OFFSET-SIMULATION OF COMPARATORS

Achim Graupner, Udo Sobe
ZMD AG, Dresden
achim.graupner@zmd.de

ABSTRACT: A simple methodology for determining the input referred offset voltage of comparators is presented. This in general is difficult as the output of a comparator is discrete valued. The method relies on a Monte-Carlo-Simulation with certain comparator input values and some postprocessing of the comparator output data. The comparator is always operated in its intended environment, there is no modification of the comparator itself nor some unusual stimuli required. There is also no known restriction for the type of comparators to be analyzed.

INTRODUCTION

Comparators are a frequently used building block in analog circuit design. One of its most important properties is its input referred offset. For the most simple implementation, an amplifier with a high gain, this is a simple task that can be carried out with the help of a feedback network and some Monte-Carlo simulation.

However, most circuit implementations of comparators are much more complicated. Comparators often employ some hysteresis or some clever clocking scheme to reduce power dissipation or offset. In those cases however the determination of the input referred offset is no longer simple at all. In the latter case DC simulations are no longer sufficiently as the operation depends on the circuit’s transient behavior. In the case of hysteresis a clever DC analysis has to be carried out because the circuit provides multiple operating points.

An method for determining the offset is described in [1]. Its basic idea is to build a first-order sigma-delta-modulator comprising the comparator and an ideal integrator. When steady state is reached, i.e. the high and low times of the comparator output are equal, then the mean input value equals the sum of the comparators threshold value plus the comparators offset. The method is quite simple but to achieve a meaningful accuracy the simulation time has to be considerably long until sufficient settlement. If the comparator employs hysteresis then this method does not compute the two switching levels and their offset but a mean threshold value.

This short paper presents a very simple methodology to determine the offset of arbitrary comparators with the help of transient simulations. It requires a very simple testbench and some simple postprocessing only. In the following the method is introduced and illustrated with the help of an example.

SIMULATION SETUP

For the proposed method a testbench of the comparators with is normal surroundings as supply and clock is used. Additionally a dedicated input has to be provided. The input value of the comparator has to be ladder-shaped, see figure 1. This signal can be generated with a triangular shaped pulse followed by some ideal sample-an-hold or with a dedicated block in an analog hardware description language (AHDL). For each input value \( x_i \) the comparator is activated and its output value \( y_i \) is stored. For an ideal comparator for all input values \( x_i \) below the threshold value \( x_{th} \) the output value \( y_i \) is stored. For all values greater it is \( 1 \). This setup is shown in figure 1.

Taking device parameter mismatch into account this behavior might randomly change. This means the comparator output might be \( 0 \) even if \( x_i > x_{th} \) or may be \( 1 \) although \( x_i < x_{th} \).

The behavior of the comparator eventually depends on the actual device parameters, especially their matching. This effect can be modeled with an error signal \( x_{\text{off}} \) with is added to the comparator’s input with all device parameters of the comparator at their nominal value, see figure 2.

To evaluate the influence of the random device parameter variation Monte-Carlo-analysis is used. The result of each Monte-Carlo-iteration is collected. Now for all Monte-Carlo-samples for each input value the probability

\[
P(y_i = 1) = P((x_i - x_{th}) > x_{\text{off}}) = \frac{n_i}{N} = z_i
\]

(1)

can be estimated, where \( N \) is the number of Monte-Carlo-iterations and \( n_i \) is the number of runs where the comparator output is 1 when \( x_i \) is applied. This function is the integral of \( x_{\text{off}} \)’s probability density function. Hence from (1) the statistical properties from \( x_{\text{off}} \) can
be easily computed. In order to do so (1) can be plotted in a normal probability plot. This is a graphical technique for assessing whether or not a data set is approximately normally distributed. The data are plotted against a theoretical normal distribution in such a way that the points should form an approximate straight line. From this line the mean value and the standard deviation can be determined.

In order to generate a normal probability plot the inverse of the cumulative normal distribution function is applied to the $z_i$

$$v_j = \Phi^{-1}(z_i)$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{t^2}{2}\right) dt$$

$$= \frac{1}{2} \left[ 1 + \text{erf} \left( \frac{x}{\sqrt{2}} \right) \right]$$

Now a first order polynomial $z = p_1 \times x + p_2$ is fitted through the $x_i$ vs. $z_i$ plot. From the polynomials coefficients the mean value and the standard deviation of the distribution can be calculated:

$$\mu_{x_{\text{off}}} = -p_2 / p_1 - x_{\text{th}}$$

$$\sigma_{x_{\text{off}}} = 1 / p_1$$

This method is illustrated in the following example.

**EXAMPLE**

The figure 3 shows the functional principle of a non-clocked comparator incorporating hysteresis and a output latch.

The result of a 700-runs Monte-Carlo simulation is shown in figure 4. At the input a ramp with 1mV steps is applied. For each step of the ramp a comparison cycle is carried out and the comparator’s result is stored (in our case into a file with means of an AHDL-block). In a subsequent postprocessing for each ramp value the number of comparator output 1 is counted and normalized to the number of runs.

It can seen both the hysteresis of the comparator as well as its sensitivity to device parameter mismatch. For the rising slope and an input voltage of 1.64V in about 60% of all simulations the comparator output had an output of „1“; in the rest it remained „0“.

The simulation data of the rising slope have been plotted in a normal distribution plot, see figure 5. It is seen that the result is to a good approximation a straight line. Hence, the offset of the comparator is normal distributed and its mean value as well its standard deviation can be calculated using (4).

When normalizing the y-axis to a cumulative normal distribution then the statistical properties of the compar-
ator’s threshold value can be easily read out from the diagram (16%, 50% or 84%-values respectively), see figure 6. In this case \( x_{\text{off}} + x_{\text{th}} = 1.64 \) and \( \sigma_{\text{voff}} = 0.018 \).

In order to obtain reliable results the values for the \( x_i \) have to be selected carefully. The difference between two values should be less than the standard deviation of the offset. Also the minimum and maximum values of \( x_i \) should always include the respective comparator threshold, so the interval should span several times the standard deviation around the comparator threshold. Those numbers have to be roughly guessed before starting the simulation.

**SUMMARY**

A very simple methodology to determine the offset of arbitrary comparators with the help of transient simulations has been presented. It requires a very simple testbench and some simple postprocessing only. The comparator can be simulated in its intended operating regime.

**APPENDIX**

In the following Matlab/Octave-Code the above described data processing has been implemented:

**TABLE 1. Matlab/Octave Code**

<table>
<thead>
<tr>
<th>Step</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>% it is assumed that the simulation results are provided % in the matrix &quot;ydata&quot; % and the array &quot;xdata&quot;</td>
</tr>
<tr>
<td>2.</td>
<td>% (1) ni/N, figure 2</td>
</tr>
<tr>
<td>3.</td>
<td>% y(1:21) is this case contains the data % for the rising slope</td>
</tr>
<tr>
<td>4.</td>
<td>% select values within -2.5 ... 2.5 sigma</td>
</tr>
<tr>
<td>5.</td>
<td>% fit line through selected values % figure 5</td>
</tr>
<tr>
<td>6.</td>
<td>% also Matlabs statistic toolbox functions could be used % (normplot, qqplot, normfit)</td>
</tr>
</tbody>
</table>

**REFERENCES**