LETTER

Unit Cell Mismatch Scrambling Method for High-Resolution Unary DAC based on Virtual 3D Layout


Abstract The paper studies a unit cell mismatch scrambling method for a high-resolution unary DAC based on virtual 3-dimensional (3D) layout, to improve its spurious-free dynamic range (SFDR) for communication applications. This can be implemented with relatively simple interconnections and scrambling circuits, compared to that based on the 2-dimensional (2D) layout.

Keywords: Unary DAC, Mismatch Scrambling, Dynamic Element Matching, Virtual 3D Layout of Unit Cells, SFDR

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

Digital-to-Analog Converter (DAC) for communication applications requires good Spurious Free Dynamic Range (SFDR) performance [1, 2]. The SFDR of the unary DAC is degraded due to static characteristics mismatches among its unit cells as well as its dynamic characteristics.

In this paper, we focus on the care for the static characteristics mismatches using the Dynamic Element Matching (DEM) for the unary DAC. We consider here the signal bandwidth of the DAC is from DC to the Nyquist rate (half of the sampling frequency) or higher frequency, so that the DEM technique only needs to spread the spectrum of the spurious tones caused by the mismatches in the entire signal band uniformly without need for mismatch shaping; this is called as mismatch scrambling [3].

However, direct implementation circuit of such mismatch scrambling for the high-resolution unary DAC becomes complicated. Then, in [4] a mismatch scrambling technique utilizing the features of 2D regular (square or rectangular) layout and routing for the unit cells was investigated, utilizing row and column decoders features [5].

Here, we investigate a more hardware efficient method for a higher resolution DAC, by considering a virtual 3D case, considering X-, Y- and Z-decoders.

Notice that most of DEM algorithms are for low resolution DACs and for mismatch shaping [3, 6-30]; to our knowledge, there is no research for small hardware mismatch scrambling for high resolution DACs except for [4].

2. Configuration and Problems of Unary DAC

Fig. 1 shows a 6-bit unary DAC with unit current sources. Its binary digital inputs (B6, B5, B4, B3, B2, B1) are converted into the thermometer codes (S63, S62, ..., S2, S1) through the decoder. Here we assume the following:

\[ I = I_1 = I_2 = \ldots = I_{63} \]

Then the k-th current switch for S_k=1 turns on. For example, if the digital input 000111, then S_k=1 (k=1, ..., 7), S_m=0 (m=8, ..., 63), and the analog output is \( V_{\text{OUT}} = 7 R I \).

Next, we consider mismatches among current sources in Fig. 1. \( I_n \) is the unit current source corresponding to \( S_n \). Ideally all values of \( I_n \) \( (n=1, 2, ..., 63) \) are identical, but in reality, these can be different [6]:

\[ I_n = I + \Delta I_n \]

\[ I_n = \left( \frac{1}{2^2} \right) (I_1 + I_2 + \ldots + I_{63}) \]

These mismatches \( \Delta I_n \) cause the overall DAC nonlinearity and generate spurious components of the DAC output in frequency domain and thus degrade the SFDR.

Then the mismatch scrambling method is considered to spread the spurious spectrum components caused by the mismatches and improve the SFDR. The mismatch scrambling is a technique to change the unit cell selection at each sampling time in a pseudo random manner.

For example, consider the case that the digital input is the DC value of 00101. Then a mismatch scrambling algorithm may work as follows:

At time \( k \), \( S_5=S_4=S_3=S_2=S_1=1 \) and the analog output \( V_{\text{OUT}}(k) = R (5 I_1 + \Delta I_2 + \Delta I_3 + \Delta I_4 + \Delta I_5) \).

At time \( k+1 \), \( S_6=S_5=S_4=S_2=S_1=1 \) and \( V_{\text{OUT}}(k+1) = R (5 I_1 + \Delta I_2 + \Delta I_3 + \Delta I_4 + \Delta I_5 + \Delta I_6 + \Delta I_7) \).

At time \( k+2 \), \( S_7=S_6=S_5=S_3=S_2=S_1=1 \) and \( V_{\text{OUT}}(k+2) = R (5 I_1 + \Delta I_2 + \Delta I_3 + \Delta I_4 + \Delta I_5 + \Delta I_6 + \Delta I_7 + \Delta I_8) \).

Here the current mismatch effects are dynamically changed.
3. Mismatch scrambling technique for 2D regular layout

The circuit to implement conventional mismatch scrambling increases exponentially as the number of unit cells increases if it is directly implemented. Hence its simple implementation utilizing the features of the 2D matrix layout and routing of the unit cells is discussed in [4]. Our interpretation for 2D regular layout mismatch scrambling method is as follows:

We consider here a 6-bit unary DAC with digital binary inputs of $B_6$, $B_5$, $B_4$, $B_3$, $B_2$, $B_1$ and their thermometer codes are $S_{64}$, $S_{63}$, $S_{62}$, ..., $S_3$, $S_2$, $S_1$. Also consider the 2D square layout and routing of the unit cells as shown in Fig. 2. There, each component is given as follows:

i) Row-decoder: binary inputs $B_6$, $B_5$, $B_4$, thermometer outputs $R_7$, $R_6$, $R_5$, $R_4$, $R_3$, $R_2$, $R_1$.

\[
R_7 = B_6 B_5 B_4, \quad R_6 = B_6 B_5, \quad R_5 = B_6 (B_5 + B_4), \quad R_4 = B_6, \\
R_3 = B_6 + B_5 B_4, \quad R_2 = B_6 + B_5, \quad R_1 = B_6 + B_5 + B_4.
\]

ii) Column-decoder: binary inputs $B_3$, $B_2$, $B_1$.

thermometer outputs $C_7$, $C_6$, $C_5$, $C_4$, $C_3$, $C_2$, $C_1$.
\[
C_7 = B_3 B_2 B_1, \quad C_6 = B_3 B_2, \quad C_5 = B_3 (B_2 + B_1), \quad C_4 = B_3, \\
C_3 = B_3 + B_2 B_1, \quad C_2 = B_3 + B_2, \quad C_1 = B_3 + B_2 + B_1.
\]

See Table 1. Also we define $R_0 = C_0 = 1$, $R_g = C_g = 0$.

We use a local decoder for each unit cell with the following logical expression (Table 2):

\[
S_n = R_{p+1} + R_p C_q
\]

$s_n$ is the switch for $n$th current source. Here,

For $1 \leq n \leq 8$, \quad $p = 0$.

For $9 \leq n \leq 16$, \quad $p = 1$.

For $17 \leq n \leq 24$, \quad $p = 2$.

For $25 \leq n \leq 32$, \quad $p = 3$.

For $33 \leq n \leq 40$, \quad $p = 4$.

For $41 \leq n \leq 48$, \quad $p = 5$.

For $49 \leq n \leq 56$, \quad $p = 6$.

For $57 \leq n \leq 64$, \quad $p = 7$.

For $1 \leq \text{mod}_8 (n) \leq 7$, \quad $q = \text{mod}_8 (n)$.

For $\text{mod}_8 (n) = 0$, \quad $q = 8$.

Now let us consider the mismatch scrambling circuit suitable for 2D layout as shown in Fig. 3. Table 3. For each $n (n=1, 2, 3,..., 63, 64)$, we have

\[
S_n = R_{p+1} + R_p C_q
\]

and $p$, $q$ are changed dynamically in a pseudo random manner ($p, q = 1, 2, ..., 7, 8$). Let us consider one-to-one mapping between $(C_B, C_7, C_6, C_5, C_4, C_3, C_2, C_1)$ and $(C_B, C_7, C_6, C_5, C_4, C_3, C_2, C_1)$, and their relationships are changed at each sampling time.

One example is as follows:

At time $k$:

\[
(C_B, C_7, C_6, C_5, C_4, C_3, C_2, C_1) = (C_B, C_7, C_6, C_5, C_4, C_3, C_2, C_1)
\]

At time $k+1$:

\[
(C_B, C_7, C_6, C_5, C_4, C_3, C_2, C_1) = (C_1, C_3, C_5, C_6, C_B, C_4, C_7, C_2)
\]

At time $k+2$:

\[
(C_B, C_7, C_6, C_5, C_4, C_3, C_2, C_1) = (C_4, C_1, C_6, C_2, C_B, C_3, C_5, C_7)
\]
Similarly, let us consider one-to-one mapping between $(R'_8, R'_7, R'_6, R'_5, R'_4, R'_3, R'_2, R'_1)$ and $(R_8, R_7, R_6, R_5, R_4, R_3, R_2, R_1)$, and we change their relationships dynamically. Also, the following relationship is maintained:

If $R'_{p+1} = R_{p+1}$, then $R'_p = R_p (p = 0, 1, 2, 3, 4, 5, 6, 7)$.

One example is as follows:

At time $k$:

$$
\begin{align*}
(R'_8, R'_7, R'_6, R'_5, R'_4, R'_3, R'_2, R'_1) &= (R_8, R_7, R_6, R_5, R_4, R_3, R_2, R_1) \\
(R''_8, R''_7, R''_6, R''_5, R''_4, R''_3, R''_2, R''_1) &= (R_7, R_6, R_5, R_4, R_3, R_2, R_1, R_0)
\end{align*}
$$

At time $k+1$:

$$
\begin{align*}
(R'_8, R'_7, R'_6, R'_5, R'_4, R'_3, R'_2, R'_1) &= (R_5, R_4, R_3, R_2, R_1, R_0, R_8, R_7) \\
(R''_8, R''_7, R''_6, R''_5, R''_4, R''_3, R''_2, R''_1) &= (R_6, R_5, R_4, R_3, R_2, R_1, R_0, R_8)
\end{align*}
$$

At time $k+2$:

$$
\begin{align*}
(R'_8, R'_7, R'_6, R'_5, R'_4, R'_3, R'_2, R'_1) &= (R_6, R_5, R_4, R_3, R_2, R_1, R_0, R_8) \\
(R''_8, R''_7, R''_6, R''_5, R''_4, R''_3, R''_2, R''_1) &= (R_7, R_6, R_5, R_4, R_3, R_2, R_1, R_0)
\end{align*}
$$

These interconnection changes can be realized with multiplexer arrays (Row Scrambler, Column Scrambler in Fig. 3) following the column and row decoders, as well as pseudo random signal generators (RNG_R, RNG_C in Fig. 3) for control of their selection signals; RNG_R, RNG_C can be realized with linear feedback shift registers (LFSRs).

To understand the above mismatch scrambling for the 2D square layout, we explain the 4-bit unary DAC case for simplicity. Fig. 4 (a) shows its 2D matrix layout, whereas Fig. 4 (b) shows the one with scrambler circuits following the row and column decoders, and also it explains one example of the unit cell selection order change.

**4. Mismatch scrambling technique for virtual 3D layout**

The 2D layout mismatch scrambling algorithm in Section 3 may be enough for DACs with up-to 6-bit resolution. However, the authors often use 1D layout instead of 2D layout for a unary DAC with high resolution, e.g., 10-bit. In such a case, routing and scramblers using the 2D regular layout mismatch scrambling method becomes complicated and needs substantial chip area.

Then we propose here a virtual 3D layout mismatch scrambling circuit (Fig. 5); this can reduce the total scrambling circuit amount and the number of interconnections compared to the 2D layout mismatch scrambling circuit and hence this is more effective for higher resolution DACs; the virtual 3D layout DAC can be done as 1D layout which currently we use for our DAC design.

---

**Fig. 3. 2D unit cell layout of 6-bit unary DAC with mismatch scrambling.**

**Table 3. Logical expressions of unit cell local decoders for a 6-bit unary DAC with 2D layout mismatch scrambling in Fig. 4.**

**Fig. 4.** (a) 4-bit unary DAC with 2D regular layout of unit cells. (b) 4-bit unary DAC mismatch scrambling circuit and operation example.
Next, we show our proposed virtual 3D layout mismatch scrambling in Fig. 7. There, scramblers or multiplexer arrays with selection signal control circuits of LFSRs (RNG_X, RNG_Y, RNG_Z) are added.

i) Scrambler_X:
- inputs \((X_4, X_3, X_2, X_1)\)
- outputs \((X'_4, X'_3, X'_2, X'_1)\)

ii) Scrambler_Y:
- inputs \((Y_4, Y_3, Y_2, Y_1, Y_0)\)
- outputs \((Y'_4, Y'_3, Y'_2, Y'_1, Y'_3, Y'_2, Y'_1, Y'_0)\)

Also, the following relationship is maintained:
- If \(Y'_{p+1} = Y_{n+1}\), then \(Y''_p = Y_n\) \((p, n = 0, 1, 2, 3)\).

iii) Scrambler_Z:
- inputs \((Z_4, Z_3, Z_2, Z_1, Z_0)\)
- outputs \((Z'_4, Z'_3, Z'_2, Z'_1, Z'_3, Z'_2, Z'_1, Z'_0)\)

Also, the following relationship is maintained:
- If \(Z'_{p+1} = Z_{n+1}\), then \(Z''_p = Z_n\) \((p, n = 0, 1, 2, 3)\).

These inputs and outputs are one-to-one mappings, and have the following inter-connections (Table 6):

\[
S_n = Z'_{n+1} + Z''_{n+1} + X''_{n+1} + X''_{n+1} + Z''_{n+1} + X''_{n+1}
\]

Let us consider one-to-one mapping between \((X'_4, X'_3, X'_2, X'_1)\) and \((X_4, X_3, X_2, X_1)\), and their relationships are changed dynamically (Figs. 7, 8).

One example is as follows:

At time \(k\):
\[
(X'_4, X'_3, X'_2, X'_1) = (X_4, X_3, X_2, X_1)
\]

At time \(k+1\):
\[
(X'_4, X'_3, X'_2, X'_1) = (X_4, X_3, X_2, X_1)
\]

At time \(k+2\):
\[
(X'_4, X'_3, X'_2, X'_1) = (X_4, X_3, X_2, X_1)
\]

At time \(k+3\):
\[
(X'_4, X'_3, X'_2, X'_1) = (X_4, X_3, X_2, X_1)
\]

Similarly, let us consider one-to-one mapping between \((Y'_4, Y'_3, Y'_2, Y'_1)\) and \((Y_4, Y_3, Y_2, Y_1)\), and we change their relationships dynamically.

One example is as follows:

At time \(k\):
\[
(Y'_4, Y'_3, Y'_2, Y'_1) = (Y_4, Y_3, Y_2, Y_1)
\]

Next, let us consider one-to-one mapping between \((Y'_4, Y'_3, Y'_2, Y'_1)\) and \((Y_4, Y_3, Y_2, Y_1)\), and we change their relationships dynamically.

One example is as follows:

At time \(k\):
\[
(Y'_4, Y'_3, Y'_2, Y'_1) = (Y_4, Y_3, Y_2, Y_1)
\]
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_3, Y_2, Y_1, Y_0)
At time k+1:
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_3, Y_2, Y_1, Y_4)
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_2, Y_1, Y_0, Y_5)

At time k+2:
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_2, Y_1, Y_4, Y_3)
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_1, Y_0, Y_3, Y_2)

At time k+3:
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_1, Y_0, Y_2, Y_3)
(Y^r_4, Y^r_3, Y^r_2, Y^r_1) = (Y_0, Y_1, Y_2, Y_3)

Also, consider one-to-one mapping between (Z'_4, Z'_3, Z'_2, Z'_1) and (Z_4, Z_3, Z_2, Z_1), and their relationships are changed dynamically. One example is as follows:
At time k:
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_4, Z_3, Z_2, Z_1)
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_4, Z_2, Z_1, Z_0)

At time k+1:
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_3, Z_1, Z_0, Z_4)
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_3, Z_2, Z_0, Z_4)

At time k+2:
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_3, Z_4, Z_1, Z_2)
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_2, Z_3, Z_0, Z_1)

At time k+3:
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_2, Z_1, Z_4, Z_3)
(Z'_4, Z'_3, Z'_2, Z'_1) = (Z_1, Z_0, Z_3, Z_2)

Table 5. Logical expressions of unit cell local decoders for a 6-bit unary DAC with virtual 3D layout in Fig. 5.

5. Simulation Verification

We have simulated a 6-bit unary DAC with some unit current mismatches and compared the results without mismatch scrambling, with the one for 2D regular layout and with the proposed one for virtual 3D layout. Their output power spectrum and SFDRs are shown in Fig. 9. We see that both mismatch scrambling methods improve the SFDR compared to the case without mismatch scrambling, and both mismatch scrambling effects are comparable.

6. Conclusion

In this paper, we have proposed unary DAC mismatch scrambling circuit based on virtual 3D layout concept, but there 1D layout is actually used. Its circuit implementation requires only small amount of circuits compared to the conventional methods. Simulation results show that its SFDR can be improved compared to the case without the mismatch scrambling. We finally remark that its extention to virtual 4D or higher dimension layout is possible.
Table 6. Logical expressions of unit cell local decoders for a 6-bit unary DAC with virtual 3D layout mismatch scrambling in Fig. 11.

<table>
<thead>
<tr>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01 = Z1' + Z0'*Y1' + Z0'*Y0'*X1'</td>
</tr>
<tr>
<td>S33 = Z3' + Z2'*Y1' + Z2'*Y0'*X1'</td>
</tr>
<tr>
<td>S02 = Z1' + Z0'*Y1' + Z0'*Y0'*X2'</td>
</tr>
<tr>
<td>S34 = Z3' + Z2'*Y1' + Z2'*Y0'*X2'</td>
</tr>
<tr>
<td>S32 = Z2' + Z1'*Y4' + Z1'*Y3'*X4'</td>
</tr>
<tr>
<td>S64 = Z4' + Z3'*Y4' + Z3'*Y3'*X4'</td>
</tr>
</tbody>
</table>

Fig. 9. Simulated 6-bit DAC output power spectrum. (a) Without mismatch scrambling. (b) With mismatch scrambling for 2D regular layout. (c) With mismatch scrambling for virtual 3D layout.

Acknowledgments

This work is supported by Adaptable and Seamless Technology Transfer Program through Target-Driven R&D (A-STEP) from Japan Science and Technology Agency (JST) Grant Number JPMJTR201C, as well as JSPS KAKENHI Grant Number 21K04190.

References