Using Different Weights in DACs

Mark Vesterbacka[1], K. Ola Andersson[1,2], Niklas U. Andersson[1,2], and J. Jacob Wikner[3]

[1]Dept. of E.E., Linköping University, SE-581 83 Linköping, Sweden, fax +4613139282

[2]LiDC, Ericsson Microelectronics AB, Box 1554, SE-581 15 Linköping, Sweden

[3]SDC, Ericsson Microelectronics, Westmead Dr., Swindon, SN5 7UN, UK, fax +441793490151

E-mail: {olaa, niklasa, markv}@isy.liu.se, jacob.j.wikner@mic.ericsson.se

Summary

In this paper we discuss some properties of different codes with their respective sets of weights to be used in digital-to-analog converters (DACs). The thermometer (unratioed) code is widely used instead of a binary code in the most significant bits of a segmented DAC to reduce errors due to weight and timing mismatch. The binary and thermometer codes are two extremes, where the first-offers a small digital hardware cost and the latter a large cost. We have investigated some of the properties of these codes and codes with properties in-between; such as linear, polynomial, and segmented codes. Some new ideas and results on using different sets of weights and how to generate them are presented. We present simulation results for some low-order polynomial codes.

Keywords: DACs, nonlinearity measures, codes

1. Introduction

In digital-to-analog converters (DACs) we add a number of analog reference levels or weights that are controlled by the digital input bits. The input bits select which weights to add to represent a certain digital input at the analog output. In the memory-less case, a DAC performs the operation

$$A(nT) = \sum_{k=0}^{K-1} w_k \cdot b_k(nT), \tag{1}$$

where w_k is the weight, $b_k(nT)$ is the k-th bit, $A(nT)$ is the analog output, and K is the number of weights. The weights $\{w_k\}$ can be chosen arbitrarily as long as we are able to represent all values of A between its minimum and maximum. Typically, we refer to an N-bit converter when the number of levels that can be represented is 2^N. For the *offset binary code*, we use a set of weights given by

$$\{w_k\}_{k=0}^{K-1} = \{2^k\}_{k=0}^{K-1}. \tag{2}$$

The most common way to make the circuit less sensitive to mismatch errors is to use the thermometer (unratioed) code for which all weights are equal, e.g., $w_k = 1$, where $K = 2^N - 1$ in (1). Thermometer code requires a large digital encoder – the amount of hardware grows exponentially with the number of bits – but the converter becomes less sensitive to typical errors in the analog domain, since layout "tricks" can be applied more easily [1].

Typically, glitches due to timing mismatch, and nonlinearity due to weight mismatch, become more dominating if a code has a large spread in weights. For the binary code, weight w_k is twice the size of weight w_{k-1}, but for the thermometer code they would be equal.

Due to the large hardware cost for the thermometer code, segmentation is normally used where, e.g., the M most significant bits (MSBs) are thermometer coded (unratioed) and the $N-M$ least significant bits (LSBs) are binary coded. We have $w_k = 2^k$ for $k < N-M$ and $w_k = 2^{N-M}$ for $k \geq N-M$ and the total number of weights becomes $K = N - M + 2^M - 1$. To find a reasonable trade-off between performance and hardware cost, optimization algorithms are applied to solve the problem.

In this work, we discuss different codes that end up in-between, or as variations of, the cases above. We have previously considered a set of linearly increasing weights [2] and seen that it can be modified to yield very good glitching properties [3]. For this case, the weights are increasing as $w_k = 1 + k$ for $k = 0, 1, \ldots, K-1$, where the number of bits is $K \approx 2^{(N+1)/2}$ when N is high.

In the following we extend this approach by considering more generalized codes that have weights, w_k, that are determined by arbitrary polynoms, e.g.,

$$w_k = \sum_{m=0}^{\infty} a_m k^m \tag{3}$$

or given by, e.g., a difference equation

$$w_k = \sum_{m=1}^{k} a_{k-m} w_{k-m}. \tag{4}$$

In Sec. 2 we discuss some boundaries and requirements on the weights to be used in a DAC. In Sec. 3 we discuss a generalized polynomial set of weights, and codes generated with difference equations. In Sec. 4 we discuss the special case of combined DACs and how different set of weights can be used. Some simulation results are given in Sec. 5. The work is concluded in Sec. 6.

2. Boundaries and properties

To simplify our notation of the weights, we use an ordered set where

$$w_k \geq w_{k-1} \text{ for } k = 2, \ldots, K. \tag{5}$$

Further, we restrict the weights to be positive integers, $w_k \in Z_+$. To be able to represent all values between 0 and $2^N - 1$, we also require that

$$w_k \leq 1 + \sum_{m=0}^{k-1} w_m. \tag{6}$$

Both requirements are fulfilled by the codes described in Sec. 1.

Note that some codes are redundant, i.e., different combinations of weights can be used to represent the same value at the output. This redundancy can, and in most cases should, be used to apply dynamic matching or calibration techniques to the DAC [5, 6].

Besides the benefit from matching and calibration techniques, we do also motivate this work by noting that the digital technology is expected to scale faster than analog and therefore we can allow a rather high digital hardware complexity to reduce the sensitivity to typical analog errors in a DAC. Hence, by investigating different codes we will have a higher degree of freedom in the trade-off between digital complexity and impact of typical analog errors in the future.

Previously we touched upon the issue that if the weights of the DAC have a large spread, the converter will be more sensitive towards mismatch errors. Therefore, we introduce the variance of the weights as a rough measure on the quality of the chosen weight set. (This measure is only valid as long as no randomization is applied.) We define this variance as

$$V_w = \frac{1}{K} \sum_{k=0}^{K-1} (w_k - \overline{w_k})^2, \tag{7}$$

where $\overline{w_k}$ is the mean value of all weights. For the binary case, this becomes approximately $V_w = 2^{2N}/3N$ for large N. For the thermometer code it is $V_w = 0$, and for the linear code, we get $V_w = 2^{N-1}/3$.

If we have a 14-bit converter, the variance for the binary case will be approximately 5×10^6, for the linear case 3000, and for a segmented converter with the 6, 7, 8 MSBs unratioed, we get 5000, 600, and 70.

These values (roughly) indicates that the binary case is worse than the thermometer case, and that the linear and segmented codes end up between the two extremes.

3. Weight sets

In this section we describe a generalized weight set for DACs that we call a *polynomial* weight set. We also discuss weight sets generated from a *difference equation*.

Polynomial weight sets

If we choose polynomials to describe the strength of the weights, we assume that the weights are given by a general expression as

$$w_k = a_0 + a_1 \cdot k + a_2 \cdot k^2 + \ldots \tag{8}$$

where a_i are characteristic coefficients.

For the thermometer coded case we get the coefficient a_0 to be 1, and all other coefficients are 0. For the linear code [2], we get all coefficients equal to zero, except $a_0 = a_1 = 1$, and for the binary code we would have an infinite number of coefficients described by

$$a_i = \frac{(\ln 2)^i}{i!}. \tag{9}$$

For example, we may assume a set of weights that is described by a second-order polynomial

$$w_k = a_0 + a_1 \cdot k + a_2 \cdot k^2. \tag{10}$$

If we let $w_k = 2^k$ for the first few k, all other weights will be integer numbers, since they can be written as linear combinations of the inital weights. Note that the choice of binary weights ensures that we will be able to represent all values. Another feasible choice is to let the three first weights be 1, 2, and 3. If we choose the first three weights to be e.g. 1, 2, and 5, we can obviously not represent the number 4. With $w_k = 2^k$ for $k = 0, 1, 2$ we get the coefficients $a_0 = 1$ and $a_1 = a_2 = 1/2$. Hence, the weights in our example are given by

$$w_k = 1 + \frac{k}{2} + \frac{k^2}{2} = 1 + \frac{k(k+1)}{2}, \tag{11}$$

which become $w_k = 1, 2, 4, 7, 11, 16, \ldots$.

If we choose the first three weights to be 1, 2, and 3, we end up with $a_o = a_1 = 1$ and $a_2 = 0$, which yields the linear code.

A third-order polynomial description gives us the weights

$$w_k = a_0 + a_1 \cdot k + a_2 \cdot k^2 + a_3 \cdot k^3. \tag{12}$$

In this case we have to set the first four weights according to, e.g., the exponential values $w_k = 2^k$ for $k = 0, 1, 2, 3$. This inserted in (12) gives us the coefficients $a_0 = 1$, $a_1 = 5/6$, $a_2 = 0$, and $a_3 = 1/6$. The weights become $w_k = 1, 2, 4, 8, 15, 26, \ldots$. This approach can be extended to any polynomial order.

However, the polynomial might not describe a convex function for all initial values, hence in some cases the weights will diverge and typically the requirement in (5) is not met. The feasibility of the derived coefficients has to be checked.

Difference-equation weight sets

An alternative way to describe the weights is to use a difference equation. A general difference equation would be

$$w_k = a_0 + a_1 w_{k-1} + a_2 w_{k-2} + \dots , \tag{13}$$

whereto we also add required initial conditions. Then we set restrictions on the coefficients a_i to achieve ordered integer weights. We have, e.g., for the binary code that

$$w_k = 2w_{k-1} \text{ and } w_0 = 1 . \tag{14}$$

For the linear code, we have

$$w_k = 1 + w_{k-1} \text{ and } w_0 = 1 . \tag{15}$$

It is obvious that the codes from (14) and (15) requirements (5) and (6) are met. An example on a set of weights generated from a difference equation is Fibonacci numbers, where the weights is determined by the difference equation $w_k = w_{k-1} + w_{k-2}$.

4. Weights in combined dual DACs

In [3] and [4] we have discussed how the outputs of two DACs can be combined in order to improve on the impact of glitches and finite output impedance. Typically, we use two DACs and define the total output signal as the sum or the difference between between the individual DAC outputs.

Binary Weights

If we use binary weights $(w_k = 2^k)$ in the two DACs we are able to use redundancy, since we can represent e.g. the number 10 by using $10 = 12 - 2 = [8 + 4] - [2]$ or $10 = 13 - 3 = [8 + 4 + 1] - [2]$, etc. The numbers within brackets denote the bit weights used in each DAC. For this case, we can apply randomization techniques to smooth distortion into noise as described in for example [4].

Ternary Weights

Another option is to choose a weight set where the weights are given by $w_k = 3^k$, i.e. 1, 3, 9, 27, etc. This will give us a code which is non-redundant. Hence, there is only one single subset of weights allowed to represent a certain number. So, the only way to represent 10 is to use the combination $10 = 10 - 0 = [9 + 1] - [0]$. In this case no randomization can be applied due to lack of redundancy.

The output range of an M-bit ternary code used in two DACs with their outputs subtracted is given by the range between the minimum and maximum values: $-(3^M - 1)/(3 - 1), \dots, (3^M - 1)/(3 - 1)$. Hence,

$$R_3 = 2 \cdot \frac{3^M - 1}{3 - 1} = 3^M - 1 . \tag{16}$$

The (combined) range of the binary N-bit coded DACs is given by

$$R_2 = 2 \cdot \frac{2^N - 1}{2 - 1} . \tag{17}$$

Comparing the two ranges, $R_2 = R_3$, gives us a relation between the number of required bits for the two cases

$$N = \log_2 \frac{3^M + 1}{2} \text{ and } M = \log_3(2^{N+1} - 1) . \tag{18}$$

For larger M and N we have the approximate result that

$$N \approx 1.58M - 1 \text{ and } M \approx 0.63 \cdot (N + 1) . \tag{19}$$

To represent the same range as a 13-bit binary code, we need 9 bits in the ternary code, etc., to the cost of less redundancy.

Golomb Ruler

From the ternary code (as well as, e.g., the binary and thermometer codes) it can be noticed that different weights may be used to represent an output value (as also was shown in the previous examples). This will make the code sensitive to timing mismatch between internal switches associated with different weights. One set of weights that allow us to use only one single weight (from each DAC) to generate a code is determined by the marks on a Golomb ruler [7].

The Golomb ruler is divided into a number of marks. To measure a certain unique distance we use two and only two unique marks (except for the special–and obvious–case of 0). This corresponds to using two unique weights, one from each DAC, and the difference is the “distance”. This will make each DAC insensitive to internal timing mismatch between switches.

For example, we may use the weights $w_k = 6, 4, 1$ (and 0) in each DAC. This set of weights is the marks of one Golomb ruler. The difference between the DAC outputs is given by all combination of pairs of weights between each DAC. Hence

$$\begin{aligned} &6 = 6 - 0, 5 = 6 - 1, 4 = 4 - 0 \text{ and} \\ &3 = 4 - 1, 2 = 6 - 4, 1 = 1 - 0 \end{aligned} \tag{20}$$

More Golomb rulers can be found in [7].

5. Simulated impact on performance

We have simulated four DACs in Matlab, where one was segmented, one was linear-coded, and two were based on low-order polynomial codes. In the simulations, the bits for the different codes are generated with a simple MSB-first successive approximation algorithm.

To measure the performance the static nonlinearity is investigated, where the differential and integral nonlinearity (DNL and INL) typically are used. Assuming the same set of mismatch errors, these measures will be differently distributed when different codes are applied. In Fig. 1, the DNL of a segmented 14-bit DAC is plotted as a function of the number of thermometer-coded most-significant bits. The absolute DNL and INL has been averaged and the stochastic mismatch variation for each element in the DAC has a standard deviation of 1%. The plot shows that the maximum DNL decreases with the number of thermometer-coded bits. In Fig. 2 the INL is plotted for the same simulation setup.

The same type of simulation has been performed for the other codes. We have investigated the DNL for the linear code and two of the polynomial codes described by (11) and (12). We find that the linear code has improved linearity over other codes, except for the case of a segmented code with more than 7 unratioed MSBs in the DNL case. For INL the corresponding number of unratioed MSBs is 4.

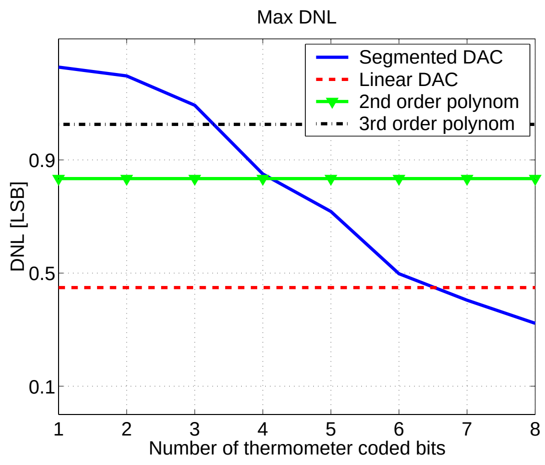

Figure 1: Simulated maximum DNL vs. number of unratioed bits in a 14-bit DAC.

6. Conclusions

We have presented a brief discussion on alternative codes or weight sets to be used for D/A conversion in single and dual DACs.

Acknowledgments

The authors would like to thank Prof. Lars Wanhammar at Linköping University, Sweden and Dr. Gunnar Björklund, Ericsson Microelectronics AB, Stockholm, Sweden.

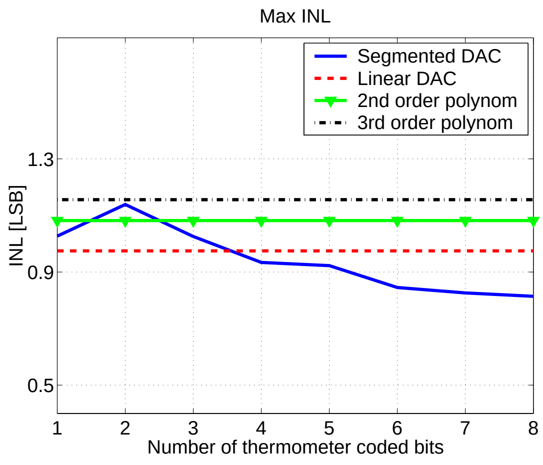

Figure 2: Simulated maximum INL vs. number of unratioed bits in a 14-bit DAC.

References

[1] C.-H. Lin, K. Bult, “A 10-b, 500-MSample/s CMOS DAC in 0.6 mm^2,” *IEEE Journal of Solid-State Circuits*, Vol. 33, No. 12, Dec. 1998, pp. 1948-58.

[2] J.J. Wikner, M. Vesterbacka, "D/A Conversion with Linear-Coded Weights," Proc. *2000 Southwest Symp. on Mixed-Signal Design, SSMSD’00,* pp. 61-66, San Diego, CA, Feb. 27-29, 2000.

[3] M. Vesterbacka, "Linear-Coded D/A Converters with Small Relative Error Due to Glitches," Proc. *IEEE 2001 Midwest Symp. on Circuits and Systems, MWSCAS’01*, vol. 1, pp. 280-283, Fairborn, Ohio, Aug. 14-17, 2001.

[4] K.O. Andersson, et al. “Combining DACs for Improved Performance,” accepted to the *4th Intern’l Conf. on Advanced AD and DA Conversion Techniques and their Applications*, Prague, 2002.

[5] I. Galton, “Spectral shaping of circuit errors in digital-to-analog converters,” *IEEE Transaction on Circuits and Systems-II*, Vol. 44, No. 10, Oct. 1997, pp. 808-17

[6] S. Boiocchi, et al., “Self-calibration in high speed current steering CMOS D/A converters,” in Proc. *2nd Intern’l Conf. on Advanced AD and DA Conversion Techniques and their Applications*, 1994, pp. 148-52.

[7] J.P. Robinson, “Optimum Golomb rulers,” *IEEE Transaction on Computing*, vol. C-28, pp. 943-4, Dec. 1979.