rest of event-driven techniques. This is more significant when the mathematical complexity of the neural models increases (see **Figures 5–8**).

- The main factor that finally constrains the computational performance of all these event-driven methods is the number of events that need to be processed. These events are mainly internal and propagated spikes (Ros et al., 2006a) that linearly increase with the neural activity. Time-driven integration methods are preferred rather than event-driven integration methods for those neural networks with high levels of neural activity (see **Figures 7**, **8**). Conversely, there are particular cases in which the event-driven integration methods can be the best option. There are, actually, biologically realistic SNNs in which parts of their inner layers present a very low and sparse neural activity, such as the granular cells in the cerebellum (D'Angelo et al., 2016) or the mushroom bodies within the olfactory system in Drosophila (Serrano et al., 2013). The importance of these particular networks cannot be overlooked (i.e., just the granular cerebellar layer accounts for half of the neurons of the whole brain, its neurons receive between three and six input synapses with a low and very sparse activity, with most of them remaining silent and barely generating spikes). In these cases, event-driven integration methods perform better than time-driven integration methods.

Time-Driven Main Functional Aspects

The main functional aspects in relation to the time-driven integration methods can be summarized as follows:

- Hybrid CPU-GPU integration methods perform better than CPU methods. This is specifically relevant when the mathematical complexity of the neural models increases. GPU hardware architecture performs better computing parallel tasks than CPU architecture. The computation of the neural dynamics is a pure parallelizable task and consequently, GPU-friendly. In a hybrid CPU-GPU platform, the GPU only processes the neural dynamics, whilst the spike generation and propagation are processed in the CPU. When the mathematical complexity of the neural models increases, the workload assigned to the GPU increases, whilst the workload of the CPU remains equal. For this reason, CPU-GPU neural models perform better than purely CPU neural models, especially when the mathematical complexity of the neural models increases. This increase in performance is shown in **Figures 5–8**.
- Bi-fixed-step integration methods outperform fixed-step integration methods for both CPU and GPU platforms when the mathematical complexity of the neural model increases (see **Figures 5–8**). Complex neural models usually demand small integration step sizes to better cope with the stiffness of their neural model equations during the spike shape generation. **Figures 5E,F** show how the maximum step size on a fixed-step integration method is constrained due to the differential equation stiffness (HH model). The adaptation mechanism used by the CPU bi-fixed-step integration methods improves the simulation performance by enlarging the simulation step size during those neural dynamic intervals out of the spike phase.
- The adaptation mechanism of the integration step size for GPU bi-fixed-step integration methods increases performance thanks to the minimization of the time spent in the synchronization and transfer of data between the CPU and GPU processors.
- Whilst CPU integration methods are better suited for small-medium groups of neurons (from one neuron to several thousands of neurons, depending on the mathematical complexity), the GPU integration methods are better suited for larger numbers of neurons (from thousands to millions of neurons). The computation time invested in the synchronization period and data transferences between CPU and GPU platforms dominates over the computation time invested in solving the neural dynamics when the number of neurons within the network is small (see **Figure 6**). In this case, the computational performance of the GPU integration methods reaches a plateau.
- The adaptation mechanism that the bi-fixed-step integration method uses in CPU may decrease the computational performance when the mean firing rate over the neural network is quite high. When the neural activity increases, the ratio of use between the local and global step also increases. The computational workload for the neural dynamic increases and the performance drops (see how the computation time increases in **Figure 7**).

EDLUT Hybrid Architecture into Perspective

EDLUT is a simulator mainly oriented to efficiently simulate medium-scale neural networks (tens of thousands of neurons) pursuing real time simulations. EDLUT uses point neural models, such as LIF, AdEx or HH. EDLUT information transmission relies on spike timing rather than on the particular spike shape. What matters is when the spike is emitted rather than how the spike is generated. Neurons are just means to an end needed toward understanding the behavior of the neural network behind. The neural communication mechanisms are deployed at network level at very high simulation speeds on a single multicore computer, thus facilitating real time embodiment experiments (Carrillo et al., 2008; Luque et al., 2011a,b, 2014a,b, 2016; Garrido et al., 2013a; Casellato et al., 2014; Antonietti et al., 2016). In these neurorobotic experimental set-ups the neural network and the body are coupled as a single entity.

Conversely, NEURON (Hines and Carnevale, 1997) is mainly designed for the simulation of very complex and detailed neural models. What matters here is how the spike was generated rather than when it was emitted. Understanding neurons themselves is the goal. To be as biologically plausible as possible, NEURON is conceived to deal with high levels of mathematical complexity that usually require time-driven simulation methods (either fixed- or variable-step integration methods). The computational cost here highly depends on the mathematical complexity which makes the simulation of hundreds or tens of hundreds neurons conforming a network almost computationally intractable. Using