a systematic way you can usually avoid too much stupid brute-force verification and too much trial-and-error. There are several examples which show that manual design can outperform computer simulations—and a clever mix is often the best. For instance, computer programs usually work completely numerical, whereas designers can apply analytical hand calculations to find at least an approximate solution (which is often good enough). This typically leads also to deeper design understanding, e.g., regarding sensitivities. For a computer, sometimes even the simplest things may become time-consuming: A designer often knows from symmetry reasons that the sensitivity to two parameters is the same, or he/she knows that the sensitivity on differential gain to many bias components is very low because they only have an influence on common-mode signals or that certain sensitivities are even zero because related transistors are not active in the current operation mode.

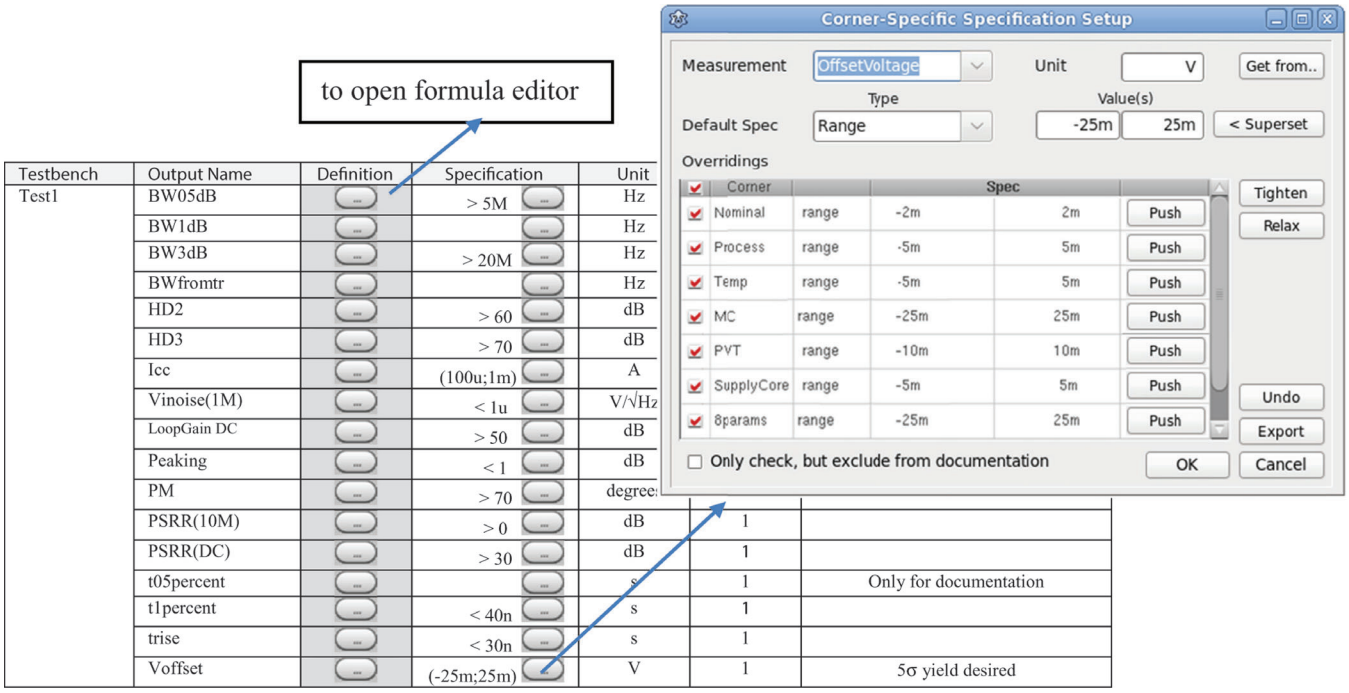
Such insights also lead to efficient design strategies. One example is that you typically first need to make sure that your analog circuit is having the correct DC operating point and e.g., achieving a certain gain, before thinking about other characteristics such as noise performance, speed, and distortion. So circuit design is often "pampering up" a circuit step by step, whereas a pure verification engineer could be already happy with finding one condition in which the design breaks.

In design, this translates usually to solving the most urgent problems first, before solving second-order problems. Of course, what is major and what is second order is not always so easy to know upfront and requires some experiments and experience. For instance, if reverse isolation is critical for your amplifier, you should consider using a cascode stage, but that might be harder to implement at low supply voltages. So you often need to inspect both variants: normal common-source stage versus cascode stage. Once you "pampered up" your design, the next step should be testing it under more difficult situations, like check whether it still works at extreme temperatures, or at minimum or maximum load and supply voltage. In this phase, designers do a lot of parameter sweeps and typically tweak their design further.

2.2 Specification Margin Approach: Fast but Risky

As explained, the brute-force simulation effort to maximize the overall yield is usually far too huge. One way to divide and conquer and for better design understanding is to focus on partial yields, because often the designer has an idea what to change if the power consumption is too large, what else is to do for better bandwidth, etc. Looking only to the overall yield would mean ignoring important information! In fact, also just looking to the partial yields is still a method with big waste of information, because looking only for yield means that we would act as a 1-bit ADC, just because when we calculate the sample yield we only check for pass or fail, but ignore e.g., the information on *how much* we fail! Let us go back to our op-amp example in more detail and analyze what could go wrong if we follow a very fast approach.

An approach with really huge speed-up would be doing just a nominal analysis and tightening the specs (see Figure 2.2). If you know from previous designs, circuit topology, reading the technology documentation, or swept simulations that sheet resistance is your major impact on supply current, and you know it is varying by $\pm 15\%$, you should tighten the current consumption spec by 15%! To get some safety margin, you may use 20% (so 80% of the original spec) to also include second-order effects like TC of the resistance and reference voltage variations. This safety margin can be called specification margin or performance margin, because it is related to the worst-case performance and the spec limit. For comparisons, it is often good to define it not as absolute measures, but in percent.



Testbench	Output Name	Definition	Specification	Unit		
Test1	BW05dB	...	> 5M	Hz		
	BW1dB	...		Hz		
	BW3dB	...	> 20M	Hz		
	BWfromtr	...		Hz		
	HD2	...	> 60	dB		
	HD3	...	> 70	dB		
	Icc	...	(100u;1m)	A		
	Vinoise(1M)	...	< 1u	V/√Hz		
	LoopGain DC	...	> 50	dB		
	Peaking	...	< 1	dB		
	PM	...	> 70	degree		
	PSRR(10M)	...	> 0	dB	1	
	PSRR(DC)	...	> 30	dB	1	
	t05percent	...		s	1	Only for documentation
	t1percent	...	< 40n	s	1	
	trise	...	< 30n	s	1	
	Voffset	...	(-25m;25m)	V	1	5σ yield desired

Figure 2.2 Flexible corner setting and typical design environment functions.

In our op-amp example, mentioned a similar margin approach when simply adding the worst-cases from the MC analysis and the corner analysis. In MC, the performance delta from average performance to the WC corner can be often expressed in terms of standard deviation sigma σ, and later we pick up the margin method when discussing the process capability index C_{PK}.

Also the structure of many datasheets supports a margin approach, like defining a tight spec at 27°C and a relaxed spec for full temperature range. Also for PCB designs the developer starts typically with spec margin methods, just because the tolerances are often well-documented, and executing sweeps on temperature, supply, etc. is quite time-consuming. However, there are also problems with specification margin approaches:

- You need to determine the design sensitivities quite carefully, either by hand calculations or by simulations.
- The approach usually only works fine with almost linear relationships and if no strong correlations (so-called mixed terms) are present.
- If many effects are important, they can add up too much, like close to ±100%. In such cases, the margin approach is too conservative, but in other cases, it is often too optimistic!
- You cannot directly debug the design at the point of spec violation!
- The margin approach might be suitable to *center* a design on specs, but making the design really better and reducing the performance spreads is difficult. Here, and for asymmetric tolerances a corner approach can be much better.

In conclusion, the margin approach works fine only for few performances like current consumption or maybe bandwidth or noise figure, but seldom for difficult specs like phase margin, settling time, or IP3. So you should use it mainly in the planning phase or in the starting phase of circuit design, but not for careful verifications. For instance, an RF designer may know from experience that gain is typically 1 dB worse than simulated without layout parasitics. Here, a good way to go is to improve the modeling, e.g., to include at least expected or hand-calculated wiring, package, and substrate parasitics! Maybe this degrades the gain by 0.7 dB—so that you can safely reduce the "fear" design margin to 0.3 dB to be protected against the "unknown."

One big advantage of modeling enhancement is better debugging, and another is that you can also improve on other unwanted effects, e.g., the parasitics may also cause stability problems or can cause cross talk.

How to treat tolerances? This is a big problem in the spec margin approach, and doing it wrong, it could fail. However, unfortunately doing it right often ends up in over-pessimism! If we simply do sweeps and add the magnitudes of the ranges, like $\sum|\Delta y_i|$, it only looks as if we would do a worst-case analysis! This is because we typically do the sweeps of one parameter with the other parameters kept fix, like at nominal. However, it could easily happen that putting such a fix parameter to another value ends up in a bigger Δy_i! In conclusion, this is actually no WC method. If a sensitivity analysis is simple or if the sensitivity S is just known quite well, e.g., current is PTAT or TC of a BJT V_{BE} is app. –1.8 mV/K, then we could also estimate Δy as $S\Delta x$. However, again we would need to find the maximum sensitivity! Actually, doing it this way, the whole approach tends to become both more complex and also often far too pessimistic, unfortunately. So better use the spec margin approach if you are allowed to overdesign (like spending quite a lot of area and current) and if your design is not too nonlinear. Unfortunately, we have seen that even for a classical linear circuit like an op-amp, OFAT can fail for WC finding for that reasons.

For statistical variables, the classical approach is to add quadratic, so add the variances $V = \sigma^2$ and then take the square root. This leads to quite a realistic error propagation. It should be mentioned that the approach is correct if there are no correlations and if you really only aim for standard deviations. For real worst-cases, it is only suited if you have pure Gaussian distributions. Statements like "beyond $\mu + 6\sigma$ there are only 1ppb samples" are critical, because if you do not have found a 6σ sample by simulation, you actually make a risky extrapolations [Schmid]. Beside all these problems, it is also extremely important to clearly state, what is meant when e.g., writing $\pm 10\%$. Clearify if it is a hard limit or only $\pm$1sigma!

Besides all the criticism in the concept of design margin, it is a good starting point. For instance, one outcome from a swept analysis is the sensitivity, but another one could be the point in which the design starts to "break." Understanding the design behavior in this "breaking" point (e.g., caused by saturation or breakdown) helps usually a lot in finding where and how to improve the circuit! Figure 2.3 gives a design example; for the sketched transistor stack, we need $V_{DD} > V_{GS} + 2V_{Dsat}$, but you need to do a similar analysis also for many other stacks, like to obtain the range for the input common-mode voltage and for the output voltage.

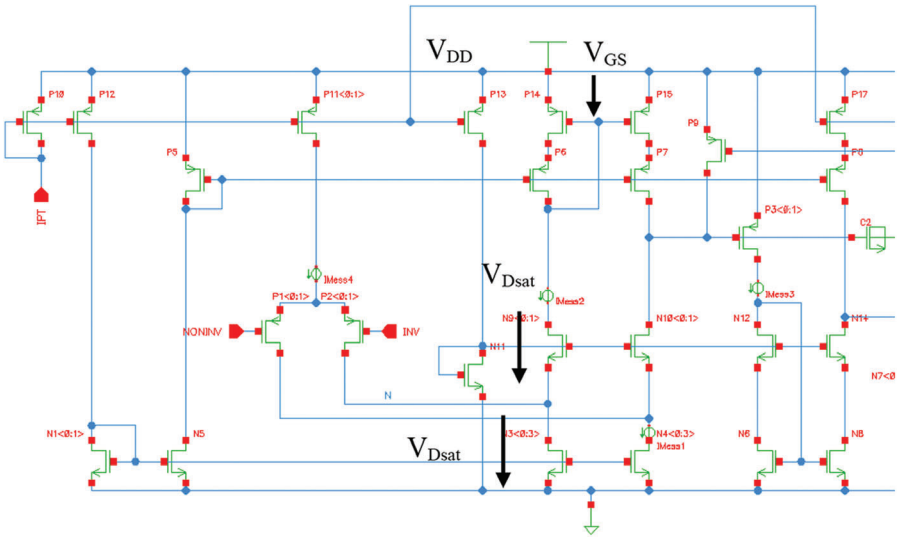


Figure 2.3 Typical transistor stack in an analog circuit to show the worst-case on saturation.

In conclusion, this method of doing parameter sweeps till the design really breaks should apply intensively in early in design stages, but not only. Of course, as extension, you can also apply *combined* sweeps or execute the sweeps around an expected worst-case like temperature sweep at Low V_{DD}+SS corner for saturation-critical specs.

Note: Shifting the break points more and more to the true wanted spec ranges (and beyond) is also a method to make difficult optimizations *feasible*! It acts like g_{min} or source ramping in the DC analysis by a circuit simulator! In difficult circuits and DC simulations, it may happen that the circuit equations are too hard to solve for supply $V_{\mathrm{DD}} = 3$ V, so the idea is to start with a simpler problem, like finding the circuit solution for a smaller value (like 1 V or even 0 V; in these the nonlinearities are often lower) and then increasing V_{DD} till we reach the full value. Also in the laboratory do not directly apply the full supply voltage immediately after building the prototype.

If the simple but also very efficient spec margin strategy can be successful depends on the design itself, e.g., phase margin PM is usually uncritical for one-stage op-amps, but might be difficult topic for multistage amps. Such performances can never be treated by spec margins, here we really need to simulate additional critical corner cases!

Also a mixed approach can be created for reducing the set of worst-case corners, the "stretched parameter" method: Often *different* parameters can

change the circuit operation in a very *similar* way like a decrease of 100 mV in V_{DD} or $\Delta T = -100°\mathrm{C}$, or switching from FF process corner to SS, could have a similar impact on saturation effects and minimum possible supply voltage! So instead of combining all sweeps (giving unfortunately many combinations!), we can also pick one variable only and make its range more extreme! In automotive designs, often the temperature effect is the strongest one, so we can make it e.g., 25 K wider and cover this way also smaller effects like threshold voltages & V_{DD} tolerance! Often you know that V_{TO} changes over process by 100 mV and the TC is –1.5 mV/K, so (for a given circuit) you can directly "translate" it to a change in temperature. This way the designer has still a good overview and can focus on the most important problems, e.g., debugging the circuit at the breakpoint (now in a simple temperature sweep). Problems can appear if there are multiple failure mechanisms, so again this method is typically used in earlier design stages.

In conclusion, if we would know about the set of "worst-case conditions," i.e., the combination giving the worst performance *within* allowed (valid) parameter ranges, we could easily check against the specification directly (without margin) and we could also speed-up design, verification, and maybe also the production test dramatically, with much less risk than in using the specification margin approach! We have to pay a very little price: It cannot be as fast as an approach purely based on design margins, but it can treat quite strong nonlinearities and correlations.

A general compromise for design is also to do design tweaks not at typical conditions but already at an expected worst-case corner, like minV_{DD}, WC load for stability and total harmonic distortion THD, and highest clock frequency. This way the performance margins and the errors in estimating them becomes smaller. Of course, at some point like when moving to layout using the real worst-case corners is much better.

2.3 The Worst-Case Approach

Checking the design at the most extreme parameter combinations is intuitively a good verification method as it is (much) less risky than spec margin approaches. Finding the worst-cases is also a method to speed-up the whole design flow against the exhaustive method with running full corners and full MC. Figure 2.4 shows such flow in alignment with sensitivity-driven design.

The testbench setup can be done in the same way as in any flow, like the brute-force flow. Such testbenches run in a circuit simulator, and typically, an automated calculator tool can extract key performances (like bandwidth, rise

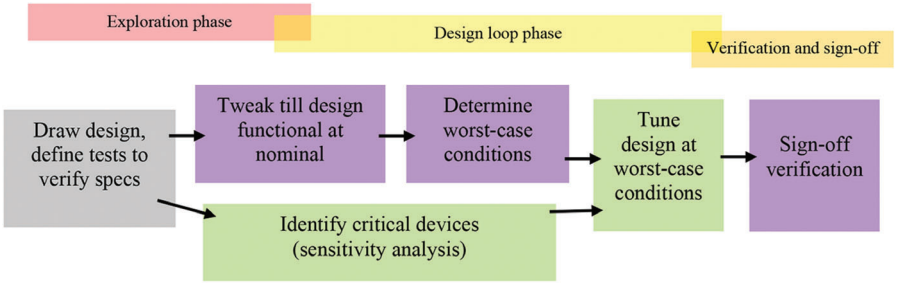


Figure 2.4 Flowchart for efficient circuit design.

time, and phase margin) from the simulation raw data (like voltage signals versus frequency or time usually saved in a binary format).

Finding the critical conditions (represented by $\boldsymbol{x}_\mathrm{S} \cdot \boldsymbol{x}_\mathrm{R}$) is helpful for both spec checking and debugging. To improve the design, we usually have to identify which parameters $\boldsymbol{x}_\mathrm{D}$ are influencing and critical and tune them. Typically, the designer knows quite well about the *major* sensitivities, but often only qualitatively, and also surprises are possible, caused by unwanted resonances, "dirty" circuit tricks, etc. Once the designer is happy, he can proceed with more detailed sign-off simulations, layout, etc.

Looking only to the set of deterministic parameters $\boldsymbol{x}_\mathrm{R}$: The worst-case is given by the worst performance $\boldsymbol{f}$ defined by a certain value combination of environmental parameters $\boldsymbol{x}_\mathrm{R}$, each being in its *valid* parameter range, whereas the design parameters $\boldsymbol{x}_\mathrm{D}$ are fix. Note that we do not need exact specifications, and the decision whether an upper or a lower spec limit would be set is enough.

This way finding the worst-case (WC) is possible with a simple grid-based approach and is similar to parameter search or an optimization. Often the worst-case occurs at the most extreme parameter settings, but not always (e.g., for a bandgap output voltage, having a quadratic behavior). We can expect at least one WC combination for each performance, and of course sometimes WC for different specs appears at the same condition (at least approximately). Due to correlations, also the overall worst-case is often not simply the combination of the individual worst-cases obtained from individual parameter sweeps!

If we want to include statistical parameters $\boldsymbol{x}_\mathrm{S}$, the definition of worst-case becomes more difficult; because parameters following a normal Gaussian distribution do not have a finite "allowed" range, they may vary (theoretically) from $-\infty$ to ∞! What we can do is to assign a certain minimum yield

(like 99.85%) to our "worst-case." And this way we can calculate also the worst-case statistical parameter sets.

The major problem is that many older design environments have no support for this, and unlike normal environmental range parameters $\boldsymbol{x}_{\mathrm{R}}$, designers have no way to even set the statistical parameters directly—even if we know the worst-case combination on $\boldsymbol{x}_{\mathrm{S}}$.

Interval Analysis. There would be also other mathematical ways to treat that problem, e.g., the so-called interval analysis assumes finite ranges for parameters and tries to calculate from that the variations of circuit performances. However, unfortunately, this approach tends to be far too conservative, leading to strong much overdesign, and it is hard to apply for nonlinear circuits, and many variables, or even statistical variables.

Looking to the overall variations in a design, the performance *f*, e.g., DC gain, delay, or bandwidth, is usually quite amazingly good at nominal conditions and without mismatch. However, process variations cause significant degradations, like 20%. To reach the worst-case across all corners (Figure 2.5), you have to accumulate also all environmental variations like supply voltage and temperature (classical PVT analysis). For CMOS logic, the individual variations are usually in a similar range. In analog design, it depends highly on circuit type, but with a few tricks, the supply changes can often be reduced quite a lot compared to logic design. On the other hand, quite some effort is needed to make analog functions stable against mismatch, e.g., because in modern technologies you often need to operate at quite small supply voltages, so that the ratio between offsets and V_{DD} is not so small as one might expect.

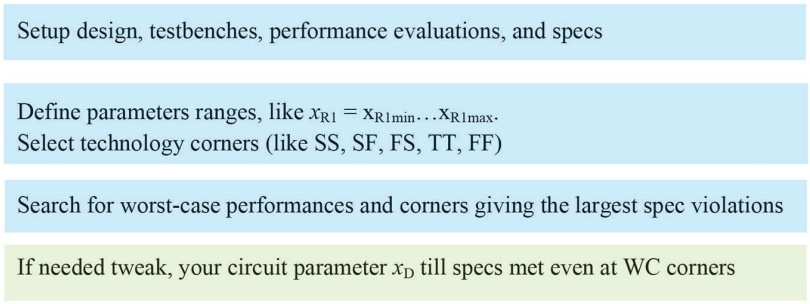


Figure 2.5 Design based on worst-case corners.

2.4 Worst-Case Corner Finding

Solving the problem about the worst-case is a key point in design, and the other one is finding ways to improve the design efficiently. Modern design environments offer several methods *beyond* pure corner analysis and Monte Carlo (next subchapter), and the worst-case corner creation is a good starting point to inspect advanced *automated* design techniques. Based on manual techniques we compose some worst-case search strategies, and let them run against standard methods, and adaptive methods in modern EDA environments.

Let us start with observations from typical design situations. Often it is possible to make a design running well at nominal conditions, and often even for all points in a *sweep*, like design is in-spec for both a temperature and a supply sweep. However, this does *not* guarantee that the design works also if we combine the environmental variations. It is also not sure that the worst-case is already given by the individual worst-case for each variable! Designers address this difficulty usually by not only performing single parameter sweeps but also running multidimensional sweeps and the so-called corner simulations. A corner is a set of parameter values, usually including environmental variables but often also technology parameters $\mathrm{R_{sheet}} = (\mathrm{minR_{sheet}}, \mathrm{nomR_{sheet}}, \mathrm{maxR_{sheet}})$.

The key problems are unfortunately also quite manifold, e.g., there is no single worst-case like the combination of maximum temperature, minimum supply, and maximum load capacitance—almost for sure it is different for each key performance f! Also it can easily happen that the worst-case conditions change if you change the design, which means that design tweaking and verification at worst-case conditions are tasks that *influence* each other!

So how designers typically solve that problem of finding the worst-case set of corners and variables? Let us do it now at least for the *deterministic* variables like temperature T, supply voltage V_{DD}, and load resistance R_{L}. We also include technology corners like slowNMOSfastPMOS (SF) and fastNMOSfastPMOS (FF), and such process corners are usually predefined by the foundry and cover some worst-case combinations (e.g., regarding CMOS speed, important to avoid timing violations in digital circuits). The user can access them usually by setting a string variable. In opposite to the typical range variables $\boldsymbol{x}_{\mathrm{R}}$ (like T or V_{DD}), they are *discrete*, and often there is no real ordering or ranking on the process corner string variables (e.g., FS vs. SF).

If we have only one variable, then the WC search problem is the problem of a (absolute) minimum or maximum search regarding the performance function

f under inspection. And basic math shows that such extreme value is either at the end points of an interval or at a point with zero derivative—if the relation between parameter x and performance f is continuously differentiable. If the variable x is discrete like a (binary) bus value (e.g., with eight values or for process corners) and if the relationship is highly nonlinear, then we would indeed need to run a full sweep of all values and sort the results to find the most extreme values. However, if f is fully linear, then the WC will be always at an extreme corner, and never somewhere in the middle. For nonlinear cases, we should have at least 3 levels for the variable! The more values we take, the more time we need, but a benefit would be that this way some error estimations become possible.

A further problem is that we usually have to deal with multiple operational range parameters $\boldsymbol{x}_{\mathrm{R}} = (x_{\mathrm{R1}}, x_{\mathrm{R2}} \ldots x_{\mathrm{Rn}})$. A simplified overall worst-case finding algorithm would be to sweep each of the n parameters (so-called factors) alone, look for the individual worst-cases, and then just combine the individual ones to compose the overall worst-case set. This method is called OFAT, one factor at time, and it needs $2n + 1$ combinations to simulate (if we use three values for each variable of the n variables: minimum, nominal, and maximum). As mentioned, unfortunately there is (by far) no guarantee that we will really find the true overall WC with this technique; even if you use more than 3 levels! If the circuit behavior is highly nonlinear and has strong parameter interactions, then OFAT can easily fail! Experiences show that OFAT typically can only find perhaps roughly 50% of the true WC combinations (or e.g. with other combinations, like all extreme pairs, as done in Table 2.1). So more robust algorithms are desirable, especially if you need to treat many corner variables. Of course, almost all these advanced methods unfortunately require some *more* variable combinations to run compared to OFAT. The exhaustive brute-force approach is just running all combinations and is called full-factorial method. It guarantees ending up in the true worst-case (if the sweeps are dense enough), but it is not efficient at all. Typically, you switch back to this only in small corner sets, fast-running testbenches, for a golden sign-off run, if you have a bad feeling regarding some aspects of the design or if you have anyway enough time (e.g., executing an overnight run).

Is WC finding an optimization problem? Mathematically a clear yes, because both are minimization or maximization! Just the goals and variables differ compared to a circuit optimization.

On the other hand, there are quite many differences in the problem characteristics and on how designers treat corners and circuit optimization.

The optimum value of a resistor or a capacitor is often somewhere in the middle of the interval of the possible values, whereas the WC circuit behavior appears much more often at the interval ends, like at maximum temperature or minimum V_{DD}. In both cases, there are of course also counter examples, but these are less likely. For instance, the maximum bandgap voltage occurs usually somewhere at moderate temperatures, or also $\mathrm{L} = \mathrm{L}_{\mathrm{min}}$ is often an optimum value, when only looking for speed.

In addition, it is quite native to treat corners with a quite raw grid, whereas for optimization, e.g., on filters, few percent changes in a parameter are very critical. So often designers just check for three variable values in a corner analysis, but do really dense sweeps for parameter optimizations. A further difference is that the number of corner variables is quite fix and often all combinations have really to be verified, but for optimization the designer can simply decide whether he/she wants to optimize it or not.

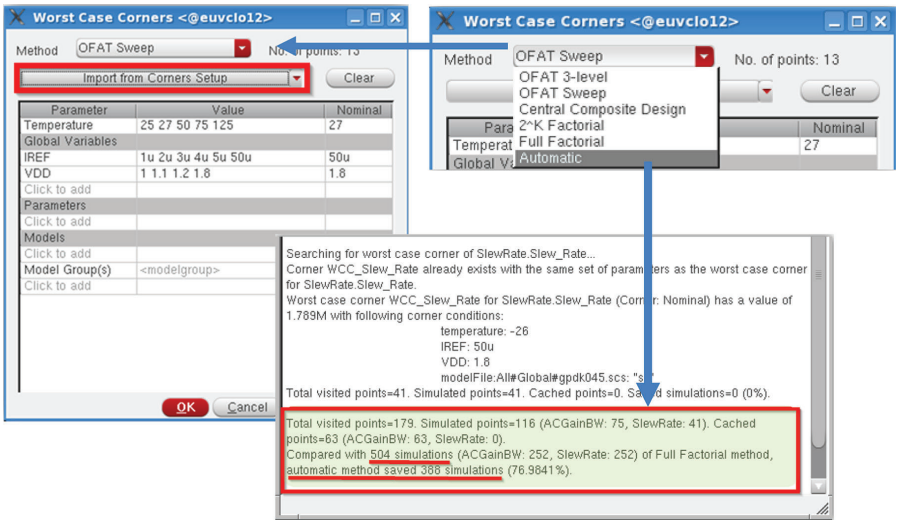
Therefore, also numerically the optimizers for both tasks differ a lot, e.g., WC corner search is focused on quite few discrete variables (like 3 to 10) and on the global extremum for each performance $\mathrm{f_i}$. Actually, in both applications finding the *global* extremum is desirable, but in WC search finding it is even more important.

Circuit optimization, is usually more challenging in other aspects, because we need to treat *all* the different spec-relevant performances *simultaneously*, and often there are much more variables, more correlations, and quite *flat* goal functions (small gradients make the optimum usually harder to find). So, often already finding *some* improvement in overall performance, just finding a better trade-off is enough. More in Chapter 8.

Actually the situation is a bit strange: If f is linear, than OFAT would surely find the WC with linear rising effort $\mathrm{n_{OFAT}} = 2{\cdot}\mathrm{n}{+}1$. However, for the 3-level full factorial corner set (in math, any such combination is a so-called "design") we need $\mathrm{n_{3full-fac}} = 3^{\mathrm{n}}$ ($+1$ point for including the nominal case). So people invented also further fix or adaptive algorithms (see Table 2.1, showing a $3\times$ speed-up of an automatic method against full-factorial [Weber2015]). Those algorithms are typically regarding speed and accuracy essentially in between OFAT and full-factorial (see Figure 2.5).

One example of such a fix algorithm is named 2-level full-factorial, and it uses all possible parameter combinations, but only the most extreme value

Table 2.1 Automatic worst-case corner search for an op-amp [Weber2015]

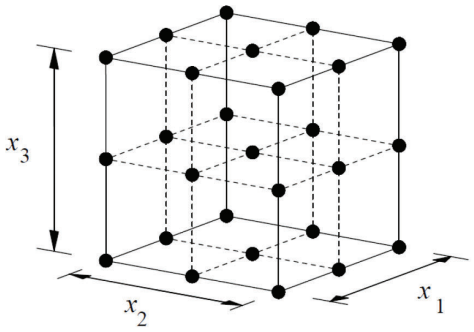


(so we skip the combinations with nominal values). For linear functions, this would still work fine, but *not* for significant second-order nonlinearity (because here the WC is often somewhere in the middle of the interval). Another classical method is called central-composite CC, It is combining the OFAT sweeps e.g., with a 2-level full factorial (or e.g. with other combinations, like all extreme pairs, as done in Table 2.1). Later we will also discuss even more advanced methods, but for now let us check how we can improve the fastest method, OFAT.

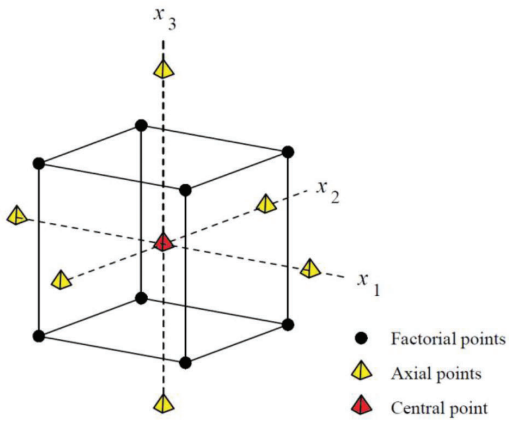
2.4.1 Worst-Case Corner Example and Heuristics

As mentioned, the simple, intuitive OFAT method can easily fail for circuits with strong nonlinearities and/or strong parameter interactions. For complex designs or already for an op-amp, this is not always easy to anticipate from circuit understanding. Luckily, there is one nice example showing some difficulties immediately: Already for a basic CMOS inverter and looking for its delay, you can see one major reason why OFAT fails surprisingly often.

The major corners for inverters are process corners (SS, NN, FF, FS, SF), temperature T, and supply voltage V_{DD}. In most of our cases, designers are intuitively right: Beside maximum load capacitance, the usual WC combination on speed is *maximum* temperature (due to mobility degradation), *minimum* V_{DD}, and of course the *slow* corner SS. However, if V_{DDmin} is really very small,



a) 3-level full-factorial n=3^s=27



b) 3-level central composite (n=2^s+2s+1=15)

Figure 2.6 Visualization of different frequently used corner methods (s = 3 dimensions).

like for hearing aid applications, it could happen that the gate-overdrive V_{DD}-V_{TO}, and thus the speed, becomes very small, especially at low temperatures! This is because $|V_{\mathrm{TO}}|$ has a negative TC, close to the famous –2 mV/K.

Actually, the *same* circuit could have the WC corner combination either at T_{min} (as shown in Figure 2.7) or at T_{max} (as usual for large V_{DD}—so when the upper blue curve is not of interest) just the *ranges* for V_{DD} and T matter mostly (and a bit also the transistor sizes and the technology)!

So let us inspect the more difficult case of a very low minimum V_{DD} and how OFAT would act on this scenario:

Plain OFAT would make three sweeps around the *nominal* values, and we would find that the individual worst-cases are at SS and minimum V_{DD},

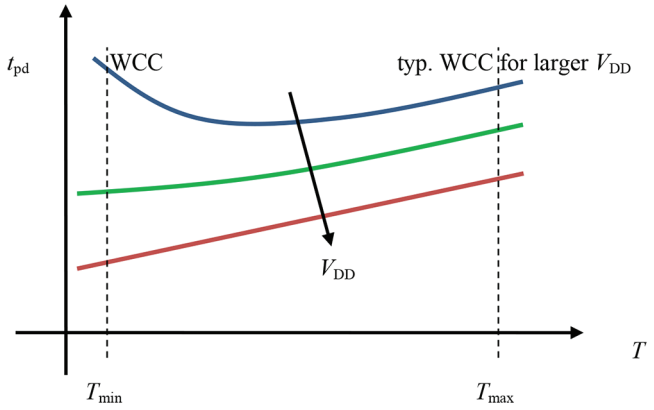


Figure 2.7 CMOS inverter delay versus temperature T for different V_{DD} at SS corner.

but on temperature OFAT would still get maximum T (look at the *middle* green curve of Figure 2.7)! So the *composed* overall WC by OFAT is simply wrong, and only two parameters (V_{DD} and process) are correctly treated! The OFAT-WC on T is wrong due to nonlinearities (in his case mostly quadratic) and correlations (mixed functional terms)! What bothers also is that the WC combination could indeed "jump" in circuit designs, even if the design is robust and meaningful. This is also the case for many other circuits besides CMOS inverters. What helps is that the "jump" is *not* very critical regarding performance f (the y-value might be still quite similar even between T_{min} and T_{max}). To some extent, such difficulties could also happen in other tasks like finding the worst-case for statistical corners or during an optimization.

One question is of course:

> *How would a clever designer address this problem?*

And another one is:

> *Are there clever mathematical algorithms with both: proven efficiency and reliability?*

Which options to improve, e.g., the setup, do we have? EDA tools have usually only access to information about the design if potentially time-consuming simulations have been done, and if the results are available in suitable form. The designer has also his experience and know-how, e.g., from remembering the problems in older often similar circuit designs or from reading the

technology manual. He/she can influence the analysis setup and can exploit this by making an educated guess for the worst-case corners, like:

- Worst-case for leakage current is almost for sure at fast technology corner and high temperatures and supply voltages.
- Lower worst-case on RC filter cutoff frequency is usually given at slow-resistor and slow-capacitor corner.
- Worst-case on CMOS gate delay is at slow MOS corner and minimum supply, and at least usually at maximum temperature. However, we have seen that there is an exception for ultralow supply voltages, because here also minimum temperature can be critical! So just check both of them, which is still better than running all PVT combinations.
- Analog circuits suffer from saturation mostly at minimum supply and slow MOS corner. Again the most critical temperature is often harder to predict (e.g., depending on bias scheme and transistor sizes).
- Of course static performances such as leakage current or DC gain are not impacted by corners from load capacitance or package inductance.
- Worst-case on noise is usually at high temperature and sensitivities on supply or load are usually small.

Following these assumptions may come with some risks, and by far, not in all cases, you can completely determine the full WC combination by experience, but often you can reduce the simulation effort significantly, e.g., ending up in a short sweep instead of running many combinations. Table 2.2. gives some more examples on typical worst-case corners; you should inspect them early in the design phase.

2.4.2 Advanced OFAT Methods

There are many options to improve searches, e.g., exploring the space adaptively or exploiting *a-priori* knowledge. Just making an educated guess for the WC corners and simply taking them is (very) risky. However, we can also try to find out more general concepts of taking designer know-how into account. One simple method for combining *a-priori* knowledge with a mathematical algorithm is this: A WC corner search is typically faster if you have a good starting point. So OFAT around the nominal conditions has usually a bigger chance to fail than doing OFAT around the "expected *a priori* estimate" worst-case! For the ultra-low supply CMOS inverter, plain OFAT can *fail* as we have already seen, but let us inspect what will happen if we "expect" the WC is at minV_{DD}, process at SS, and e.g., maximum T (which is wrong for the ultra-low V_{DD} application!). We would indeed find in the sweeps around this "expected WC" the correct over-all WC! The formerly critical sweep on T would be

Table 2.2 Typical overall WC corner combinations for some circuit classes

Circuit	Performance	Range Parameters	Process/ Statistical Parameters	Comments
CMOS inverter	Delay	minV_{DD}, maxT or minT	SS	Might be difficult to find by standard OFAT
	Leakage current	maxV_{DD}, maxT	FF	Easy to find or anticipate
	Dynamic current	maxV_{DD}, minT	FF	Easy to find or anticipate
	Threshold	Depends on application	SF, FS	Easy to overlook the mixed corners
Op-amp	Speed	lowBias	SS	
	Stability		FF	Often load impedance is critical too, but it is hard to anticipate the WC. Low temperatures critical for constant current biasing.
	Supply current	highT, highBias	FF	
	Offset	highT		Often the offset increases with T
	DC gain	highT		g_m drops via T, and with PTAT bias you often get no 100% compensation
	Noise	highT	SS	FF might be critical regarding current noise
	Saturation	minV_{DD}	SS	WC for T is hard to predict
Wide-band amplifiers (resistive load)	DC gain	highT	SS + lowR	Due to A = g_mR. In CML gates, this might be critical too
	BW	lowBias	SS, highC, highR	Quite obvious, but stability problems can cause deviations
	Distortion	lowT, lowV_{DD}	SS, lowT, lowR	The WC may differ for different distortion mechnisms.
LNA	Supply current	highT	lowR	If PTAT biasing
	Gain	minV_{DD}, highT	SS	
	IP3, P1dB	minV_{DD}		For PTAT bias lowT gives often bad linearity
	NF	maxT	SS	
	Stability	maxV_{DD}		Hard to predict further

now at minV_{DD} and SS (instead of nom$\mathrm{V_{DD}}$ and TT), and we would nicely find the *correct* worst-case temperature, with even *no more* simulations to execute.

Another very fast OFAT variant would be to do OFAT, but to start in the direction of the expected WC; if the expected WC is indeed worse than the starting point (like nominal conditions), there is usually no need to run the opposite OFAT points, because the later would at least typically give the best case, not the worst-case! Such "OFAT with shortcuts" would give a speed between pure guessing and normal OFAT, but it would be not *much* more accurate than OFAT, and so it would be still risky (e.g., if strong second-order nonlinearities are present).

A good variant that *reduces* the OFAT risk is doing a "preordered stepwise OFAT." Let us try to solve the CMOS inverter WC problems manually: As mentioned, the delay WC on V_{DD} and process is almost trivial (even OFAT would have no problems!), but the temperature characteristic is (quite often) more difficult, and here, two effects (TC of mobility and TC threshold voltage) can "reverse" the overall sensitivity; so we have a sign change of the TC, resulting in a strong quadratic behavior. We can modify OFAT further: we should start with low-sensitivity and almost linearly behaving variables, because here the WC is almost trivial, i.e. we should run a sweep on V_{DD}, finding min V_{DD} as WC. Then, we should take this setting and run the sweep on process corners, finding SS as most critical. Last we should sweep T with $V_{\mathrm{DD}} = V_{\mathrm{DDmin}}$ and process corner set to SS. We will find the correct WC on T and also the overall WC as desired!

The consequence for the general case is that OFAT is more successful if we do it stepwise, with using the currently found WC combination, and we should sweep difficult variables in late sweeps!

Actually, the stepwise OFAT is exactly what designers do in the circuit construction phase, and they just follow the golden rule of only changing one parameter at the time; if the change was good (i.e., we moved indeed a bit to the worst-case), then keep it, think a bit, and consider the next improvement step! It would be usually no good idea to consider the next improvement, but not taking already the first improvement!

Note: The resulting algorithm works like a very simple coordinate-based optimizer, and it would be only a local optimization algorithm. Keep this only in mind, we discuss optimization in Chapter 8 in much more detail.

A small disadvantage of these enhanced OFAT methods is that the setup effort is a bit higher, because the user needs to decide *upfront* which variables are critical (large sensitivities, high nonlinearity, strong correlations) and/or

what the expected WC corner combinations are. In addition, if we use "too" clever shortcuts, we may take too much risk. An extreme speed-up example would be making a step Δx and using it immediately, just if it makes indeed the behavior worse. Of course, it is more fool-proof to double-check with an opposite step like $-\Delta$x, although it might be "waste" of time in quite many cases. Better avoid risky assumptions in circuit verification. In statistics, one example for a large-risk method is using the C_{PK} without checking for normality. Also in corner analysis there are several well-known critical cases:

- Actually, it is quite typical that temperature is often the most critical parameter, especially for circuits which should work over a big temperature range (like automotive designs). This is also because circuit designers apply usually some "tricks" to make the circuit robust against temperature changes, e.g., by implementing a clever bias scheme (bandgaps, PTAT current generators, replica circuits, resistors with opposite TC, etc.). This often ends up in nonlinear behavior, like the well-known quadratic bandgap voltage characteristic.
- Another difficult parameter might be load impedance, e.g., an op-amp might be (quite) stable for both very small and very big load capacitances, but truly unstable for moderately large capacitances, and this characteristic might be also related to other parameters like load current or the ESR of the output capacitor.
- A third example could be the gain setting parameter in a variable-gain amplifier VGA; if the circuit is tricky, maybe the WC (e.g., on third-order intermodulation point IP3) is at an intermediate position where you may not expect it. Such difficult variables should be treated with quite dense sweeps. For an ordered OFAT such variables should be treated in late sweeps, because they decide mostly on the overall WC, whereas the more linear or less sensitive variables should be set earlier to the WC search.

Of course, if the designer would know that his design is indeed brand-new or difficult, he/she would go indeed for a full-factorial analysis, and by sorting the results table, you can manually find all WC combinations correctly. Unfortunately, for many corner variables, this is not efficient and *much* slower than OFAT.

2.4.3 Advanced Fitting Methods and Adaptive Search Methods

Fix sampling schemes can provide only a certain compromise between speed and accuracy, and one way to improve was to include *a-priori* know-how.

There are some similarities to fully *adaptive* algorithms; e.g., we could just run OFAT and use the obtained information as knowledge to decide on the next sampling points to be simulated. Actually our *stepwise* OFAT is already adaptive, but we can do even better, and such advanced adaptive methods have been implemented very successfully in several custom-design environments already.

Figure 2.8 gives an overview on different worst-case corner search methods. Note that we just show the *typical* behaviors. The enhanced OFAT methods are not shown there to provide a better overview: they could maintain or even exceed the OFAT speed, in addition, due to the *a-priori* know-how reflected in the setup, the *risk* compared to pure OFAT can be even reduced significantly (*if* the setup is done well).

All methods have in common that designers can get rid of boring repetitive work! You need specs, and you just have to define the variables and their values (like in a normal corner analysis) and press the run button!

Automatic methods usually offer a very *good* compromise between speed and accuracy. They usually start with the OFAT method to identify the most important variables; and then more detailed search and space exploration methods are used, e.g., with the inclusion of correlations on these more important variables. Internally, a kind of model-based search algorithm is performed, e.g., from OFAT results, we can create a linear model.

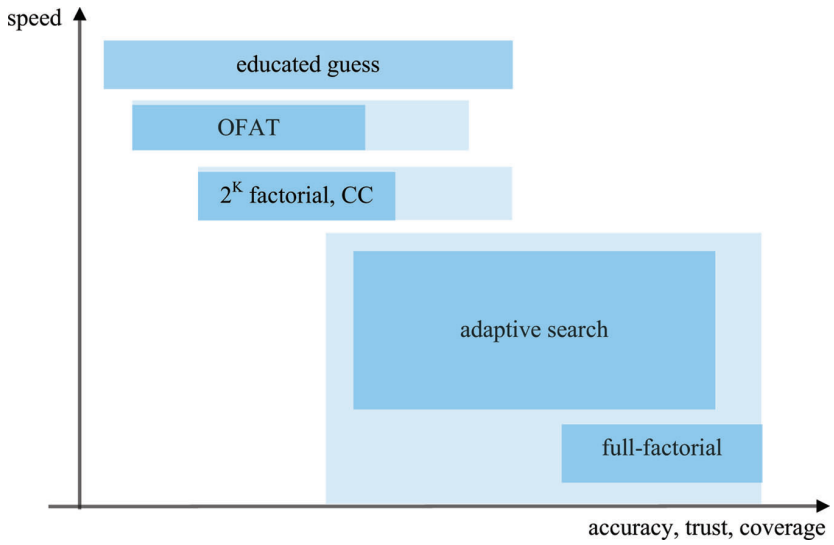


Figure 2.8 Comparison of some important worst-case corner finding methods.

If you think of the next better method beyond linear interpolation and extrapolation; then probably splines are a good choice. Splines also act piecewise, so if we make a local change (e.g., adding new simulation points), only a local recalculation is needed. Splines also "behave" better than high-order polynomials; the later tend to generate severe "oscillations" (even much more than cubic splines, see Figure 2.10a). However, interestingly also somewhat even better methods are available! Indeed, splines very popular and easy to calculate for 1D to 3D cases, but at *higher* dimensions (so more corner variables) several positive characteristics get lost [Neamtu2001]. Look at Figure 2.9 (from [Erikson 2012]) for a difficult 2D application of so-called thin-plate splines. Here the "overshooting" problem is so severe that the spline fit is not bijective anymore.

With splines we assume that a low-order polynomials fits well to data, and we e.g., force certain continuity characteristics (like continuity in f and first derivative f'). This way different splines can be defined, like cubic splines or Akima splines. Even smoothing splines, which can fit to noisy, statistical data, can be used. However, there is actually little *theoretical* foundation for the use of splines, and on several functions we can outperform splines. At the truly *simulated* points (and using a good model, e.g., splines, high-order polynomials [Daems2002] or so-called Gaussian process models (GPM) [Jones2001, McConaghy]), we can make the fitting error ε arbitrary small, but in the "simulation gaps", any model is surely less accurate. So having not only a fit, but also an error indication (a kind of confidence interval, see Chapter 3) would be a clear advantage.

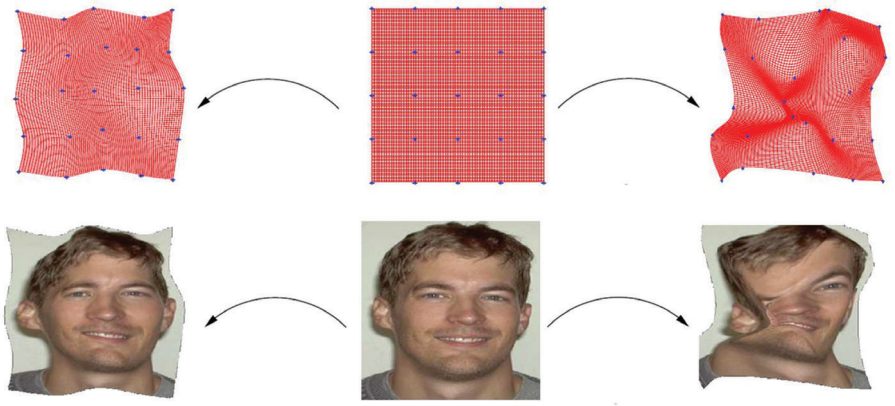


Figure 2.9 Bijective and non-bijective result from spline fitting [Erikson 2012].

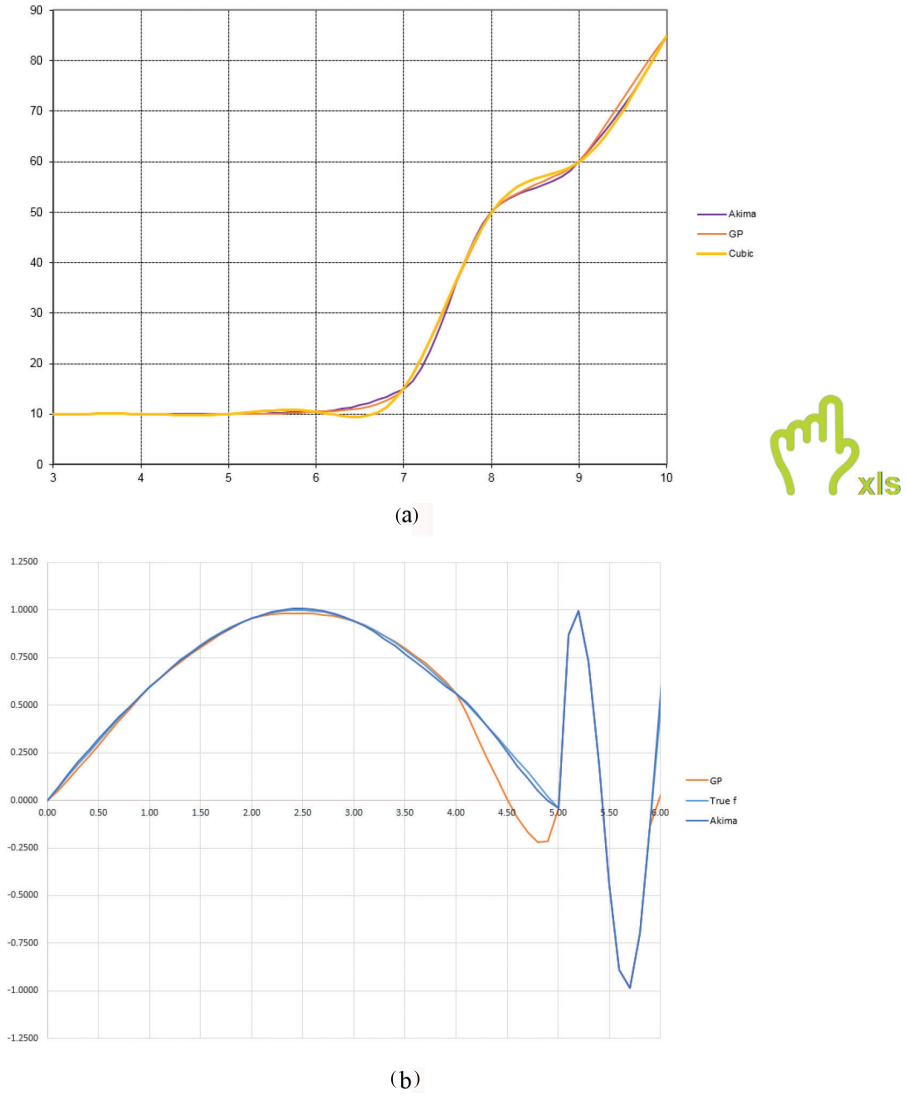


Figure 2.10 Comparison different fitting methods on a nonlinear ramp function (data sampled with $\Delta x = 1$) and for a sine function of two different frequencies (10 samples per period).

Note: WC search has aspects of optimization, interpolation and model fitting, but actually also to *learning*! The available simulation points are a kind of training for our model generation, and when we run further simulations we actually test the model, and our learning success.

Splines use low-order polynoms, but what is a Gaussian process model GPM? GPMs use no piece-wise approach, but we try to maintain the advantages of acting quite local at least to some degree. And this near-local behavior, using the Gaussian bell-shape function e^{-x^2}, allows also fits to almost any data. GPM are also very flexible in other aspects, e.g., there is no need that the sampling points are on a certain grid; even random placements would be allowed.

$$f_{GPM}(\mathbf{X}) = a(\mathbf{X}) + \sum b_i \cdot \exp(-|x - x_i|/c_i)^2 \tag{2.1}$$

Note: In the statistical part of the book we come back to Gaussian distributions in more detail. Here this formula is currently not much more than just a mathematical attempt, just an approach: We start with it, and are just happy that it works quite fine, like splines. For now there is not much to understand, which will become quite different in statistics!

The 2nd term in Equation (2.1) is the Gaussian part, and because the Gaussian pdf diminishes for large Δx, we can make the local approximation at x_i very good, without impacting much the fit at the other sampling points, and we generate no oscillations (as high-order polynomial would do). For more flexibility often the exponent is not set to $p = 2$ but is allowed to be fitted within 1 and 2. The function e^{-jxj} is a peaky function; and this will also allow fits to true peaks; this would be much harder for splines. Also the first part $m(\mathbf{X})$ might be different in different GPMs, e.g., it could be a constant or e.g., a linear function. One nice GPM feature is that these little Gaussian peaks look a bit like a landscape with little mountains. And knowing the height h at one point x_i obviously helps us to make estimations for the height in the neighborhood; and the correlation between (x_i) to $h(x_i + \Delta x)$ is obviously the higher the smaller $|\Delta x|$ is. And this is also the case for our Gaussian functions. Interestingly, the whole idea has been created for "earth modeling", and these and similar methods are called *kriging*.

One simple model fit verification method is cross-validation by splitting the data into two equal parts and double check the estimates ($f_{fit1}(\mathbf{X}_i)$ versus $f_{fit2}(\mathbf{X}_i)$). An often used variant is "leave-out-one": We can fit a model to n points, and then we compare the model predictions from this full model to models just using one point less. This way an error estimation is possible, actually for all fitting methods. With such an error indication, e.g., an adaptive auto-stop for the WC corner search can be implemented, so that the worst-case combinations can be found in a reliable *and* quite fast way. Gaussian process models are nowadays often preferred, because for statistical applications they have a certain foundation, and offer basic built-in error estimation

capabilities. The latter is desirable, because cross-validation and "leave-out-one" can be time-consuming and surprisingly inaccurate (see also Chapter 6 on bootstrap).

However, look up! How can *statistics* help in computer *corner* simulations? In WC corner search we have the situation, that indeed our simulated points are fully reproducible, so there are almost no further (statistical) errors on top (in opposite to lab measurements)! Here the GPM advantage regarding better error estimation would not matter so much! However, in the corner simulation *gaps* there is a "risk", a model error, which we should limit; and that makes the good GPM error estimation also for corner analysis useful, although such error estimate is not the same as the true error.

Actually the situation is tricky: In a normal least-square fit (like linear regression, see Chapter 5), we have a unknown errors, and we assume that typically the error is everywhere (independent on $\mathbf{X}$), with the same *distribution* and *standard deviation*, but here in corner performance modeling, we have an error $\varepsilon = \varepsilon(\mathbf{X})$, which is zero for all $\mathbf{X} = \mathbf{X}_i$ (ith simulated point = sampling value).

So to some degree it is often a matter of taste to use GPM or splines, in many cases the results will be very similar (see Figure 2.11). Just having a solid concept with GPM, and at least the option to include statistical errors, is a certain advantage. For instance, you can see in Figure 2.10b that the GPM fit is not perfect, actually the Akima spline works better here. This is because the hyper parameter p is set to 1.5 as in the other examples, but due to the sharp edge (at x = 5, when the sine wave frequency is increased by 10×), e.g., p = 1.1 would give a much better fit. Using GPM and solid estimation methods such "mistakes" are very rare, and actually not necessary. In addition, our 1D example is not 100% representative for complex situations; in many dimensions all methods become more difficult, but spline fitting degrade faster regarding calculation time and accuracy than GPMs! That is an important factor when you need to treat 3 to 20 corner variables.

It is often said that GPM come *without* prerequisites, but actually this is not completely true, usually some internal so-called hyper parameters have to be tweak for an optimum fit (in advanced GPMs it is not only the exponent p). However, indeed the GPM assumptions are very mild ones, i.e. just having (tweaked) Gaussian error distributions; so there are usually applicable with very low risks.

With splines we more or less let the data "speak", and we "only" *interpolate* when we talk about range parameters $\boldsymbol{x}_R$. In this aspect GPM is very similar, and therefore both work quite well in the circuit design context! This is no big surprise, because many similar algorithms have been even developed

Samples

x	f(x)
0.00	0.00
1.00	0.62
2.00	0.97
3.00	0.91
4.00	0.46
5.00	-0.19
6.00	-0.76
7.00	-1.00
8.00	-0.81
9.00	-0.28
10.00	0.37

Interpolation from data

x	Kriging, Beta = 1.5	Akima	f	Taylor3rd	CubSpline
0.00	**0.0000**	**0.0000**	**0**	**0**	0
0.10	0.0528	0.0735	0.066617	0.066617	0.066542
0.20	0.1119	0.1439	0.132939	0.132938	0.132799
0.30	0.1741	0.2115	0.198669	0.198667	0.198485
0.40	0.2380	0.2763	0.263517	0.263506	0.263317
0.50	0.3029	0.3386	0.327195	0.32716	0.327007
0.60	0.3682	0.3986	0.389418	0.389333	0.389272
0.70	0.4332	0.4563	0.449912	0.449728	0.449826
0.80	0.4973	0.5121	0.508407	0.508049	0.508383
0.90	0.5597	0.5661	0.564642	0.564	0.56466
1.00	**0.6184**	**0.6184**	**0.61837**	**0.617284**	0.61837
1.10	0.6665	0.6684	0.66935	0.667605	0.669249
1.20	0.7105	0.7152	0.717356	0.714667	0.717113
1.30	0.7518	0.7588	0.762175	0.758173	0.7618
1.40	0.7909	0.7993	0.803608	0.797827	0.803146
1.50	0.8278	0.8364	0.841471	0.833333	0.840988
1.60	0.8625	0.8703	0.875595	0.864395	0.875164
1.70	0.8948	0.9008	0.905829	0.890716	0.905509
1.80	0.9244	0.9280	0.932039	0.912	0.931862
1.90	0.9507	0.9517	0.954108	0.927951	0.95406
2.00	**0.9719**	**0.9719**	**0.971938**	**0.938272**	0.971938
2.10	0.9811	0.9881	0.98545	0.942667	0.985366
2.20	0.9852	0.9996	0.994583	0.94084	0.994342
2.30	0.9858	1.0062	0.999298	0.932494	0.998893
2.40	0.9834	1.0080	0.999574	0.917333	0.999048
2.50	0.9784	1.0047	0.995408	0.895062	0.994838
2.60	0.9707	0.9963	0.98682	0.865383	0.986289
2.70	0.9604	0.9826	0.973848	0.828	0.973432
2.80	0.9472	0.9636	0.956549	0.782617	0.956295
2.90	0.9307	0.9392	0.935	0.728938	0.934907
3.00	0.9093	0.9093	0.909297	0.666667	0.909297
3.10	0.8766	0.8756	0.879555	0.595506	0.879524
3.20	0.8393	0.8400	0.845905	0.51516	0.845766
3.30	0.7988	0.8022	0.808496	0.425333	0.808231
3.40	0.7559	0.7620	0.767496	0.325728	0.767129

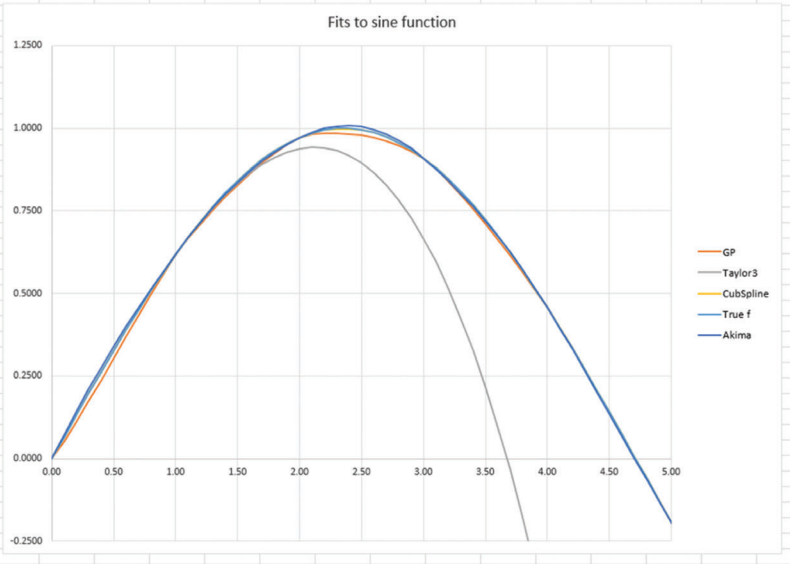


Figure 2.11 Comparison different fitting methods on the sine function.

since years for problems which are even *more* difficult than most IC blocks, e.g., for geostatistical applications or earth modeling. Almost any kind of nonlinearity and number of parameters can be treated with such (almost) non-parametric methods.

So-called kernel density estimation (KDE) is a further example for such a "parameter-free" method (see Chapter 3), but actually any method comes at least with some assumptions. One tricky part KDE is the "bandwidth" setting, the setting on how smooth the fitting result should be; and for GPM there are similar issues. In low dimensions, GPMs are harder to calculate than splines, but for more complex corner setups this situation reverses. Generally runtime is not a big issue anymore; and usually the circuit simulation times are still much larger. Several benchmarks are available on different GPM algorithms, in [Guerra-Gomez2015] you can even find a circuit-related benchmark. In the examples, the rms error was in the order of 2%–10% for the better modeling algorithms. This sounds good, because the rms error at the training points would be even almost zero, so we talk about the error in the testing points. On the other hand, the local peak errors might be much larger. In GPM we can at best say that such cases are very rare, statistically. Typically the model *creation* times are in the order of seconds to minutes, whereas the model *evaluation* times (just executing Equation (2.1)) are much shorter.

In theory, you can always construct extremely difficult cases where only full-factorial would be reliable, but modern mathematical algorithms almost act with almost proven efficiency and high reliability; actually, as these methods can detect the internal errors (Figure 2.12), they will simply switch back to full-factorial in such rare cases, but keeping good speed in most real cases!

Note: The error estimate $\varepsilon(x)$ is zero at the sampling points x_i, and ε is quite large in regions with low sampling density. In addition, ε is rising quickly in *extrapolation* regions, so GPM's offer directly what designers would also assume. To some degree, the GPM error estimation is similar to the error estimation with Taylor polynomials, so it is not really magic that such error estimations are possible. If the data itself has uncertainties (right Figure 2.12), its variance V can be just add to the model tolerance ε^2.

What does this all mean to circuit design? Indeed, such adaptive "pure mathematical" algorithms are in competition to clever "heuristic" methods, like expected WC or parameter rankings to the setup and following a certain "strategy". As usual, with a clever testbench setup, more speed is possible too, e.g., if we divide our tests into fast and time-consuming ones, we could run

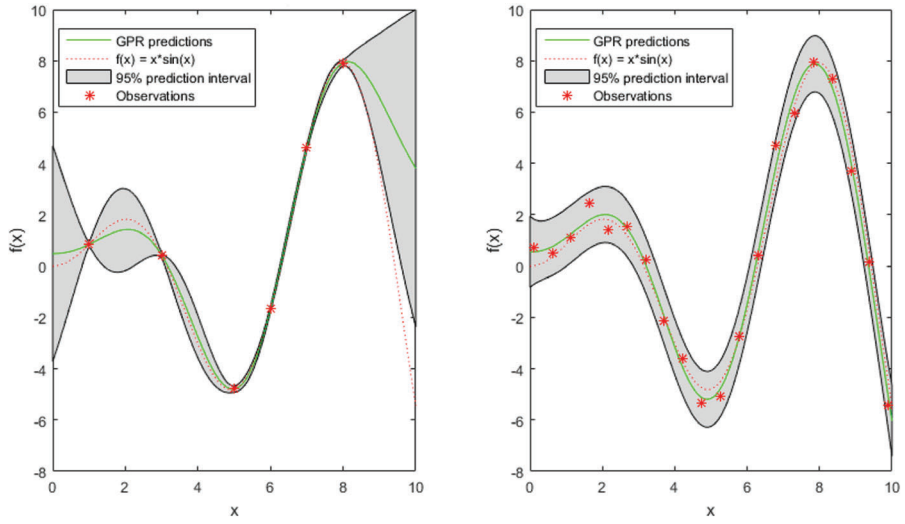


Figure 2.12 GPM applied to non-noisy and non-noisy data (Matlab® tutorial on GPM) [Matlab].

a fast method for the slow tests, but use a more intensive search method for the fast ones. However, in conclusion, the WC corner searching problem in $\boldsymbol{x}_{\mathrm{R}} = [\boldsymbol{x}_{\mathrm{Rmin}}, \boldsymbol{x}_{\mathrm{Rmax}}]$ can be regarded as almost solved. Doing it manually can be a time-consuming and boring work, and unfortunately, it is needed from time to time after circuit changes or for testbench extensions. So let the computer do it for you! In [Woods2015] you can find a further benchmark on mathematical examples, and (more important) detailed descriptions for combining sampling methods with parameter screening and modeling techniques.

As mentioned, finding the worst-case among statistical variables $\boldsymbol{x}_{\mathrm{S}}$ for given yield is important as well, but it is significantly more difficult. So let us soon inspect the statistical behavior of designs in more detail (Chapters 3 to 7). In Chapter 9, we will come back to the worst-case topic again, when we combine it with optimization techniques. Some basic local optimization techniques look similar to our different OFAT variants, whereas global optimizers often have elements of a full factorial analysis, but for efficient circuit optimization we will improve the algorithms further.

Too much praise for mathematical WCD search methods? Check out our little men versus machine showdown in subsection 2.8.2.

2.5 Monte Carlo and Mismatch

Even many corner simulations do not cover the full worst-case condition and also give usually no good yield indication. This is because there are not only variations in *global* parameters (like temperature or R_{sheet}), but also parameters could vary from *instance to instance* (like the two transistors in a differential pair, e.g., on threshold voltage). In digital designs with older technologies, typically the process variations dominate, but in analog designs and when using ultra-deep submicron technologies, also the mismatch becomes very critical. Due to this and lower supply voltages, all kinds of circuit design become more and more difficult. *Even* for a perfect layout, you have to accept a certain *mismatch*, unfortunately; actually, the statistical models even assume that you indeed strictly follow best-practice layout rules [Hastings]. Physical reasons for such statistical variations are variations in channel doping concentration, gate oxide thickness, or line roughness.

Putting all global and all instance parameters together, you often have thousands or more parameters in a single real-world block. Simulating all parameter combinations is usually simply impossible, so a 100% verification becomes extremely hard. Usually, statistical methods jump in here! The easiest and also most general form of statistical analysis is a Monte Carlo simulation. MC requires a statistical model for each variable, and based on that, random samples will be created and each parameter set will be simulated. Actually, this is similar to a corner simulation, just the parameter changes are now random in MC. MC is one method to capture the performance dispersions from statistical parameter variations. One key problem with MC is that all results depend on chance, so finding the true parameters, like the sigma σ of a normal Gaussian distribution from MC data, can be tricky. A typical situation comparing *multiple* MC runs is shown in Figure 2.13; for a *large* MC count, we can expect that most MC results will be in a very *small* tolerance region, hopefully close to the true value. However, for lower MC counts, the variations are wider, so our estimation is more uncertain. On top of that there is even no guarantee at all that the average from MC runs with a moderate count is really identical to the one taken from a huge MC runs! There could be a general bias and a bias for finite samples. A famous example (checkout Google) is that $1/(n-1)\sum(x-x_{\mathrm{m}})^2$ (with sample mean x_{m}) is a so-called unbiased estimator for the variance V, but actually $\sqrt{(1/(n-1)\sum(x-x_{\mathrm{m}})^2)}$ is no such "beautiful" unbiased estimator for standard deviation σ! In addition, it is seldom that your distributions look so nicely bell-shaped and symmetrical; they could be even discrete and highly asymmetric!

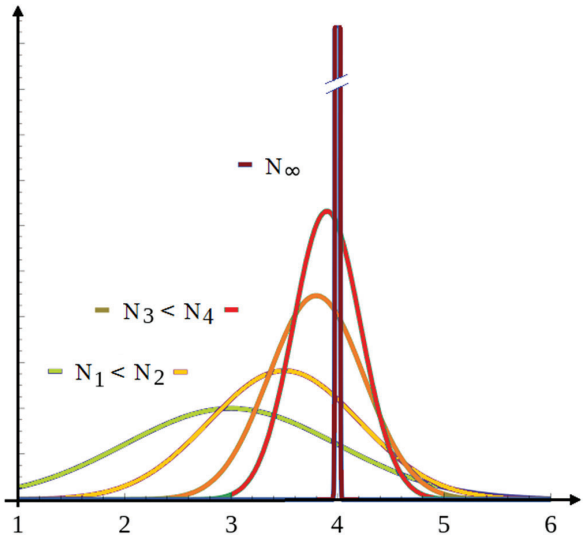


Figure 2.13 Typical behavior of MC estimates for different MC counts *N*.

For luck, as engineer, you can often relax and accept an error of 10% in sigma (like you do not care much whether the offset standard deviation is 5 mV or 5.5 mV); often such errors are on that level already for moderate MC counts (like n = 100). However, there are several examples where you need a much higher precision (like for verifying a yield loss on 0.01%) and/or even with 1000 samples, the bias is well above several percentages. In Chapter 3, we care more about the foundations of Monte Carlo and statistics.

Both process and mismatch parameters are basically statistical parameters $\boldsymbol{x}_{\mathrm{S}}$. And the verification has to run on these plus the usual operational range parameters $\boldsymbol{x}_{\mathrm{R}}$ (like temperature and supply voltage). However, in older IC technologies, usually the technology parameters are often split into process parameters treated by corners only, and mismatch parameters treated by MC. So to limit the modeling effort and to speed-up the verification, the foundries supplied full technology corner sets like "nominal," "fast," "slow," or "slowNMOS-fastPMOS" to address the worst-case on global process parameters. For this reason, it would be (in principle) enough to run a PVT corner analysis and a mismatch-only MC analysis to take mismatch into account. Luckily, mismatch is often quite independent from PVT parameters, so often it is good enough if you run the MC mismatch analysis at nominal PVT conditions.

A more severe problem is that usually the foundry-provided process corners often do not fit well for analog designs. In fact, they are only composed to address the worst-case speed for CMOS logic cells! From a foundry, you will simply never get a worst-case process set for IP3 or noise figure or phase margin! Even the meaning of FF or SS for analog circuit is not fully clear, e.g., fast NMOS transistors have typically a lower threshold and thinner gate oxide (leading to larger I_{DSAT}), but the thinner oxide would also lead to a larger gate capacitance C if we use such NMOS as a filter capacitor. However, *larger* capacitance means *lower* bandwidth and thus *slow*—but the corner is named "fast"! Similar problems arise in many kinds of circuits, even in logic circuits. In current-mode logic (CML), we use NMOS differential pairs and resistive loads, and the worst-case on gain $g_{\mathrm{m}}R_{\mathrm{L}}$ is for slowNMOS and fastR, so a foundry-provided "worst-case" corner set with fast transistors and fast resistors would not hit the gain worst-case. So you may create your own "expected analog" critical corners to improve the situation, but whatever you do, you would need (quite many) more corners, so (much) more verification time.

In addition, you have to check to which yield such process corners are related, and it could happen that the corners are set to a 5σ limit, but in your design, you may aim for 3σ yield. In such cases, a PVT simulation would stress your design too much, and in other cases, the PVT simulation will still show less variations than the real production!

A solution is only possible if you have access to full *statistical models* for both process and mismatch, as in all modern PDKs. This would enable you to find the correct statistical worst-case sets for *your* circuit and for *your* performances of interest in your operating range!

This way statistical techniques enable full-yield verification, but one pity remains: MC results depend on chance, so in addition to model errors and numerical inaccuracies, we also have to deal with a statistical sampling error. MC has the advantage that statistical variance becomes smaller and usually acceptable if you just increase the MC sample count enough (e.g., to 1,000), and it can provide a direct yield estimate. The pity is that sometimes you may really need quite a large MC count, so a lot of time.

For luck, MC is by far not the only technique to evaluate the statistical behavior of a design. In the remaining chapters, we also talk about more advanced, more complex methods. For special cases, like pure DC analysis, advanced simulators with built-in methods to address mismatch can be used. They work similar to noise analysis or statistical hand calculations – in both

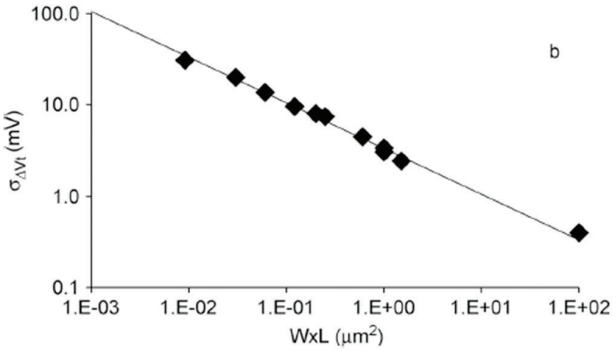


Figure 2.14 Pelgrom's scatter plot obtained from measurements in 90 nm [Tuinhout].

the individual effects add up quadratically, for noise you work with the spectral noise densities and for statistics you work with the standard deviations e.g., described by Pelgrom's area law for mismatch. Such analysis is typically much faster, but often less accurate compared to a full MC analysis.

Pelgrom's law (Figure 2.14):

$$sV_T = k/\sqrt{A}$$
$$\text{with mismatch constant } k[Vm] \tag{2.2}$$

Example: Mismatch calculation by hand

To fit an analog circuit into the supply rails, you need to keep an eye on the MOS transistor saturation voltages. To achieve a certain V_{DSAT} for a given technology and current, you may need a certain minimum W/L ratio of r $= W/L \geq 10$, so $W = 10 \cdot L$ at the limit case. The question is still how large should be *W* and *L*, and often you can obtain both according to the desired threshold voltage accuracy like $\sigma V_{\mathrm{T}} \leq 10$ mV and this is related to the device area $A = W \cdot L$ and to the process and device-specific mismatch constant *k* (e.g., it is highly impacted by the oxide thickness). A typical value is $k = 5$ mVum (quite valid for 180 nm CMOS, modern processes are better due to lower gate oxide thickness, e.g., giving 2.5 mVum, but older processes like 1 um CMOS are worse, e.g., showing 20 mVum). So for $A = 1$ um^2, we get $\sigma V_{\mathrm{TO}} = 5$ mV, and with $A_{\mathrm{min}} = 0.25$ um$^2 = W \cdot L = 10 \cdot L^2$, we would be at the spec limit of 10 mV; and we can now calculate the desired $L_{\mathrm{opt}} = \sqrt{(A_{\mathrm{min}}/10)} = 0.158$ um and $W_{\mathrm{opt}} = 10\, L_{\mathrm{opt}}$.

Note: It looks that already a small MOS transistor can give a good matching, but actually multiple transistors can have an impact on the overall mismatch, and also your spec limit on offset voltage is usually not just one sigma, but for instance 5σ—all this leads quickly to quite large transistors. The consequence? The very newest technologies offer extremely small transistors; this which might be translated to big area savings, but due to the matching problems this advantage is hard to realize in analog blocks. This does not mean that no area improvements are possible anymore, but design becomes harder and "going" digital or mixed-signal is more and more a native choice.

How about finding statistical WC corners from MC? What we can do in several design environments is "picking" a certain MC sample like the most extreme point in a MC analysis (e.g., on offset or supply current) and using it as an *approximate* statistical WC corner. Next, we can even put them into the set of WC corners on range parameters $\boldsymbol{x}_{\mathrm{R}}$ to form a set of approximated WC overall corners. In Chapter 3, we explain MC in much more detail, and in Chapter 7, we put several techniques together to be able to effectively find statistical worst-case corners with well-defined sigma level with good accuracy, so-called worst-case distances (WCD).

Note: If we pick, e.g., after a 100-point MC run, just the worst sample, then such sample might be equivalent to maybe 2.8σ or 3.1σ, so it would depend pretty much on chance. And the chance to get a 5σ sample is very low. To get such extreme sample, we may need millions of points, so pure MC is not efficient for this task.

Mismatch on single transistor or pairs? To measure the mismatch in the laboratory, it is good to use transistor *pairs*, because using pairs the global variations and temperature gradients will cancel out. It is best to use many such pairs for a stable statistic. So many people think mismatch simulation has also to be done on pairs or you even need to "define" pairs of transistor instances. This is not really true! Also for one *single* transistor, you can have a certain mismatch like $\sigma(V_{\mathrm{TO}}) = 10\,\mathrm{mV}$. Having two such transistors in a differential pair gives a total sigma that is $\sqrt{2}$ higher, so you have to look up carefully when looking for the concrete values. Almost all tools work with instance-specific mismatch constants, but in your circuit the measured mismatch for a certain output may depend on multiple devices.

2.6 Moving to a Robust Circuit Design

Once you found the overall worst-case on deterministic and statistical variables—with manual methods you typically can only approximately reach it—you can be in different situations like the design passes all specs even at all worst-cases or not. In the latter case, the design should be improved, but how can we do it? You need to minimize the sensitivities to many parameters by carefully choosing the circuit topology and by carefully sizing the circuit. Many techniques can and should be applied in general, not only if the specs are very tight.

From the circuit viewpoint, these are the major techniques for a robust design:

1. Avoid dependence on absolute parameters, e.g., by the use of current mirrors, differential pairs, and feedback.
   Choose suited topology, reduce systematic errors, increase loop gain, etc.
2. Reduce the variations caused by mismatch
   e.g., by increasing the component sizes and by applying dynamic matching techniques.
3. Apply calibration techniques
   e.g., using replica circuits in the bias part and putting VCO in a PLL loop.
4. Reduce variations
   by adding a voltage regulator, by using special bias schemes (e.g., PTAT bias gives near-constant g_m) or by the addition of cascade stages (for better PSRR or CMRR), etc.
5. Use of components with higher stability
   e.g., use certain resistor combinations to cancel TC and reduce statistical variations and use external references (like crystals or high-accuracy resistors).
6. Relax block specs, which may lead to more difficult specs in other blocks
   e.g., often a function can be easily implemented with spending more supply current.
7. To optimize the yield, you should "center" the design
   e.g., try to tweak the design so that spec violations appear with similar probability for competing specs.
8. Even in case of spec failures, the design should be still functional, and small changes in parameters (like temperature) should only cause small variations in circuit behavior (like gain).

Use less risky, well-proven circuits, and avoid "dirty" tricks (like relying on second-order effects as the TC of a certain resistor type).

9. Of course also digital designers have their tricks, like the use of dynamic voltage and frequency scaling, dynamic body bias, and using redundant circuits.

Obviously, some design techniques can be automated by using parameter optimizers (e.g., let the optimizer tweak W & L to find a combination giving a good compromise on speed and matching), whereas others (like using clever mixed-signal techniques) are much more difficult to automate. State-of-the-art EDA tools are suited for parameter optimizations, but true topology optimization is still at an academic level and limited to quite simple blocks.

Fighting Mismatch? The Pelgrom's law puts quite strong constraints on many analog designs. For instance, in a flash ADC, the input capacitance is a critical factor, because 2^{n-1} comparators are connected in parallel at the input. So getting one bit more requires $2\times$ smaller offset voltages, and this leads to $4\times$ more area. For the same speed, we end up in (roughly) $4\times$ more power! What can we do in such situations? And how else can we fight against mismatch, doing more than just spending more area? This is not a book on advanced circuits, and we mentioned already some general techniques. Especially suited for fighting against mismatch are chopping, correlated double-sampling, and dynamic matching. All this is very popular in sensor design, but also switching techniques have their limitations, e.g., due to nonlinear charge injection and kT/C noise. A less well-known method is this: Instead of using two transistors for a diff pair input stage, we could simply insert eight transistors in the layout. In a calibration phase, using switches we can try each pairwise combination (there are 28) and measure the offset. Then, just use the best one! One can easily show that this minimum pairwise offset is significantly better than a 2×4 transistor diff pair! Also the input capacitance is lower, so this technique is not bad for high speed too. The major effort is of course the implementation of switches and the calibration part. Other statistical circuit tricks? Yes, e.g., on-chip resistors may have a tolerance of $\pm 20\%$, but if you combine two resistor types which are statistically independent, then the overall tolerance reduces a bit. The worst-case would be the same, but it has now a lower probability!

2.7 An Efficient General Design Strategy

As mentioned, a simple but extremely time-consuming way to check the design for yield would be to run a large enough Monte Carlo analysis including all testbenches at all (environmental) corner combinations. The huge effort is usually only acceptable for a final sign-off verification or for simple circuits. Even looking at the huge amount of created data requires quite some time to display and to interpret the results, whereas "direct" inspection methods usually give immediately a better understanding.

Efficient design (i.e., search in $\boldsymbol{x}_\mathrm{D}$)—in opposite to pure verification (fix $\boldsymbol{x}_\mathrm{D}$)—requires several simplifying strategies, mainly based on divide and conquer. So let us reinvestigate our op-amp design example and refine the flow further.

At the end, we will see that we obtain a flow with high efficiency, but also with some risk and this will be quantified and minimized in the following chapters.

Let us inspect the extended flow proposal in more detail (Figure 2.15); of course in each step there is some iteration, and also overall, and there is some *overlap* among the tasks. For instance, it makes a lot of sense to include parasitics as soon as possible to your simulations (especially in RF, high-speed or low-power designs), not just when a full layout is already available. This can highly reduce the number of iterations, even if the estimated parasitics are not 100% correct. Also modern layout tools allow us to run LVS checks and DRC much earlier, or offer design rule-driven layout creation. In addition, there is also a lot of work on preparations, e.g., we assume that the design team received a process development kit (typically from the foundry) and has installed all tools, auxiliary libraries, etc. Also making a small testdesign upfront, and running through all phases makes highly sense.

Besides these preparations, also in testbench creation and modeling is truly a big part of the work too! The hand-sizing is usually done by formulas and tools such as Smith chart, math toolboxes, linear equivalent circuits, scattering parameters, CMOS quadratic IV law, filter catalogs, and symbolic calculations. For many standard circuits like LNAs, op-amps, bandgaps, and OTA, you will find several step-by-step instruction guides for design. Parasitics may come from package models, or derived from rough formulas like L = 1 nH/mm or microstrip transmission line formulas. More on this topic and related design tools is presented in Chapter 10.

One difference to the basic flow is that we do a split, i.e., we decide for a design phase with focus on speed (using few corners, like 3–10) and

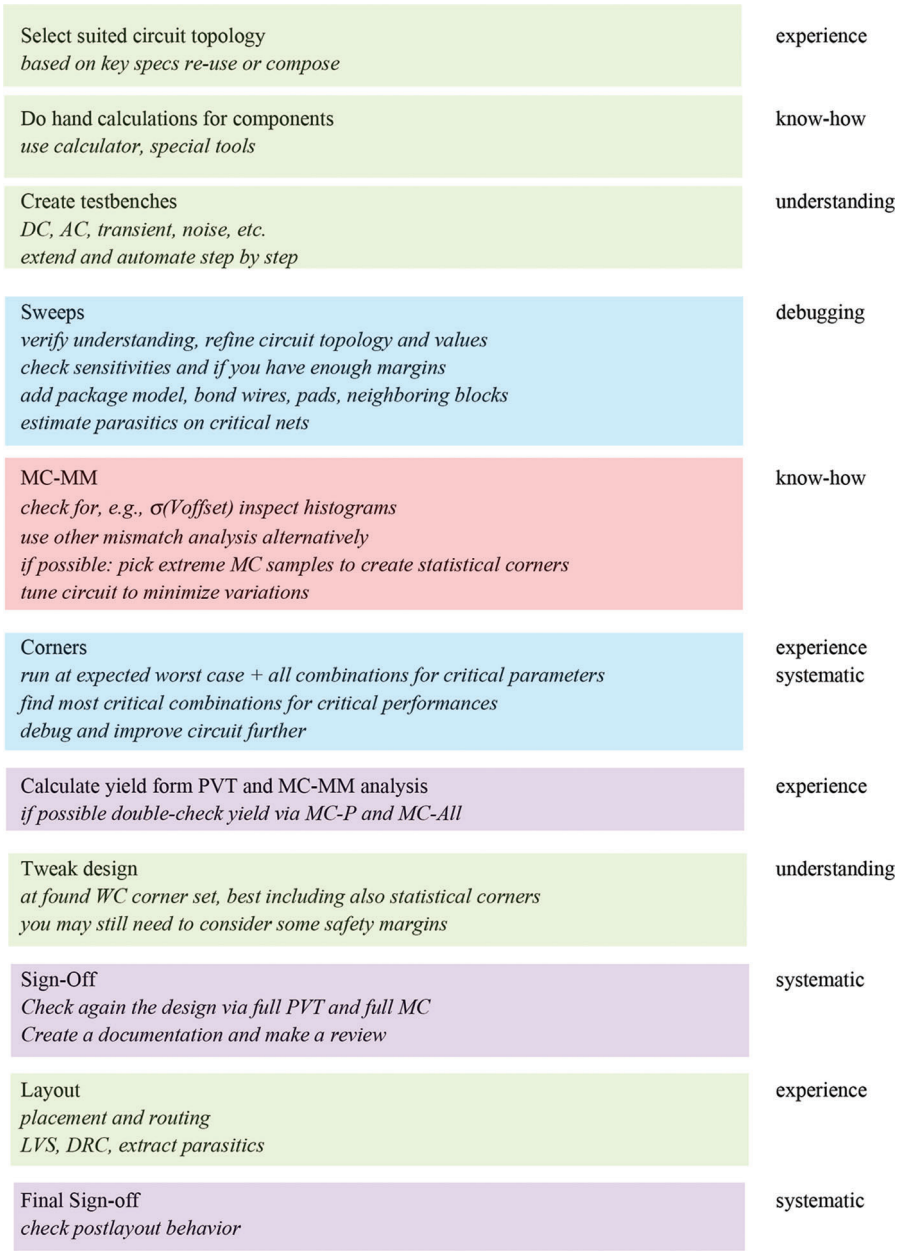


Figure 2.15 Typical manual analog flow and main challenges.

understanding (doing sweeps), and we have a sign-off verification step with focus on coverage and reliability. This way we can reduce the simulation effort in the tweaking phase. For instance, the most critical corner in analog designs is often lowVdd+slowMOS+fastR+Tsweep, for stability, it might be fastMOS+lowT+CLsweep, and for speed, it is often slowMOS+highT+maxCL. The sweep phase is done not only on range parameters $\boldsymbol{x}_{\mathrm{R}}$, but typically also on design parameters $\boldsymbol{x}_{\mathrm{D}}$ (usually a subset of the most important variables). Also model parameters might be swept, e.g., the ground inductance of a package model and the wiring capacitance on critical nets for checking the design robustness. Especially in RF designs, it is not so clear which parameters can be assumed as given and fix (like lead frame inductances) or are actually design parameters.

In the "design" loop, where we modify $\boldsymbol{x}_{\mathrm{D}}$, we have a lot of iterations. For understanding, it is best to use parameter sweeps intensively. One problem is that by tweaking the design, we can also change the critical corner combinations, so we need an update on them from time to time.

Mismatch (MM) is typically not much corner-dependent, so we can do the MC-MM analysis at typical corner (or expected WC). This analysis should be done early, because in the pure sweeps done before maybe the circuit is optimized too much for speed, so that the transistors might be too small for low offset. One advantage is that often a good interpretation of MC-MM results is possible, and usually 100 points are often enough, so quite a short MC analysis can help a lot in tweaking the design. Unfortunately, a PVT run together with such a MC-MM analysis can only give rough yield estimations. So if available, you should also do a MC analysis with "process only" for additional insights and also a MC analysis with all parameters—on the most critical VT corner. The later can also give good yield estimates, or at least an option to double-check the results.

Many modern design environments allow to save the worst MC sample as a statistical corner. Instead of (or in addition to) only making a hand calculation (like on $\pm 3\sigma$ offset), you can just put such statistical corner into the set of the usual PVT corners and size the design over it. This is a good technique and should be done for the most impacted specs and for those with significant variations from mismatch (like offset or CMRR, but usually not for phase margin or NF).

Note that some additional design margins are needed, because foundry-provided corners like SS, FS, and FF usually do not cover well analog worst-cases like phase margin PM and IP3.

The simulation effort is quite moderate, and both corner and MC analyses can run in parallel on a compute server. This is often only partially possible for more complex algorithms like gradient optimizers or worst-case distance methods (Chapter 7).

A more detailed PVT and MC analysis is usually possible for a fix design. It is usually very time-consuming on the postlayout netlist, so best include expected parasitics as soon as possible and do the postlayout simulations only at typical and few critical corners (like those on speed and stability). The technique is simple: The layout parasitics have usually no strong statistical variation, so usually you get a shift in the absolute offset voltage, but the sigma remains, and the bandwidth gets reduced maybe by some ten percent. With that qualitative *a priori* knowledge and your prelayout result, you could just compose the total postlayout behavior with few postlayout simulations only (in extreme cases just a nominal simulation). By "borrowing" information, you can get quite reliable information with much lower effort. Of course, you cannot do this for effects highly dominated by parasitics, such as cross talk.

Over-all, we need to pamper up our design, so starting the simulation part at nominal conditions is a native choice. Incrementally, add all types of variation (process, voltage, temperature, parasitics, etc.) that may matter to get an understanding. As design tweaking for optimum yield and performance requires many re-simulations, it has to be done in an efficient way with focus on the worst-case combinations.

2.7.1 Desirable Improvements

The described flow can be applied in many commercial design environments for some years, but it has also its limitations. If we do only a PVT and MC-MM analysis, we ignore that foundry-provided corners typically only cover the WC on CMOS speed, but not typical analog measures and circuits!

Options to improve:

- *Extend the foundry corner set, e.g., to include more complex cases like FF_slowRslowC. Use them for complex cases like analog filters. Another example is having mixed MOS corners for thin and thick gate (high-voltage) MOS transistors, using them is often need in pad cells or level-shifters.*

- *Consider PDK and model extensions, e.g., to be able to set the desired yield value (in sigma) for the PVT corners.*
- *Provide corner templates for all designers in the project to ensure a minimum coverage.*

We do a short MC analysis on mismatch only, and usually, the output performances will be normally distributed, but not always. Using $\pm 3\sigma$ and combining the PVT and MC-MM results is essentially an extrapolation method with some risks. The risk is often too large for high-sigma yield targets!

Options to improve:

- *Extend the MC analysis to get also high accuracy for the yield estimation; unfortunately, this may require many more MC samples. Check out Chapter 3 for more background information and examples. An easy-to-apply MC speed-up method is low-discrepancy sampling LDS (see Chapter 6).*
- *Even no change in simulation setup is needed, if you use enhanced mathematical methods to address also non-normal cases, as described in Chapter 4.*

If we run a MC analysis and create statistical corners, we cannot know how accurate these corners will be. In addition, to get a 4σ sample, we would also need a very large MC count!

Options to improve:
Consider to create really accurate statistical corners to get rid of the extrapolation risk as shown in Chapter 7. Such statistical corners have the advantage that they can be (almost) accurate for your circuit and performances, so they do not lead to under- or overdesign as the standard foundry corners. Also they can include mismatch.

Such a flow is not only for design and verification; it can also provide additional design insights.

Options to improve:
Apply multivariate analysis to obtain sensitivities from MC results. Chapter 5 describes the methods and many examples, e.g., we can apply a mismatch contribution analysis with minimum overhead on simulation time.

Do correlation analysis to understand trade-off and to better estimate the total yield from the partial yields (Chapter 5).

This flow requires of course some experience and careful setting of design margins.

Options to improve:
To get an overview and concrete numbers, let the design environment do a sensitivity analysis, and this is often easier than doing sweeps manually. Many environments also support corner-dependent spec settings, so we can include design margins if we want.

As mentioned, modeling is better than margins. Modern tools offer support to add estimated parasitics to your design and can also calculate layout-dependent effects and quite accurate parasitics estimates from a partial layout.

Having a complete verification setup is also a perfect preparation for an *automatic* optimization.

Options to improve:
Define which parameters to tweak and let an optimizer do the sizing job. This is more efficient than optimization by plain sweeps (see Chapter 8).

When is a design good? When is a flow optimum? We will care for optimization in Chapters 7 and 8, but with a strong focus on circuits. A general question is indeed what we want to achieve! For instance, if we treat in a layout each individual wire length as important and want to minimize it, we can hardly reach an optimum. At best you can reach something called a Pareto optimum. Such Pareto optimum point is already reached if you have a situation where you can only improve one goal if you get worse on another goal. Obviously, in such situations, a certain Pareto optimum can be far away from your ultimate design goal where it is more important to minimize just the length of the real critical nets or maximize the amplifier bandwidth. Luckily, most optimum conditions are quite *flat* naturally, so spending too much time to exactly reach an optimum makes little sense. So often you need to be pragmatic, like "A design is good if you cannot improve it significantly anymore, if it is in spec, and if your boss likes it."

An optimum flow is also difficult to implement; it is possible only for simple problems like finding the optimum for a parabola. The worst-case might be found by running all parameter combinations and searching for the most extreme result. This is good for small problems, but often it is better to exploit your design know-how and use a slightly riskier but much faster method. In conclusion, your responsibility as designer is mostly on setting the right goals, choosing efficient methods, and improving the design step by step, e.g., starting with ideal current sources, then implementing a real MOS current mirror, adding auxiliary functions,

and aligning with other blocks. At best you have always "something that works." For sure, an optimum design execution is almost only possible if you would know the result in advance! So realistically many things depend on anticipation, on experience and on your capability to exploit the information you already have—as good as possible! Plus, you should avoid redundant work or even dead ends.

2.7.2 Mr. Murphy and Mr. Beckmesser

Sometimes specific person stands for something quite specific, like Mr. Giacomo Casanova for womanizers, Robinson Crusoe for a lonesome person disconnected to the world, Mr. Edward Murphy is often made responsible if something gets wrong; and Mr. Sixtus Beckmesser is an annoying person, someone who knows everything better, without really being able to do it. However, in math and engineering, you should have some sympathy to Beckmesser, because we hope in some sentences so far some of the readers get almost a heart attack.

We devided our variables into three catagories, and we one of them is the range variable category $\boldsymbol{x}_{\mathrm{R}}$, another one the statistical parameters $\boldsymbol{x}_{\mathrm{S}}$, and last not least the design variables $\boldsymbol{x}_{\mathrm{D}}$. So far so good, but when explaining corner analysis we but the technology corners like FF or SS into the $\boldsymbol{x}_{\mathrm{R}}$ category. Is this correct, just because SS is obviously neither pat of $\boldsymbol{x}_{\mathrm{S}}$ nor $\boldsymbol{x}_{\mathrm{D}}$, and because the design environments let us do so?

Let us go for a simple example:
If you pick a sample from production and run the temperature *corners* according to your spec limits, then a sample needs to be in spec for all 3 temperature values to really pass the spec. If all samples pass, we have 100% yield. And if we get a fail e.g., only at $T_{\min}$, the yield would already zero!

However, if we simulate three technology corners, like slow, typ, fast, and we have one fail e.g., at slow. What is then the yield? It is obviously not 100%, but realistically it is also not 0%! If we assume a uniform distribution we may apply the rule of proportion (leading to a kind of margin approach). Or we may simply assign each category to 33%? Or we may simply say that it is simply not possible from a corner analysis to conclude on the yield?!?

Indeed at some point we need to be a bit pragmatic: Following the classical corner approach, we would say, that one fail over $\boldsymbol{x}_{\mathrm{R}}$ could already to a fail. However, realistically this is a bit too conservative, and we may overdesign a bit!

Better than nothing? But what about this: If we say a corner is standing for 5σ, but we want 7σ yield for our block. Are we then still overdesigning? Obviously not really, but what is realistic? Actually the *100% correct* way would be to model the process tolerances parameters as belonging to $\boldsymbol{x}_\mathrm{S}$, so they would require a *statistical distribution*, a pdf. Typically the pdf would be neither uniform, but often also not 100% normal Gaussian! This is because, if the process parameters are too bad, then the foundry would not process the chips further, so these samples would be sorted out! This typically leads to cut distributions, like a normal distribution which pdf is zero, e.g., beyond 5σ (or whatever). All these things may lead to some extra margin for you as designer, but better do not exploit this too much.

In addition, we mentioned that we can cover the statistical variables $\boldsymbol{x}_\mathrm{S}$ by statistical models, which are known to the design environment and the simulator. Also this is not 100% true, e.g., also simulation errors e.g., due to rounding, time step limitations, etc. have a statistical nature, but they are not so easy to quantify. Uncertainty and risk quantification is a nice topic on which you should know the basics. Read the next chapters, check out further literature, have a talk with your quality and technology experts.

2.8 Design with Pictures Part One

In the past, graphical methods have been intensively used for design, like the load-line method for power amplifiers or the Smith chart for matching network design. Nowadays, there is a trend to use numerical methods, simulators, and more advanced models, but thinking in pictures often helps a lot to understand the design (e.g., its limitations, like matching network design becomes difficult if you need to start far away from the Smith chart center) and also to understand numerical algorithms (e.g., for yield analysis or for finding a DC operating point).

We will pick up the idea of designing with pictures, because "thinking in pictures" is very helpful also for understanding many statistical methods like worst-case distances (WCD). These have also a straightforward geometrical interpretation! Also normality can be easily checked graphically in a so-called normal quantile plot. For a first design example with pictures let us focus on a power amplifier (PA).

2.8.1 CMOS RF-PA Example

So, what is really important in a radio frequency PA design? Well, it is important that the PA can provide a given output power with a constant

gain at a particular frequency range. The application essentially dictates these parameters, i.e., most systems have a predefined power range and frequency allocation that the transmitters should comply. For instance, a Bluetooth transmitter should be able to operate between approximately 2.40 and 2.48 GHz and should not exceed 20 dBm (100 mW) of output power. Usually also a certain minimum output power level is required, because with too small energy no reliable transmission is possible.

In fact, although only one transistor is needed to perform the required DC-to-RF conversion in a PA, the difficulties arise on doing it efficiently and, at the same time, in a linear enough way. There is a classical compromise between power efficiency and linearity, and how much linearity is really required depends also on the modulation scheme. Bluetooth has no very high demands, so usually a class AB amplifier fits. The frequency of operation should not be an issue here as a 45-nm CMOS process node will be used in our simulations. As target, we specify an operating frequency of 1.2 GHz and the 1 dB compression point at 15 dBm. Design for larger output power will often trigger a lot of problems, e.g., thermal problems, gain and stability degradation by ground and package inductances, reliability problems caused by breakdown mechanisms and electromigration, etc. In our starting example we want to avoid having one specification much more difficult to satisfy than others! Table 2.3 summarizes a small list of specifications for the design.

For the design of a class PA, a designer needs to know at least some RF basics, e.g., that an RF choke is used to provide the DC current, which will be modulated by the MOSFET. At the limit of class AB, i.e., class B, the resulting drain current should be *half sinusoidal* (transistor active during 180° out of the full 360° cycle) with some peak value $I_{\mathrm{p.}}$So, a first step in assessing the design would be inspecting on the IV characteristics of the transistor and relating them to the output power. For the present example, we will make use

Table 2.3 Specifications for our RF PA design

Symbol	Description	Value	Unit
f_{c}	Operating center frequency	1.20	GHz
Δf	Bandwidth	>20.0	MHz
$P_{1\mathrm{dB}}$	Compression point	>15	dBm
G	Gain	>30	dB
OIP_3	Third-order interception point	>15	dBm
$V_{\mathrm{DD,nom}}$	power supply voltage	2.2	V
η_{pk}	drain efficiency the $P_{1\mathrm{dB}}$	>15.0	%

of external inductors to simplify the design, although the pad models and bond wires need to be considered also.

For an ideal switching transistor (no saturation voltage, i.e. $V_{\mathrm{DSsat}} = 0$) the drain-source voltage would be between 0 and $2V_{\mathrm{DD,nom}} = 2.2$ V; and the maximum voltage is important for reliability reasons and it will often restrict our choice for a certain transistor type (like thin vs thick gate transistor). A finite V_{DSsat} helps actually a bit on voltage stress, but not much. Modern CMOS technologies have unfortunately quite low breakdown voltages, so usually the question arises if a cascode topology or a single transistor should be used. We can also think if special combinations, like a cascode with thick-gate device at the top and a thin-gate oxide device at the bottom. Such combination guarantees a high gain and can treat large output voltage swings, but has of course a saturation voltage given by the sum of both devices; and a large V_{DSsat} degrades the efficiency. So there are some trade-offs *in each* option that a designer should consider, for instance, thick-gate devices allow you to improve the voltage capability and reliability, but the f_{T} is reduced because, in general, the minimum length L_{min} is larger than the L_{min} of thin-oxide devices for the same process node. What about increasing L? This would typically improve on output conductance, but for an RF PA it makes little sense, because power, gain, and efficiency matter much more. To find a good design we must optimize each solution as much as possible, or we need to exclude solutions which are impractical. So the designer must know whether a certain issue (like breakdown) is in fact restrictive in the application or if it is merely a minor side effect. For instance, our frequency of operation is relatively low, so using thick-gate transistors cannot be excluded. Also some effects might be not fully covered by the target spec. For instance, isolation is not part of the spec yet, and to improve the isolation between input and output, the designer may adopt a cascode topology, maybe using thin-gate oxide devices only.

Here and in many other design aspects, the simulator can assist the designer on deciding the values of some parameters. Obviously, that needs basic understanding of what is really in play, at least at a qualitative point of view. Sometimes by visual inspection with some rough calculations, one can get a very feasible starting point, leaving the fine-tuning for a later procedure. To determine the transistor dimensions, a designer can perform a quick DC sweep and obtain the i_{DS} of a (now chosen) cascode circuit—Figure 2.16. We assume $L = L_{\mathrm{min}}$ to achieve the highest f_{T} possible and start by guessing a plausible device width based on experience (at least on the order of magnitude).

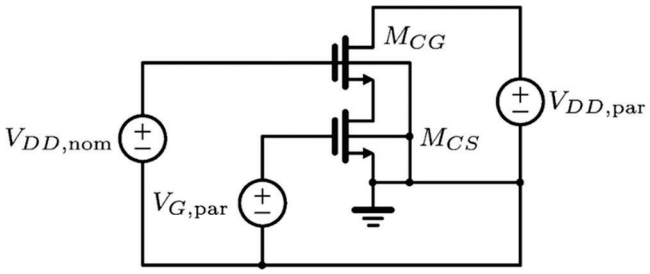


Figure 2.16 Cascode cell for DC sweeping of power supply and gate voltage at the common source.

To find the required i_{DS}, a designer can change the width in a linear way, i.e., supposing it is needed 2.5 times more current, they just increase the width of your device by the same factor. For instance, one may assume the IV of the cascode circuit in Figure 2.16 in which the transistors are equal sized—other possible solutions with unequal sizes may be explored afterward, in optimization.

Figure 2.17 shows the output of the DC simulation. If we consider the knee voltage V_{K} as the $V_{\mathrm{DS,sat}}$ (obtained from the BSIM output), in this case around 600 mV, we can assume $i_{\mathrm{DS,max}} \simeq 14$ mA for a class B PA.

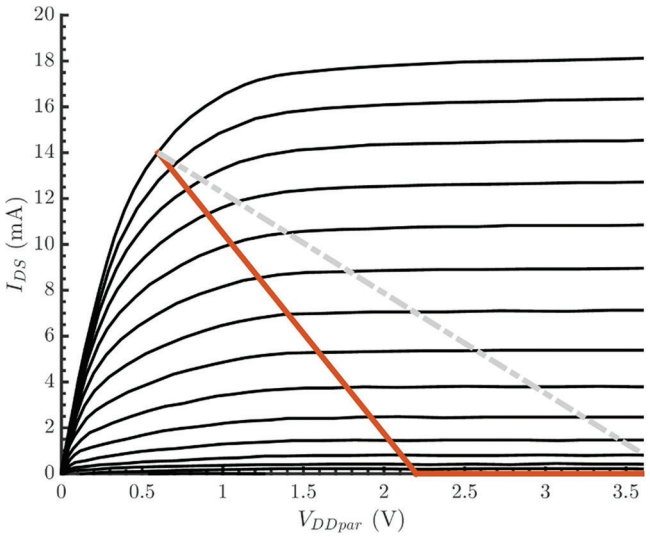


Figure 2.17 DC sweep simulation results for the cascode cell.

Under these conditions, we can get roughly the peak power $P_{\max} = V_{\mathrm{o,pk}} \times I_{\mathrm{o,pk}}/2 = 5.6\ \mathrm{mW}(7.5\ \mathrm{dBm})$, with $V_{\mathrm{o,pk}} = V_{\mathrm{DD}} - V_{\mathrm{K}} = 1.6$ V and $I_{\mathrm{o,pk}} = i_{\mathrm{DS,max}}/2 = 7$ mA. This means we can estimate a large signal load $R_{\mathrm{L}} = 1.6\ \mathrm{V}/7\ \mathrm{mA} \simeq 229\ \Omega$. Note, that this approach is quite different from small-signal design techniques, required e.g., for low-noise amplifier (LNA) design, where you focus on small-signal S-parameters and noise.

Figure 2.17 shows the draft load line (in dashed black) for the class B at 7.5 dBm output power. If we want to achieve a given peak power $P_{\max}$ that is ρ times higher than the present peak power, we need to improve the current by about ρ times, so we will need ρ MOSFETs in parallel (in both transistors of the cascode) to see whether we can achieve such peak power. The load will need to be reduced accordingly, $R_{\mathrm{L}} = 229\ \Omega/\rho$. For a peak power of 15 dBm, ρ should be in the order of 6, implying $R_{\mathrm{L}} \simeq 38\ \Omega$. However, for such a case, the peak power is already at some compression level. So, a better option is to give an extra-margin of $P_{\max}$ to $P_{1\ \mathrm{dB}}$, for instance $\rho = 10$. Based on these numbers, one can at least roughly predict the drain efficiency (η). Other non-ideal elements will come, but at least to have a reference, one can estimate whether it will be too far from the specifications. Taking into account the knee voltage, $\eta = (\pi/4) \times (V_{\mathrm{DD,nom}} - V_{\mathrm{K}})/V_{\mathrm{DD,nom}} \simeq 57\%$ (with $V_{\mathrm{K}} = 0$ we get the famous theoretical optimum of 78.5%).

One can quickly set up a circuit like in Figure 2.18, which is a simplified approach to validate the targeted power and load, just using an ideal input

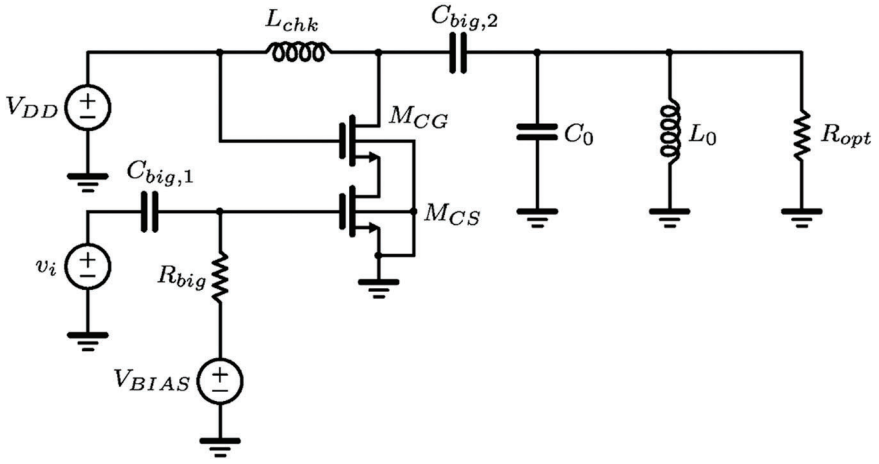


Figure 2.18 Circuit for studying basic parameters.

drive (no matching input matching network). The designer can start building a list of formulas to define some parameters—some may even be kept until the end of the design. For second order parameters we can often apply a rule of thumb. Consider for instance Table 2.4 where we listed several parameter values and respective equations. There we assumed the choke impedance as 50 times larger than the optimum load, whereas for the RF coupling capacitor, the impedance was considered 100 times smaller. Other derivations are at somehow related to specifications, e.g., the loaded quality factor (Q_{L}) of the output RLC network was assumed as 5. Also here the absolute value does not matter much as long $Q_{\mathrm{L}} \ll Q_{\mathrm{spec}} = f_{\mathrm{c}}/\Delta f$.

It is advised to include some non-ideal aspects already from the early start. For instance, the use of finite unloaded quality factors for the inductors gives a more realistic performance (e.g., in output power and efficiency) and also helps the simulator to converge. For the present example, we will assume external inductors, so we can assume intrinsic quality factors (Q_{u}) in the order of 60 to 80; furthermore, it is not difficult to find inductors with a self-resonating frequency much higher than 1.2 GHz. Hence, for each inductor, one can include at least a series resistor-valued ESR $= \omega_0 L/Q_{\mathrm{u}}$ (the *ESR* is not seen in schematic as a component because it is included in the inductor definition). Another important value to take into account is the bias level of the transistor. At the gate, the voltage should be above V_{T0} to achieve a class AB operation. A DC analysis will be sufficient to obtain this parameter. Note, that

Table 2.4 Parameter definition for studying the performance of the class AB PA

Parameter	Equation	Value	Unit
V_{BIAS}	$>V_{\mathrm{T0}}$	575	mV
V_{K}	–	0.6	V
V_{o}	$V_{\mathrm{DD}} - V_{\mathrm{K}}$	1.6	V
$R_{\mathrm{L,opt}}$	${V_{\mathrm{o}}}^2/(2\,P_{\mathrm{max}})$	38.0	Ω
Q_{L}	(arbitrated)	5.0	–
ω_0	$2\pi f_{\mathrm{c}}$	7.540	Grad/s
L_0	$R_{\mathrm{L,opt}}/(\omega_0\, Q_{\mathrm{L}})$	1.0	nH
C_0	$1/({\omega_0}^2\, L_0)$	17.4	pF
L_{chk}	$50 \cdot R_{\mathrm{L,opt}}/\omega_0$	252.0	nH
C_{big}	$100/(R_{\mathrm{L,opt}}\omega_0)$	349.0	nF
R_{big}	(arbitrated)	10	kΩ
$Q_{\mathrm{u}}(L_{\mathrm{chk}})$	(arbitrated)	80	–
$Q_{\mathrm{u}}(L_0)$	(arbitrated)	60	–
ESR(L_{chk})	$\omega_0\, L_{\mathrm{chk}}/Q_{\mathrm{u}}(L_{\mathrm{chk}})$	23.7	Ω
ESR(L_0)	$\omega_0\, L_0/Q_{\mathrm{u}}(L_0)$	127	mΩ

it also a good method to build up the testbench step by step, to immediately see if one change has a surprisingly large impact (e.g., in the package inductance at the source is not yet included, and it would have some impact on the PA gain).

Once all the parameters have been established, the PA circuit can be simulated using transient analysis or periodic steady-state simulations (PSS). Figure 2.19 shows the results from a single-tone continuous-wave (CW) test using a PSS simulation. It depicts two time-domain representations of the PA signals, one for the current through the channel of the transistor i_{DS} and the output voltage at the load. This time domain allows to see whether everything is as expected, e.g., whether the i_{DS} is nearly a half-sine waveform (50% conduction over the period) and the output is sinusoidal, although the spectral analysis output will provide more accurate information. In the present case, there is some influence of the triode operation, but still if one simulates the circuit with infinite and finite quality factors, one gets an output power of 14.7 and 13.4 dBm, respectively, and an efficiency of 49.5 in the ideal case against 40% with finite Q, i.e., one can identify already some performance degradation. Naturally, the circuit requires some optimization, but at least the designer can predict some rough numbers for a quick start.

Even though one can use transistor width scaling to set your peak power, such does not assure you that a different load value could in fact benefit the circuit performance, so often we require a kind of multi-parameter optimization

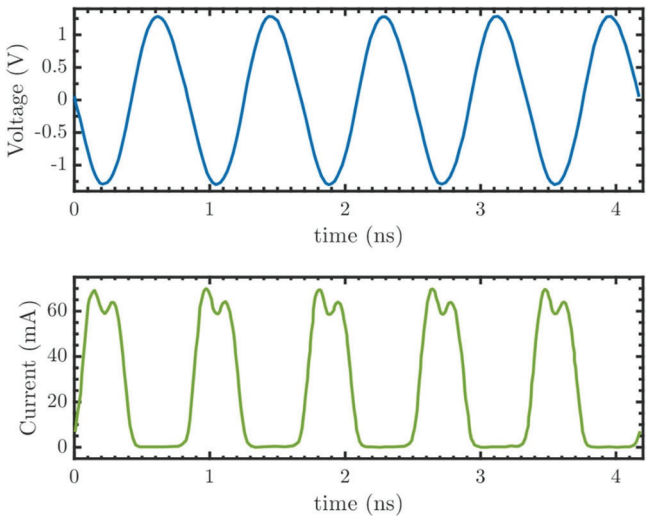


Figure 2.19 Testing the output voltage (*top*) and maximum i_{DS} of amplifier (*bottom*).

for circuits. A typical design technique for PAs is the "load pull" method, in which the complex load impedance is swept in terms of both real and imaginary part. This allows the designer to get an idea of the performance under different loads, and helps finding the optimum load. In discrete implementations, load-pull techniques (based on output tuners to vary the load) are very useful, because we would include all the parasitic elements as they are, such as lead inductance or other parasitics resultant from interconnections. However, when it comes to IC design, the simulator gets a hard job, because we need one simulation for each complex impedance! The post-processing and the plot looks the same in real world and simulation. Figure 2.20 shows the contour plots from a load-pull simulation. In Figure 2.20(a), each contour represents the load values for a given constant power level, whereas Figure 2.20(b) depicts constant PAE contours. In both cases, the lower the radius of the load locus, the higher the power (or the PAE).

As shown in Figure 2.20, to achieve the peak power and peak PAE, completely *different* load values are required, implying a noticeable performance trade-off. Also, the drive strength must be properly analyzed, so that the power gain can be optimized. Source-pull simulations can be also included to this end, but a very complex simulation setup must be implemented (4D sweeps instead of 2D sweeps). Moreover, corner analysis can also provide additional information, but the time required to perform load pull is in fact an issue. In terms of simulation, load- and source-pull techniques are excessively time expensive because the parameter sweep domain is quite vast. If we decide for a certain "compromised load" impedance, we could extend our

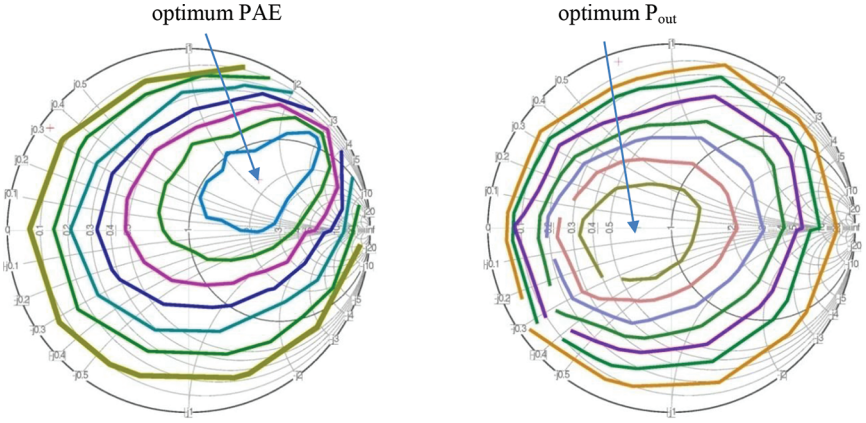
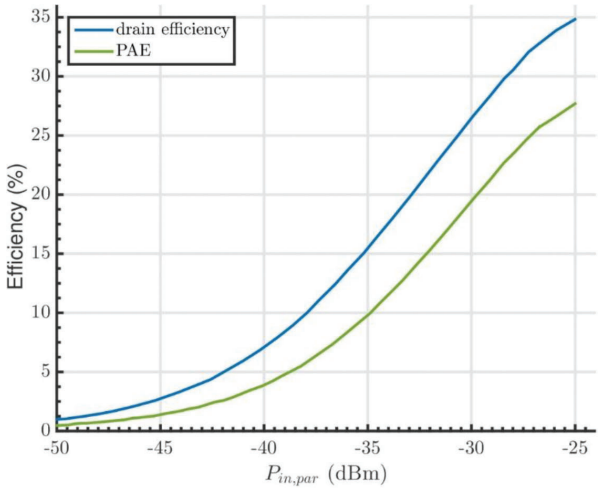
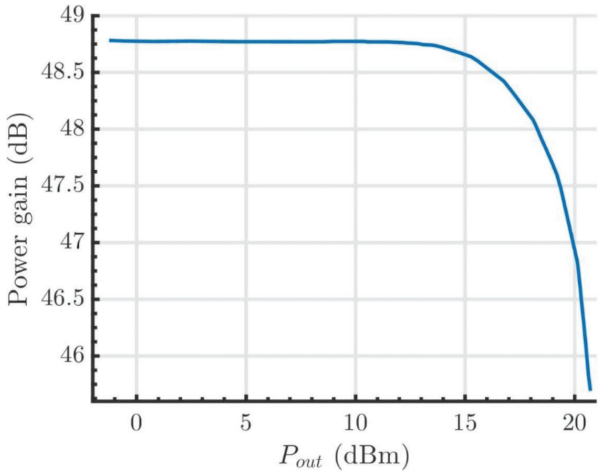


Figure 2.20 Load pull (a) constant PAE contours and (b) constant power contours.



(a)



(b)

Figure 2.21 Simulations for the nominal conditions—(a) power efficiency and (b) gain.

PA design, and include a matching network, e.g., from 50 Ω standard antenna impedance to this selected impedance. Another design option would be to select a flexible enough matching network topology, and to directly tweak the *component values* in the network (together with other relevant parameters, such as transistor width), instead of tweaking the complex impedances.

After some efforts on manual tuning, a solution arises. Some nominal results such as efficiency and gain, are shown in Figure 2.16.

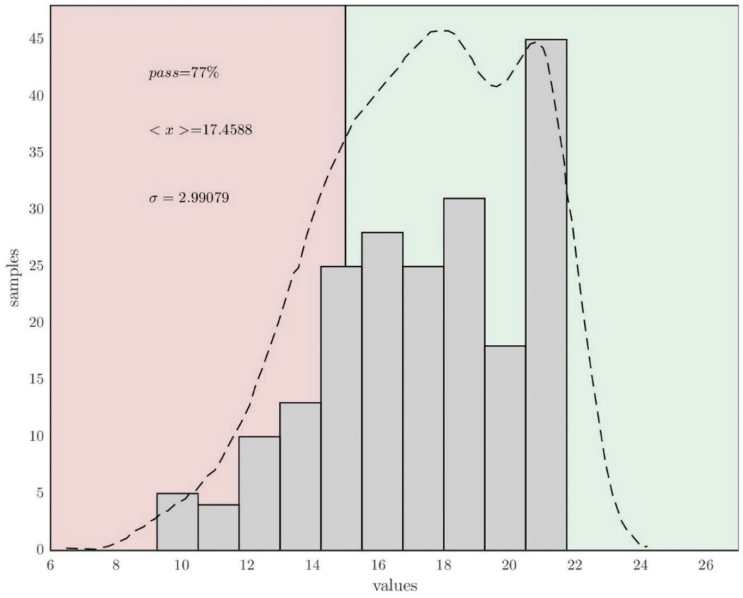
Nonetheless, when subject to process variations, the performance differs a lot, unfortunately! For the present example, the V_{T0} value depends on the process corner, and if we assume a constant gate bias, in some cases the operation can be class C (for instance in SS corner, because the V_{T0} is higher) or in less-deep class AB (e.g., in FF). So, the signal excursion will be different and will have e.g., some impact on P_{out} and PAE. One can choose SS to establish the starting point with some margin, or give some adequate performance margins, having in mind a yield optimization to be performed later. A sweep with different sizes of the cascode (equal widths always for the common gate and common source transistors), with the gate bias kept fixed will indicate that the number of fingers of the transistor above 20 is required. It turns out that the compression point differs in about 4.5 dB for FF and nominal.

Figure 2.17 depicts Monte Carlo results for a first manual design of the PA for some performance metrics. Although most of the samples fall in the safe side, there is still room for yield improvement. That will eventually sacrifice some performance parameters for others in order to have multiple specifications with higher yield. Such optimization procedure will be addressed in later chapters.

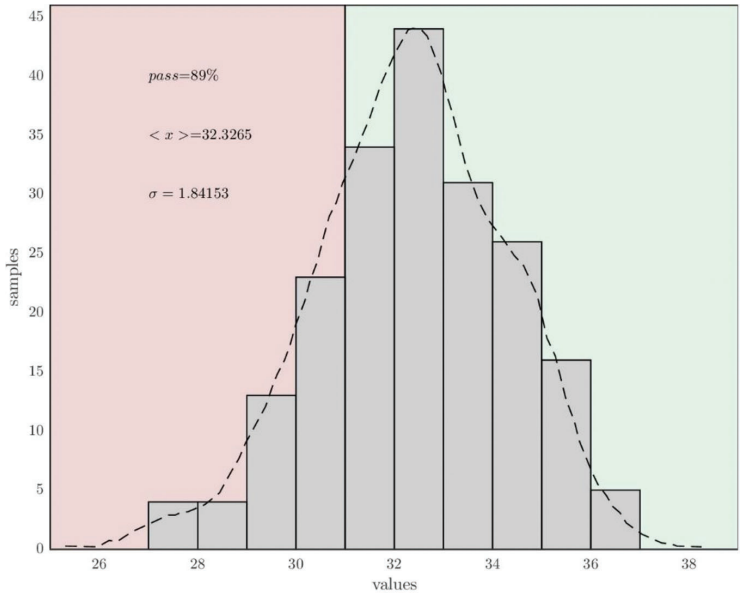
2.8.2 Worst-Case Search Showdown

When we introduced different algorithms, we used a CMOS inverter as example. We could be easily extend it and include further specs, like on leakage current, dynamic average current, static cross-current, area, input capacitance, output resistance, jitter, and DC gain. So even such a simple inverter can have quite many specs and many different WC combinations. And we could easily extend further, e.g., for a CMOS Schmitt trigger, we would have the same problem plus a big interest for the input switching threshold voltage and hysteresis. Also the corner set might be extended for inclusion of generator resistance and load capacitance. Or we may add a second inverter or a level shifter. In the later case, we should add the second supply as further corner variable. In an uplevel shifter, it could easily happen that not the min $V_{\mathrm{low}} + \min\ V_{\mathrm{hi}}$ case is most critical but min $V_{\mathrm{low}} + \max V_{\mathrm{hi}}$ case; mixed cases are often overlooked.

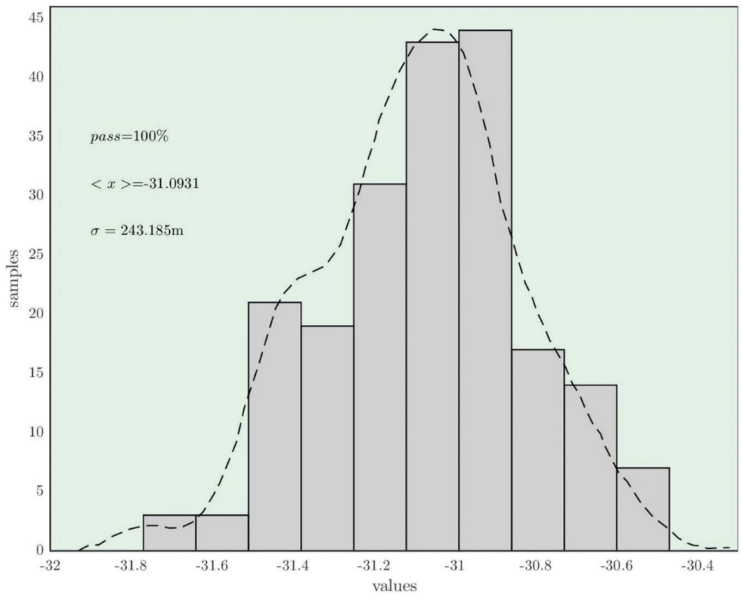
As we mentioned, sometimes simple circuits like an inverter can be difficult for WC finding. However, we do not want to focus so much on near-digital


pass=77%
< x >=17.4588
σ = 2.99079
samples
values
0
5
10
15
20
25
30
35
40
45
6
8
10
12
14
16
18
20
22
24
26

(a)


pass=89%
< x >=32.3265
σ = 1.84153
samples
values
0
5
10
15
20
25
30
35
40
45
26
28
30
32
34
36
38

(b)



(c)

Figure 2.22 MC results for (a) IP_3, (b) $|S_{21}|$, and (c) $|S_{11}|$.

circuits, so for our showdown on worst-case search methods, we picked a second difficult example. It is a CMOS op-amp, the one which we will use later also for a statistical analysis (see Chapter 4). Actually, it is often not so clear whether "difficulty" is in the circuit complexity or because of other tricky things. For normal WC corner search, OFAT typically fails in roughly 20–60% of the specs, and one tricky performance in amplifiers is often the closed-loop gain peaking, because it is critical to several variables (like load capacitance, frequency compensation elements, etc.) and many dependencies are highly nonlinear. On stability, our dedicated example op-amp circuit itself is indeed tricky, because the frequency compensation scheme of this amplifier works with an advanced pole-zero cancellation to achieve a bandwidth as high as possible (in MC, you can also see issues, e.g., the histogram is becoming bimodal (see Figure 4.14).

To check how good the different methods work, we run the different WC corner search algorithms on this op-amp, for all specs and five corner variables. Table 2.5 shows the results of different methods applied to the closed-loop gain peaking (as very *difficult* measure). As expected, the fast OFAT method is also the least accurate one. Luckily, the user gets at least a warning:

Table 2.5 Worst-case corner search with different methods on op-amp gain peaking

WC method	Peaking	WC combination	#simulations	#simulations in total	Comments
Full-factorial	15.61 dB	C_{max}, $I_{biasmax}$, V_{DDmin}, T_{max}, SF	324	324	Simulations can run in parallel if a large compute server is available
Standard OFAT	10.32 dB (2.93 dB)	C_{max}, **$I_{biasnom}$**, **V_{DDnom}**, **T_{nom}**, SF	14	14	User gets a warning that OFAT solution is inaccurate
Standard OFAT around expected WC	4.8f or 13.3*	C_{max}, $I_{biasmin}$, V_{DDnom}, T_{min}, FF*	14	14	*Result depends also on user entries! User gets a warning that the solution is inaccurate
Preordered OFAT	12.74 dB	C_{max}, $I_{biasmax}$, **V_{DDmax}**, T_{max}, SF	13	n.a.	Error is in supply, because that was set as first variable
Preordered OFAT with one further iteration	15.61 dB	C_{max}, $I_{biasmax}$, V_{DDmin}, T_{max}, SF	15	n.a.	Refining on the first set variable

Internally, OFAT is just composing the overall WC combination from the individual worst-cases, and using a linear model, it can also estimate the performance value (2.93 dB) for this combination. In a verification step, this can be verified by just executing this composed WC combination: We get 10.32 dB which indicates that the (actually linear) extrapolation was not really accurate. One surprising result is that on this performance also OFAT around an *expected* WC does not work well, although it worked *fine* in our also quite difficult CMOS inverter example! If we e.g., start the search around $V_{\mathrm{DDnom}}, T_{\mathrm{min}}, C_{\mathrm{Lnom}}, I_{\mathrm{biasmax}}$, FF, which is a meaningful set for many op-amp designs, we get almost 0 dB as worst-case, which is completely wrong! Of course, the method could also work fine, so for another meaningful start set we get e.g., 13.33 dB, which is now indeed better than the standard OFAT method around nominal. However, although OFAT around an expected WC should work better, also for good theoretical reasons, it does not really provide high robustness.

Table 2.6 shows the corner analysis results (324 corners, nominal process corner NN was only simulated in combination with the other variables at nominal), and based on that (best by using the xls file provided at the River webpage), you can try for yourself to find the worst-cases with your own methods; you can truly double-check why certain methods do not provide the absolute worst-case.

One problem in the gain peaking behavior is that it is simply constant at zero for many corner combinations, because the op-amp is very stable and behaves like a first-order low-pass filter. Another difficulty is that peaking is quite sensitive to small parameter changes for certain parameter regions. All these special characteristics, mathematically equivalent to *high-order* functions, are the reason why OFAT (being not tool-specific) fails significantly and also older automatic search algorithms do not fully end up in the true absolute WC (although being much better than OFAT!). Fur luck, all the other op-amp performances (actually more than ten) indeed caused almost no such severe difficulties.

As a further example let us check in detail our stepwise preordered OFAT search method. The user would need to define which parameters are most critical, so the accuracy depends also on user-provided settings. Let us assume that process corners will be regarded as most important, then temperature, then load capacitance (as we know this op-amp is tricky on frequency compensation), then bias current, and last supply voltage (because analog circuits often have good PSRR). In addition, we can take an estimated WC combination into account, and for op-amp, stability this is usually FF,

Table 2.6 Op-amp corner results to showcase different WC search methods

	$C_{\mathrm{L}}/\mathrm{F}$	$I_{\mathrm{ref}}/\mathrm{A}$	$V_{\mathrm{DD}}/\mathrm{V}$	Process	$T/^{\circ}\mathrm{C}$	Peaking/dB	$t_{\mathrm{rise}}/\mathrm{s}$	$I_{\mathrm{DD}}/\mathrm{A}$
Nominal	1p	10u	1.5	NN	27	2.092	1.87E–08	1.43E–04
Spec	–	–	–	–	–	< 3	$< 30\mathrm{n}$	$< 250\mathrm{u}$
Corner-ID								
0	10f	9u	1.5	SS	–40	0	1.56E–08	1.27E–04
1	10f	9u	1.5	SS	27	0	1.77E–08	1.26E–04
2	10f	9u	1.5	SS	100	9.66E–16	1.95E–08	1.25E–04
79	1p	11u	1.7	SS	27	5.07	8.02E–09	1.57E–04
80	1p	11u	1.7	SS	100	6.193	7.48E–09	1.55E–04
81	10f	9u	1.5	FF	–40	0	1.78E–08	1.36E–04
105	10f	11u	1.7	FF	–40	0	1.42E–08	1.72E–04
106	10f	11u	1.7	FF	27	0	1.54E–08	1.69E–04
107	10f	11u	1.7	FF	100	0	1.70E–08	1.66E–04
132	100f	11u	1.7	FF	–40	0	1.42E–08	1.72E–04
133	100f	11u	1.7	FF	27	9.58E–16	1.53E–08	1.69E–04
134	100f	11u	1.7	FF	100	0	1.69E–08	1.66E–04
141	1p	9u	1.7	FF	–40	0	1.62E–08	1.44E–04
142	1p	9u	1.7	FF	27	1.43E–01	1.87E–08	1.40E–04
143	1p	9u	1.7	FF	100	1.315	2.20E–08	1.38E–04
150	1p	10u	1.7	FF	–40	2.91E–15	1.45E–08	1.58E–04
151	1p	10u	1.7	FF	27	3.41E–01	1.71E–08	1.54E–04
152	1p	10u	1.7	FF	100	1.427	2.03E–08	1.52E–04
153	1p	11u	1.5	FF	–40	0	1.56E–08	1.64E–04
154	1p	11u	1.5	FF	27	4.32E–01	1.74E–08	1.62E–04
155	1p	11u	1.5	FF	100	1.727	1.95E–08	1.60E–04
156	1p	11u	1.6	FF	–40	0	1.47E–08	1.68E–04
157	1p	11u	1.6	FF	27	4.06E–01	1.68E–08	1.65E–04
158	1p	11u	1.6	FF	100	1.599	1.94E–08	1.63E–04
159	1p	11u	1.7	FF	–40	0	1.34E–08	1.72E–04
160	1p	11u	1.7	FF	27	6.07E–01	1.54E–08	1.69E–04
161	1p	11u	1.7	FF	100	1.73	1.91E–08	1.66E–04
162	10f	9u	1.5	SF	–40	3.87E–15	1.61E–08	1.33E–04
235	1p	11u	1.5	SF	27	11.88	1.65E–08	1.59E–04
236	1p	11u	1.5	SF	100	15.61	1.80E–08	1.58E–04
237	1p	11u	1.6	SF	–40	8.786	1.43E–08	1.64E–04
238	1p	11u	1.6	SF	27	11.24	1.67E–08	1.62E–04
239	1p	11u	1.6	SF	100	13.33	8.06E–09	1.60E–04
240	1p	11u	1.7	SF	–40	8.802	1.41E–08	1.68E–04
241	1p	11u	1.7	SF	27	9.88	1.68E–08	1.65E–04
242	1p	11u	1.7	SF	100	12.74	7.37E–09	1.62E–04
243	10f	9u	1.5	FS	–40	0	1.70E–08	1.29E–04
322	1p	11u	1.7	FS	27	0	1.14E–08	1.59E–04
323	1p	11u	1.7	FS	100	0	1.19E–08	1.58E–04

maximum I_{bias}, and maximum C_{L}, whereas for other variables it is hard to say, so keep them at nominal (like in standard OFAT). The search would start now with a V_{DD} sweep (least important variable), with the other parameters at the expected WC. Putting the full-factorial data into Excel and using the filtering option, you can do this by hand (see Table 2.5). In a similar way you can create any WC finder you can think of, and test it! The V_{DD} sweep is picking the points #154, #157, and #160, and the WC on peaking is maximum V_{DD}. With this, we would next sweep on bias current, so running points #142, #151, and #160. The last one is redundant, so we could skip now the simulation that point. We would find now maximum I_{bias} as current WC.

Now, we go for C_{L} and run points #106, #133, and #160 (again available from cache). As our WCC guess on C_{L} was correct, still #160 remains the WC, and we continue with a T sweep. Here, we run #159 and #161 and get $T_{\max}$ as WC, and last, we run the process corners (#80, #242, and #323 as new points). And ultimately, we found #242 and SF as WC giving a peak of 12.74 dB. We used 12 simulations only (speed-up 27×), and the result is in between the standard OFAT (10.32 dB) and the true WC (15.61 dB). Looking to the corner combination, we are only wrong regarding supply voltage, whereas OFAT is wrong on two further variables! Actually, also this result is no real surprise, because the V_{DD} WCC was already in the *first* sweep, so it has the highest chance to fail; an obvious improvement would be to re-iterate: Running points #236 and #239 would end up in the correct overall WC of 15.61 dB—in only 14 simulations (speed-up 23×)! (Table 2.6).

A further nice experiment is to check how much the result depends on user-provided WC guess. If we assume the same variable ranking, but make no assumptions on WCC, we would get still the correct WCC! Of course, this is no proof, but shows at least to some degree the robustness of the stepwise OFAT method, especially with iteration. The reliability might be further improved at the expense of some speed reduction, by checking for multiple expected WCC or just the expected WCC and nominal settings (Table 2.7).

Overall, the op-amp example nicely shows that it is indeed extremely difficult for a designer to make a *good* educated guess for the worst-cases! Designers typically either follow the OFAT idea (which is wrong on three of five variables!) or rely on experience (which maybe does not help much for new designs and new technologies). In the stepwise preordered OFAT method, the designer can at least partially include his know-how, and we can improve accuracy and speed significantly.

Note, that also this method does not pick up all information being available from simulations: If each performance simulation would run individually,

Table 2.7 Excel screenshot for the last step in ordered OFAT plus refinement

Paramete	CL/F	Iref/A	Vcc/V	gpdk09	T/°C		Peaking	t
Spec							< 3	<
							fail	f
PVT_236	1,00E-12	1,10E-05	1,5	SF	100		15,61	
PVT_239	1,00E-12	1,10E-05	1,6	SF	100		13,33	
PVT_242	1,00E-12	1,10E-05	1,7	SF	100		12,74	

we can hardly improve it further, but usually several performances can be obtained from one test and one simulation, e.g., you can easily get static and dynamic supply current in one simulation. So when looking to the WC for the different specs step by step, e.g., starting with gain-peaking spec, we could use the previously obtained results (in this case from peaking WC search) for the *remaining* WC searches on rise time or supply current! Having this information and putting it into a performance model, we can improve speed and accuracy further. For instance, it makes sense to start the WC search with the most linear and least correlated performance. In that, the risk for finding the wrong WC combination is low, and the gathered information could be used later for the more critical specs—as it was the case for gain peaking in this example op-amp.

These are indeed the principles of work in most modern design environments! They follow an idea called "design of experiments" (DOE); circuit designers would probably call it "testbench setup". DOE aims for gaining a maximum of information for a performance model (relating the inputs, like corner variables $\boldsymbol{x}_{\mathrm{R}}$, to our outputs $\boldsymbol{f}$) with *minimum* effort. Actually DOE covers also lab experiments, and e.g., dealing with measurement errors. In computer simulations, nowadays often "replacing" breadboarding, the errors are much smaller, or at least the repeatability is much better, so there is even a new scientific field for such methods, so-called DACE, design and analysis of computer experiments! Here the focus is e.g., more on complexity and space filling, not so much on making stable estimations in the presence of measurement errors.

Several reliable and efficient standard DOE methods exist, and most require only a minimum user input. For instance, note that the stepwise OFAT relies mainly on ranking only, not on true quantitative modeling; so at some point, clever mathematical methods can indeed exploit the obtained simulation results better than humans, and advanced algorithms will usually outperform manual approaches; especially if the problems get really complex

(large number of variables, high degree of nonlinearity, strong correlations). Actually, this situation for worst-case finding is a nice similarity to circuit optimization (see Chapter 7)! Also DOE has become a wide field in math, including many more methods than corner analysis and Monte-Carlo. For instance, for linear model fitting we need to run two points, if we *know* the relationship is linear, but to check linearity we need at least three points; and these should be placed *not* too close, if there are influences like numerical noise or nonlinearities. This means better go *early* to the extremes; and this is what designers do since years in corner analyses! Another classical result is that for a polynomial fit over a fix interval the optimum point placement is often related to Chebyshev polynoms. So to some degree DOE and math can also help designers in testbench design; and some further DOE results will be presented in the chapter on Monte-Carlo sampling methods, where we go *beyond* pure random MC. In Chapter 6 we will see that DOE is also useful for statistics, but note, there is no free lunch: If we optimize a set of points for a certain task, like modeling of a second order system, then we may have to make some trade-offs regarding other goals, like yield analysis.

Note that unfortunately any adaptive or stepwise ordered approach will take overall, for many specs, more points than standard OFAT. Also the OFAT speed-advantages will reduce if we have many performances, and if want to hit their worst-cases. This is just because for different performances we will usually find different WC combinations. This is obvious, but it does happen quite slowly, e.g., in realistic corner setup we can easily have hundreds of corner combinations, so even if we have 30 specs and maybe 20 different WC combinations this "fill-in" effect will be quite moderate.

In OFAT you actually make a (linear) model around a center point, and some worst-cases might be far away from that. This leads to difficulties because for the modeling of *large* deviations (starting from the center) you really need high-order models, and all kinds of models for this task have a larger "extrapolation" risk than a model which starts already a point close to the worst-case. Also here, we will find similarities in the following chapters, like when comparing response surface models and the WCD methodology (Chapter 7). The most advanced methods are adaptive, so the previously obtained simulation results are used to make decisions for the next simulations. This gives many more options for better speed and accuracy, like you have more design options when using feedback techniques or other clever circuit design concepts!

2.9 Questions and Answers

1. Compare your typical design approach with the one shown in Figures 2.1, 2.4 and 2.9. In which parts do you spend most of the time? Where is reuse possible?
   The answer to these questions depend probably highly on block type and technology. Sometimes 50% is in testbenches and finding a topology and sometimes 90% in design tweaking an almost fix topology, and you need a lot of experience to quantify this upfront. Often there is a trade-off, like you can set up complex testbenches more efficiently by first running them with Verilog-A models, but the model creation also takes some time.
2. We mentioned that verification is only a subproblem of design: Which design is better suited for operating from 0°C to 85°C: one that works till 105°C and then fails completely for strange reasons, or one that works fine till 95°C and then slightly leaving spec, but still being acceptable till 175°C?
   Obviously, pure verification at corners is not enough, e.g., if you run your simulations from 0°C to 85°C, you would never ever have found a problem!
3. Discuss the typical analog design trade-offs on different examples like ADC, op-amp, or just a single common-source stage!
   Look at subchapter on biasing and transistor sizing.
4. If you see a certain mismatch in a MC analysis, can you improve it with a better layout?
   Usually not, the mismatch constants are already obtained from an almost perfect layout, using best practices like dummies and common-centroid placement! So you can hardly improve it further.
5. Can you "beat" Pelgrom's law on mismatch?
   Yes, with a DAC and a digital calibration unit, you can compensate any offset. Or you can apply switched capacitor or dynamic matching techniques! A nice new method is "combinatorial matching," e.g., split your diff pair into 2x4 subtransistors, using all 2x4 devices gives you the usual 2× offset improvement, but using the best 3-tuple out of four gives an even lower offset (like 20% beyond Pelgrom)! Extending this to 8 out of 12 can give you an even larger improvement rate!
6. How much speed-up can you expect by using modern adaptive worst-case corner finding methods?

The more complex the problem, the more room for improvements, so speed-up depends mainly on the number of variables and the number of points for each variable, but also on the variable and performance behavior and on correlations. OFAT is the fastest and riskiest strategy, and it is often 100× faster than full-factorial; so adaptive algorithms give typically a speed-up against full-factorial of 2× to 20×. High nonlinearity can slow down advanced methods, so avoid. binary outputs (they are also bad regarding optimization (see Chapter 7).

7. Discuss design trade-off in different circuits like op-amps, active filters, or ADCs. Which trade-offs are very hard, for which you may find a workaround typically?
   Usually things related to power, noise, and speed have hard physical limits, but technologies with higher breakdown voltages and mobilities would still give an improvement. Search in IEEE papers for figure of merits!
8. Discuss the transistor sizing approach for our RF PA design. Which sizing criteria make sense, which not?
   If your technology is slow, and you have to go to the limits, then probably inspecting f_T is the best starting point. Make the transistor large enough a achieve an on-resistor low enough for good efficiency, but making the width larger, will often lower the f_T, so you need to find a good compromise. As w_T is $g_\mathrm{m}/C_\mathrm{in}$ this type of sizing is also highly connected to capacitance and g_m based biasing; and, as mentioned, the starting point is clearly R_on based sizing. Flicker noise or mismatch can be often ignored, but mismatch has an impact on the design of the bias network, at least if no power control loop is available.
9. Try to visualize and understand the different corner setting methods. Checkout literature on design of experiments (DOE). Are there (beside OFAT) designs which show no exponential rise in number of points regarding the corner variables?
   If you need to model a polynomial behavior, then the number of corner points should rise with the number of coefficients, so quadratic for 2nd order functions. One design doing so is the Box-Benken design (Figure 2.23). Here we create block-wise variable combinations and have indeed a quadratic effort.

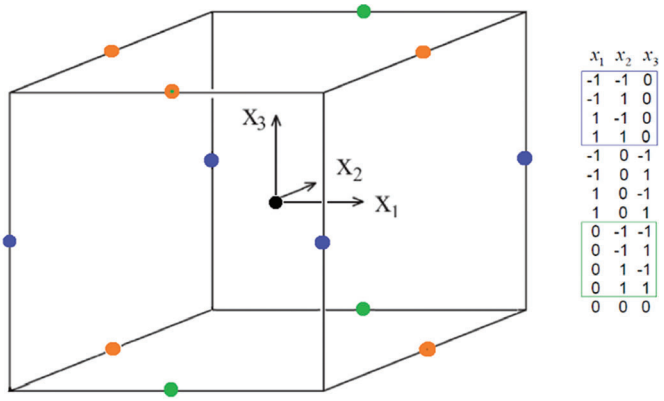


Figure 2.23 Box-Benken point set for three variables.

2.10 Rules for Corner Analysis

We described typical manual semi-automated design flow approaches. Later, we will pick up many of these ideas, and in the next chapters, we dig into the details, because also the simple-looking individual tasks like MC analysis have different faces and are far from trivial.

Rules for Corner Analysis:

- Remember the limitations that a corner analysis cannot really give a yield estimate. It cannot treat mismatch, and it cannot replace statistical analysis like Monte Carlo.
- Try to get a full overview of circuit performances and influencing parameters ASAP. Solve problems step by step, best starting with the most critical ones.
- Use OFAT sweeps for circuit understanding and debugging, but also consider sweeps with the remaining other parameters not at their nominal values, but at the real critical ones.
- Consider using sweep features directly offered by simulator analysis; this reduces the netlisting overhead and sometimes it also leads to better convergence.
- Do not forget variables to sweep, and use enough values, especially for difficult variables like temperature. Note that still such one-dimensional sweeps do not cover correlations! For this, a more detailed corner analysis, beyond OFAT, is required.

- Also sweep circuit parameters to check how much you can influence the circuit performances. Also check for variations in external components, like SMD elements. Sometimes (like for AC or DC performances) you can use also the simulator sensitivity analysis for this task.
- Make the ranges extreme enough to really let the circuit fail. Try to understand why the circuit stops to work, like "transistor N8 goes out of saturation."
- Make sure that all specs are fully understood and that all designers in the team use the same (minimum) ranges on temperature, supply voltage, etc.
- Include known important worst-case combinations ASAP, like lowV_{DD}+SlowMOS+fastR which are often most critical for saturation.
- In case of convergence problems, consider a transient analysis and sweep the parameter over time (like temperature or supply voltage). Sometimes you need special features to do so, the so-called dynamic parameter or Verilog-A models.
- Make your testbenches as realistic as possible, but step by step, e.g., include neighboring blocks, add a package model, and insert estimated wiring capacitances.

2.11 Summary on Worst-Case Corner Search

In subsection 2.8.2 we described methods which combine designer's knowledge with standard techniques, and we get some improvements on speed and reliability, e.g., demonstrate on a CMOS inverter. However, also these methods can fail, e.g., in difficult cases, like our op-amp gain peaking example. In this example, we have also nicely seen that too *greedy* methods (like OFAT or OFAT around an expected WC) can fail even quite dramatically, whereas modern adaptive methods work (at least) almost satisfactorily. So for the topic of WC finding, tools can outperform designer's anticipation capabilities; that is just why we have them.

On the other hand we have seen that clever tool setup *can* provide big improvements on time and accuracy. So if the circuit simulation runtimes are long, such methods "inspired by manual techniques" make still sense, because in some cases methods *without* any *a-priori* knowledge cannot really compete on speed, even if they are adaptive. Related to simulation effort, we have a linear grows with number of parameters for OFAT methods, whereas the reliable full-factorial method has an exponential effort. For moderately

difficult problems, we can expect that advanced, adaptive methods have *roughly* a quadratic behavior; so the advantages over full-factorial become larger the *higher* the *complexity*. This is again a strong argument of just using such methods.

Of course, one can think of performing a much bigger, more representative benchmark [Guerra-Gomez2015], but the result would not be so much different compared to our few examples; and also the criteria weightings might be different. For instance, if your company has a huge compute server, the speed in terms of *number* of simulations of a WC search algorithm would not matter so much. Here, the ability to run the simulations *in parallel* matters more, and in this case, fix (non-adaptive) algorithms like full-factorial have clear advantages. When the problem is *extended* to include also the statistical worst-case (Chapter 7), also big servers will be pushed to their limits—even more with the inclusion of circuit optimization. Only fix strategy algorithms have the advantage of *full* parallelization capability for simulations, so with a huge compute server full-factorial would be even faster than the fanciest adaptive algorithm. However, usually practical reasons prevent using the full-factorial method, e.g., you typically do not want to bother your colleagues too much by taking the exhaustive approach and occupying the full server for a "stupid" block verification.

In Chapter 11 we will give an outlook on further techniques, not yet available in commercial design environments, but the question is usually: Aren't the universal, pragmatic solutions we have already good enough? Or is the problem so difficult to design and to simulate (like finite element simulations) that more specific methods are worth thinking?

For circuit design the already available WC corner methods are indeed fine for all pragmatic engineers. Therefore probably more research and development efforts will go into other directions, for dealing with more complex problems!