electronics

MDPI

Article

Toward Memory-Efficient Analog Design Using Precomputed Lookup Tables

Hesham Omran

Integrated Circuits Laboratory (ICL), Faculty of Engineering, Ain Shams University, Cairo 11517, Egypt; hesham.omran@eng.asu.edu.eg

Abstract: Analog design productivity remains a challenge in the digitally driven semiconductor chip design field. Knowledge-based and simulation-based analog automation approaches have not achieved widespread acceptance in the analog design community. Systematic analog design using precomputed lookup tables (LUTs) is a promising approach that can address the design productivity challenge. Although modern computing systems have powerful memory capabilities, which make the LUT approach viable, reducing the memory footprint of the LUTs remains a challenge. A memory-efficient design technique using LUTs is proposed by using an incomplete grid in the MOSFET degrees-of-freedom (DoFs) space. An efficient indexing technique for the incomplete grid is also proposed, using a precomputed offset array in various scenarios, such as two-sided constraints and three-dimensional LUTs. The results show that the proposed technique can achieve up to a 67% reduction in memory footprint, in addition to improving LUT generation time and query performance.

Keywords: analog design; analog design automation; precomputed lookup tables (LUTs); g_m/I_D methodology; memory efficiency

check for updates

Citation: Omran, H. Toward Memory-Efficient Analog Design Using Precomputed Lookup Tables. *Electronics* **2024**, *13*, 3776. https://doi.org/10.3390/electronics13183776

Academic Editors: Leonardo Pantoli, Costas Psychalinos, Gaetano Palumbo, Esteban Tlelo-Cuautle, Egidio Ragonese and Paris Kitsos

Received: 8 August 2024
Revised: 14 September 2024
Accepted: 21 September 2024
Published: 23 September 2024

1. Introduction

The number of transistors that can be integrated into a single chip is still steadily increasing, as predicted by Moore's law. The FinFET has powered the industry throughout the previous decade, and the vertically stacked gate-all-around nanosheets will power the industry for two more decades. Although CMOS analog/mixed-signal blocks usually represent a tiny fraction of the billions of transistors on-chip, they are the dominant players from the perspective of design time and design effort. This can be explained by noting that, while digital design automation has been the industry standard for decades, analog design automation is still lagging, and the majority of the analog/mixed-signal circuits are still hand-crafted at the transistor level [1–3].

Traditionally, analog designers used the concept of overdrive voltage (V_{ov}) to define the transistor bias point in the strong-inversion region. The overdrive voltage is based on the long-channel square-law MOSFET model. But as the technology minimum feature size has steadily scaled down, transistors have deviated from these simple models, and the designers resorted to lengthy iterations using simulation tools. Not only is this a tedious, time-consuming process but it also leads to sub-optimal designs. The overdrive voltage has also gradually been replaced with a simulation-based design knob, the drain saturation voltage (V_{dsat}). Since the transition from the triode region to the saturation region is a gradual process, V_{dsat} is an ill-defined parameter that mimics the legacy of V_{ov}. With the increased importance of low-voltage and low-power design, more transistors are now biased in moderate and weak-inversion regions. Consequently, the overdrive voltage concept has lost its significance. Other design knobs that describe the MOSFET bias point across all operating regions, such as the gm/ID and the inversion coefficient, have been proposed [4,5].

Analog design automation efforts have been ongoing for more than 30 years. These efforts can generally be divided into two categories: knowledge-based approaches and simulation-based approaches [1–3,6]. In the knowledge-based approach, the designer tries to turn their design procedure into a computer program. This can be useful for "personal" design automation, where the designer automates some of their own time-consuming operations. However, it is difficult to generalize and lacks the accuracy of real simulation models. Moreover, it does not lead to optimal solutions. On the other hand, the simulation-based approach relies on invoking the simulator in an optimization loop. This has the advantages of accuracy and optimal solutions, but invoking SPICE in the loop results in long computation times. Moreover, the resultant point may not make sense from a designer's perspective; thus, the optimization sampling process must be guided by detailed and careful constraints. Hybrid approaches and machine learning/artificial intelligence approaches have also been proposed, but they suffer from accuracy issues when surrogate models are used, or from long computation times when the simulator is invoked in the loop [7–11].

The design of analog circuits using precomputed lookup tables (LUTs) is a promising approach that bridges the gap between the two distant islands of classical handcrafting and black-box simulation-based optimization [12–18]. The precomputed LUTs are generated by a simulator to abstract the complexity of modern device models; thus, simulation accuracy is preserved. The device data in the LUTs can be manipulated in different scenarios to enable the seamless integration of different design methodologies, such as the gm/ID design methodology, which enables intuitive biasing of transistors across all inversion levels [4,5]. The LUTs can be used in a knowledge-based approach to address the accuracy problem at the transistor level. The accuracy problem at the circuit level can be addressed by LUT-based custom solvers in an optimization loop or in a design-space exploration setting. Compared to SPICE-in-the-loop approaches, this will solve the long computation time problem.

The precomputed LUTs are generated once for a given technology; thus, the overhead of the generation process (a few hours per device) is tolerable. However, the memory footprint of the LUTs, which can be up to a few GBs per device depending on the LUT accuracy, is a drawback that the user encounters with every usage. The memory capabilities of computing devices have significantly improved in recent decades, making the usage of large LUTs viable. However, it is still desirable to minimize the memory footprint of the LUTs, especially when designing a circuit that involves a large number of different device types. In this paper, an incomplete-grid LUT is proposed to reduce the MOSFET LUT memory footprint by up to 67%.

The rest of this paper is organized as follows. Section 2 presents an overview of the MOSFET LUTs. Section 3 describes the proposed incomplete-grid memory-reduction technique. Section 4 presents the results and discussion. Section 5 concludes this paper.

2. The MOSFET Lookup Table (LUT)

Figure 1 shows the testbench used to characterize the MOSFET and build the lookup tables (LUTs). An N-type MOSFET is used for illustration purposes, but the discussion applies to P-type MOSFETs as well. The MOSFET has five degrees of freedom (DoFs). The DoFs are divided into two groups: first, the three terminal voltages V_{GS}, V_{DS}, and V_{SB}, and second, the sizing parameters, channel width (W), and length (L). To build an LUT that captures these five DoFs, the LUT would need to be 5D and would have a large size. Fortunately, the MOSFET parameters are directly proportional to the width regardless of the inversion level (weak/moderate/strong inversion) or the mode of operation (triode, pinch-off, velocity saturation). Thus, the LUT can be constructed for a single reference width $\left(W_{ref}\right)$, and linear scaling can be used to calculate the MOSFET parameters at any other width. This linear scaling may incur errors due to stress and narrow-width effects, but these errors can be corrected using small auxiliary LUTs [15].

In the context of the gm/ID design methodology, the V_{GS} DOF can be replaced with the gm/ID ratio in order to zoom in on the region of interest for analog design and, consequently, reduce the LUT size. However, the gm/ID vs. V_{GS} characteristics depend on other variables, such as V_{SB}, and the extreme values of V_{GS} are needed for analog/mixed-signal design (e.g., sizing a sampling switch). Thus, building the LUT using the full range of V_{GS} is necessary.

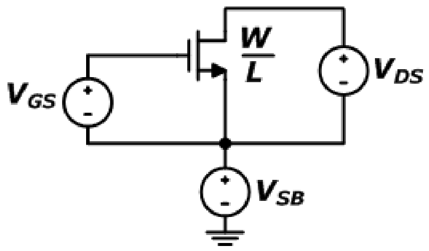


Figure 1. A schematic of the testbench used to build the MOSFET LUT showing the five DOFs of the MOSFET.

By exploiting the linear scaling property, the LUT is reduced to a 4D array, as depicted in Figure 2. Each 4D array stores one of the MOSFET parameters, e.g., drain current, small signal parameters, capacitances, noise parameters, etc. The LUT generation process is automated using a computer program that generates the testbench netlists, parses the simulation results, and stores the data in the appropriate structure. The process can be applied to devices of different types and at multiple temperatures and process corners to fully characterize a technology node. The LUT generation process can take up to a few hours per device, but it is carried out only once for a given technology. Thus, it is a one-time sunk cost to generate the LUTs, which can then be used by multiple designers across different projects.

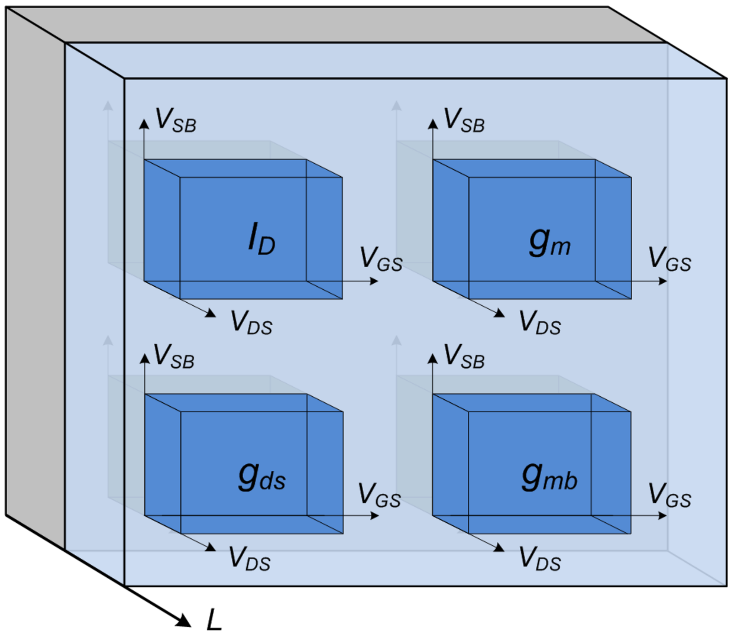


Figure 2. A simplified illustration of a device's LUT structure. The structure contains a 4D LUT for every MOSFET parameter.

An interpolation operation is required when an off-grid point is queried. This represents an inherent accuracy limitation. Thus, building the LUTs involves a size–accuracy trade-off. The step size used for every DoF is the knob that controls this trade-off. Using a fine step size improves the accuracy but results in a large memory footprint. Although the smart interpolation techniques proposed in [14] can relax this trade-off, the overall memory footprint is still significant, especially when designing a circuit that involves many device types across different corners.

3. Memory Reduction Using an Incomplete-Grid LUT

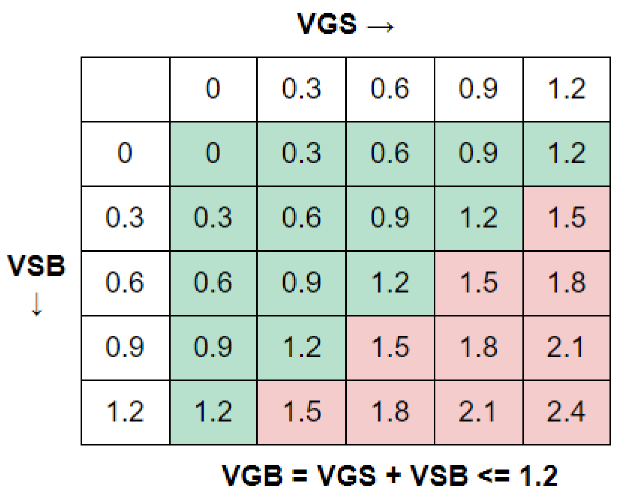
3.1. The Incomplete-Grid LUT

The process of building the LUT involves sweeping the four MOSFET DoFs to create a 4D array for every MOSFET parameter. Each array has gridded data, meaning a value exists at every grid point, where the grid points are defined by the DoFs' sweep points, i.e., the grid vectors. A key observation that can lead to substantial memory savings is that a full grid is not really required. The full grid may contain many values that are not practically needed or may violate the MOSFET safe operating region.

As an example, assume that a MOSFET has the voltage ratings given in Table 1. It is important to note that the sweep variables used to build the LUT are V_{GS}, V_{DS}, and V_{SB}. Thus, the information given for V_{GB} and V_{DB} in Table 1 is not utilized in the case of a full-grid LUT. Figure 3 shows an illustration of a 2D grid using the V_{GS} and V_{SB} grid vectors, with a coarse step of $0.3V$ for illustration purposes. The grid points that do not satisfy the condition of $V_{GB} \leq 1.2V$ are marked in red, where $V_{GB} = V_{GS} + V_{SB}$. It is clear that filtering out these invalid points will result in a substantial saving in the LUT memory footprint. While the observation that an incomplete grid should be used may seem obvious, it has not been reported in previous works discussing LUT-based analog design [12,14,17].

Table 1. Example of a MOSFET device's voltage ratings.

	Minimum (V)	Maximum (V)
V_{GS}	0	1.2
V_{DS}	0	1.2
V_{SB}	0	1.2
V_{GB}	0	1.2
V_{DB}	0	1.2


VGS →
VSB ↓
VGB = VGS + VSB <= 1.2

	0	0.3	0.6	0.9	1.2
0	0	0.3	0.6	0.9	1.2
0.3	0.3	0.6	0.9	1.2	1.5
0.6	0.6	0.9	1.2	1.5	1.8
0.9	0.9	1.2	1.5	1.8	2.1
1.2	1.2	1.5	1.8	2.1	2.4

Figure 3. A 2D full grid using V_{GS} and V_{SB} grid vectors. The value of V_{GB} is shown at every grid point. The green and red colors show the valid and invalid points, respectively.

3.2. Indexing the Incomplete-Grid LUT

The process of evaluating the LUT output at a given query point involves querying the surrounding grid points and then performing an interpolation operation. The query of a value in an N-dimensional full grid requires defining the index of each grid vector, i.e., the index is an N-dimensional vector. For the 2D example shown in Figure 3, it is necessary to define the V_{GS} index (column index) and the V_{SB} index (row index) to identify the query grid point. When the invalid points are removed from the array, it is no longer possible to use the full-grid indexing methodology.

To address this problem, the full-grid array is unrolled into a 1D vector, as shown in Figure 4. The full-grid 1D index (FGi) can be calculated as follows:

$$FGi = V_{GSi} + (V_{SBi} - 1) \times FGS \tag{1}$$

where V_{GSi} and V_{SBi} are the indices in the V_{GS} and V_{SB} grid vectors, and the full-grid stride (FGS) is the number of columns per row, i.e., the length of the V_{GS} grid vector. Note that it is assumed that the first index is equal to 1.

VGS	0	0.3	0.6	0.9	1.2	0	0.3	0.6	0.9	1.2	0	0.3	0.6	0.9	1.2	0	...
VSB	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.6	0.6	0.6	0.6	0.6	0.9	...
VGB	0	0.3	0.6	0.9	1.2	0.3	0.6	0.9	1.2	1.5	0.6	0.9	1.2	1.5	1.8	0.9	...
FGi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**...**

Figure 4. The 2D array in Figure 3 unrolled into a 1D vector. The full-grid index (FGi) is shown for each grid point.

Next, the incomplete grid is unrolled into a 1D vector, as shown in Figure 5. The cells marked in red are shown for clarity, but they do not exist in the incomplete grid. Thus, their index is represented by a dash (-), i.e., they do not have an index. The incomplete-grid 1D index (IGi) cannot be calculated using (1). The stride in this case is not constant and depends on the number of points discarded in each row. The IGi can be expressed as

$$IGi = V_{GSi} + \sum_{i=1}^{j-1} IGS_i \tag{2}$$

where IGS_i is the incomplete-grid stride of row_i, i.e., the number of valid points in row_i. The IGS vector, as illustrated in Figure 6, is an additional overhead vector that is crucial for correctly mapping elements from the original array to the incomplete-grid 1D vector.

VGS	0	0.3	0.6	0.9	1.2	0	0.3	0.6	0.9	1.2	0	0.3	0.6	0.9	1.2	0	...
VSB	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.6	0.6	0.6	0.6	0.6	0.9	...
VGB	0	0.3	0.6	0.9	1.2	0.3	0.6	0.9	1.2	1.5	0.6	0.9	1.2	1.5	1.8	0.9	...
IGi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**-**	**10**	**11**	**12**	**-**	**-**	**13**	**...**

Figure 5. The 2D array in Figure 3 unrolled into a 1D vector. The incomplete-grid index (IGi) is shown for each grid point.

VGS →

VSB ↓

	0	0.3	0.6	0.9	1.2	**IGS**
0	0	0.3	0.6	0.9	1.2	**5**
0.3	0.3	0.6	0.9	1.2	1.5	**4**
0.6	0.6	0.9	1.2	1.5	1.8	**3**
0.9	0.9	1.2	1.5	1.8	2.1	**2**
1.2	1.2	1.5	1.8	2.1	2.4	**1**

VGB = VGS + VSB <= 1.2

VGS →

VSB ↓

	0	0.2	0.4	0.6	0.8	1	1.2	**IGS**
0	0	0.2	0.4	0.6	0.8	1	1.2	**7**
0.3	0.3	0.5	0.7	0.9	1.1	1.3	1.5	**5**
0.6	0.6	0.8	1	1.2	1.4	1.6	1.8	**4**
0.9	0.9	1.1	1.3	1.5	1.7	1.9	2.1	**2**
1.2	1.2	1.4	1.6	1.8	2	2.2	2.4	**1**

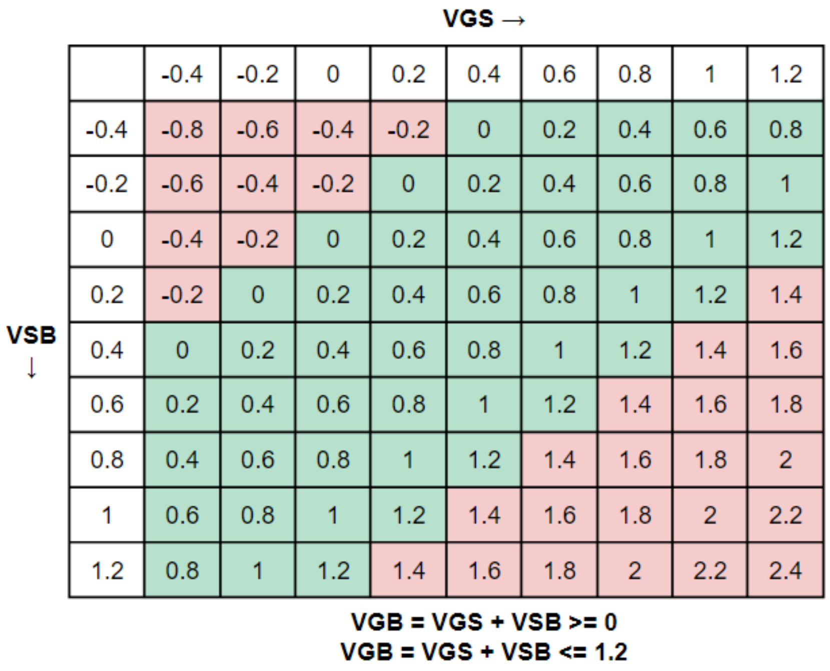
VGB = VGS + VSB <= 1.2

Figure 6. Two examples of the incomplete-grid stride vector (IGS).

3.3. Incomplete Grid with Two-Sided Constraints

The previous incomplete-grid example considered a one-sided constraint, e.g., $V_{GB} \leq V_{GBmax}$. In some practical scenarios, it is necessary to apply a two-sided constraint, e.g., $V_{GBmin} \leq V_{GB} \leq V_{GBmax}$. For example, it may be necessary to extend the characterization range of the device to include negative values of V_{GS} and V_{SB}. Negative values of V_{GS} may occur naturally in stacked devices and can be used to decrease the leakage current if the device does not suffer from gate-induced drain leakage (GIDL). Negative values of V_{SB} may be also used in a body biasing scheme to tune the MOSFET's behavior.

Figure 7 shows an example of a 2D (V_{GS}, V_{SB}) array with a two-sided constraint applied to V_{GB}. It is assumed that both V_{GS} and V_{SB} can be negative, but V_{GB} is always positive. The solution given by (2) will not work for this case since there can be invalid elements at the beginning of every row; thus, another solution has to be developed.



VGS →

VSB ↓

	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1.2
-0.4	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8
-0.2	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1
0	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1.2
0.2	-0.2	0	0.2	0.4	0.6	0.8	1	1.2	1.4
0.4	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6
0.6	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8
0.8	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
1	0.6	0.8	1	1.2	1.4	1.6	1.8	2	2.2
1.2	0.8	1	1.2	1.4	1.6	1.8	2	2.2	2.4

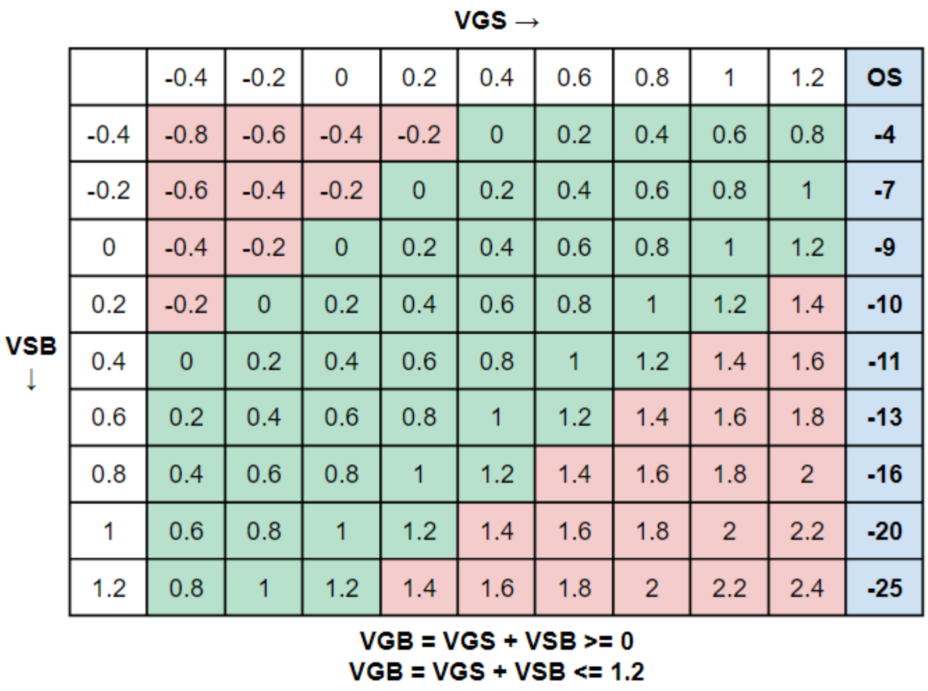
VGB = VGS + VSB >= 0
VGB = VGS + VSB <= 1.2

Figure 7. A 2D full grid using V_{GS} and V_{SB} grid vectors. The value of V_{GB} is shown at every grid point. The green and red colors show the valid and invalid points, respectively. A two-sided constraint is applied to V_{GB}.

An effective solution to this problem is to use an offset vector to replace the IGS vector. The offset vector will simply store the difference between the FGi and the IGi. In this case, the IGi can be expressed as

$$IGi = FGi + OS_i \tag{3}$$

where FGi is given by (1) and OS_i is the offset of row_i. Figure 8 shows the offset vector, where an offset value is calculated and stored for every row in the array depicted in Figure 7. The offset vector is calculated during the LUT generation process and represents an additional overhead vector that is crucial for correctly indexing the incomplete-grid 1D vector.



VGS →

VSB ↓

	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1.2	**OS**
-0.4	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	**-4**
-0.2	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	**-7**
0	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1.2	**-9**
0.2	-0.2	0	0.2	0.4	0.6	0.8	1	1.2	1.4	**-10**
0.4	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	**-11**
0.6	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	**-13**
0.8	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2	**-16**
1	0.6	0.8	1	1.2	1.4	1.6	1.8	2	2.2	**-20**
1.2	0.8	1	1.2	1.4	1.6	1.8	2	2.2	2.4	**-25**

VGB = VGS + VSB >= 0
VGB = VGS + VSB <= 1.2

Figure 8. The 2D array in Figure 7 with the additional offset vector (OS), which stores the cumulative offset at every row.

3.4. Three-Dimensional Incomplete Grid

The previous sections considered the case of a 2D grid. Practically, the constraint applied to V_{GB} in the V_{GS} vs. V_{SB} space can be similarly applied to V_{DB} in the V_{DS} vs. V_{SB} space, as shown in Figure 9. Thus, by combining the two spaces, a 3D (V_{GS}, V_{DS}, V_{SB}) array is formed. The unrolled 1D full-grid index (FGi) for the 3D array is given by

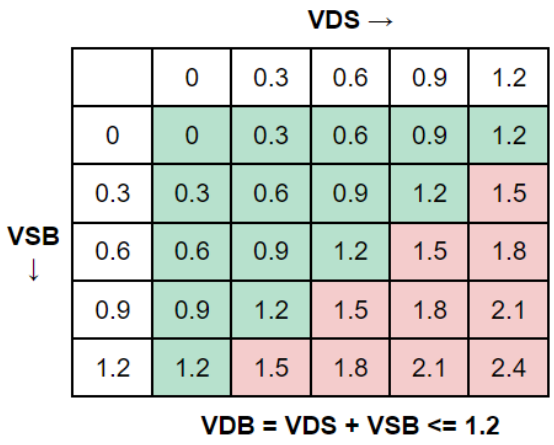
$$FGi = V_{GSi} + (V_{DSi} - 1) \times S_{VDS} + (V_{SBi} - 1) \times S_{VSB} \tag{4}$$

where $(V_{GSi}, V_{DSi}, V_{SBi})$ are the indices of the (V_{GS}, V_{DS}, V_{SB}) grid vectors, S_{VDS} is the V_{DS} stride, defined as

$$S_{VDS} = length(V_{GS}) \tag{5}$$

and S_{VSB} is the V_{SB} stride, defined as

$$S_{VSB} = length(V_{GS}) \times length(V_{DS}) \tag{6}$$



VDS →

VSB ↓

	0	0.3	0.6	0.9	1.2
0	0	0.3	0.6	0.9	1.2
0.3	0.3	0.6	0.9	1.2	1.5
0.6	0.6	0.9	1.2	1.5	1.8
0.9	0.9	1.2	1.5	1.8	2.1
1.2	1.2	1.5	1.8	2.1	2.4

VDB = VDS + VSB <= 1.2

Figure 9. A 2D full grid using V_{DS} and V_{SB} grid vectors. The value of V_{DB} is shown at every grid point. The green and red colors show the valid and invalid points, respectively.

One important point to consider is that the V_{GB} and V_{DB} constraints must be applied concurrently to check the valid points. Thus, the condition of the valid points is given by ANDing all the following constraints:

$$V_{GS}(V_{GSi}) + V_{SB}(V_{SBi}) \leq V_{GBmax} \tag{7}$$

$$V_{GS}(V_{GSi}) + V_{SB}(V_{SBi}) \geq V_{GBmin} \tag{8}$$

$$V_{DS}(V_{DSi}) + V_{SB}(V_{SBi}) \leq V_{DBmax} \tag{9}$$

$$V_{DS}(V_{DSi}) + V_{SB}(V_{SBi}) \geq V_{DBmin} \tag{10}$$

These conditions are also used when a query point is requested from the LUTs to ensure that the eliminated points are not included in the automated design flow.

For every V_{GS} row, the incomplete-grid offset is calculated and stored in the offset array (OS). Thus, the offset becomes a 2D array with a total number of elements equal to

$$numel(OS) = length(V_{DS}) \times length(V_{SB}) \tag{11}$$

The incomplete-grid index (IGi) for the query point $(V_{GSi}, V_{DSi}, V_{SBi})$ is now given by

$$IGi = FGi(V_{GSi}, V_{DSi}, V_{SBi}) + OS(V_{DSi}, V_{SBi}) \tag{12}$$

which can be used to access the unrolled 1D incomplete grid.

4. Results and Discussion

The proposed incomplete-grid implementation for the LUT can result in significant savings in the LUT memory footprint. An overhead exists due to the offset array (OS), but this overhead is negligible, especially when the length of the V_{GS} grid vector is large. The offset array overhead, expressed as a percentage of the device LUT size, is given by

$$OS\ overhead\ (\%) = \frac{1}{N \times C \times length(L) \times length(V_{GS})} \times 100 \tag{13}$$

where N is the number of LUTs, i.e., the number of parameters stored for a given device, e.g., I_D, g_m, g_{ds}, etc., and C is the number of process and temperature corners. Practically, if it is assumed that $N > 10$ and the length of the L and V_{GS} grid vectors > 10, the overhead will be less than 0.1%. Note that the offset array is shared among all the LUTs of a given device because all the LUTs use the same grid vectors and the same constraints to filter the full grid and create the incomplete grid. It should be noted that the offset array does not add a performance overhead because it is precomputed during the LUT generation process and stored with the LUT structure.

Figure 10 compares the sizes of the full-grid LUT and the incomplete-grid LUT. The number of points of the three grid vectors (V_{GS}, V_{DS}, V_{SB}) is assumed to be the same, i.e., a 3D square array. The figure clearly shows that the incomplete grid can result in significant memory savings. The percentage of memory saving can be calculated as

$$Memory\ Saving\ (\%) = \frac{Full\ Grid\ Size - Incomplete\ Grid\ Size}{Full\ Grid\ Size} \times 100 \tag{14}$$

and is plotted in Figure 11. As the number of grid vector points increases, the memory saving approaches 67%, which means reducing two-thirds of the full-grid LUT size.

The previous results assume that the grid vectors have a uniform step. Practically, the step may be variable, with a small step used at the beginning and then an increased step afterward. The justification for the variable step is that, for analog circuits, the devices are usually biased with low to moderate values of (V_{GS}, V_{DS}, V_{SB}); thus, more accuracy is needed in this region. Consider the case where the number of points used in the $0 \rightarrow V_{max}/2$ range is double the number of points used in the $V_{max}/2 \rightarrow V_{max}$ range. The memory savings in this case are plotted in Figure 12. As expected, the memory savings are less than in the uniform step case, but a significant memory saving (around 50%) can still be achieved.

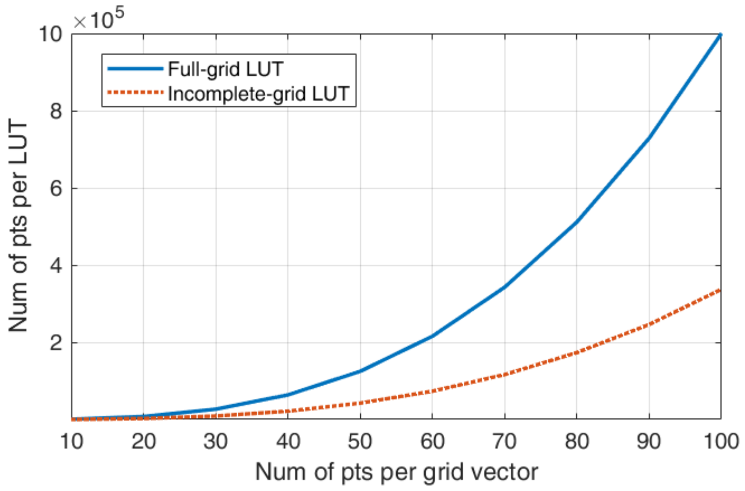


Figure 10. The number of points per LUT vs. the number of points per grid vector. The incomplete-grid LUT can achieve significant memory savings.

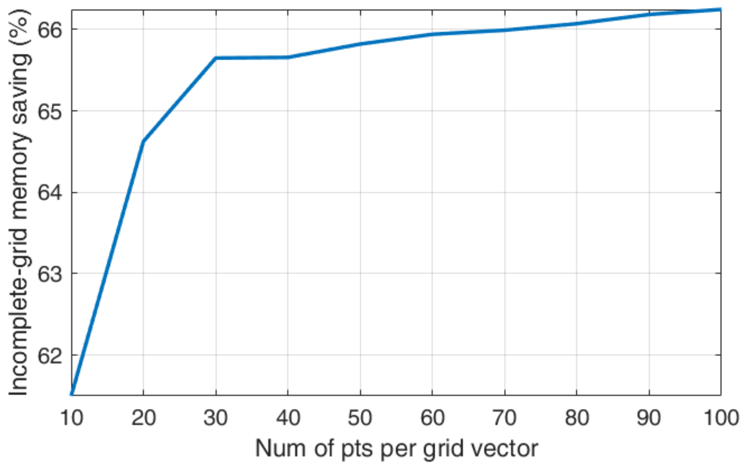


Figure 11. The percentage of memory saving due to using the incomplete grid vs. the number of points per grid vector for grid vectors with a uniform step.

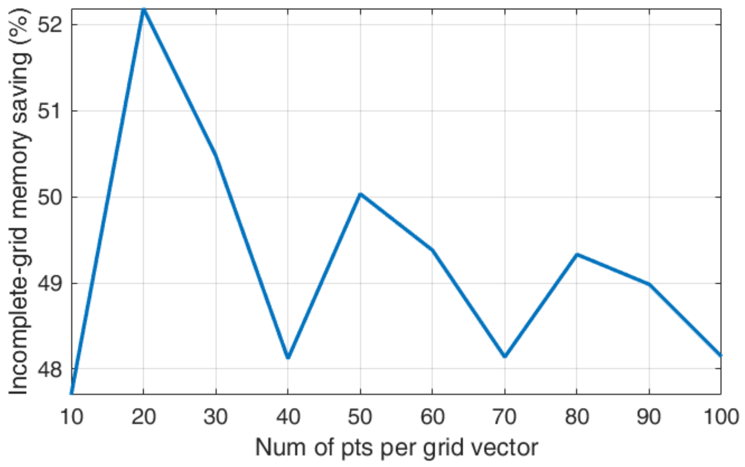
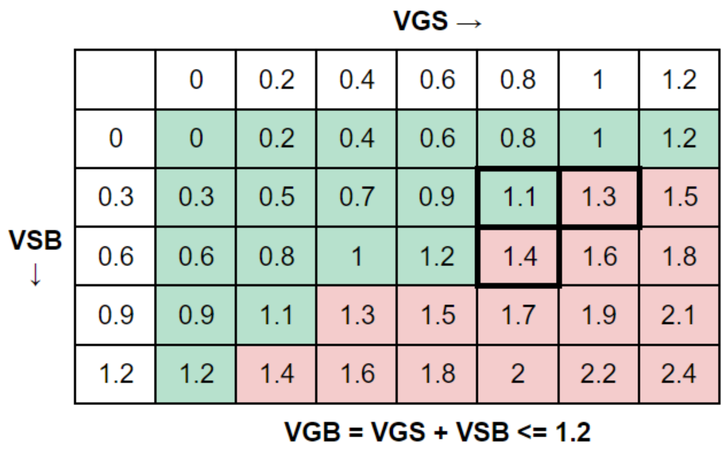


Figure 12. The percentage of memory saving due to using the incomplete grid vs. the number of points per grid vector for grid vectors with a non-uniform step.

For rectangular arrays, the incomplete grid may suffer from data loss at the array edges. For example, consider the case shown in Figure 13, where the number of points in the V_{SB} vector is less than the number of points in the V_{GS} vector. Assume it is necessary to query a point that has $V_{GS} = 0.85$ and $V_{SB} = 0.3$. This point is valid because $V_{GB} = 1.15 < 1.2$. However, since the neighboring point from the right $(V_{GS} = 1, V_{SB} = 0.3)$ has been removed, the interpolation process will fail to obtain the grid points that surround the query point. The same problem applies to the case of $V_{GS} = 0.8$ and $V_{SB} = 0.35$ but in the downward direction. A simple solution to this problem is to always add one extra point after the last valid point. This will reduce memory savings, especially when the grid vector is small. However, the memory footprint is problematic when the grid vectors are large, and in this case, the extra grid point will not add significant overhead.


VGS →
VSB ↓
VGB = VGS + VSB <= 1.2

	0	0.2	0.4	0.6	0.8	1	1.2
0	0	0.2	0.4	0.6	0.8	1	1.2
0.3	0.3	0.5	0.7	0.9	1.1	1.3	1.5
0.6	0.6	0.8	1	1.2	1.4	1.6	1.8
0.9	0.9	1.1	1.3	1.5	1.7	1.9	2.1
1.2	1.2	1.4	1.6	1.8	2	2.2	2.4

Figure 13. An example of a rectangular array showing loss of data at the edges.

It is worth noting that the advantage of using the incomplete-grid LUT is not only the memory savings. The device characterization testbenches can also utilize only the valid points; thus, the LUT generation time will be reduced by the same memory-saving factor. Another advantage is that the time to query data from the LUT will also be reduced due to the smaller LUT size.

In order to demonstrate a design example using the incomplete-grid LUTs, we consider the two-stage CMOS Miller amplifier shown in Figure 14. The design space of the circuit is generated by substituting the device parameters from the LUTs into symbolic expressions for the circuit's performance metrics. A database containing 100k design points is generated in 24 s using a standard machine with a quad-core processor and 8 GB of RAM. Figure 15 illustrates the circuit design space showing the gain-bandwidth product (GBW) vs. the total bias current while applying constraints on the circuit's DC gain and phase margin (PM). Such a design chart can be used to evaluate the feasibility limits of a given circuit in a given technology and to obtain the Pareto optimal fronts for the design performance metrics.

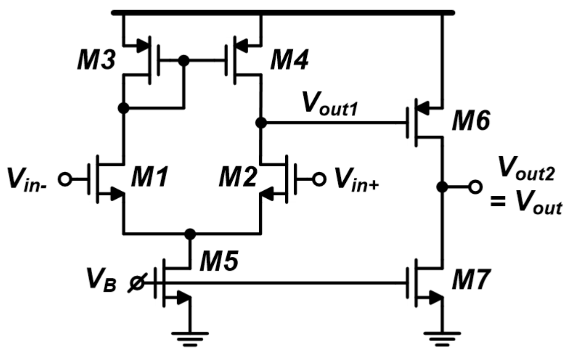


Figure 14. Schematic of the two-stage CMOS Miller amplifier used as a design example.

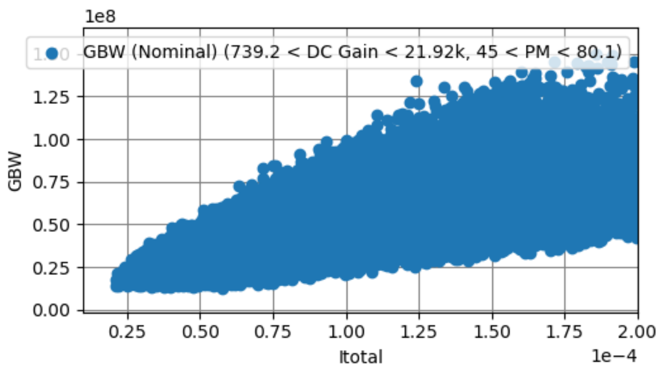


Figure 15. Design space of the two-stage Miller amplifier showing the GBW vs. the total bias current under DC gain and PM constraints.

5. Conclusions

This paper presented a technique to reduce the memory footprint of lookup tables (LUTs) used in CMOS analog design by using an incomplete grid. The one-dimensional unrolled grid can be indexed using a precomputed offset array that typically represents less than 0.1% overhead. The proposed technique supports one-sided/two-sided constraints and 2D/3D arrays. A 67% reduction in the LUT size can be achieved, along with a similar speedup in LUT generation time. Handling the loss of information at the valid/invalid boundary of the incomplete grid was also discussed.

Funding: This research was funded by Egypt's Information Technology Industry Development Agency (ITIDA) grant number ARP2022.R33.4.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The author would like to thank the Analog Designer's Toolbox (ADT) team at Master Micro LLC for supporting this work.

Conflicts of Interest: The author declares no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Scheible, J. Optimized is Not Always Optimal—The Dilemma of Analog Design Automation. In Proceedings of the 2022 International Symposium on Physical Design; Association for Computing Machinery, New York, NY, USA, 27–30 March 2022; pp. 151–158. [CrossRef]
2. Barros, M.F.; Guilherme, J.M.; Horta, N.C.G. *Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques*; Springer: Berlin/Heidelberg, Germany, 2010.
3. Rutenbar, R.A.; Gielen, G.G.E.; Antao, B.A. (Eds.) *Computer-Aided Design of Analog Integrated Circuits and Systems*; Wiley-IEEE Press: Hoboken, NJ, USA, 2002.
4. Silveira, F.; Flandre, D.; Jespers, P.G. A gm/ID based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA. *IEEE J. Solid-State Circuits* **1996**, *31*, 1314–1319. [CrossRef]
5. Jespers, P. *The gm/ID Methodology, a Sizing Tool for Low-Voltage Analog CMOS Circuits: The Semi-Empirical and Compact Model Approaches*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010.
6. Crossley, J.W. BAG: A Designer-Oriented Framework for the Development of AMS Circuit Generators. Ph.D. Thesis, EECS Department, University of California, Berkeley, CA, USA, 2014.
7. Linán-Cembrano, G.; Lourenço, N.; Horta, N.; de la Rosa, J.M. Design Automation of Analog and Mixed-Signal Circuits Using Neural Networks—A Tutorial Brief. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *71*, 1677–1682. [CrossRef]
8. Sajjadi, S.A.; Sadrossadat, S.A.; Moftakharzadeh, A.; Nabavi, M.; Sawan, M. DNN-Based Optimization to Significantly Speed Up and Increase the Accuracy of Electronic Circuit Design. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2024**, *71*, 1273–1284. [CrossRef]
9. Choi, Y.; Park, S.; Choi, M.; Lee, K.; Kang, S. MA-Opt: Reinforcement Learning-Based Analog Circuit Optimization Using Multi-Actors. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2024**, *71*, 2045–2056. [CrossRef]
10. Budak, A.F.; Gandara, M.; Shi, W.; Pan, D.Z.; Sun, N.; Liu, B. An Efficient Analog Circuit Sizing Method Based on Machine Learning Assisted Global Optimization. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2021**, *41*, 1209–1221. [CrossRef]

11. Li, Z.; Carusone, A.C. Design and Optimization of Low-Dropout Voltage Regulator Using Relational Graph Neural Network and Reinforcement Learning in Open-Source SKY130 Process. In Proceedings of the 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), San Francisco, CA, USA, 28 October–2 November 2023; pp. 1–9. [CrossRef]
12. Jespers, P.G.; Murmann, B. *Systematic Design of Analog CMOS Circuits Using Pre-Computed Lookup Tables*; Cambridge University Press: Cambridge, UK, 2017.
13. Elmeligy, K.; Omran, H. Fast Design Space Exploration and Multi-Objective Optimization of Wide-Band Noise-Canceling LNAs. *Electronics* **2022**, *11*, 816. [CrossRef]
14. Youssef, A.A.; Murmann, B.; Omran, H. Analog IC Design Using Precomputed Lookup Tables: Challenges and Solutions. *IEEE Access* **2020**, *8*, 134640–134652. [CrossRef]
15. Mohamed, K.; Yasseen, K.Y.; Murmann, B.; Omran, H. Capturing Layout Dependent Effects in MOSFET Circuit Sizing Using Precomputed Lookup Tables. *IEEE Access* **2023**, *11*, 41205–41217. [CrossRef]
16. Mohamed, K.; Nafea, S.; Omran, H. Design Automation of Low Dropout Voltage Regulators: A General Approach. *Electronics* **2022**, *12*, 205. [CrossRef]
17. Murmann, B. Practical Aspects of Script-Based Analog Design Using Precomputed Lookup Tables. In Proceedings of the 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, 19–22 May 2024; pp. 1–5. [CrossRef]
18. Arevalos, D.; Marín, J.; Herman, K.; Rojas, C.A. Leveraging Lookup Tables for Efficient LDO Design Exploration using Open-Source CAD Tools and IHP-Open130-G2 PDK. In Proceedings of the 2024 31st International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Gdansk, Poland, 27–28 June 2024; pp. 124–129. [CrossRef]