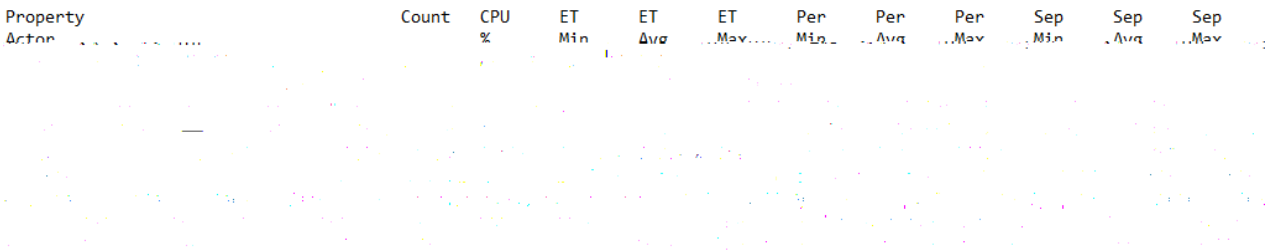
Percepio Application Note PA-025, 2018-10-25

Today's powerful vision processors allows for great performance, but how do you ensure your solution really makes efficient use of the hardware? Perhaps some OpenVX graph node requires much more processing time than expected and overloads a core, while the other cores are mostly idle? Perhaps the application is spending a lot of time waiting for DMA transfers to complete? Perhaps you have tried to

The Actor Instance Graph can show various timing properties, not just execution time, but also e.g. separation and periodicity. You can change what property that is displayed in the dropdown menu that reads “Execution Time” by default. Some of the properties are however not very relevant for OpenVX traces. For

With the Actor Statistics report you can not only find the extreme values, but with the links you can also see what was going on in the system at that time, i.e. what caused it. Although all details are not recorded you can still get valuable clues. For instance, using the Actor Instance Graph you can find other cases with similarly high execution time and check for correlations in the trace. Perhaps the high execution time only occurs under particular circumstances, e.g. due to intense activity on other CPU cores saturating the bus?

Note that you can export and save the the statistics reports, either as formatted HTML files (like above) or as tabbed text files (example below). The latter allows for easy data import into other tools and is done by checking the option “Data Export” in the Actor Statistics Report dialog.



This allows you to run measurements on alternative designs and compare the resulting performance metrics in a systematic way. For instance, in the above example we can see that IDLE0 is running 40.7% of the time, meaning that Core 0 is only 59.7% loaded, while IDLE1 only runs for 17% of the time meaning that Core 1 is 83% loaded. Perhaps that idea you got last night would give higher and more balanced utilization? Just record traces and compare!

Tracealyzer allows for adding your own , i.e. custom events logged from the application code. User events allows you to visualize just about anything in your application, like diagnostic messages, variable values and states. This can be displayed in several ways.

In the below screenshot, we see an example where user events have been logged on two user event channels, ”MyVariable” showing values of an integer variable and ”MyState” showing state names. Tracealyzer can display such user events in several ways, e.g. as event labels in the trace view (1) and as entries in the Event Log (2). The User Event Signal Plot (3) allows for plotting numerical data from user events.

Moreover, if you have important state variables in your system, you can log the state changes as user events and define a State Machine in Tracealyzer to see the states in the trace view timeline (4). You can also see a summary of the state changes as a state machine graph (5). You can even get statistics on the time spent in each state, or the time between any two events by defining a ”custom interval”.

To add user events into your trace, see the below examples.

Use evTraceCustomEvent()00320025002D002800260037000300460048004B001400030020000300480059003700550044005

https://percepio.com

Visit https://percepio.com/download and make sure to select OpenVX/Synopsys as Target Platform. The download link will be sent to the provided email address. If you don't have a Tracealyzer license already, sign up for evaluation in the download registration form. You will then receive an addtional email with an evaluation license key. The evaluation time is 30 days for this version of Tracealyzer.

To install on Windows, just launch the executable installer. On Linux, extract the Tracealyzer .tgz archive and read "RunningOnLinux.txt" for further instructions.

When Tracealyser is started for the first time, select "Activate License" to enter your license key.

You can begin exploring the features right away using the example trace shown in this document. This is included with Tracealyzer and available directly from the initial "Welcome to Tracealyzer" screen.

To record a trace from your OpenVX application, follow these steps:

1. In your application code, add the following include statement:

   $ #

2. Add a