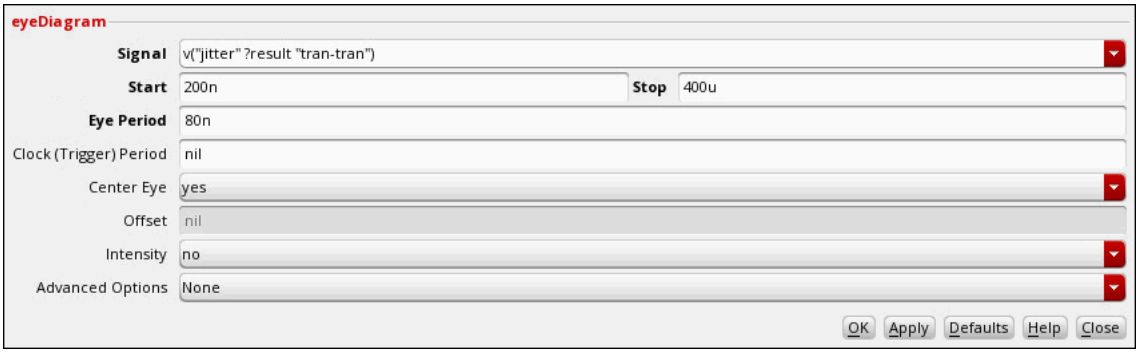
eyeDiagram

Gives an eye-diagram plot in which the waveform signal is divided into fixed time periods, which are then superimposed on each other. The result is a plot that has many overlapping lines enclosing an empty space known as the eye. The quality of the receiver circuit is characterized by the dimension of the eye. An open eye means that the detector can distinguish between `1's` and `0's` in its input, while a closed eye means that a detector placed on `Vout` is likely to give errors for certain input bit sequences.

This function is available only in the SKILL mode.



This function includes the following arguments:

- *Signal*—Name of the signal to measure.
- *Start*—X-axis value from where the eye-diagram plot is to begin.
- *Stop*—X-axis value where the eye-diagram plot is to end.
- *Eye Period*—Time period for the eye diagram.
- *Clock (Trigger) Period*—Time interval between triggering events.

  > If the trigger period is not specified, the eye period is used as the trigger period.

- *Center Eye*—Specifies whether to place the eye diagram in the center of the graph, which means the center of the eye opening is placed at the half of the eye period. Valid values: `yes` or `no`. If you select `yes`, the eye diagram is centered based on the crossing times across the average signal value between the start and the stop time.
- *Offset*—Specifies the offset value that is used to shift the eye. If you specify a positive value, the eye-right is shifted to the right according to the specified value. If you specify a negative value, the eye-left is shifted to left. This field becomes inactive if you select `no` from the *Center Eye* option.
- *Intensity*—Specifies whether to highlight the intersection points in the eye diagram Valid values: `yes` or `no`. If you select `no`, the eye digram is plotted with the default color scheme.
- *Advanced Options*—Specifies whether the vertical (`Max Vertical Opening`) or horizontal opening (`Max Horizontal Opening`) of the eye is to be calculated.

Additional Information

Calculating Horizontal and Vertical Eye Width:

The waveform is folded on the X-axis between the start time (`n_start`) and stop time (`n_stop`) by the length `n_period`.

The function performs the following steps to calculate the horizontal eye opening:

- Calculates all the X points are where the folded waveform intersects the line `y = yMid`
  where, `yMid = ymax(o_waveform) + min(o_waveform))/2`
- From these calculated X points, returns the two consecutive X points, which have the maximum distance between them, as the horizontal eye opening.
  `X[k]` and `X[k-1]` for which the (`X[k]-X[k-1]`) value is maximum.

The function performs the following steps to calculate vertical eye width:

- Calculates the horizontal eye width to find the consecutive X points, `X[k]` and `X[k-1]` having maximum distance between them.
- Calculates all Y points where the folded waveform intersects the following line:
  `x = (X[k]-X[k-1])/2`
- From these calculated Y points, returns two consecutive Y points, which have the maximum distance between them, as the vertical eye opening.
  `Y[k]` and `Y[k-1]` for which (`Y[k]-Y[k-1]`) is maximum.

Assumptions

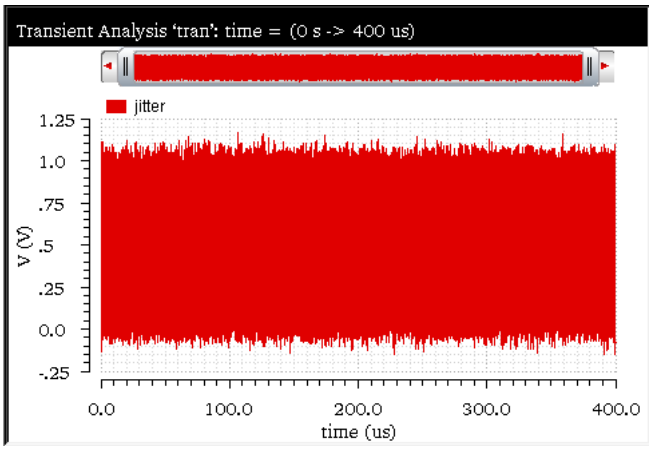
The following assumptions have been made while calculating the advance option values:

- The opening of an eye approximately lies on following:
  `(ymax(o_waveform)+ymin(o_waveform))/2`
- Only one eye opening exists in the area mentioned above in which the waveform is folded.

For more information about eye diagram, see the Eye Diagram assistant.

Example

This example shows the eye diagram plot generated when you apply the `eyeDiagram` function on the following input signal:



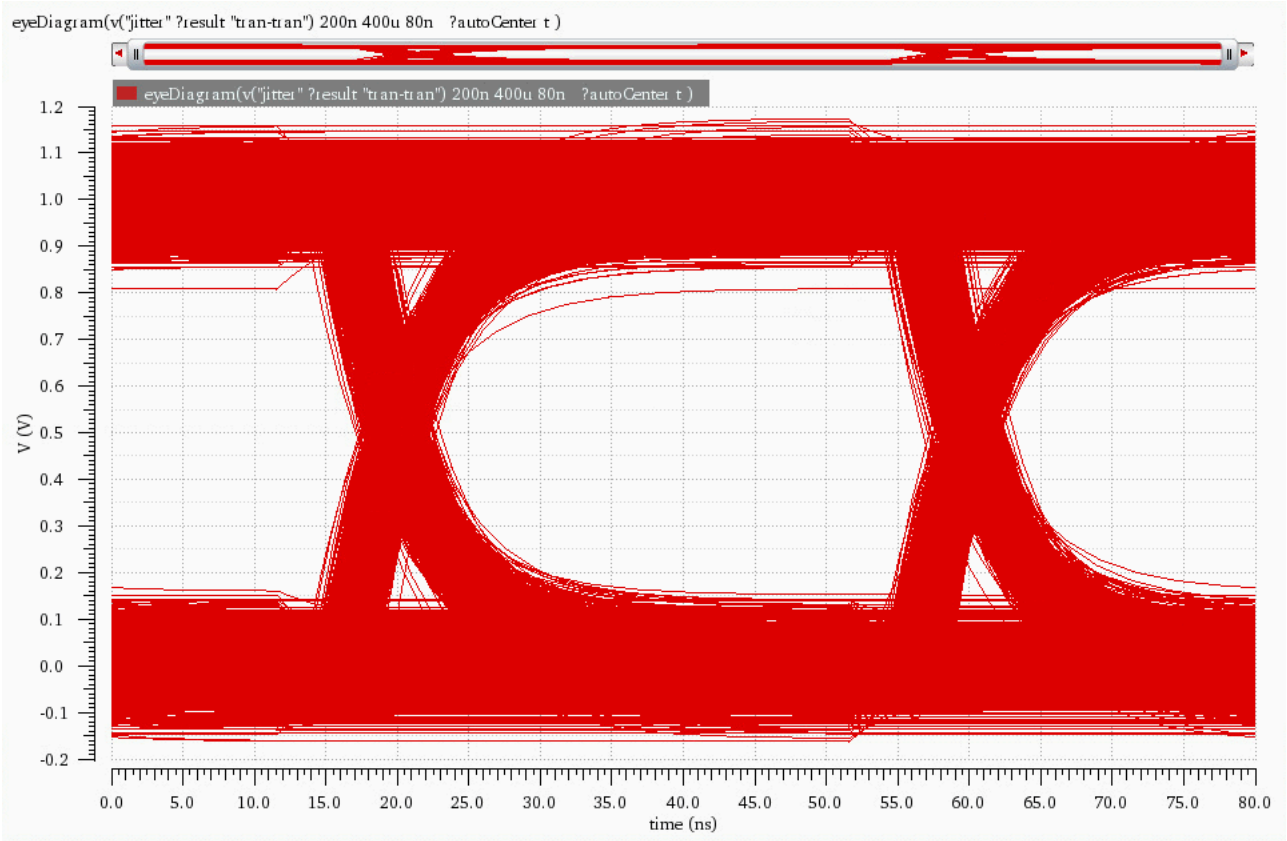
The following arguments are specified in this example:

- *Signal*—`v("jitter" ?result "tran-tran")`
- *Start*—`200n`
- *Stop*—`400u`
- *Eye Period*—`80n`
- *Clock (Trigger) Period*—`nil`
- Center Eye—`yes`
- Offset—`nil`
- Intensity—`no`
- *Advanced Options*—`None`

The following expression is created in the Buffer:

```
eyeDiagram(v("jitter" ?result "tran-tran") 200n 400u 80n   )
```

Now, when you evaluate this expression, the following eye diagram plot is displayed in the graph window:



Related OCEAN Function

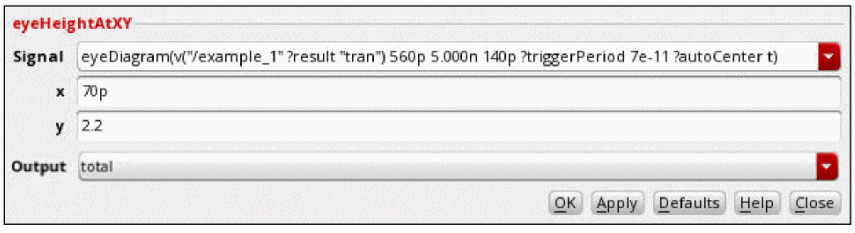
The equivalent OCEAN command for `eyeDiagram` is:

```
eyeDiagram ( o_waveform n_start n_stop n_period ?advOptions t_advOptions )
     => o_waveform/nil
```

eyeHeightAtXY

Calculates the height of an eye at the specified point (x,y) inside the eye diagram.

This function is available only in the SKILL mode.



This function includes the following arguments:

- *Signal*—The eye diagram waveform that is used to calculate the eye height.
- *x*—The x-axis value that is used to calculate the eye height.
- *y*—The y-axis value that is used to calculate the eye height.

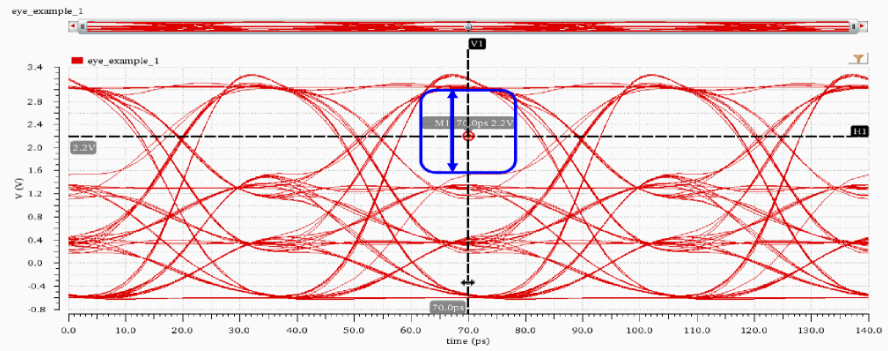
  > The specified coordinates (x,y) must lie within the open eye whose height you want to calculate.

- *Output*—Specifies whether you want to calculate the absolute eye height or the eye height relative to the specified y-axis value. Possible values are:
  - *total*: (Default) Calculates the total eye height.
  - *above*: Calculates the eye height that is above the specified y-axis value.
  - *below*: Calculates the eye height that is below the specified y-axis value.

> You can watch a video demonstration on how to use this function at Enhancements in eyeHeightAtXY and eyeWidthAtXY Functions. Also, read the related blog at Virtuoso Video Diary: Do More With eyeHeightAtXY and eyeWidthAtXY Calculator Functions in Virtuoso Visualization and Analysis XL.

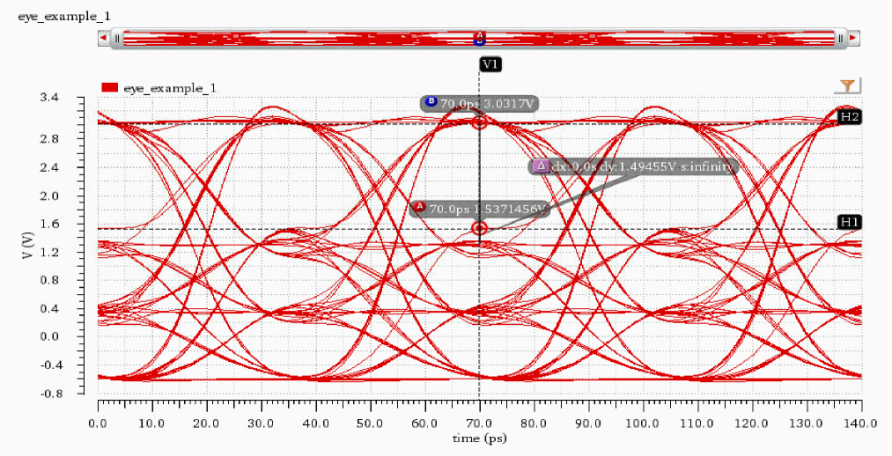
Example 1

This example shows the height calculation of the highlighted eye at point `M1` (`70ps`, `2.2v`). Note that the point, `M1`, lies within the eye whose height is to be calculated.

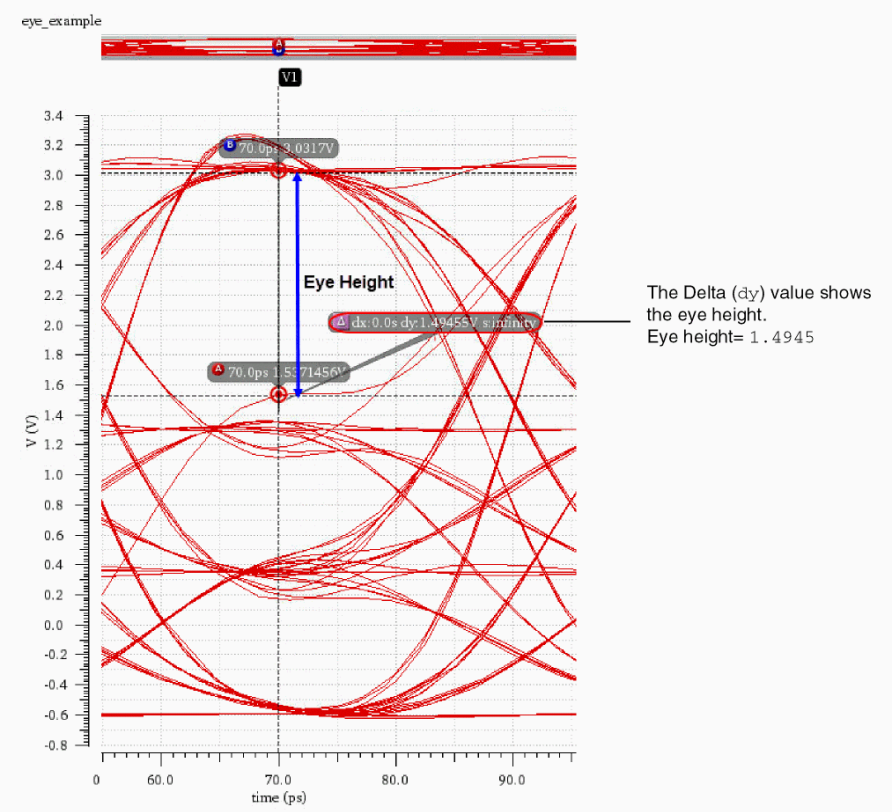


Eye height is the difference of two intercepts made with the innermost traces of the eye in the Y-axis direction. You can calculate the eye height at `70ps` and `2.2v` with the help of an AB marker.

Create a vertical marker at `70ps` and then insert the points, `A` and `B`, where the vertical marker intercepts the innermost traces of the eye. The delta of the AB marker (`dy`) indicates the difference of the Y-axis values of points `B` (`3.0317v`) and `A` (`1.537145v`), which is equal to `1.49455v`. This is the eye height, as shown in the following figure.



The following snippet shows a magnified version of the information displayed in the previous figure.



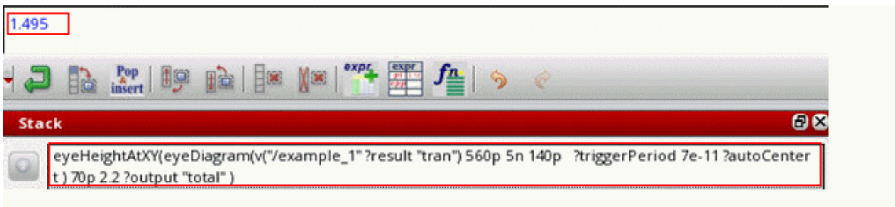
Now, calculate the total eye height using the `eyeHeightAtXY` function. Specify the following arguments:

- *Signal*—`eyeDiagram(v("/example_1" ?result "tran") 560p 5.000n 140p ?triggerPeriod 7e-11 ?autoCenter t)`
- *x*—`70p`
- *y*—`2.2`
- *Output*—`total`

The following expression is created in the Buffer:

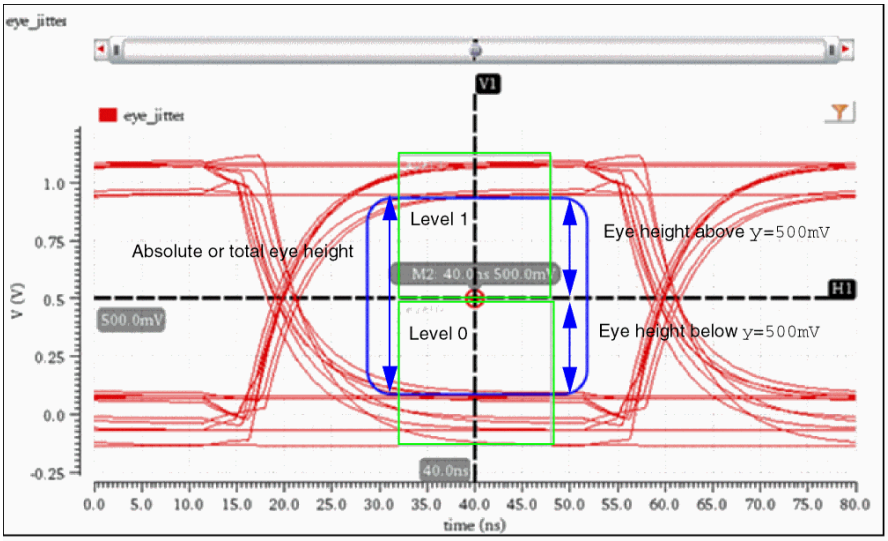
```
eyeHeightAtXY(eyeDiagram(v("/example_1" ?result "tran") 560p 5.000n 140p ?triggerPeriod 7e-11 ?autoCenter t) 70p 2.2 ?output "total" )
```

When you evaluate this expression, the function returns the value of eye height, as shown in the following figure.



Example 2

This example shows the height calculation of the highlighted eye at the point `M2` (`40ns`, `500mV`). Note that the point `M2` lies within the eye whose height is to be calculated.



$$eyeHeightAtXY_{total} = eyeHeightAtXY_{above} + eyeHeightAtXY_{below}$$

$eyeHeightAtXY_{below}$ = Eye height in the level 0 region

$eyeHeightAtXY_{above}$ = Eye height in the level 1 region

Now, we will calculate the total eye height at `M2`, eye height above `y=500mV`, and eye height below `y=500mV` by using the `eyeHeightAtXY` function.

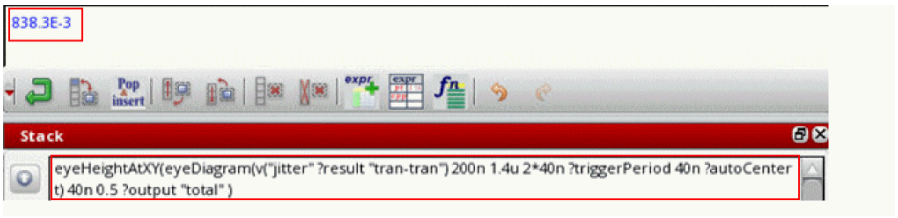
To calculate the total eye height at `M2` (`x=40ns`, `y=500mV`), specify the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`
- *x*—`40n`
- *y*—`0.5`
- *Output*—`total`

The following expression is created in the Buffer:

```
eyeHeightAtXY(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 40n 0.5 ?output
```

When you evaluate this expression, the function returns the value of total eye height, as shown in the following figure.



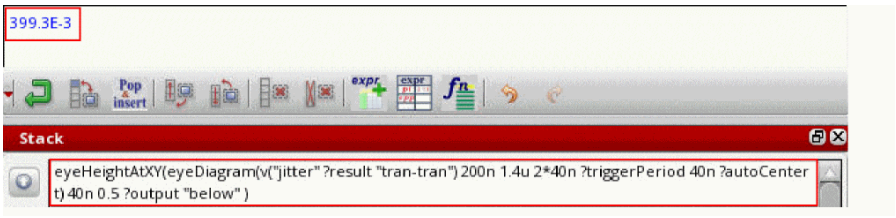
To calculate the eye height at `M2` (`x=40ns`, `y=500mV`) below `y=500mV` (eye height in the region Level 0), specify the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`
- *x*—`40n`
- *y*—`0.5`
- *Output*—`below`

The following expression is created in the Buffer:

```
eyeHeightAtXY(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 40n 0.5 ?output
```

When you evaluate this expression, the function returns the value of eye height that is below `y=0.5V` (`500mV`), as shown in the following figure.



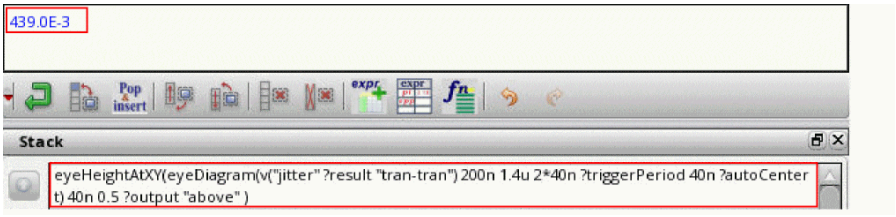
To calculate the eye height at `M2` (`x=40ns`, `y=500mV`) above `y=500mV` (eye height in the region Level 1), specify the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`
- *x*—`40n`
- *y*—`0.5`
- *Output*—`above`

The following expression is created in the Buffer:

```
eyeHeightAtXY(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 40n 0.5 ?output
```

When you evaluate this expression, the function returns the value of eye height that is above `y=0.5V` (`500mV`), as shown in the following figure.



Observe that the total eye height at `M2` is the sum of eye heights in the level 0 and level 1 regions.

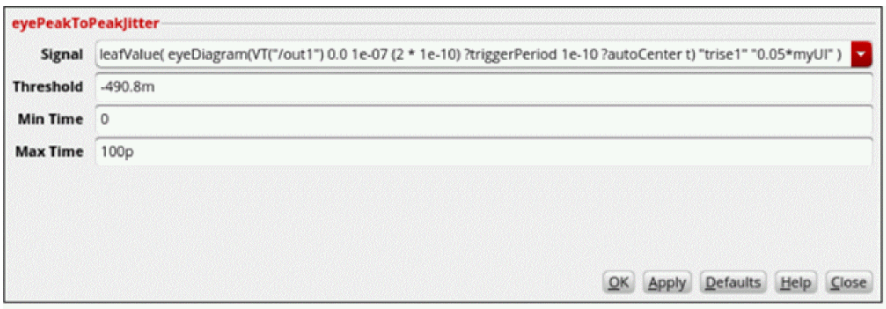
Related OCEAN Function

The equivalent OCEAN command for `eyeHeightAtXY` is:

```
eyeHeightAtXY ( o_eyeDiagram f_x f_y ?output t_output)
     => f_eyeHeight/nil
```

eyePeakToPeakJitter

Calculates the peak-to-peak jitter at the specified threshold within the region of an eye diagram. The peak-to-peak jitter is the time between the first and the last crossing.

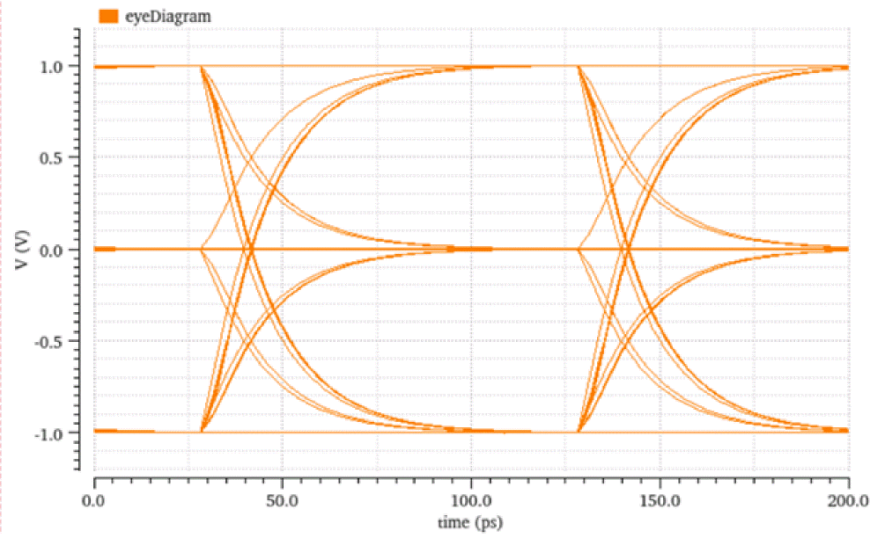


This function includes the following arguments:

- *Signal*—The eye diagram waveform for which the peak-to-peak jitter is to be calculated.
- *Threshold*—Threshold value at which the peak-to-peak jitter is to be calculated.
- *Min Time*—Start time of the crossing.
- *Max Time*—End time of the crossing.

Examples

The following figure shows the input waveform of the eye diagram for which peak-to-peak jitter is to be calculated.



To calculate the peak-to-peak jitter the threshold value `-490.8mV` of this eye diagram using the calculator function `eyePeakToPeakJitter`, specify the following arguments:

- *Signal*: `leafValue( eyeDiagram(VT("/out1") 0.0 1e-07 (2 * 1e-10) ?triggerPeriod 1e-10 ?autoCenter t) "trise1" "0.05*myUI" )`
- *Threshold*: `-490.8m`
- *Min Time*: `0`
- *Max Time*: `100p`

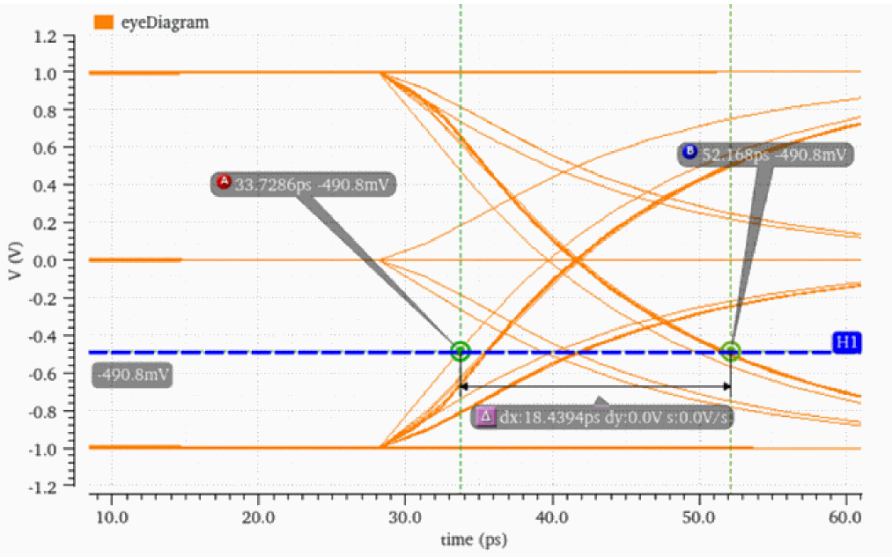
The following expression is created in the Buffer:

```
eyePeakToPeakJitter(leafValue( eyeDiagram(VT("/out1") 0.0 1e-07 (2 * 1e-10) ?triggerPeriod 1e-10 ?autoCenter t) "trise1" "(
```

When you evaluate this expression, the function returns the value of the peak-to-peak jitter at the threshold value `-490.8mV` of the eye diagram. The peak-to-peak jitter is the distance between the first (`0p`) and last crossing (`100p`)at the threshold value `-490.8mV` of the eye diagram.

```
=> 18.44E-12
```

The calculated value of peak-to-peak jitter can be verified by placing the `A` and `B` markers on the first and last crossing at the threshold value of `-490.8mV`. The distance between these two markers is `18.4ps`.



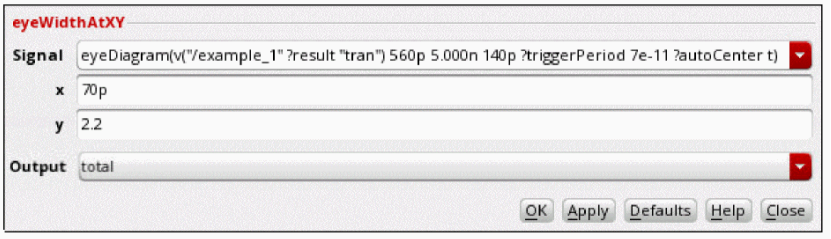
Related Topics

eyePeakToPeakJitter

eyeWidthAtXY

Calculates the width of an eye at the specified point (x,y) inside the eye diagram.

This function is available only in the SKILL mode.



This function includes the following arguments:

- *Signal*—The eye diagram waveform that is used to calculate the eye width.
- *x*—The x-axis value that is used to calculate the eye width.
- *y*—The y-axis value that is used to calculate the eye width.

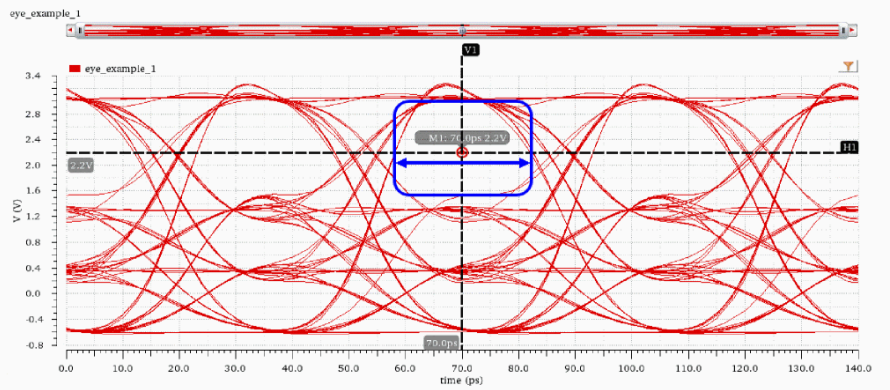
  The specified coordinates (x,y) must lie within the open eye whose width you want to calculate.

- *Output*—Specifies whether you want to calculate the total eye width or the eye width relative to the specified x-axis value. Possible values are:
  - *total*: (Default) Calculates the total eye width.
  - *left*: Calculates the eye width that is left to the specified x-axis value.
  - *right*: Calculates the eye width that is right to the specified x-axis value.

You can watch a video demonstration on how to use this function at Enhancements in eyeHeightAtXY and eyeWidthAtXY Functions. Also, read the related blog at Virtuoso Video Diary: Do More With eyeHeightAtXY and eyeWidthAtXY Calculator Functions in Virtuoso Visualization and Analysis XL.

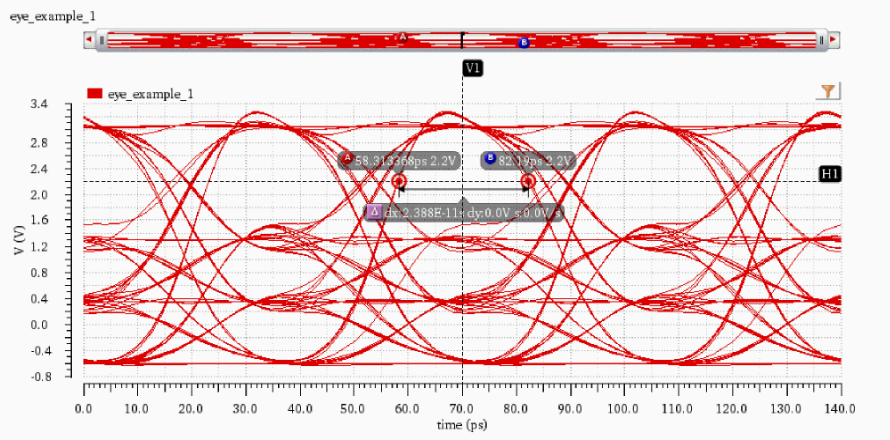
Example 1

This example shows the width calculation of the highlighted eye at point `M1`(`70ps`, `2.2v`). Note that the point `M1` lies within the eye whose width is to be calculated.

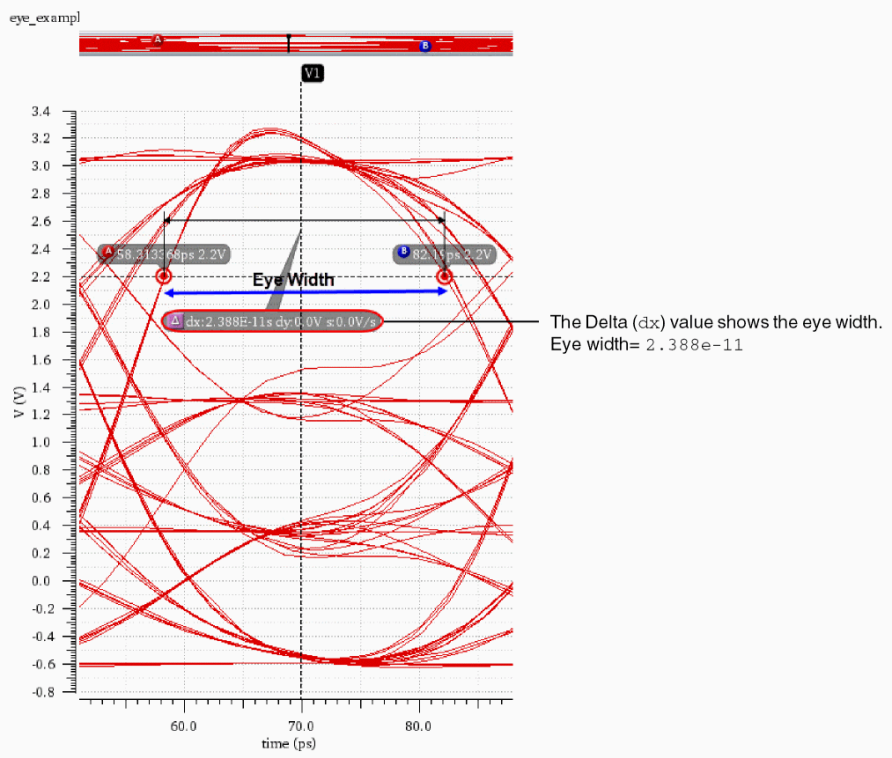


Eye width is the difference of two intercepts made with the innermost traces of the eye in the X-axis direction. You can calculate the eye width at `70ps` and `2.2V` with the help of an AB marker.

Create a horizontal marker at `2.2V` and then insert the points, `A` and `B`, where the horizontal marker intercepts the innermost traces of the eye. The delta of the AB marker (`dx`) indicates the difference of the X-axis values of points `B` (`82.19ps`) and `A` (`58.31ps`), which is equal to `23.88ps` or `2.388e-11` second. This is the eye width, as shown in the following figure.



The following snippet shows a magnified version of information displayed in the previous figure.



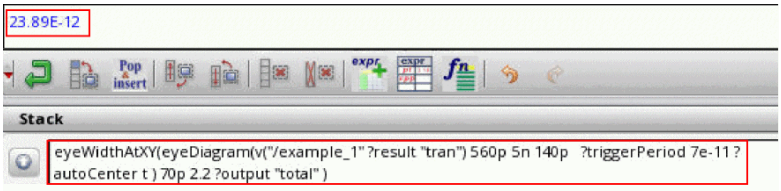
Now, calculate the total eye width using the `eyeWidthAtXY` function. Specify the following arguments:

- *Signal*—`eyeDiagram(v("/example_1" ?result "tran") 560p 5.000n 140p ?triggerPeriod 7e-11 ?autoCenter t)`
- *x*—`70p`
- *y*—`2.2`
- *Output*—`total`

The following expression is created in the Buffer:

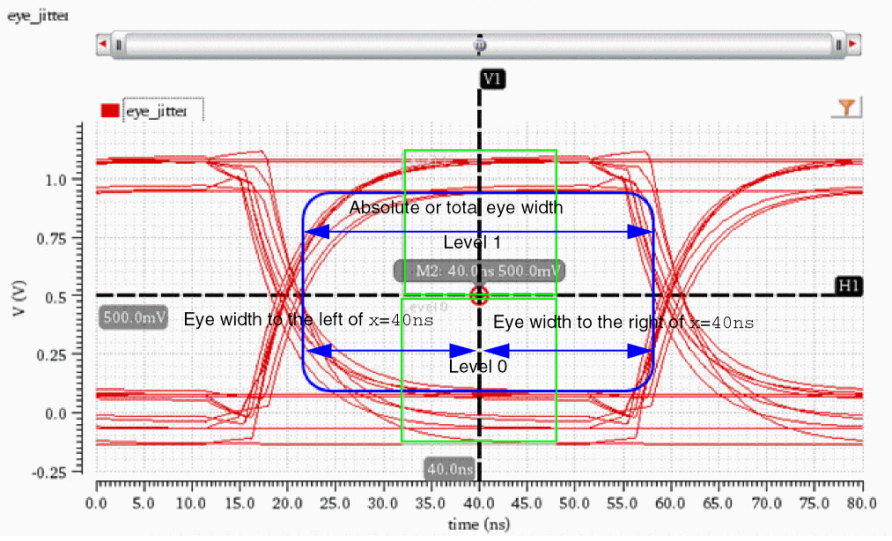
```
eyeWidthAtXY(eyeDiagram(v("/example_1" ?result "tran") 560p 5.000n 140p ?triggerPeriod 7e-11 ?autoCenter t) 70p 2.2 ?output "total")
```

When you evaluate this expression, the function returns the value of eye width, as shown in the following figure.



Example 2

This example shows the width calculation of the highlighted eye at point `M2` (`40ns`, `500mV`). Note that the point, `M2`, lies within the eye whose width is to be calculated.



$$eyeWidthAtXY_{total} = eyeWidthAtXY_{left} + eyeWidthAtXY_{right}$$

Now, calculate the total eye width at `M2`, eye width to the left of `x=40ns`, and eye width to the right of `x=40ns` by using the `eyeWidthAtXY` function.

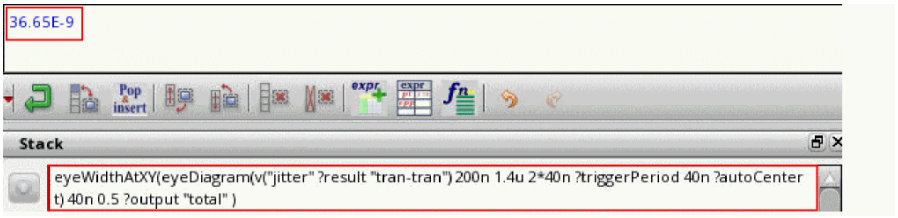
To calculate the total eye width at `M2` (`x=40ns`, `y=500mV`), specify the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`
- *x*—`40n`
- *y*—`0.5`
- *Output*—`total`

The following expression is created in the Buffer:

```
eyeWidthAtXY(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 40n 0.5 ?output '
```

When you evaluate this expression, the function returns the value of total eye width, as shown in the following figure.



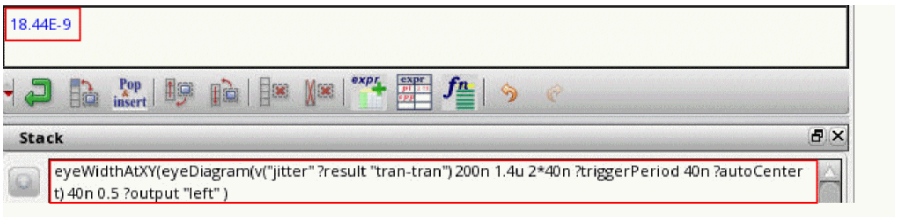
To calculate the eye width at `M2` (`x=40ns`, `y=500mV`) to the left of `x=40ns`, specify the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`
- *x*—`40n`
- *y*—`0.5`
- *Output*—`left`

The following expression is created in the Buffer:

```
eyeWidthAtXY(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 40n 0.5 ?output '
```

When you evaluate this expression, the function returns the value of eye width that is left to `x=40ns`, as shown in the following figure.



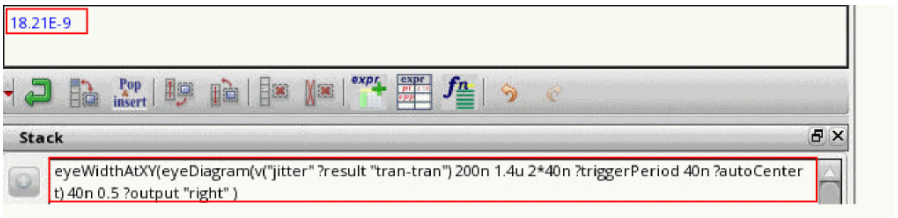
To calculate the eye width at `M2` (`x=40ns`, `y=500mV`) to the right of `x=40ns`, specify the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`
- *x*—`40n`
- *y*—`0.5`
- *Output*—`right`

The following expression is created in the Buffer:

```
eyeWidthAtXY(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 40n 0.5 ?output '
```

When you evaluate this expression, the function returns the value of eye width that is right to `x=40ns`, as shown in the following figure.



Observe that the total eye width at `M2` is the sum of eye widths to the left and to the right of the specified x value.

Related OCEAN Function

The equivalent OCEAN command for `eyeWidthAtXY` is:

```
eyeWidthAtXY ( o_eyeDiagram f_x f_y ?output t_output)
     => f_eyeWidth/nil
```

eyeAperture

Returns the aperture of the input eye diagram signal.

This function includes the following fields:

- *Signal*—Input signal, which is an eye diagram waveform for which the eye aperture is to be calculated.
- *Vref*—Reference voltage value.
- *AC Height*—AC height, which specifies the height of the left side of the aperture window.
- *DC Height*—DC height, which specifies the height of the right side of the aperture window.
- *Display Aperture*—Specifies whether to display the aperture in the eye diagram or calculate the eye width. Valid values: `yes` or `no`. When this argument is set to `yes`, the eye aperture is displayed in the output plot. When set to `no`, the eye aperture width is returned. Default value: `no`.
- *Optimize*—Specifies whether to calculate the reference voltage that can be used to achieve the maximum eye aperture width. Valid values: `yes` or `no`. Default value: `no`.

Note the following points:

The `eyeAperture` function results in an evaluation error in the following cases:

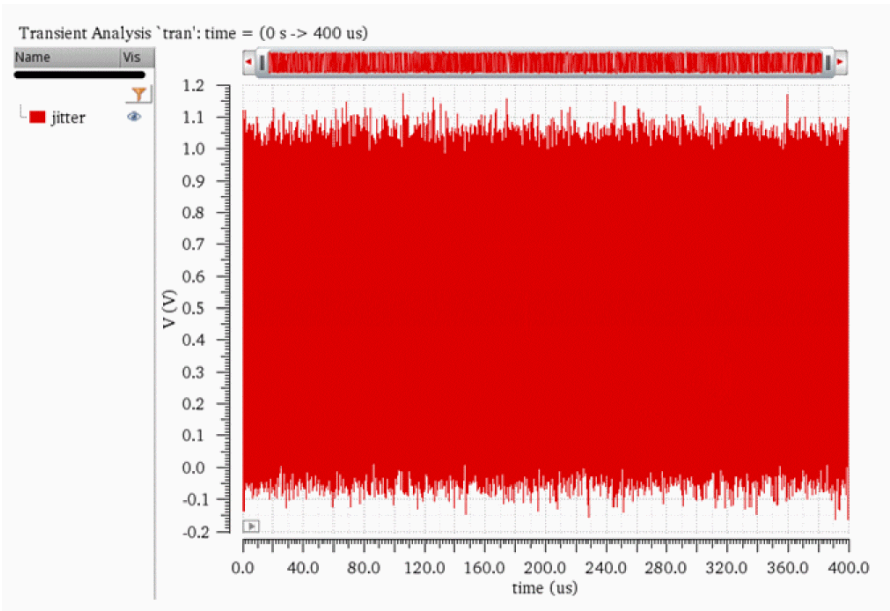
- For open eye diagram
- Eye period is not set as `2UI`
- Value of the AC or DC height exceeds the height of the eye
- Value specified for `Vref` is incorrect

Examples

Part1: Calculating eye aperture on a single eye diagram waveform

Consider the following input signal from the transient analysis:

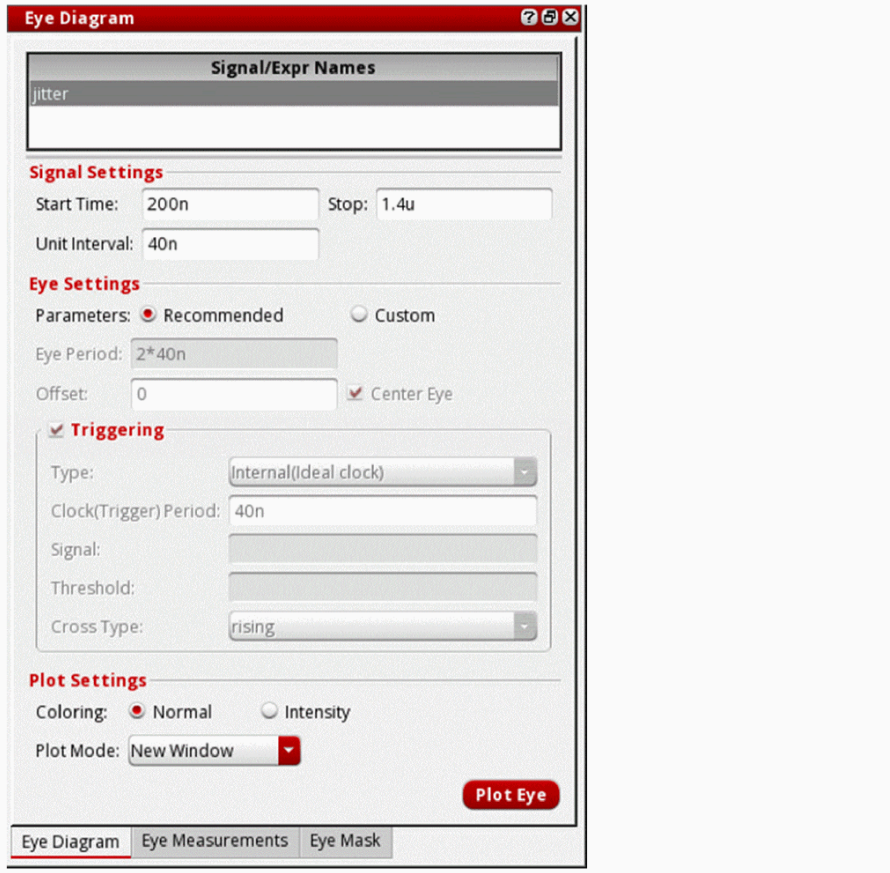
```
v("jitter" ?result "tran" ?resultsDir "./ViVA_Workshop_IC618/design/prbs.raw")
```



> The `eyeAperture` function requires an eye diagram waveform as an input; therefore, firstly, you need to plot an `eyeDiagram` from this input signal. To plot the eye diagram, specify the following field values in the *Eye Diagram* assistant:

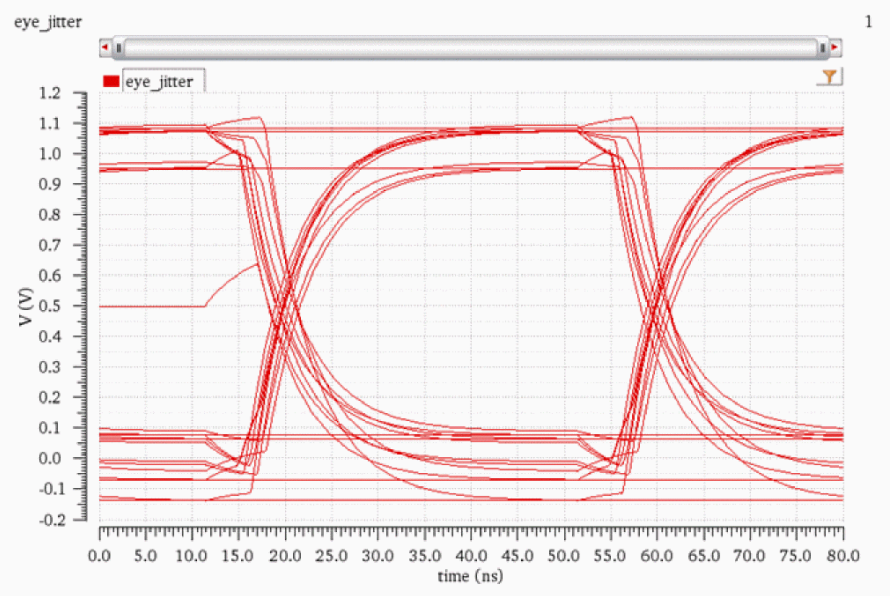
- *Start Time*—`200n`
- *Stop*—`1.4u`
- *Unit Interval*—`40n`
- *Eye Period*—`2*40n` (2UI)

The following figure shows these settings configured in the *Eye Diagram* assistant.



Now, click the *Plot Eye* button. For more information about how to use the *Eye Diagram* assistant, see Eye Diagram Assistant.

The following eye diagram is plotted in a new window.



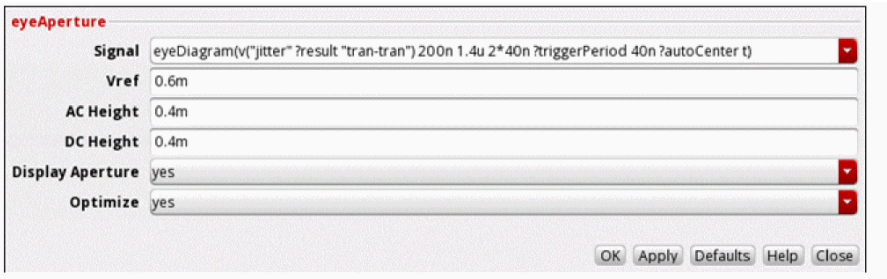
Now, when you send this eye diagram to Calculator, the following expression is displayed in the Buffer:

```
eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)
```

Now, apply the `eyeAperture` function on this Buffer expression with the following arguments:

- *Signal*—`eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)`

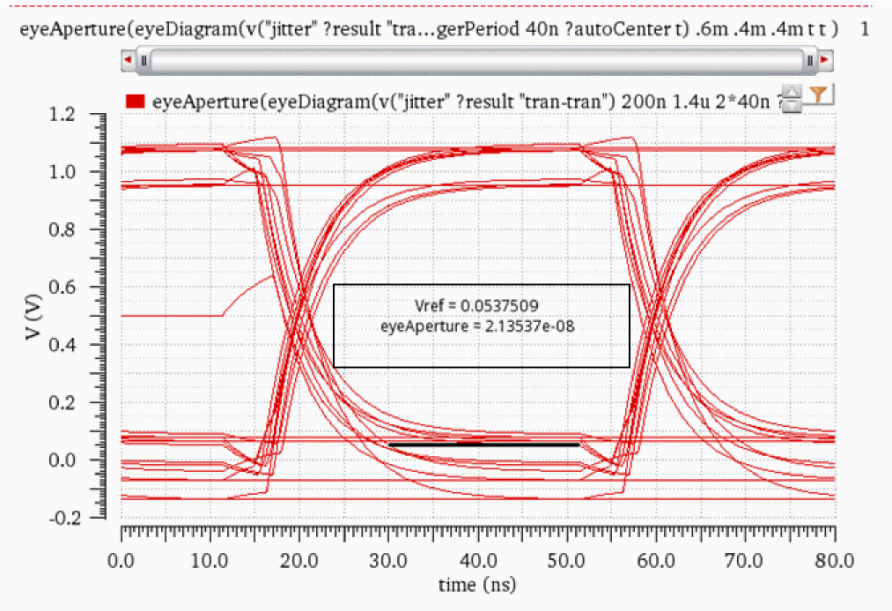
- *Vref*—`0.6m`
- *AC Height*—`0.4m`
- *DC Height*—`0.4m`
- *Display Aperture*—`yes`
- *Optimize*—`yes`



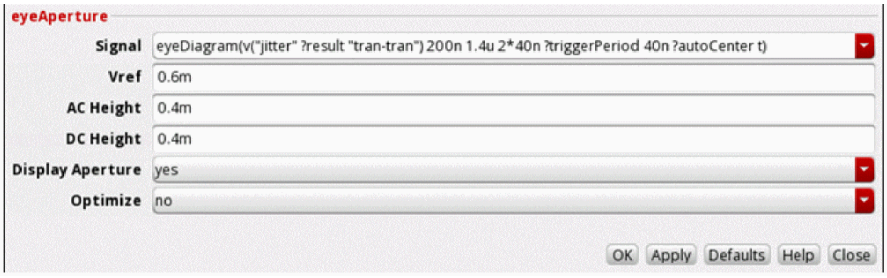
After you apply the function, the following new expression is created in the Buffer:

```
eyeAperture(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) 0.6m 0.4m 0.4m t
t )
```

When you evaluate this expression, the following output is displayed in a new window:



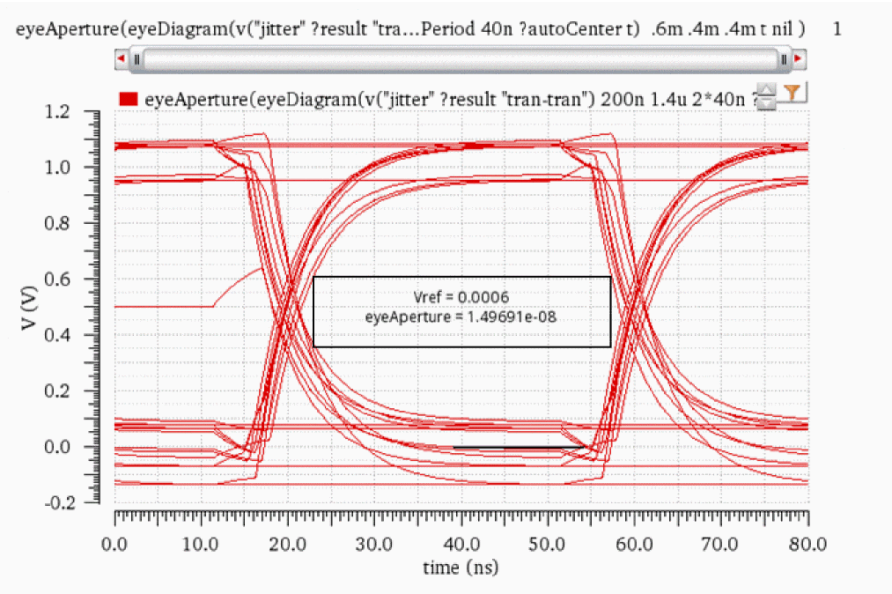
Now, apply the `eyeAperture` function again with the *Optimize* value as `no`.



The following expression is created in the Buffer:

```
eyeAperture(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t)  0.6m 0.4m 0.4m
t nil )
```

When you evaluate this expression, the following output appears in a new window:

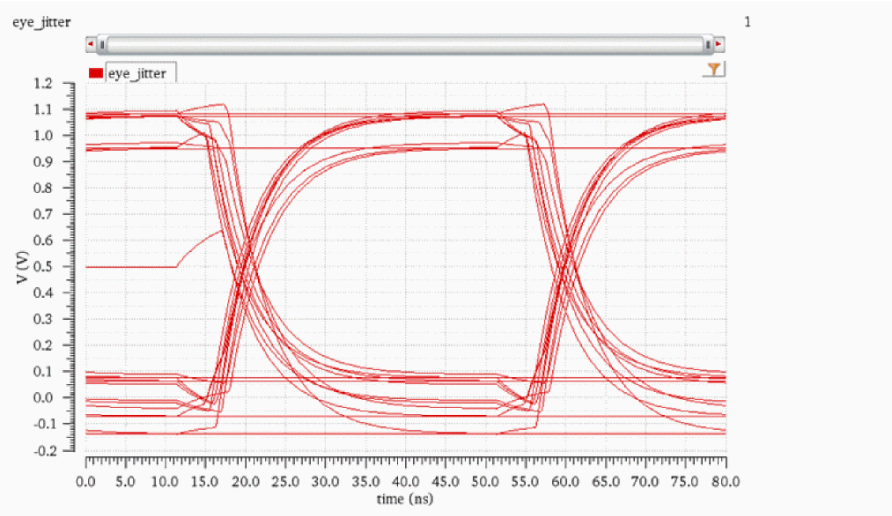


Part2: Calculating eye aperture on two eye diagram waveforms

Eye Diagram1: Create an eye diagram waveform for the `jitter` signal by using the *Eye Diagram* assistant with the following field values.

- *Start Time*—`200n`
- *Stop*—`1.4u`
- *Unit Interval*—`40n`
- *Eye Period*—`2*40n` (2UI)

The following eye diagram plot appears:

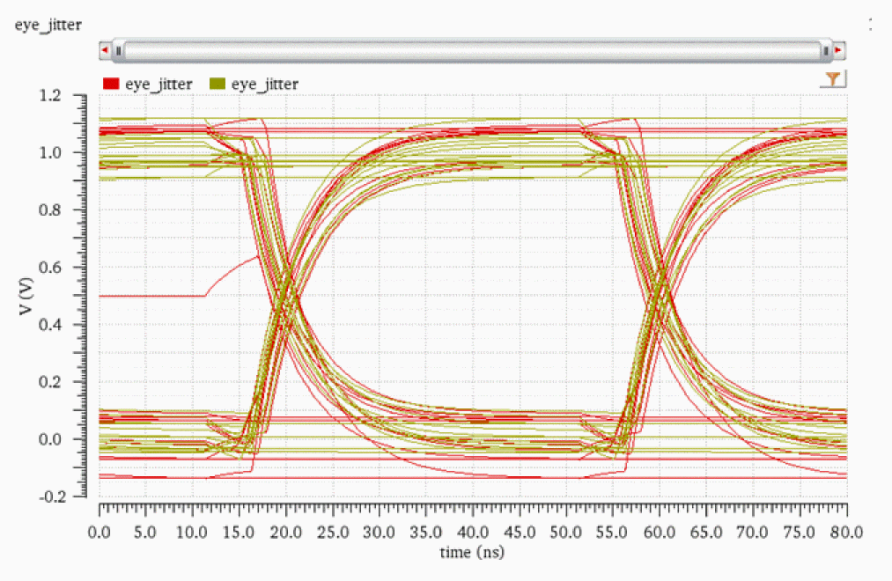


Eye Diagram2: Create the second eye diagram for the `jitter` signal by using the *Eye Diagram* assistant with the following field values:

- *Start Time*—`1.4u`
- *Stop*—`2.8u`
- *Unit Interval*—`40n`
- *Eye Period*—`2*40n` (2UI)

Plot the eye diagram in `Append` mode. This new eye diagram is plotted in the same window in which the first eye diagram (created in part1) was plotted.

The following eye diagrams are displayed in the append mode.



Now, select both the eye diagram waveforms by holding down the `Ctrl` key, right-click and choose *Send To – Calculator as a list*.

The following expression is created in the Buffer:

```
list(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) eyeDiagram(v("jitter" ?result "tran-tran") 1.4u 2.8u 2*40n ?triggerPeriod 40n ?autoCenter t) )
```

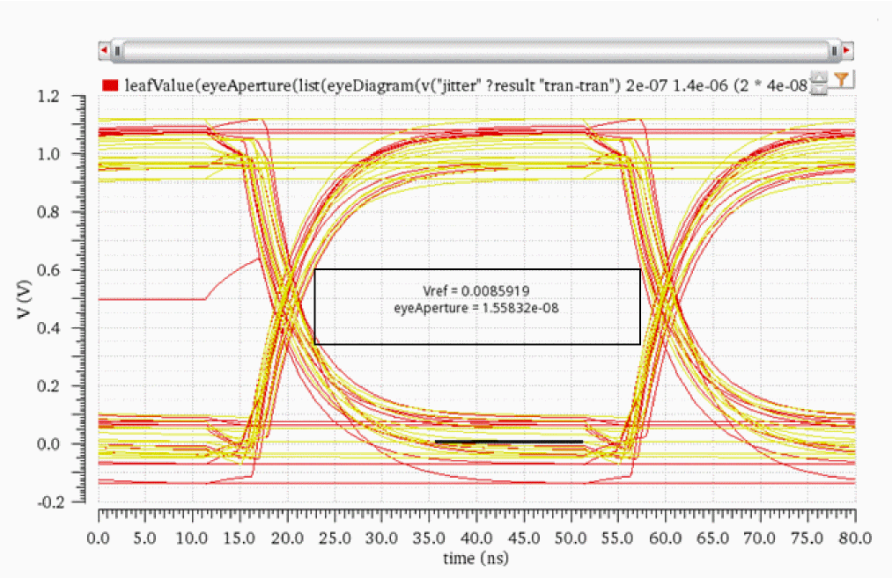
Now, apply the `eyeAperture` function on this expression with the following argument values:

- *Signal*—`list(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) eyeDiagram(v("jitter" ?result "tran-tran") 1.4u 2.8u 2*40n ?triggerPeriod 40n ?autoCenter t) )`
- *Vref*—`0.6m`
- *AC Height*—`0.4m`
- *DC Height*—`0.4m`
- *Display Aperture*—`yes`
- *Optimize*—`yes`

When you apply the function, the following expression is created in the Buffer:

```
eyeAperture(list(eyeDiagram(v("jitter" ?result "tran-tran") 200n 1.4u 2*40n ?triggerPeriod 40n ?autoCenter t) eyeDiagram(v("jitter" ?result "tran-tran") 1.4u 2.8u 2*40n ?triggerPeriod 40n ?autoCenter t) ) 0.6m 0.4m 0.4m t t )
```

When you evaluate this expression, the following output is displayed in a new window:



Additional Information

If you specify an output expression using `eyeAperture` and `eyeDiagram` functions in the ADE Outputs Setup pane, the output waveform for eyeAperture is generated after the simulation is complete. In addition, the scalar values for VREF and eyeAperture are displayed in CIW.

For example, create the following expression in ADE Outputs Setup:

```
eyeAperture(eyeDiagram(VT("/net3")0 VAR("a") 1e-10)0 (VAR("a")/100)
```

Now, when you run a simulation, the following output waveform is generated that displays the `eyeAperture` and `Vref` values.

Also, the Vref and eyeAperture scalar values are displayed in CIW:

```
Loading paraplot.cxt
vref = 0.000000e+00. Eye aperture width = 9.357741e-11
net /net3 selected but not highlighted
net /net2 selected but not highlighted
Loading stats.cxt
```