How to Utilize a Windowing Technique for Accurate DFT

Product Version IC 6.1.5 and MMSIM 12.1
December 6, 2013

By
Michael Womac
Copyright Statement

© 2013 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc. All others are the property of their respective holders.
## Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>4</td>
</tr>
<tr>
<td>Application</td>
<td>5</td>
</tr>
<tr>
<td>Audience</td>
<td>5</td>
</tr>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>Simple DFT Window Comparisons</td>
<td>5</td>
</tr>
<tr>
<td>Sources of Error in a Transient Simulation</td>
<td>8</td>
</tr>
<tr>
<td>Sources of Error in a Discrete Fourier Transform</td>
<td>9</td>
</tr>
<tr>
<td>First DFT Error Source – Interpolation Error</td>
<td>9</td>
</tr>
<tr>
<td>Second DFT Error Source – End Point Error</td>
<td>13</td>
</tr>
<tr>
<td>Understanding the Frequency Placement of the Tones</td>
<td>14</td>
</tr>
<tr>
<td>Understanding the Amplitude of the Frequency Bins</td>
<td>19</td>
</tr>
<tr>
<td>Multiple Tone DFTs</td>
<td>29</td>
</tr>
<tr>
<td>Setting up the Simulation and Plotting the Resulting DFT</td>
<td>31</td>
</tr>
<tr>
<td>Real Circuit Example</td>
<td>32</td>
</tr>
<tr>
<td>Increasing Real Circuit DFT Dynamic Range</td>
<td>37</td>
</tr>
<tr>
<td>Summary</td>
<td>42</td>
</tr>
<tr>
<td>References</td>
<td>42</td>
</tr>
</tbody>
</table>
Purpose

This application note enables you to utilize a transient simulation to perform quick frequency analysis using a rectangular window. This can be accomplished by:

- Defining a precise window for the time to be analyzed
- Defining the frequencies that have an integer number of cycles within the window of interest
- Using Discrete Fourier Transform (DFT) within the rectangular window to capture all frequency data to be analyzed

To achieve these results, you must correctly address the following sources of error that mask the window’s capability:

- Interpolation errors of the signal in the simulator due to time points not located at the times required for the FFT algorithm in the calculator DFT function
- Non-integer number of cycles available for the DFT

Additionally, if set up properly before a transient simulation starts, a rectangular windowed DFT is straightforward and yields a predictable result. This technique is especially useful under the following circumstances:

- If Periodic Steady State (PSS) or Harmonic Balance (HB) simulation is not available
- If a design engineer needs accurate integrated energy or closely-spaced frequencies with minimal simulation time

This application note provides a brief introduction of the sources of error for a DFT, and explains how to set up a transient simulation in ADE using clear fundamentals to achieve repeatable simulation results for periodic signals by defining only 3-4 values in an ADE-L state.

The purpose of this application note is not to replace the Spectrum assistant in ViVA. Its purpose is to help you define the necessary settings prior to the simulation for achieving good DFT results within a transient simulation for circuits that have an integer number of cycles for all frequencies. The techniques described in this application note are applicable to the Spectrum assistant.
Application

This technique is not appropriate for randomized or chaotic signals, such as those resulting from Delta Sigma Modulated circuits or voltage controlled oscillators which require other windowing techniques available in the built-in VIVA DFT calculator function and automated by the Spectrum assistant. This technique is appropriate for driven circuits.

Audience

This application note is intended for users who need to perform frequency response and analysis during a transient simulation. It is also intended for users who need the ability to evaluate high dynamic range requirement for one or many tones, which are closely located and contain both amplitude and power information.

Introduction

The benefit of using a rectangular window for a DFT is generally not well known. Defining a state in Virtuoso ADE-L and MMSIM properly can be an effective tool to quickly run a DFT with little setup information. Since MMSIM has strobing available in the transient which can be controlled by an ADE-L state variable, it enables you to have absolute control of where a time point is placed. This can serve as a reusable setup template.

Simple DFT Window Comparisons

To begin, an example of two sine waves summed together will be used. One sine wave located at 2 Hz and one at 4 Hz. In this example, we will focus on the “fake” harmonics that show up due to setup errors.

In the following example, a comparison of a Rectangular, Cosine2, Cosine4, and Hanning (or Cosine) windows with a simple signal is shown.

![Figure 1. This is a 1Hz and 2Hz sine wave at 1V each summed together that is simulated for 1 second.](image)

These windows provide a good dynamic range, but have some artifacts to consider. Figure 1 displays a plot of the “simple signal” that will be used to compare the windowing techniques mentioned previously.

Further investigation on this approach is required for frequency-dependent circuits, for example, PLLs.
Figure 2 shows the resultant frequency plot when the transient signal of Figure 1 is applied to the aforementioned DFT windows – only the window type is changed from plot to plot. Note that in the Rectangular window, there are two tones located at 2 Hz and 4 Hz with an amplitude of 1 Vp (or 0 dBV) which is exactly what is defined in the source. Also note that all the other windows have more than two tones and different amplitude values for the tones.

Consider the Cosine4 trace in Figure 2. It displays the following results:
- A 1 Hz tone that shows -1.92 dBV
- A 2 Hz tone that shows 2.922 dBV
- A 3 Hz tone that shows 4.0 dBV
- A 4 Hz tone that shows 2.922 dBV
- A 5 Hz tone that shows -1.93 dBV
- A 6 Hz tone that shows -13.9 dBV

Without prior knowledge of the signals that are present, you would not be able to determine the frequencies that are present or their correct amplitudes. In cases where non-linearities create intermodulation distortion and frequency conversion, it is not always known what frequencies are present or what the amplitudes are. Therefore, it is required to have an accurate amplitude and frequency result from the DFT window.

Observe the three windowing analysis techniques and note the smear energy from one frequency bin to another. The rectangular window is unique because it can produce results with little or no smearing. A detailed discussion on the smearing effect is provided later.

The smoothing windows can produce exceptional results at the expense of the simulation time by reducing the resolution bandwidth. In Figure 3, you can see that with four cycles of the resolution band with frequency, Cosine2 can resolve the frequencies and have the correct amplitudes. However, you should take care if you are integrating over a bandwidth with Cosine2 because it will have more energy than the rectangular window, which is accurate. You will have to apply the correct offset to get accurate integrated results.
Figure 3. 2 Hz and 4Hz tones plotted with both Rectangular and Cosine2 Windows with 4 cycles needed of the lowest frequency needed in the previous example.

Sources of Error in a Transient Simulation

Below is a list of the errors present in a transient simulation. Some or all of these errors are present for real circuits at each time point which can accumulate over the simulation time creating small (depending on tolerances) errors that ultimately limit the DFT dynamic range.

---

<table>
<thead>
<tr>
<th>Type of Transient Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double precision error</td>
<td>15 significant digits or -300dB</td>
</tr>
<tr>
<td>Matrix solver error</td>
<td>limited by the matrix condition number; depends on matrix reordering and pivoting</td>
</tr>
<tr>
<td>Iteration error</td>
<td>depends on the number of Newton iterations; controlled by Newton convergence tolerances</td>
</tr>
<tr>
<td>Numerical integration error</td>
<td>depends on integration method, proportional to the size of the time step, and controlled by LTE tolerance and Newton convergence</td>
</tr>
<tr>
<td>Global time integration error</td>
<td>depending on the circuit functionality and integration method, local error can cancel or accumulate over the transient simulation time interval</td>
</tr>
</tbody>
</table>

Table 1. Source of Transient Simulation Errors

Sources of Error in a Discrete Fourier Transform

The DFT sources of error are listed below.

<table>
<thead>
<tr>
<th>Type of DFT Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation error</td>
<td>DFT requires a certain number of uniformly spaced samples, while the simulator takes non-uniform time steps. This error can be resolved by forcing simulator timepoints at the required time for the DFT.</td>
</tr>
<tr>
<td>End point error</td>
<td>DFT implies periodicity in the time domain. If the signal is not periodic or does not have an integer number of cycles present, discontinuities will be present at the end points.</td>
</tr>
</tbody>
</table>

Table 2. Source of DFT Errors

While both transient errors and DFT errors are present in the DFT results, this application note will primarily be discussing just the two sources in Table2.

First DFT Error Source – Interpolation Error

The first source of error present in the transient analysis is the interpolation error. The result from a transient analysis may appear to be a continuous signal, but, in reality, it is a sampled signal. By default, the samples are automatically controlled by the transient analysis unless the user controls the time points in the transient analysis by using the Time Step options or the Output option parameters.
The **strobeperiod** parameter located under the **Output** option tab of the transient options will be used in this application note. The **strobeperiod** parameter forces a time point at an interval determined it. This allows regular points to be guaranteed in the simulation results to be available for post processing. These regular time points are necessary for DFT to yield accurate results.

### Example of Not Using Transient Strobeperiod

In **Figure 4**, a DFT with automatically determined times steps is shown to have < 60dB dynamic range. Due to the aforementioned errors in **Table 1** and **Table 2**, the dynamic range of the DFT is limited.

![Figure 4](image)

**Figure 4.** DFT resulting from irregular time points with Rectangular window of 1MHz sine wave. Note <60dB dynamic range.
Example of Using Transient Strobeperiod

In Figure 5, a sine wave with regular time steps defined by strobeperiod taken during simulation with a 1MHz sine wave from an ideal voltage source is shown. In Figure 6, the sine wave used to generate the DFT in Figure 5 can be seen. Note that the regular time steps with exactly 31.25ns spacing was determined by the ADE-L state variables. This value for strobeperiod placed the correct number of time points for the DFT such that no interpolation between time points had to be performed during the DFT through the entire period of the sine wave. This application note will discuss how compute the strobeperiod in a later section.

![Figure 5. DFT resulting from regular time points with Rectangular window of 1MHz sine wave. Note 280dB dynamic range.](image)
The difference between Figure 4 and Figure 5 is a result of consistent sampling of data points with equal spacing over the period of the signal. Using this technique, you are effectively causing the DFT to be accurate by forcing a time step where the DFT requires it to be. Therefore, there will be no time point interpolation and the time point assumption for the discrete form of the Fourier Transform analysis is satisfied.³ It should be noted that, in practice, with realistic circuits, -300dB dynamic range is not achievable – this is only limited by the double precision error as defined in Table 1. This will be discussed in a later section to show what can be expected with realistic circuits.

³ Cadence Calculator DFT help page – see section discussing Radix-2 Cooley-Tukey
Second DFT Error Source – End Point Error

End Point Error includes non-integer number of cycles and transient simulation errors. Often, the dominant source of error for a DFT is a non-integer number of cycles while transient simulation errors are usually relatively small. To understand the processes taking place during a DFT, let us examine a sine wave that is sampled and converted to the frequency domain by a DFT with a rectangular window.

The purpose of windowing is to reduce the discontinuities at the windowing edges that are caused by non-integral number of periods of a signal (or signals) in a window for a rectangular window. See Figure 7. In simulation, this is usually used for oscillators or other circuits where the frequency is not precisely known.

A rectangular window needs to capture at least one complete cycle of the lowest frequency signal of interest. Note that higher frequencies signals may be present in this window. For example, there may be a 1 MHz periodicity when 1 MHz and 10 MHz signals are applied to the system. The assumption made in the DFT is that the signal in the sample window is periodic. This means that the signal must be at steady state and is not random or chaotic in nature.

When a DFT is calculated using a rectangular window, it is equivalent to multiplying the signal, in our case a sine wave times a square pulse in the time domain. Doing so yields a sinc() response in the frequency domain. When a sine wave is multiplied by the window in the time domain, the result is a tone at $1/T$, where $T$ is the period of the sine wave. The amplitude in the frequency domain is determined by $A \cdot trbw$. See Figure 8. $trbw$ is the period of the resolution bandwidth in the frequency domain. This is the lowest frequency that can be resolved by DFT.

![Figure 7. Time Window placed around an integer number of cycles.](image)

---

Figure 8. Effect of a rectangular window in time domain when transformed into the frequency domain. Note this is only shown in positive frequency.

Figure 7 shows that the window abruptly starts and ends by clipping the signal around the specified points, tstart and tend. The window is equivalent to the rect() function (see http://en.wikipedia.org/wiki/Rect for more information). The lowest frequency that can be resolved by the DFT is the frequency 1/trbw or frbw. frbw is called the resolution bandwidth frequency. This, for example, would be encountered when looking at a signal on a spectrum analyzer in the laboratory. This terminology will make the meaning consistent with laboratory measurements with a spectrum analyzer.

Understanding the Frequency Placement of the Tones

Now, to quickly establish the basic effects of the rectangular window on the sine wave, simulation of a simple circuit is done with a 10Hz sine wave when there is an integer number of cycles included in the time window trbw. In Figure 9 below, a simple single source test bench has been created to show the effects of non-integer number of cycles in trbw.
First, a simple simulation with a 10 Hz signal with a 1Hz resolution bandwidth is shown. In Figure 10, a single tone can be seen from the DFT with an amplitude of 0 dBV (1 Vp) in the 10Hz frequency bin with all other frequency bins below -300dB. Note that there are no tones except at 1Hz intervals. Therefore, if there are frequencies smaller or greater than 1 Hz, they are forced to ‘fit’ in a 1Hz wide bin.
Figure 10. 10 Hz sine wave simulated for 1 sec then applied a rectangular DFT in dB.
Figure 11. Simulator Sampling Process for Integer Cycles in trbw. Note, only showing 6 samples to make the drawing legible. The number of samples should always be a $2^N$ points.
Figure 12. Sampling of sine wave with a rectangular window applied in the DFT. Note that all other frequencies (red dots) except for 10 Hz is multiplied by a very small number. Note also that not all samples of tsample are shown in the time domain plot. Most important for this example is the first and last time samples.
What is happening? In Figure 11, a simple example of the sampling process can be seen. Once the rectangular window is defined in the DFT, by creating a \text{sinc()} response in the frequency domain, all sampled frequencies from every frequency, other than 10 Hz, are multiplied by a small number. See Figure 12. \text{sinc()} is plotted for clarity in these results. This was documented by Ramirez\textsuperscript{5}.

**Understanding the Amplitude of the Frequency Bins**

Next, a simulation of a sine wave is performed with a non-integer number of cycles in \text{trbw=1} second. In Figure 13, both the 10 Hz signal and the 10.25 Hz signal with a 1 Hz window applied are shown. Note that only the 10.25 signal was included in this simulation. The 10 Hz plot is shown only for reference. Obviously, there is a large end point error and one would expect error in the results. Let us examine how the \text{sinc()} function will show the details how the DFT will respond to the end point error. Unlike the smoothing windows, the \text{rect()} window has less tolerance to end point error.

In Figure 13, the DFT is now shown with lines connecting the spectral points to make the difference more obvious. Note that the 10.25Hz signal has frequencies that are not really in the signal. The signal source is generating only a 10.25 sine wave. However, there are no 8, 9, 11, or 12 Hz signals present in the source – hence, they are artifacts of the DFT process when the window does not have an integer number of cycles (that is, end point error) present in the resolution bandwidth.

Next, in Figure 14, an example of how the sampling of the sine wave has an end point error and how the \text{sinc()} function is shifted by 0.25 Hz through the resulting DFT is shown. This is the key in understanding how the tones are generated in the other frequency bins. Figure 15 shows the resulting plot of the DFT of the 10.25 Hz sine wave that matches the \text{sinc()} function shifted by 0.25 Hz. This results in mapping the frequency bins to their corresponding amplitudes. Note that:

- 9 Hz bin is now ~15dB lower than the 10Hz bin
- 11 Hz bin is ~ 11 dB lower than the 10 Hz bin
- All other tones follow the amplitude of the \text{sinc()} function as well. Compare this in Figure 15 and Table 3.

In Figure 15, it can be seen that the 10 Hz signal energy is now lower than 0 dBV by 0.11dB, even though the 10.25 Hz sine wave has an amplitude of 1V. The energy was transferred from the 10 Hz frequency bin to the other frequency bins – it is not lost.

Let us use Parseval’s Theorem to prove that the results are consistent with theory. First, if you integrate just the 10.25 Hz tone in Figure 13 and compare the result with the same

integration bandwidth of Figure 10, the results will be unequal. Where did the other 10 Hz energy go? This can be found by integrating Figure 13 from 0 Hz up to $F_{Nyquist}$ (which is not shown in this figure for clarity). You will see that they have almost exactly the same value (only missing a small percentage) as if integrating just the 10 Hz tone in Figure 10.

According to Parseval’s Theorem:

$$\int_{-\infty}^{\infty} f(t)^2 \, dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\omega)|^2 \, d\omega \quad \text{equation (1)}$$

This can be calculated in the VIVA calculator by performing the following calculation:

$$\text{vin}_{\text{rms}}^2 = \frac{1}{2f_{\text{rbw}}} \int_{0}^{F_{\text{Nyquist}}} \text{abs}(\text{DFT}(\text{VT(vin)})^2 \, df \quad \text{equation (2)}$$

Since $\text{vin} = 1 \text{Vpeak}$, $\text{rms}(\text{vin}) = 0.707 \text{ Vrms}$. Parseval’s theorem is in terms of power, so $\text{vin}_{\text{rms}}^2 = 0.5 V^2$. When equation (2) above is applied to Figure 10 (or the red curve in Figure 13), the result in the VIVA calculator is 0.5 $V^2$ for both the time domain by calculating $\text{rms(vin)}^2$ in ViVA and when integrating all frequencies as defined in equation (2). The ViVA calculator function used to calculate the integrated energy is:

$$\text{integ(abs(dft(VT("/vin") VAR("tstart") VAR("tend") VAR("fft_size") "Rectangular" 1 "default")))^2 0 F_{\text{Nyquist}} / (2*frbw) \quad \text{equation (3)}$$

This proves Parseval’s theorem in both the time and frequency domains in the simulation.

If the same integration is done in the frequency domain for Figure 10, you will get the same answer of 0.5 $V^2$. Now, to show that all of the energy is in the 10Hz bin in Figure 10, the limits of integration are changed. Since all the contributions to the total energy from the other bins (that is, 8, 9, 11, 12 Hz and so on.) are so small, changing the limits of integration to only include the 10 Hz bin will result in 0.5 $V^2$. Therefore, the results do not change because DFT correctly placed all the energy in the 10 Hz bin. To prove this, you need to integrate from the 9Hz bin to the 11Hz bin to get all the energy because, half of the energy is in the 9 Hz to 10 Hz and 10 Hz to 11 Hz frequency bins. The ViVA integration equation will look like the following:

$$\text{integ(abs(dft(VT("/vin") VAR("tstart") VAR("tend") VAR("fft_size") "Rectangular" 1 "default")))^2 9 11 / (2*frbw) \quad \text{equation (4)}$$
Figure 13. 10.25 Hz sine wave (Pink curves) simulated for 1 sec the applied rectangular DFT in dB. Note that 10Hz sine wave still present for reference.
Figure 14. 10.25Hz signal with 1 Hz window applied and with tones now appearing at all frequency bins (red dots). The sinc() is now centered at 10.25 Hz. Note also that not all samples of tsample are shown in the time domain plot. Most import for these examples are the first and last time samples.
Figure 15. Shows that the 10.25 Hz tone is now 1.18 dB lower than the tone centered at 10 Hz. Note also the 7, 8, 9, 11, and 12 Hz bins compared to 10 Hz. This shows the shaping of the sinc() function now that the sinc() function is misaligned from the integer number of sine waves.
Table 3. DFT and Sinc() Response comparison Table

<table>
<thead>
<tr>
<th>Frequency (Hz)</th>
<th>DFT Results (dB)</th>
<th>Sinc() Response (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>-23.5</td>
<td>-23</td>
</tr>
<tr>
<td>8</td>
<td>-21</td>
<td>-21</td>
</tr>
<tr>
<td>9</td>
<td>-18</td>
<td>-18</td>
</tr>
<tr>
<td>10</td>
<td>-0.11</td>
<td>-0.1</td>
</tr>
<tr>
<td>11</td>
<td>-11</td>
<td>-11</td>
</tr>
<tr>
<td>12</td>
<td>-18</td>
<td>-18</td>
</tr>
</tbody>
</table>

If equation (2) is applied to the 10.25 Hz signal, you will get $0.5 V^2$ in the time domain and if equation (3) is applied to the frequency domain result, you will get $0.499 V^2$. If equation (4) is used over the range of 9-11 Hz, as in the previous example, the result will be $0.435 V^2$. This is -1.2 dB lower than 1V sine wave – which is very close to the -1.18 dB shown in the ViVA plot of the simulation for the 10.25 Hz sine wave.

If 0.25 Hz is now added to the input frequency so that the frequency becomes 10.5 Hz and the simulation is rerun, almost half the energy will move equally in the 10 and 11 Hz bins showing that the sinc() function is centered at 10.5 Hz. In Figure 16, an extra ½ cycle is plotted (turquoise trace) in addition to the 10 Hz sine wave (red trace). In the frequency domain plot of Figure 16, the energy has moved from 10 - 10.25 to the 10.5 Hz DFT traces. In this example, bin 10 and 11 would be equally spaced around the peak of the sinc() center lob centered at 10.5 Hz where the tones would be having approximately equal amplitude.
Figure 16. 10.5 Hz sine wave (turquoise curves) simulated for 1 sec the applied rectangular DFT in dB. Note that 10Hz sine wave still present for reference.
Figure 17. 10.75 Hz sine wave (green curves) simulated for 1 sec the applied rectangular DFT in dB. Note that 10Hz sine wave still present for reference.

Next, consider 10.75 Hz, which shows the same result as 10.25 Hz, but it is shifted mostly to the 11 Hz bin.
Finally, with the signal at 11 Hz, you can see that all the energy moves to the 11 Hz bin. Why is 11 Hz completely contained in one frequency bin and not spread as in the 10.25, 10.5, and 10.75 Hz examples? This is because there are an integer number of tones – no end point error! See the time domain plot (blue curve in Figure 18). After observing that 11 integer cycles now moved the energy completely to the 11Hz bin, you can image the sinc() function, as shown in Figure 14, continuing to slide across all the frequencies and the dots tracing out the amplitudes of the corresponding frequency bins for 0-20Hz.

When an integer number of cycles is present, the nulls of the sinc() function align over the integer in multiples of frbw (=1Hz). Because of this, the total energy of the signal is placed at the frequency of the sine wave contained within the window. All the other bins are left with ~0 energy as is the case with the 10 Hz example in Figure 10 and the 11 Hz example in Figure 18. It is important to understand this concept. If an integer number of cycles is not contained within the window, the amplitude of not only the tone of interest is changed but also all frequencies up to F_{Nyquist} are altered by the sinc() function. This will generate errors in the resulting calculations. If you cannot place an integer number of cycles, other windows may be required to suppress the side lobes low enough to meet the dynamic range requirements. The drawback to this is that the tones will have to be spread further apart in frequency, and the amplitudes will need to have the correct offsets applied to measure the correct amplitudes. This will change depending on the window chosen. See table 7-1 on pages 140 and 141 in [5].

Figure 18. 11 Hz sine wave (blue curves) simulated for 1 sec the applied rectangular DFT in dB. Note that 10Hz sine wave still present for reference.
Multiple Tone DFTs

In this section, we will add tones to show how this concept can be extended to many tones. The first consideration to add more tones is that all the tones must be an integer multiple of the lowest frequency tone. In the previous example, the lowest frequency is 1 Hz. A tone will be placed at 2 Hz intervals to show the capability of this technique. In Figure 19, the time and frequency plot can be seen showing tones spaced at every 2 Hz.
Figure 19. Simulation of tones spaced every 2 Hz from 2 Hz to 18 Hz. Note dynamic range is maintained.

An additional benefit of this capability is the ability to resolve a large amplitude dynamic range in the frequency domain. In Figure 20, you can compare the 1V (2, 6, 10, 14, and 18Hz tones) signals relative to the 1uV (4, 8, 12, and 16 Hz tones) signals with this rectangular window. That is, 120 dB dynamic range, which is achievable for real circuits.

Figure 20. Multi-tone with 120dB dynamic range. That is 1V vs. 1uV!
Setting up the Simulation and Plotting the Resulting DFT

The setup is simple and can be seen in Figure 21. There are a number of variables that are needed. Two variables, $t_{\text{end}}$ and $t_{\text{strobo}}$ are used in the transient analysis statement. Three parameters: $t_{\text{start}}$, $t_{\text{end}}$, and $f_{\text{ft size}}$ are used in the DFT equation during post processing. All other options are for intermediate calculations that can be used for clarity of the simulation results.

![Figure 21. ADE setup necessary to run simulation correctly.](image)
Following are the definitions of the equations in Figure 21:

- $F_1$ => Fundamental Frequency – can be $frbw \to \text{USER INPUT}$
- $F_2$ => Optional 2nd Frequency – must be an integer multiple of $frbw \to \text{USER INPUT}$
- $OSR$ => Over Sampling Ratio \to \text{USER INPUT}
- $N$ => Order of divisor to the power of $2^N$ to determine $frbw \to \text{USER INPUT}$
- $frbw$ => resolution bandwidth frequency
- $F_{Nyquist}$ => Nyquist frequency of the simulation
- $fsample$ => Simulation sampling frequency
- $T_{steadystate}$ => Time to allow circuit to settle – must be a multiple of $\frac{1}{frbw}$ (or $t_{rbw}$)
- $t_{start}$ => Starting time of rectangular window used in DFT
- $t_{end}$ => Ending time of rectangular window used in DFT and transient analysis
- $t_{strobe}$ => Strobe period that is entered into transient analysis options \to output
  \to strobeperiod options
- $fft_{size}$ => Size of DFT to be plotted

**ViVA equation to plot DFT in dBV is:**

```
dB20(dft(VT("/vin") VAR("tstart") VAR("tend") VAR("fft_size") "Rectangular" 1 "default"))   equation (5)
```

**Real Circuit Example**

In this section, a real transistor circuit is used and shown in Figure 22. This will show what is achievable with this technique. The circuit is a cascade of three simple common source amplifiers. A fundamental frequency of 100MHz for tone F1 and a second tone F2, at 101.172MHz are provided as the input frequencies. This frequency selection meets the criteria of an integer number of cycles in the resolution band width. F2 is automatically calculated in the previous ADE-L state to be exactly three resolution bandwidths away from F1. Since F1 is used to calculate $frbw$, this ensures that there is an integer number of cycles for F1 and F2 (see the setup equation $frbw$ in the ADE state in the previous section). After proper biasing is achieved, you will notice that 1 Volt tones are not possible. The signal must be much smaller than 1V to prevent compression and saturation of the multiple stages of the amplifiers. In this example, the input amplitude is 100uV (or -80dBV).
First, in Figure 23, the transient results can be seen for each node of the amplifier. Only the input and output nodes will be analyzed. The drain voltages of stage one and two ($V_d$ and $V_{d2}$) are shown to display the DC levels.
Figure 23. Transient results of the cascaded amplifier. Notice the large signal swing differences as the signal goes from Vin to Vd_tran to Vd2_tran and finally at Vout_tran responses getting larger at each node.

In Figure 24, the dynamic range of the input signal can be seen in the frequency domain. Figure 25 and Figure 26 show what can be achieved with this technique for the vin and vout nodes of this amplifier. In this example, 120 dB of dynamic range is achieved with an amplifier with >50 dB of gain from input to output. Note that the dynamic range of the output compared to the input spectrum has reduced significantly. Now, the sources of error as defined in Table 1 are present and limiting the dynamic range of the DFT with the rectangular window.
Figure 24. Input signal to amplifier. Notice dynamic range and two tones at F1=100MHz at -80dBV and F2=101.1719MHz at -80dBV.
Figure 25. Output Spectrum with input included to show gain and other harmonics generated by the amplifier not present in the input spectrum. Input spectrum is the red trace and the output spectrum is the yellow trace.
Figure 26. Output signal with intermodulation tones located very close to F1 and F2 with > 120dB dynamic range.

Increasing Real Circuit DFT Dynamic Range

The rectangular window is good but not a panacea. If you absolutely require the dynamic range, a window other than rectangular must be applied. In the transistor circuit example from Figure 22, you may obtain an extra 100dB dynamic range by longer simulation and a different window. Effectively, using the smoothing window, such as Cosine2 or Hanning, reduces the abrupt effects of the rectangular window by smoothly transitioning the end points to zero. This gains dynamic range at the expense of frequency resolution, energy integration...
error, and increased simulation time. In Figure 27, the same circuit of Figure 22 is plotted with a Cosine2 window.

Figure 27. Full spectrum plot of 3 stage transistor amplifier plotted with both a Rectangular window and a Cosine2 window.

The Cosine2 window has great dynamic range but the frequency resolution is limited. In Figure 28, the frequency resolution limitations are seen. Note that the red curve (the rectangular window) resolved lower frequency separation between tones at N-1 and N-2 resolution bandwidths and the amplitude error for the Cosine2 window is 6 dB for the fundamental tone at 100MHz.

Figure 28. 100MHz region with 1 cycle of the fundamental signal that makes the frbw.
If you simulate two cycles, the frequency cannot be resolved yet and the amplitudes are uncertain. See Figure 29.

Simulating three cycles gets closer but it is still unclear which tones are from the signal and which tones are from the window. See Figure 30 and Figure 31.

After simulating four cycles, you can now discern the signals correctly. In Figure 32, it can be seen that now there is separation between the tones of the signal and the intermodulation tones. Note that the amplitudes are now accurate. In Figure 33, you can see the benefit of the Cosine2 window.

---

**Figure 29.** 100MHz region with 2 cycles of the fundamental signal.
Figure 30. 100MHz 3 cycles of the fundamental signal.

Figure 31. 100MHz region with 3 cycles of the fundamental signal with Cosine2 highlighted.
Figure 32. 100MHz region with 4 cycles of the fundamental signal.

Figure 33. 100MHz region with 4 cycles of the fundamental with Cosine2 highlighted.
Summary

In this application note, a description of errors in the transient analysis and DFT was provided along with a description of how these errors impact DFT analysis. A simple method for setting up the equations to calculate the parameters for the DFT and the transient analysis was also shown resulting in a simple ADE-L state for you to implement which quickly sets up the simulation with as few as three input parameters needed. In Table 4 below, you will find the performance summary between the rectangular window and smoothing windows in general.

<table>
<thead>
<tr>
<th>Types of Signals</th>
<th>Rectangular DFT Window</th>
<th>Smoothing DFT Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity to End Point Error</td>
<td>High</td>
<td>Low – forces end points to zero</td>
</tr>
<tr>
<td>Sensitivity to Interpolation Error</td>
<td>Low - with (2^N) samples for the Radix-2 Cooley-Tukey Algorithm in the calculator</td>
<td>Low - with (2^N) samples for the Radix-2 Cooley-Tukey Algorithm in the calculator</td>
</tr>
<tr>
<td>Dynamic Range</td>
<td>Good</td>
<td>Best</td>
</tr>
<tr>
<td>Frequency Resolution</td>
<td>Best</td>
<td>Poor</td>
</tr>
<tr>
<td>Total Integrated Energy Accuracy</td>
<td>Best</td>
<td>Poor - Requires offset to be determined</td>
</tr>
<tr>
<td>Simulation Time Requirements</td>
<td>Shortest – 1 cycle of (frbw)</td>
<td>Longest – minimum of 4 cycles of (frbw) to resolve equivalent frequencies</td>
</tr>
</tbody>
</table>

Table 4. Performance Summary between the rectangular window and smoothing windows in general.

References


3 Cadence Calculator DFT help page – see section discussing Radix-2 Cooley-Tukey
