Capacitor Mismatch Compensation for the Quasi-Passive Switched-Capacitor DAC

P. Rombouts, L. Weyten, J. Raman, and S. Audenaert

***Abstract*— A method to eliminate the effect of capacitor mismatch for a switched-capacitor DAC is described. The method consists of two elements. The first element is the use of a compensating switching algorithm, which can eliminate the effect of capacitor mismatch only for some digital input values. The second element is that each digital word is written as the sum of two other words for which the capacitor mismatch can be eliminated. These words are converted and the corresponding voltages are summed.**

I. Introduction

At present there is much interest in switched capacitor (SC) circuits for data conversion. Probably the simplest circuit suitable for D/A conversion is the one presented by Suárez *et al.* [1].

A major limitation to the accuracy performance of SC converters is the accuracy of the capacitor ratios. To achieve 12-bit or more accuracy, this problem has to be remediated. Various solutions such as the ratio-independent method were described [2]. However, this was developped for A/D conversion and is less suitable for D/A conversion. In [3] another approach, based on an integrating circuit, is presented. This is slow. Other techniques use trimming of components, which is subject to aging, and complicates process technology. In this paper, a new analog-domain approach is discussed. The new method does not require trimming of components and is insensitive to aging. There is no calibration cycle required at startup either.

Recently, a DAC circuit was presented that uses compensating switching to reduce the effect of capacitor mismatch [4]. For many digital input values, the mismatch can be eliminated. However, for a number of values the capacitor mismatch cannot be compensated for. Here, an element is added to the compensating switching method: each digital input value y is split into two values $y = x_1 + x_2$. It is shown that for the case of a pair number of bits, every y can be written as the sum of two x_i, for which the effect of capacitor mismatch can be eliminated through compensating switching.

II. Compensating Switching

Fig. 1 shows the compensating switching DAC circuit. An n-bit D/A conversion is carried out in n steps from LSB to MSB. Every step consists of a charging phase Φ_1 and a redistribution phase Φ_2. In the charging phase, one of the capacitors is either charged to a reference voltage or discharged to ground. In the redistribution phase, the charge is redistributed over both capacitors. The new element of the compensating switching method is that every step is carried out in one of two modes. In mode I, the first capacitor is used as charging capacitor, whereas in mode II the second capacitor becomes the charging capacitor.

Let us now investigate the effect of a capacitor mismatch ε for this circuit. We define ε as

$$\varepsilon = \frac{C_2 - C_1}{C_2 + C_1}. \tag{1}$$

Manuscript received November 27, 1995; revised January 7, 1997. This paper was recommended by Associate Editor J. Franca.

The authors are with The Laboratory of Electronics and Information Systems (ELIS), University of Ghent (RUG), B-9000 Gent, Belgium.

Publisher Item Identifier S 1057-7122(98)01574-8.

Then, the output voltage for an n-bit D/A conversion may be written as [4]

$$\frac{V_{\text{out}}(x,\mathbf{t})}{V_{\text{ref}}} = \sum_{i=0}^{n-1} 2^{i-n} b_i \frac{(1 - t_i\varepsilon)}{(1 + t_i\varepsilon)} \prod_{k=i}^{n-1}(1 + t_k\varepsilon). \tag{2}$$

Here, t_i represents the mode of operation: $t_i = 1$, if the ith conversion step is carried out using mode I and $t_i = -1$ if mode II is used instead. The bits b_i form the binary representation of the input value x. They are numbered from 0 (LSB) to $n - 1$ (MSB), where n represents the total number of bits. Let us now designate $F_{(n)}(x,\mathbf{t})$ the deviation from the ideal output voltage for value x and for switching sequence described by the vector $\mathbf{t}$. The subscript (n) indicates the number of bits. Since ε is small, we can linearize (2), yielding

$$F_{(n)}(x,\mathbf{t}) = \varepsilon V_{\text{ref}} \sum_{i=0}^{n-1} b_i 2^{(i-n)} \left(-t_i + \sum_{j=i+1}^{n-1} t_j \right). \tag{3}$$

Here, the convention is used that the second summation is not carried out for $i = n - 1$.

Now the t_i can be chosen to minimize the absolute value of the error. This optimal switching sequence, which depends on the input value x, may be stored in a ROM or can be calculated by an algorithm presented in [4]. We shall designate $f_{(n)}(x)$ the value of the error for this optimal switching sequence.

The proposed new method consists of writing each digital input value y as $y = x_1 + x_2$. For both values x_i, a separate DAC with optimal compensation is used. After the conversion of the x_i, we have capacitors charged to V_1 and V_2, corresponding to x_1 and x_2. Then these voltages V_1 and V_2 are summed by an analog circuit.

A circuit which, in principle, can perform a ratio-independent addition, is shown in Fig. 2. However this circuit suffers from parasitic capacitance, which may significantly deteriorate its operation. This problem was addressed before in [2]. A similar ratio-independent addition technique can be applied here as well.

III. Splitting y into $y = x_1 + x_2$

In this section, it is shown that we can always find an x_1 and x_2 such that $f_{(2N)}(x_1) = 0 = f_{(2N)}(x_2)$. Most of the proofs of the properties in this section, however, are not interesting themselves and are therefore omitted.

A. The 2-Bit DAC

Let us first consider an example where we investigate the error with compensating switching for a 2-bit DAC. The possible input values are 0, 1, 2, and 3. It is obvious from (3) that $f_{(2)}(0) = 0$. For $f_{(2)}(1)$, we get

$$F_{(2)}(1,\mathbf{t}) = (-t_0 + t_1)\frac{V_{\text{ref}}\varepsilon}{4}$$

which equals 0 for $t_0 = t_1$. For $f_{(2)}(2)$ we get

$$F_{(2)}(2,\mathbf{t}) = 2(-t_1)\frac{V_{\text{ref}}\varepsilon}{4}$$

which cannot be eliminated, since t_1 equals 1 or -1. Finally, for $f_{(2)}(3)$, we obtain

$$F_{(2)}(3,\mathbf{t}) = (-t_0 - t_1)\frac{V_{\text{ref}}\varepsilon}{4}$$

which equals 0 when $t_0 = -t_1$.

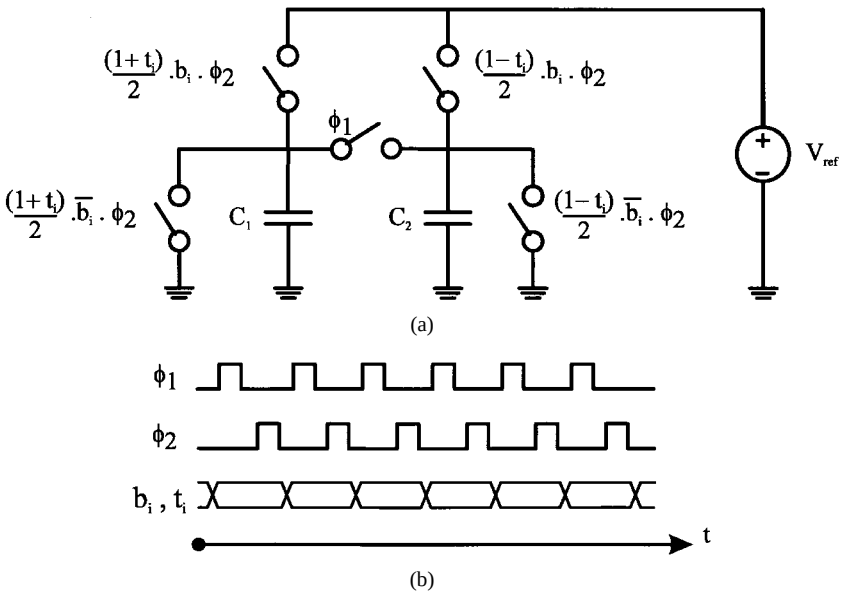

Fig. 1. (a) Compensating switching DAC circuit, with indication of the switch driving signals. (b) Required timing for a 6-bit D/A conversion.

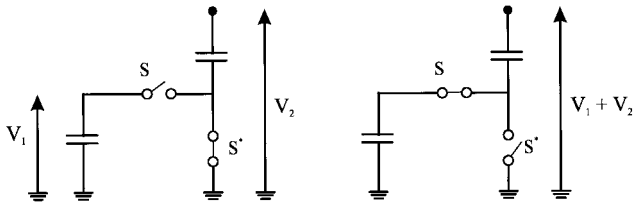

Fig. 2. Basic idea for exact analog summation circuit.

From the above we can see that for a 2-bit DAC only the values 0, 1, and 3 can be compensated. The value 2, however, can be written as $2 = 1 + 1$. Therefore the effect of capacitor mismatch can be eliminated for the input value 2 as well.

B. Input Values That Can Be Compensated

From now on we shall restrict ourselves to $n = 2N$-bit DAC's. In Fig. 3, the resulting error with optimal compensating switching $f_{(n)}(x)$ is plotted versus x for the case of a $n = 10$-bit DAC. An obvious mirror symmetry around the middle can be noticed, yielding the following property.

Property I: $\forall x$ with $0 < x < 2^n$: $f_{(n)}(x) = f_{(n)}(2^n - x)$.

After further investigation, the following properties also show up.

Property 2: If $f_{(2N-2)}(x) = 0$ for the value x, then $f_{(2N)}(x) = 0$ too.

Property 3: If $f_{(2N-2)}(x) = 0$ and $x < 2^{2N-3}$, then $f_{(2N)}(2^{2N-2} + x) = 0$ too.

With properties 1–3, an algorithm can be formulated to construct a set S_N of x_i with $f_{(2N)}(x_i) = 0$ for a $2N$-bit DAC, from a set S_{N-1} of x_i for a $2N$-2-bit DAC, and thus recursively from the set of a 2-bit DAC. A flowchart of the algorithm is shown in Fig. 4(a).

It can be verified that the number of elements in the set S_N for the $2N$-bit DAC equals $\#S_N = 3^N$.

C. Splitting Algorithm

Theorem 1: Every $y < 2^{2N}$ can be written as $y = x_1 + x_2$ with $x_1, x_2 \in S_N$.

Also here, we shall not give a complete, formally correct proof. However, to give more insight, the concept of a possible proof is briefly discussed. First the interval $[0; 2^{2N}]$ has to be partitioned in six subintervalls: $[0; 2^{2N-2}]$, $[2^{2N-2}; 2^{2N-2} + Y_{N-1}]$, $[2^{2N-2} + Y_{N-1}; 2^{2N-1}]$, $[2^{2N-1}; Y_N]$, $[Y_N; 3.2^{2N-2}]$, and $[3.2^{2N-2}; 2^{2N}]$. Here, Y_N is defined as the smallest $y > 2^{2N-1} \in S_N$. It can be verified that Y_N equals $(2^{2N+1} + 1)/3$.

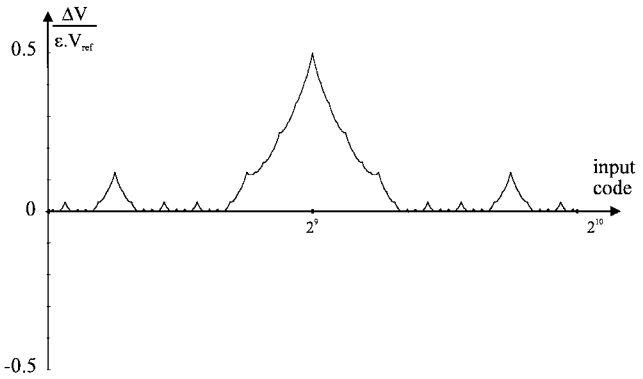

Fig. 3. Calculated error $\Delta V/\varepsilon V_{\rm ref}$ for optimal compensating switching.

Now the theorem can be proved through induction on N.

- We know from the example in Section III-A that the theorem is correct for $N = 1$.
- Suppose the theorem is correct for $N - 1 \Longrightarrow$ correct for N too?

  The basic idea is to find $y^* < 2^{2N-2}$ for every $y < 2^{2N}$. Then we know from the induction hypothesis that we can find $x_1^*, x_2^* \in S_{N-1}$ such that $y^* = x_1^* + x_2^*$. With such x_1^*, x_2^* there have to be corresponding x_1 and x_2 that can be derived from x_1^* and x_2^* through the properties 1–3. Doing such a derivation for all six subintervals proves the theorem.

Following a similar concept as mentioned for the proof, an algorithm can be constructed to split up $y = x_1 + x_2$ automatically: each $y \in [0, 2^{2N}]$ is reduced to an $y^* \in [0, 2^{2N-2}]$. This is done as follows: first the interval to which y belongs has to be determined; then a reduction is carried out corresponding to the interval, e.g., for the fourth interval $[2^{2N-1}; Y_N]$ we get $y^* = y - 2^{2N-1} = x_1^* + x_2^*$. The corresponding x_1 equals $x_1 = x_1^* + 2^{2N-2}$. To be able to reconstruct x_1, the value 2^{2N-2} has to be stored in some variable

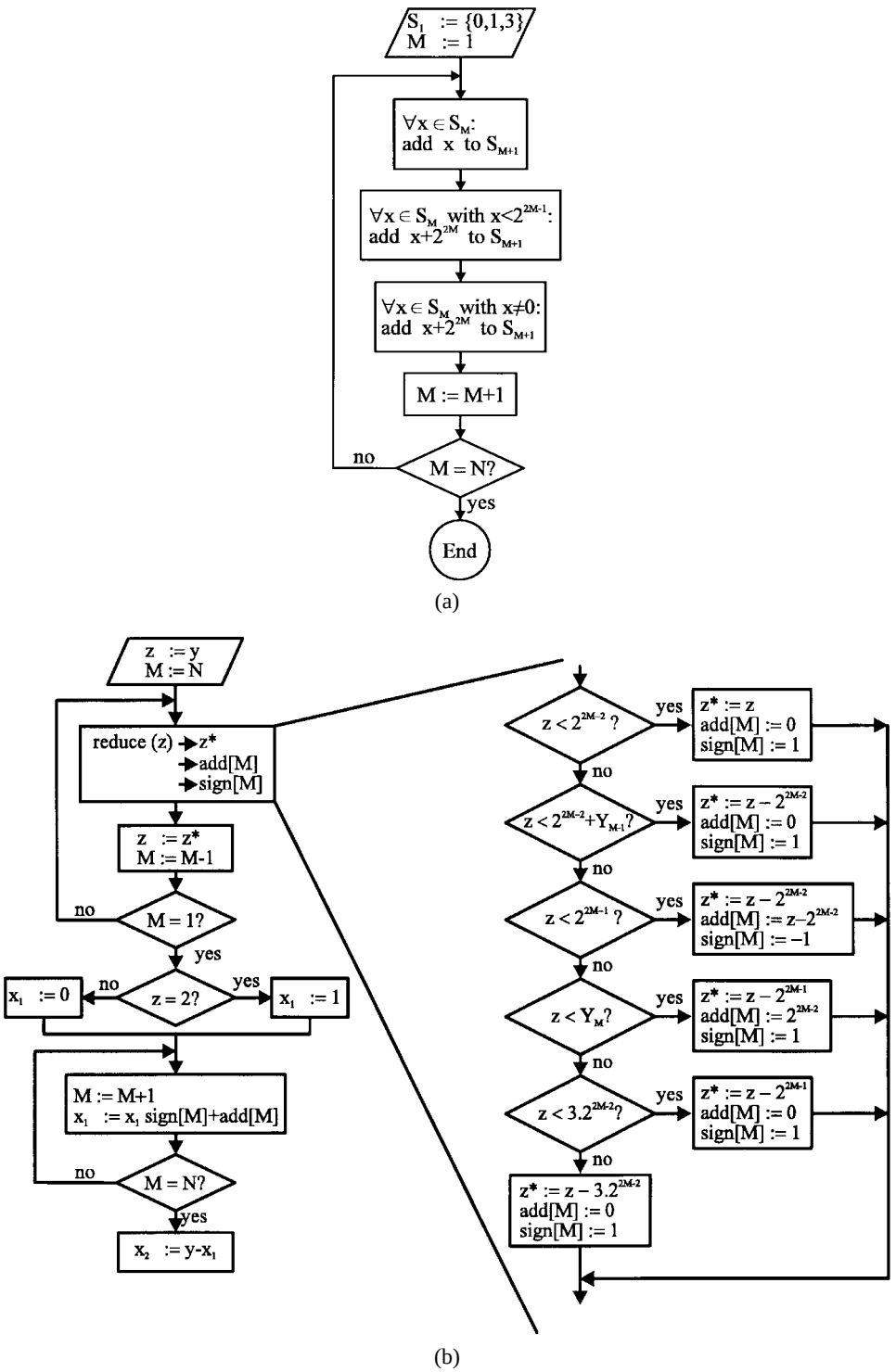

Fig. 4. Flowchart of the algorithm for (a) constructing the set S_N and (b) splitting the input value.

"add[N]." In general, the sign of x_1 also has to be stored in a variable "sign[N]." This is continued until we have $y^{*\cdots*} \in [0, 4[$. Then it is trivial to determine the value of $x_1^{*\cdots*}$. Then the value of x_1 can be constructed from $x_1^{*\cdots*}$ and the arrays "add[$2..N$]" and "sign[N]." The value of x_2 can then be determined through $x_2 = y - x_1$. A flowchart of a possible implementation is represented on Fig. 4(b).

Both the algorithm to construct the optimal switching sequence [4] and the new splitting algorithm are of order N. So we see that all data that are necessary to follow the proposed method can be calculated during one conversion cycle.

IV. Accuracy Limitations

Next to capacitor mismatch, there is another source of nonlinearity for the two-capacitor DAC, i.e., charge injection: when a FET switch opens, the charge from the conducting channel is injected in the adjacent nodes. There is an analogous effect when the switch closes [1].

Let us first investigate the effect of charge injection during the redistribution phase [Fig. 5(a)]. When switch S_1 closes, the charge to form the conducting channel of the switch-transistor Q_{channel} is injected from both capacitors. When switch S_1 opens, the channel

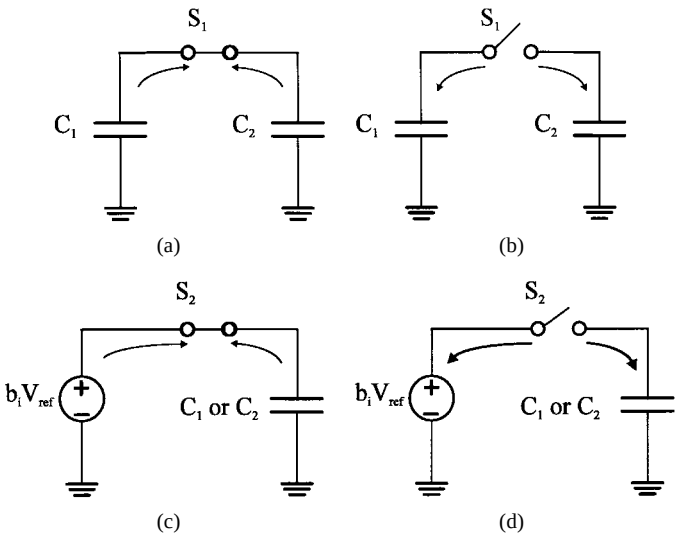

Fig. 5. (a) Charge injection during the redistribution phase. (b) Charge injection during the charging phase.

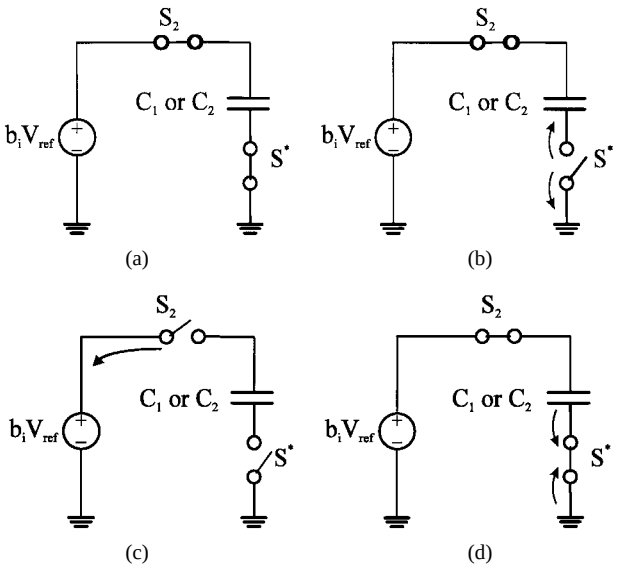

Fig. 6. Effect of the auxiliary switch S^*.

pinches off and the charge in the channel Q_{channel} is re-injected, into the capacitors. Since capacitors C_1 and C_2 are nominally equal, the injected charge to both capacitors is almost equal as well. From this analysis, we can conclude that charge-injection from the switch S_1 does not affect the circuit performance.

Let us now investigate charge injection during the charging phase [Fig. 5(b)]. When switch S_2 closes, charge injection has no effect since the capacitor is connected to the voltage source. When switch S_2 opens, however, a signal-dependent charge is injected into capacitor C. A method to eliminate the signal-dependent part of this charge injection is well known for A/D converters [2]. The same idea may also be applied to this circuit. Therefore we add an auxiliary switch S^* which is controlled by a third clock phase ϕ_3 (Fig. 6). Switch S^* is opened a little bit before switch S_2. When switch S_2 opens, the charge cannot flow to C and is injected to the low-impedance voltage source. Then switch S^* closes again. The resulting charge injection from S^* is not signal-dependent, because both source and drain voltages of the FET switch S^* are constant.

According to the previous discussion, the effect of charge injection is reduced to a constant offset, which can be tolerated from a linearity point of view. However, there may remain some small signal dependency. Indeed, we have implicitly assumed that the only charge injection comes from the transistors conducting channel. In practice, charge injection can also occur from the nonlinear gate-source and gate-drain parasitic capacitances.

Another important performance-limiting factor for high-accuracy SC circuits is noise. For the two-capacitor DAC, the main source is kT/C thermal noise from the switched capacitors. For a given reference voltage, this can easily be reduced by increasing the capacitance. For a 2.5-V reference, 10-pF capacitors are required to obtain 16-bit performance.

We have performed HSPICE simulations to verify the validity of the techniques. EUROPRACTICE simulation data for Mietec's 2.4-μm CMOS processes were used. The switches had minimum size. For numerical reasons, we used 100-fF capacitors in the simulations. With the technique explained above, the signal-dependent part of the charge injection was reduced to about 3 mV instead of 25 mV for the conventional circuit. For sufficiently large capacitors, the effect of charge injection is inversely proportional to the capacitance value. Therefore, this allows for 16-bit performance in this technology when using the kT/C limited capacitance value of 10 pF, combined with minimum sized switches. For a given technology, we can consider a speed-accuracy tradeoff: if large-width switches are used to increase the speed, the resulting charge injection becomes worse, hereby affecting the linearity. If small-width switches are used, less charge is injected and therefore higher accuracy can be obtained, but the conversion speed is limited.

At this point, it is important to note that the new technique only eliminates the first-order effect of capacitor mismatch. Therefore, at least 8-bit capacitor ratio accuracy is required for 16-bit performance. A 0.3% capacitor ratio accuracy is easily achievable in standard CMOS technology, however. At the 16-bit level, the effect of capacitor nonlinearity appears also. Based upon the known performance of existing circuits [3], however, we conclude that 16-bit linearity might be achievable with the proposed new techniques.

V. Conclusion

A new method to eliminate the effect of capacitor mismatch for a SC DAC is described. It is insensitive to aging of components and does not require a calibration cycle at startup.

However, additional data are needed for the compensating switching and for splitting the input digital words. Reading them from a ROM becomes expensive when the resolution rises. Both the data for switching and splitting can also be calculated by simple algorithms, that take $O(N)$ operations.

Parasitic effects are investigated. A method to reduce the effect of charge injection is presented. From the analysis, we can conclude that a resolution of up to 16 bit may be achieved with the proposed new method.

References

[1] R. E. Suarez, P. R. Gray, and D. A. Hodges., "All-MOS charge redistribution analog-to-digital conversion techniques—Part II," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 379–385, Dec. 1975.

[2] P. W. Li, M. J. Chin, P. R. Gray, and R. Castello, "A ratio-independent algorithmic analog-to-digital conversion technique," *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 828–836, Dec. 1984.

[3] M. J. M. Pelgrom and M. Roorda, "An algorithmic 15-bit CMOS digital-to-analogue converter," *IEEE J. Solid-State Circuits*, vol. 23, pp. 1402–1405, Dec. 1988.

[4] L. Weyten and S. Audenaert, "Two-capacitor DAC with compensative switching," *Electron. Lett.*, vol. 31, no. 17, pp. 1435–1436, Aug. 17, 1995.