

EE 391 Report
written by
Clayton Daigle
for
Dr. Boris Murmann

Winter 2006
Stanford University

A Summary of
*High-speed D/A Converters: from Analysis and Synthesis Concepts to IC
Implementation*
By
Konstantinos Doris

This report is a summary of *High-speed D/A Converters: from Analysis and Synthesis Concepts to IC Implementation*, a thesis by Konstantinos Doris at the Eindhoven University of Technology, Eindhoven, The Netherlands. This particular thesis was singled out for exploration for two reasons: first it represents state-of-the-art work in the area of high-speed CMOS DACs, and second it includes exposure and analysis of many different techniques currently employed in this area. Thus it is both an interesting study of a particular design, as well as a nice collection of wisdom on the subject in general. This report fulfills the requirements for completion of EE391, an individual study course for graduate students at Stanford University. The author of this report is pursuing research in the area of high-speed Digital to Analog conversion.

Introduction

In *High-speed D/A Converters: from Analysis and Synthesis Concepts to IC Implementation*, hereafter called the thesis, the author describes the design and performance of a state-of-the-art Current-Steering Digital-to-Analog Converter (CS DAC), implemented in 0.18 μ m CMOS.

Technology		0.18 μ m CMOS
Maximum rate		600 MHz
Resolution		12 bits
INL / DNL		1LSB / 0.6 LSB
SFDR at Nyquist	Low sample rates	78 - 80 dB
	$F_s = 100\text{MHz}$	74 dB
	$F_s = 200\text{MHz}$	70 dB
	$F_s = 300\text{MHz}$	66 dB
	$F_s = 400\text{MHz}$	65 dB
	$F_s = 500\text{MHz}$	60 dB
Power		216 mW
Area		1.13 mm ²

Figure 1. Performance summary

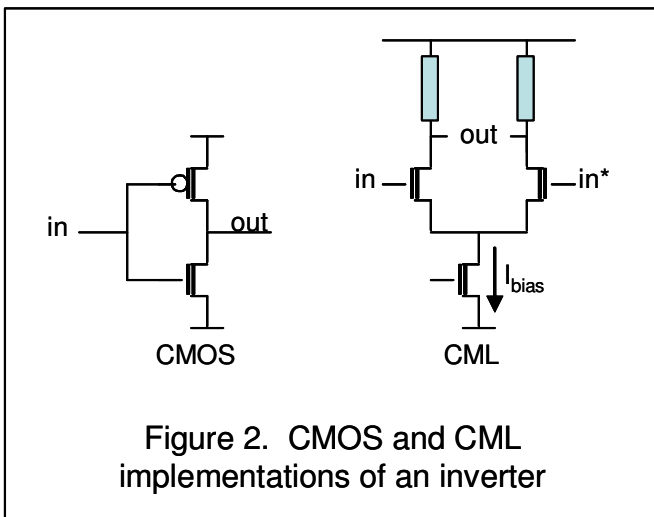
The final performance of the design, in working silicon, is quite good. Figure 1 gives some highlights, which validate the techniques and methodologies employed by the author throughout the design process. The most impressive aspect is how well performance, and most notably SFDR, is maintained at high frequencies (there is an excellent graph of this in the thesis that I cannot reproduce because I do not have access to the original data). Most DACs tend to show a much more rapid decline in performance at high frequencies than this one does. On a conceptual level, the final design is an excellent melding of contemporary thinking with innovative ideas. In fact, the author develops several techniques that are either new, or are newly applied to this space. On an equally respectable level, he does a thorough evaluation of any established ideas he borrows from literature and takes full responsibility for their correctness in his design. In short, he gives the reader confidence that all design decisions are purposeful and backed by good engineering.

In this report, we will focus more on the unique innovations contained in the design than on other contemporary ideas, though even basic concepts are mentioned for completeness. Three primary innovations seem to be most directly linked to the performance improvements achieved. These are 1) better understanding of timing errors, 2) logic implementation in CML, and 3) improved mapping schemes. Each of these are expanded upon later in the report, but I would like to quickly describe them here in the introduction in order to highlight their importance.

Timing analysis improvements: The author repeatedly expresses grief at the lack of timing considerations that accompany most DAC analysis in literature. In an attempt to bring some kind of rationalization to the dark art of DAC design (or perhaps to analog design in general) the author delves into the effects of timing error with a refusal to settle for qualitative understanding. Without this effort, he would have no way to make intelligent tradeoffs in the face of contradictory advice. Here is a simplified example of what this means: It is well known that many kinds of MOSFET matching errors can be improved by using larger devices. So to improve matching between two MOSFET switches, we should make these devices large, correct? Unfortunately, the increased gate capacitance of these larger devices slows down control signals and exacerbates timing errors. So whether or not bigger transistors are beneficial depends on whether or not the resulting timing errors are of more concern than other types of matching errors. Without quantifiable relationships among the design parameters involved, it is impossible to make an intelligent decision on this matter.

Another important distinction in this regard is the difference between global timing errors and local timing errors. Global timing errors are those that can be modeled as uncertainties in the input clock to the DAC. When an update pulse is received, there

will be a finite delay before the DAC can update its output. Global timing error refers to the variation in this delay, and it can be modeled as input clock jitter. Although analysis of clock jitter is well understood in signal processing, it really only represents a simplified picture of what is happening within a CS DAC. As you will see, CS DACs consist of many (perhaps hundreds) of current sources that are each switched on and off to generate different outputs. Local timing error refers to the timing differences of these internal switches. In some simplified analysis, it may make sense to lump all this variation into a kind of global quantity and model it as clock jitter, but when designing a DAC it is a level of detail that we do not wish to discard. As an example, it has been shown in literature that doubling the update rate of a DAC while keeping the output frequency constant (oversampling) reduces jitter-induced distortion by 3dB. But when accounting for *local* timing errors we get the reverse effect: doubling the oversample rate *increases* distortion by 3dB! This is a completely logical result because a higher sample rate increases the amount of time spent transitioning to new values, rather than producing particular ones. The author devotes considerable effort to quantifying the effect of local timing errors on the performance metrics we care about.



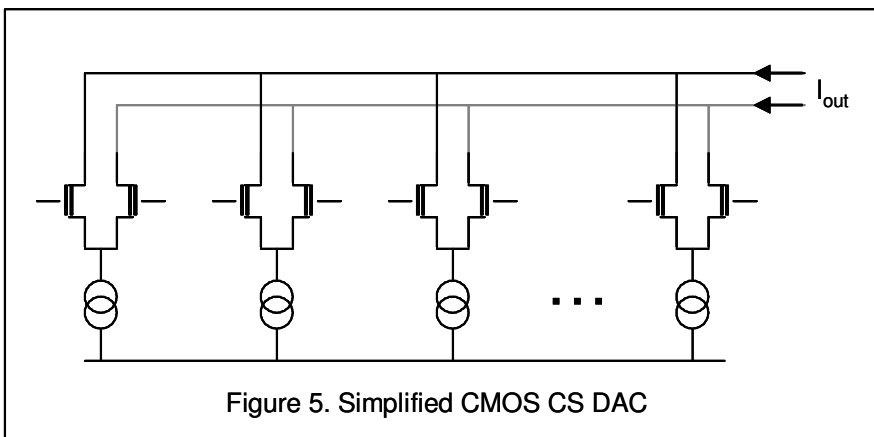
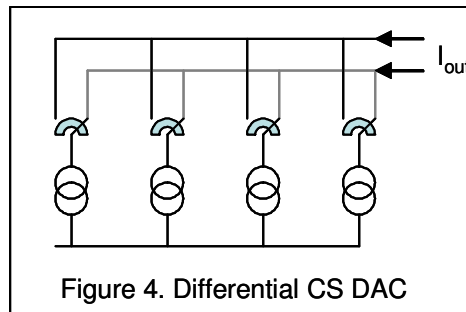
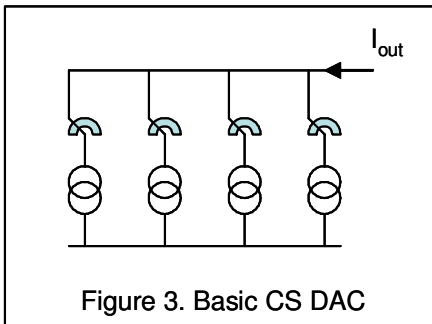
CML logic: CML stands for Common Mode Logic (the term “CML logic” is commonly used, despite the obvious redundancy). The easiest way to understand CML is to examine how its transistor level implementation differs from CMOS. Figure 2 shows an inverter implemented both in CMOS and in CML. CMOS logic is by far the more prevalent in DACs often because designers are very comfortable with it. It has the advantage of dissipating no power in a static state. CML, on the other hand, dissipates power in either logic state since the tail current, I_{bias} , always flows. It also leaves less headroom for operation due to a greater number of devices in series between the rails. What, then, makes CML attractive? First, the “zero power dissipation” tag for CMOS quickly disappears under dynamic switching conditions, and at high-speeds CML can

actually consume less power than a CMOS counterpart. More importantly in mixed signal applications, the fact that CML pulls nearly-constant current means that it does not inject as much noise into the supplies and substrate as CMOS logic. This minimizes data-dependent power supply modulation, which can cause unwanted distortion. The author's use of CML is unique for DACs in this class.

Improved Mapping Schemes: CS DACs are implemented, at least at some level, using an array of current sources. Depending on the implementation, these sources may be equally weighted or binary-weighted or some combination. At the physical level each source may be further broken into several parallel units spaced around the chip in some fashion, which is done to 'average' out spatially-correlated errors. *Mapping* refers to how these individual unit elements in the array are mapped to different input signal codes. For example, imagine a 5-bit DAC implemented with 32 identical current sources. Ideally there would be only 32 possible output values, each proportional to how many of the 32 sources are "on" at a given time. But if these sources are not identical, due to fabrication tolerances or thermal gradients or other imperfections, then there may be better or worse sources to use for each output value. For example, to generate the midscale value, half (16) of the currents need to be turned on. In theory, if the sources are identical, any 16 can be chosen with equal results. But in reality, small mismatches will cause every unique combination of 16 to yield a slightly different result. Within these results, there will be a particular combination that will optimize for a given goal, such as linearity or absolute accuracy. It is the map that decides which current sources in the array will be combined when generating any value. The author of the thesis uses a novel mapping scheme in order to overcome matching errors not only in the magnitude of the current sources, but also in the timing of current switches. This results in mapping that is optimized for improvements at both low and high frequencies.

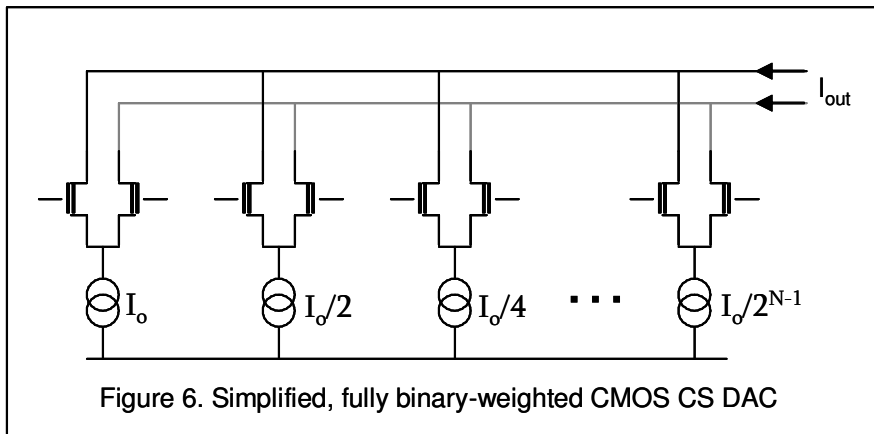
CS DAC Basics

A simple picture of a basic CS DAC is shown in Figure 3. It consists of an array of current sources, each connected to a common output via switches. Although pictorially simple, such a device would be clumsy in practice since the current sources do not have anything to pull current from when they are not being used. Fortunately, life gets a little easier (and the term “current-steering” makes a little more sense) in the context of Figure 4. In this DAC, the current sources are always operating and the switch simply steers the current to one of the two output nodes. If both output nodes are used together, then this architecture can claim the lofty title of “differential” CS DAC. Such a configuration is actually quite easy to realize in CMOS, partially because MOSFET transistors make such excellent switches. Figure 5 shows conceptually how MOSFET differential pairs can be used to realize the switches. In addition, the current sources can be easily realized in CMOS, though of course all the biasing and detail work involved in the final implementation quickly clouds this clean picture.



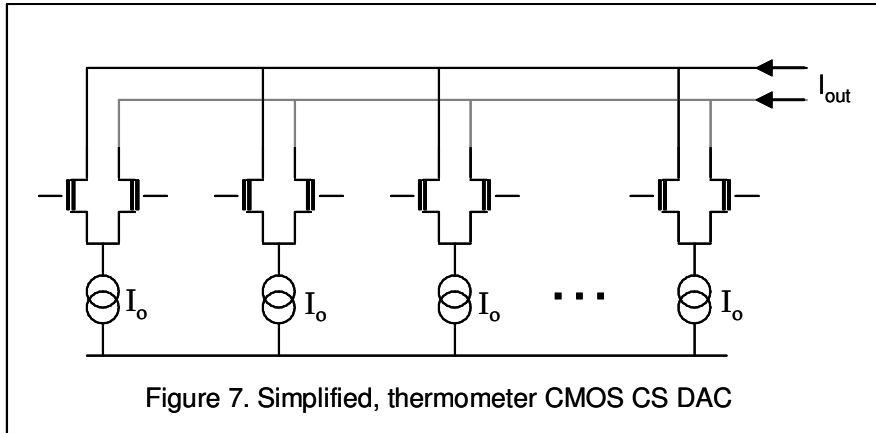
Binary-weighted DACs

The thesis covers the design of a high-speed, high-resolution CMOS CS DAC. Nearly all DACs of this type employ some degree of binary weighting in the current sources. Figure 6 shows what is called a “fully binary-weighted” CS DAC. One current source per bit is required (for example you need 12 current sources for a 12-bit DAC), and each current source is precisely twice the magnitude of the next smallest current



source. Binary-weighting allows you to generate the maximum number of unique, evenly-spaced outputs given a certain number of source elements: $2^{12} = 4096$ unique outputs for a 12 bit DAC. Unfortunately, such strict adherence to binary weighting is not useful in practice due to matching limitations. To see why, consider that the largest current source must be $2^{11} = 2048$ times the size of the smallest current source with an error margin of only ± 1 in order to guarantee monotonicity. That is, the largest current source must be exactly between 2047 and 2049 times the size of the smallest source. Such 0.1% matching isn't feasible in fabrication and certainly not across temperature gradients.

At the opposite end of the spectrum, we have what is called the thermometer DAC, pictured in Figure 7. The thermometer DAC does not quite share such a matching burden because the source elements are equally-weighted instead of binary-weighted. The primary disenchantment with it is the number of current sources required for implementation. A 12-bit thermometer DAC must contain $2^{12} = 4096$ source elements, compared to the 12 needed for the binary-weighted version. To generate a particular output, the thermometer DAC simply turns on that many sources. So if the desired output is code 1000, then 1000 of the current sources get turned on. Monotonicity in this case is trivial since turning on one more current source will always increase the output current. However, monotonicity is not always the only type of linearity with which we are concerned. Think of monotonicity as a crucial specification for very small signals. If a signal is only a few bits in amplitude, then



having codes that go the wrong way would be catastrophic. But it is sometimes the case that small non-monotonic steps can go unnoticed in a large signal spanning the full range of the DAC. In that case, we may be more concerned about the overall shape of the DAC transfer function. Integral Nonlinearity (INL) is the specification that quantifies, at least loosely, how large signals are corrupted by DAC nonlinearities. Binary weighted DACs run more risk than thermometer DACs of being non-monotonic, but neither type has any refuge from INL. INL is a function of all the accumulated uncertainties in all the current sources used to generate a given code, and it is completely unconcerned about how those sources were individually weighted. Hence, INL affects equally both binary-weighted and thermometer DACs.

In practice, most designs fall somewhere between these two extremes, using thermometer weighting for the coarse (more significant) bits and binary weighting for the fine (less significant) bits. The hard part is deciding how many bits are going to be binary and how many will be thermometer. This key architectural decision is called bit partitioning, and it will be the first thing we examine as we analyze the DAC realized in the thesis.

Architecture and Physical Implementation of a CS CMOS DAC

Bit Partitioning

Often one of the first decisions that must be made when architecting a CS DAC, is the partitioning between binary and thermometer sections. Several opposing strategies come into play, and we need to be aware of a few more things before we get too far. Generally, concepts like monotonicity and INL, as discussed earlier, are applied in the context of DC or low-frequency signals. But when generating high-frequency signals, DACs can be even more insidious due in part to glitches. Consider the mid-scale transition of a 12-bit DAC. “Mid-scale” means the step from code 2047 to code 2048, which is midway between 0 and the full-scale code 4095. This is only a 1-bit step, so the jump in output current should be very small. But consider how these values look in straight binary representation. On that level, this is a step from 011...111 to 100...000. In a strictly binary-weighted implementation of a CS DAC, this corresponds to a transition from having every current source turned *on except* the biggest one to having every current source turned *off except* the biggest one. Monotonicity is obviously in jeopardy due to matching – you must guarantee that the largest source matches the sum of all the other sources, with a difference equal to the minimum step size. But even if the current source magnitudes are all perfectly weighted, timing errors can cause trouble of their own. Think about what happens if the MSB turns on before the other bits turn off. This means that momentarily, *every* source is on – a full scale output! Repeating for clarity, when making the transition from 2047 to 2048, we accidentally generated 4095 for a brief instance. This erroneous burst of current is called a glitch. It is easy to imagine such a thing actually happening since the current switches will be different sized transistors, making it hard to achieve identical switching characteristics for all bits. In reality, glitches happen very quickly and never quite reach such obscene magnitudes as we’ve imagined here. But nonetheless, glitching dumps observable, code-dependent, unwanted charge onto the signal every time the DAC updates. The faster the update rate, the more glitch energy you must deal with, which is why glitch energy can be such trouble at high frequencies. In the past, people have employed deglitching circuits to swallow extra output current during updates, but they often cause more harm than good because they must be in the signal path.

Thermometer implementations, do not produce as much glitch energy because only one switch changes state for a step of one bit. Thus thermometer implementations may seem preferred for both linearity and glitch reasons, and in fact, some literature suggests using as many thermometer bits as space and power allows. However, this advice is only valid provided that clocking errors are constant. Having so many current sources means having them spread out farther and having more interconnect

among them. Thus, increasing the number of thermometer bits comes at the cost of synchronization. In addition, you must also account for the increased complexity of the decoding logic when implementing thermometer DACs. Binary decoding is trivial: each input bit corresponds to a particular current-steering switch. Thermometer decoding is complex because a small number of input bits must fan out to a large number of switches (according to the mapping strategy). All of this additional decoder logic causes power supply disturbances and substrate noise.

In order to help resolve the bit partitioning decision, the author first develops relationships between desired performance parameters and local timing errors. The rationale is then that we should follow established advice of increasing the number of thermometer bits, but stop at the point that the loss of internal synchronization degrades performance. Assuming we can estimate the local timing error along internal timing chains, the following expression can be used to estimate signal-to-distortion ratio (SDR).

$$\text{SDR} = 3.01(N - 1) - 10\log\left(\sigma^2 f_i \cdot f_s\right) - 9.03\text{dB}$$

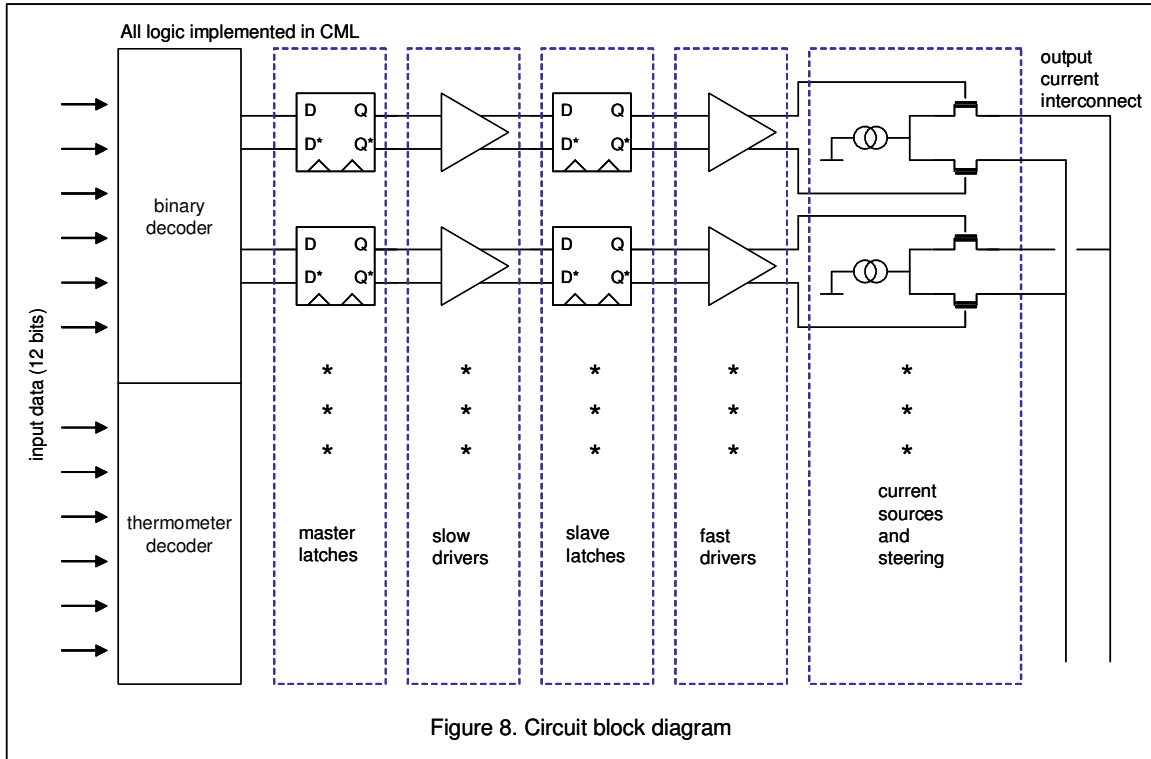
In this equation, N is the DAC resolution, σ is the local timing variation, f_i is the signal frequency and f_s is the sample rate. Local timing errors can be estimated based on floor-plan and anticipated device sizes and can be simulated using circuit subsections (much easier than having to simulate the entire DAC and measure SDR).

In the end, bit partitioning for this design balanced concerns over glitch energy, timing precision, supply disturbances and area/power consumption. The final work is partitioned into 6 binary bits and 6 thermometer bits. Complete justification for this decision is quite lengthy and was based on numerous simulations, therefore we cannot present it here. In practice, most levels of partitioning can be quickly discarded for apparent reasons, leaving only two or three up for serious consideration. In this case, the author only analyzed cases with between five and eight thermometer bits, before deciding on 6.

Buffering between digital and analog sections

Now we are ready to go through the rest of the design piece by piece. Figure 8 shows a high level diagram of the DAC. On the left are the input latches and decoding logic. This DAC is double buffered, giving it two levels of latches between the digital logic and the current-steering switches. Such isolation is highly desirable, even though it is often neglected in designs. Analysis by the author shows that pulse shapes delivered to the current switches can be corrupted due to poor isolation, which may significantly degrade performance. Carefully controlled CML is utilized in the latches

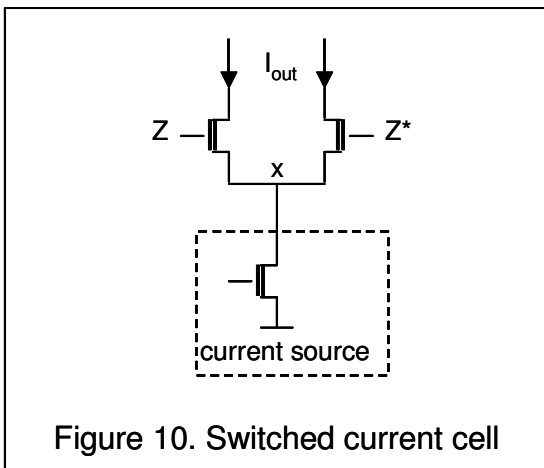
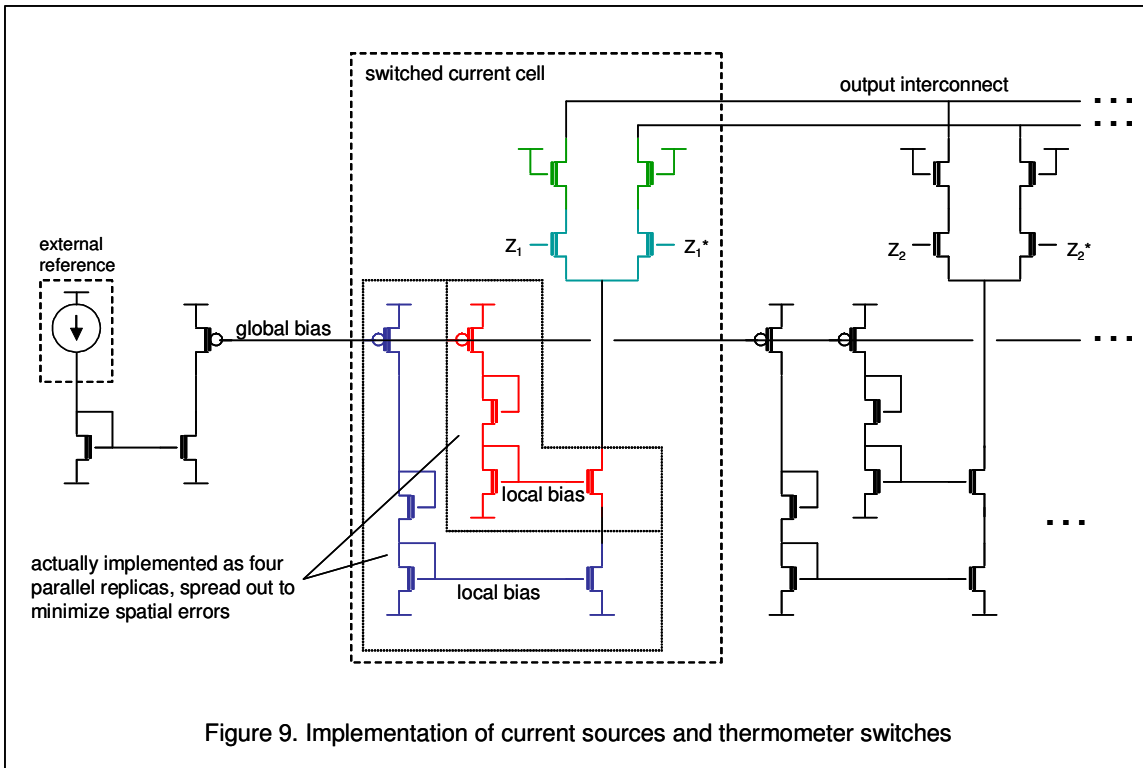
and drivers in order to provide the cleanest possible signals to the switched-current cells.



Biasing the current sources

Figure 9 gives a detailed picture of the switched-current cells. Notice that a single, global biasing node fans out to several cells in order to convey biasing information. Such global biasing is obviously necessary to achieve matched current sources across the chip, but they can also be a means for unwanted communication among various parts of the circuit. To ameliorate this situation, the author only uses the global biasing node to generate local biasing nodes, which in turn directly interface the transistor-implemented current sources. This adds a degree of separation between the global biasing nodes and the local dynamic nodes within the switched-current cells that could corrupt them.

A related area of concern is the modulation of current sources by their respective switches and by other cells. Any single-transistor current source exhibits a strong dependence on V_{ds} , thus any changes to the load characteristics driven by such a current source will cause modulation of the output current. This modulation can be examined at the X node in Figure 10. This figure represents the simplest possible implementation of a switched-current cell. The differential pair, driven by signal Z,



is the current switch and the tail transistor is the current source. The voltage at the I_{out} node is dynamic (it is the output node of the DAC, carrying a signal of 100's of MHz). As you can see, node X is not well isolated from the DAC output, nor from the switching action of Z. Modulation of V_{ds} by the output signal is a recipe for distortion, therefore greater isolation must be obtained. As a first level of defense, the author chooses to cascode the current source as shown in Figure 9.

In addition, several other axis of isolation are utilized to reduce modulation of the current source both by the output signal and by other cells. This is where local

biasing is key. The red transistors are the cascode transistor along with its local biasing, while the blue transistors are the current source transistor, along with its local biasing. The local biasing circuits keep the global biasing node from being modulated by dynamic signals within the individual cells. You may be tempted to combine the two local biasing circuits together (combine the red and blue), but that would provide a way for switching disturbances to modulate the current source.

Sizing the current sources

Sizing the current source transistors may seem like a difficult task and, well it is. But fairly basic concepts can take us a long way in understanding this issue. We know that, all other things being equal, MOSFET saturation current scales linearly with device width. This means that each of the thermometer bits can be realized using transistors of equal width, while the binary-weighted bits can be realized using transistor widths that decrease by powers of 2. Since 6 bits are binary-weighted, the largest transistors would have to be $2^5 = 32$ times bigger than the smallest ones. In practice, better matching is achieved by letting the smallest transistors become what we call the unit size and building the larger transistors from multiple parallel copies of the unit device. All of these unit devices, laid out in a grid, is what gives rise to the concept of a current array (and mapping is what decides which devices are connected in parallel to comprise each output level). Because all transistors, regardless of size, scale according to the unit device, sizing the current sources actually boils down to sizing the unit element. How do we do this?

Earlier, we mentioned that monotonicity requires the largest source to be $2^{N-1} \pm 1$ times what are now calling the unit size. For our 6-bit binary-weighted section, the largest current source will be made of 32 unit elements, and its value must be between 31 and 33 times the average unit current. Now suppose that the unit current elements have a normalized standard deviation of (σ_I/I) . If we sum M of them, then the standard deviation of the resulting current will be $(\sigma_I/I) \sqrt{M}$. In prior work, it is shown that an approximate expression for the INL of an N-bit converter based on the uncertainty of the unit element is

$$\text{INL} = \sqrt{2^{N-1}} \cdot \left(\frac{\sigma_I}{I} \right)$$

where N is the DAC resolution. Although we have not derived this equation, I believe it should be very satisfactory on an intuitive level. All we really need, then, is a way to relate the size of the unit devices to the current spread of the unit devices. Thankfully, this has already been done, yielding the following expression:

$$WL_{\min} = \frac{1}{2} \left(A_{\beta}^2 + \frac{4A_{VT}^2}{V_{ov}^2} \right) \div \left(\frac{\sigma_I}{I} \right)^2$$

Where A_{β} and A_{VT} are technology-dependent uncertainties that we do not have control over. It is immediately obvious that if we demand extremely tight tolerances on current errors (if we try to make (σ_I/I) very small) then the corresponding devices may need to be very large. We also see that increasing V_{ov} can improve matching, but this uses more power and reduces headroom. L is strongly influenced by output impedance concerns. And keeping W small limits drain capacitance, which, as we will see shortly, affects output impedance modulation. Balancing all these ideas, the final values chosen were $W = 3\mu\text{m}$, $L = 16\mu\text{m}$, and $V_1 = 0.85\text{V}$ ($V_{ov} = 0.5\text{V}$).

Finally, we should mention that even each unit element is actually made of 4 separate transistors, one placed in each of four sub-arrays. This is done to cancel certain types of matching errors and is covered in more detail in the section on mapping.

Designing the switches

There are several important things to consider when designing the switches. First, they must provide some isolation between the output node (which is again moving at 100's of MHz) and the current sources. Therefore, a buffer, which can also be thought of as a cascode, is used at the output so that the switch drives a more-constant load. Second, charge feedthrough (via mostly overlap capacitance) can couple the switch control lines to the output. One straightforward way to minimize this effect is to limit the size (and hence the capacitance) of the switches. But does this adversely affect switch matching? In order shed light on this issue, we go back to our simplified matching problem that we discussed in the introduction. We said then that the spread of certain mismatches is reduced when using larger devices. In a differential pair that is being used as a switch, we are interested in the concept of the effective offset voltage. If two differential switches have different offset voltages, then they will not switch at the same time when driven with identical signals with finite slope. It is important to have switches that can be synchronized, which means we would like to have a small spread of offset voltage. An equation that describes the spread of offset voltage in a differential pair as a function of device size is

$$\sigma_{V_{\text{off}}} = \frac{A_{VT}}{\sqrt{WL}}$$

Bigger switches, then, will have better offset matching, which helps synchronization. But what about the fact that this also slows down control signals? Remember, a large

device has much more capacitance. For a given gate drive, it will not turn on and off as sharply as a small switch. Does driving the extra gate capacitance offset the benefit of better offset voltage matching? The timing spread due solely to input capacitance is given by

$$\Delta t_s = V_{\text{off}} \frac{C_{\text{in}}}{I_{\text{drive}}}$$

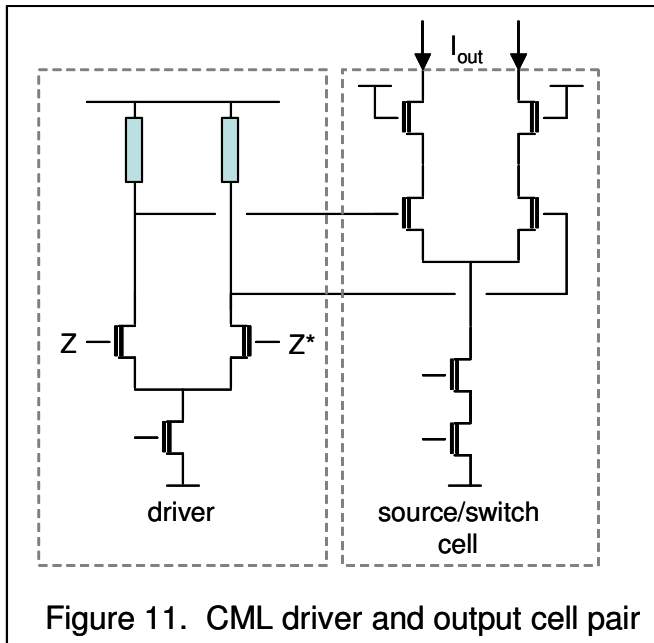
Using long channel approximations, we can combine these two equations into the following:

$$\sigma_{\Delta t_s} = A_{\text{VT}} C_{\text{ox}} \frac{\sqrt{WL}}{I_{\text{drive}}}$$

This says that increasing the size of the switches is, on the whole, actually bad for synchronization. So while using larger devices improves offset matching, it does not do so to the degree that it will overcome timing mismatch due to larger gate capacitance. The author also verified this relationship with simulations, which indeed show that increasing device width by a factor of 4 roughly doubles the timing uncertainty. Combined with the fact that a small device also limits feedthrough, we are compelled to make the switch small.

Switch Drivers

Earlier we mentioned that current switches driven by distorted pulse shapes can degrade performance, and one mechanism for this is feedthrough via gate capacitance of the switches. Noisy or otherwise unruly pulses can inject noise right into the output. But even clean pulses can generate output spikes if they are too sharp. Therefore, it is often desirable to slow down the switch signals. How can this be done? Remember all the logic is implemented in CML, and a very nice thing about CML is the control of edge rates and common mode voltage. Figure 11 shows a CML implementation of a driver-switch combination. By changing the value of the driver tail current, you can change the edge rate of the switch signal. Faster signals correspond to smaller timing skew (good) *and* increased capacitive feedthrough directly to the output (bad). These opposing constraints must be balanced in a thorough design strategy. In this case, a bias current of 50uA was chosen for the slow drivers (see Figure 8 to find the “slow drivers”) because synchronization is not critical there. A zippier 250uA was used for the fast drivers that directly interface the

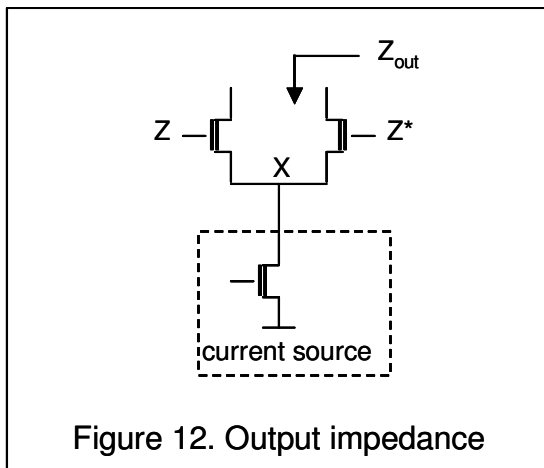


switches. As with the partitioning, full justification for these values cannot be given here since it is based on repeated simulations.

Before moving on, I would like to mention an interesting consideration. What happens if the switching is not symmetrical and both transistors in the differential switch pair are both off at the same time (the one turns off before the other turns on). If this happens, no current will be available for the current source and it will pull the tail node down until it goes out of regulation. Recovering from such an event can take a long time. On the other hand, if both transistors are on at the same time (the one turns on before the other turns off), then the penalty is far less severe. The current source will see some intermediate load, but will not go out of regulation and the common mode voltage will not be disturbed as greatly. Therefore, it is beneficial to try to make sure we err on the side of having both of them on at the same time, rather than off. This allows for another benefit of CML. The higher common mode voltage operation of CML inherently increases the likelihood that both transistors will be on momentarily. In addition, the common mode output voltage can be raised further by adjusting the loads.

Output Impedance

So far, we have not said anything about DAC output impedance. Consider Figure 12. Except for brief periods during switching, we normally have one transistor on and one transistor off. What is the impedance in each state? Looking only at capacitance, we see that for the side that is off, there is only the parasitic capacitances of c_{dtot} , though this is complicated by the dependence of C_{db} on drain voltage. And for the



side that is on, we get additional capacitance at the x node, which we simply call C_x . The reactance seen in each state is then:

$$C_{\text{off}} = C_{\text{gd}} + C_{\text{db}}(V)$$

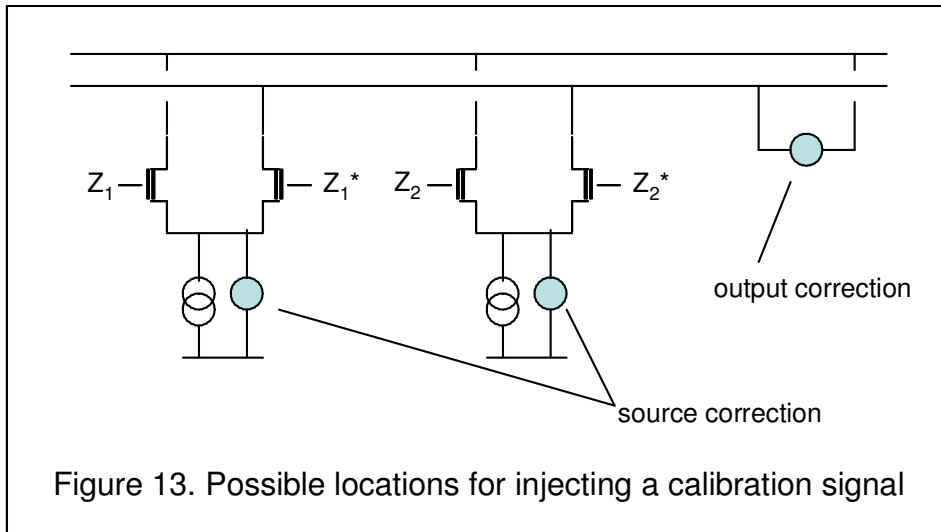
$$C_{\text{on}} = C_{\text{gd}} + C_{\text{db}}(V) + C_x$$

Considering that the output impedance is made of many of these switches in parallel, we end up with the result that the total output impedance of the DAC is strongly affected by the current you intend to generate. Data-dependent output impedance modulation is certainly undesired. The situation is mitigated to some degree by the fact that we use the output differentially. Much of the even order distortion cancels out, but this certainly does not solve all our problems. This is another reason to buffer the switch outputs with cascode transistors. In addition, dummy switches are connected in parallel that act complementary to the main switches in order to maintain a consistent output impedance. Refer again to Figure 9 to see the final implementation of the current sources and switches (dummy switches not shown).

Current Source Layout and Mapping

The layout of the current source array is closely related to mapping. In fact, the current source array is really just a patterned arrangement of transistors, and the map is what actually assigns each one to a specific conceptual “source.” Mapping was briefly described in the introduction, and we will not go into that much greater depth here as there seems to be no logical way to present any more detail without presenting all of the detail, defeating the purpose of a summary. But we can do a slightly better job of motivating the mapping concept.

Mapping is a statistical way to desensitize overall circuit performance to some types of matching errors. It works by interleaving functionality across a chip so that location-



dependent errors, such as uneven doping or thermal gradients, will affect all functions in the same way and can thus be canceled out. Since it is a statistical technique, it cannot guarantee anything. It just mixes things up enough so that hopefully most errors will cancel. Are there not more definitive ways to cancel errors, such as through calibration? How could a CS DAC be calibrated? We would need to inject some current somewhere in order to correct for errors. Figure 13 shows two places where correction currents could be injected, and we will consider each.

Injecting current at the output is difficult if you keep in mind that you have to have different correction for every output value. Thus the correction element would, in fact, need to be a high-speed DAC in its own right, capable of applying correction currents just as fast as the main DAC. A far simpler approach is to inject correction currents at each current source. Now the calibration circuitry is removed from the signal path and the correction currents do not need to be dynamic. This actually can deliver perfect adjustment to the current sources for all output values, yet this is not the solution the author chose in the end.

Mapping really is an alternative to calibration or perhaps a type of calibration. Calibration normally is explicit: we measure the output error and adjust the current sources until something meets our specification. Mapping is not explicit, we just mix things up in the best way we can think of and hope things cancel. Clearly the explicit calibration is better? Not so. There are several practical concerns that we would have to deal with, such as the enormous increase in complexity in having all these extra calibration DACs everywhere. But we will not even have to invoke the “too complicated” clause if we consider timing. The switched-current cells not only have current magnitude mismatches, but they also have timing mismatches. At high-speeds, the timing mismatches dominate. The map developed by the author both

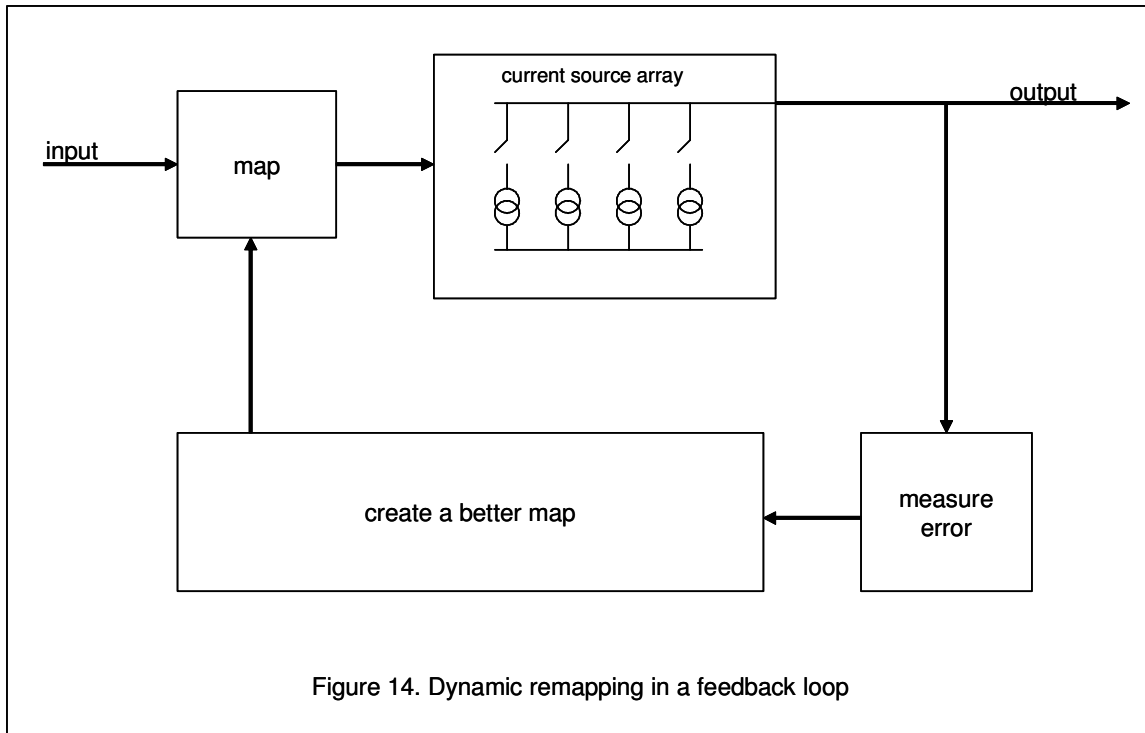


Figure 14. Dynamic remapping in a feedback loop

increases the likelihood that current magnitudes will match, and increases the likelihood that timing will match as well.

Mapping can get quite fancy. So far we have mainly had in mind static maps, but maps could also be dynamic. The concept of map hopping means a DAC changes maps repeatedly, perhaps cyclically or randomly. Each map itself may be patterned in some way or may be completely random. In fact, random map hopping is used within some sigma delta converters because random noise can usually be dealt with more easily than coherent noise. To make things more complex, a DAC could run a self-calibration routine that generates an optimal map to compensate for a new environment. For example if it were placed near a warm power amplifier or if it undergoes some package stress due to PCB flexing. An interesting possibility is shown in Figure 14, which shows how mapping could be used in a feedback loop.

In the final design, the author does choose to use only a static map. For one thing, the complexity that would result from having enough multiplexing capabilities to map any current source to any output would likely destroy dynamic performance. The final mapping strategy is fairly sophisticated. Based on prior research done by the author, a map is generated that outperforms previous attempts that were based on symmetry alone. Figure 15 shows the final chip layout. The map, which cannot be shown, determines how the sub arrays are combined.

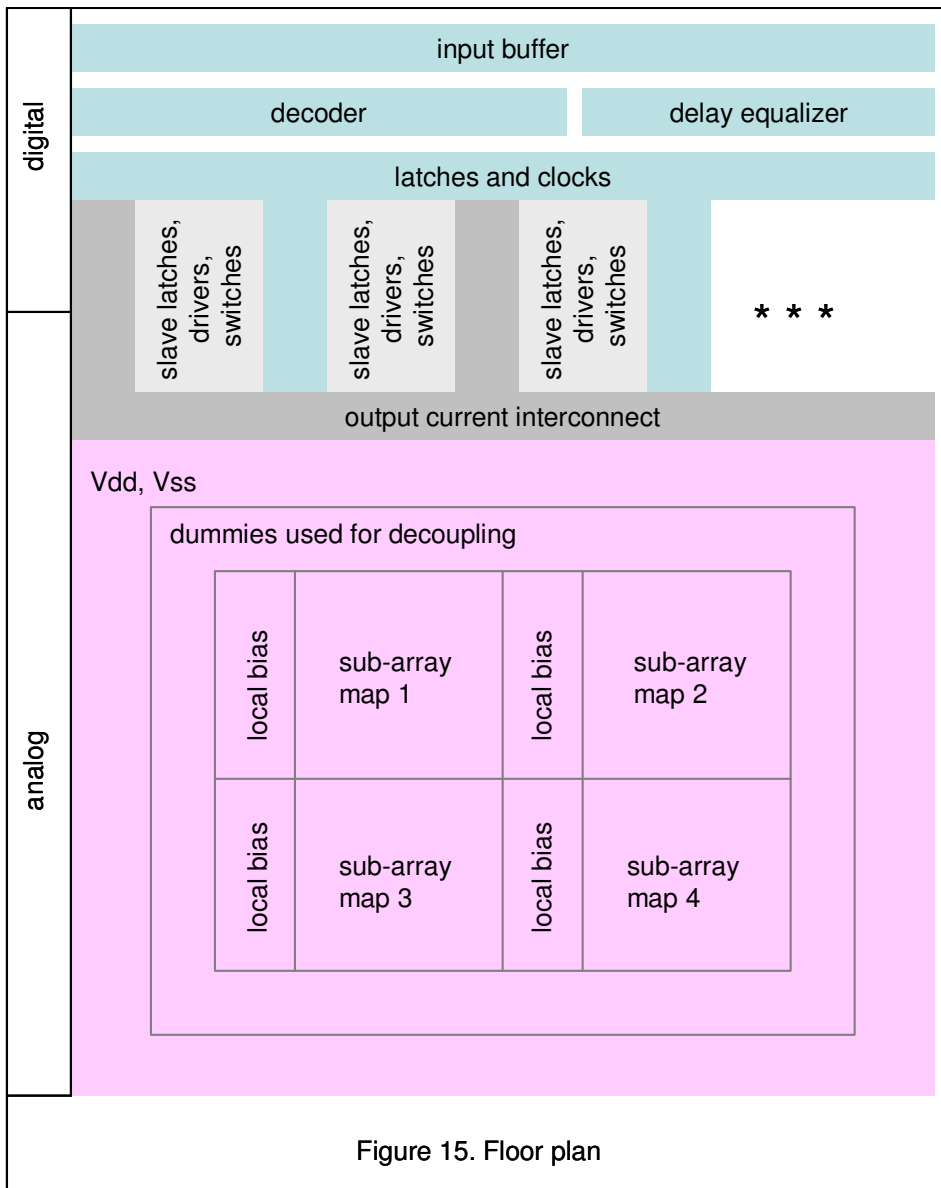


Figure 15. Floor plan

Summary and conclusions

In conclusion, the thesis is an excellent display of good design practice and innovation. Reducing it to only a few pages has proved a difficult challenge and I apologize for the many details that had to be skirted in order to suppress the length of this report. I have tried my best to keep the descriptions accurate and to point out where major simplifications are being made. Since I am, as yet, no expert on DAC design, it is certainly conceivable that I have made some logical errors in the way I have presented this material. Much of this material was new to me and writing the report was not so much an exercise in summarizing, but in learning.