Platform Orchestration with a RISC-V Tiny Controller

Sergi Alcaide

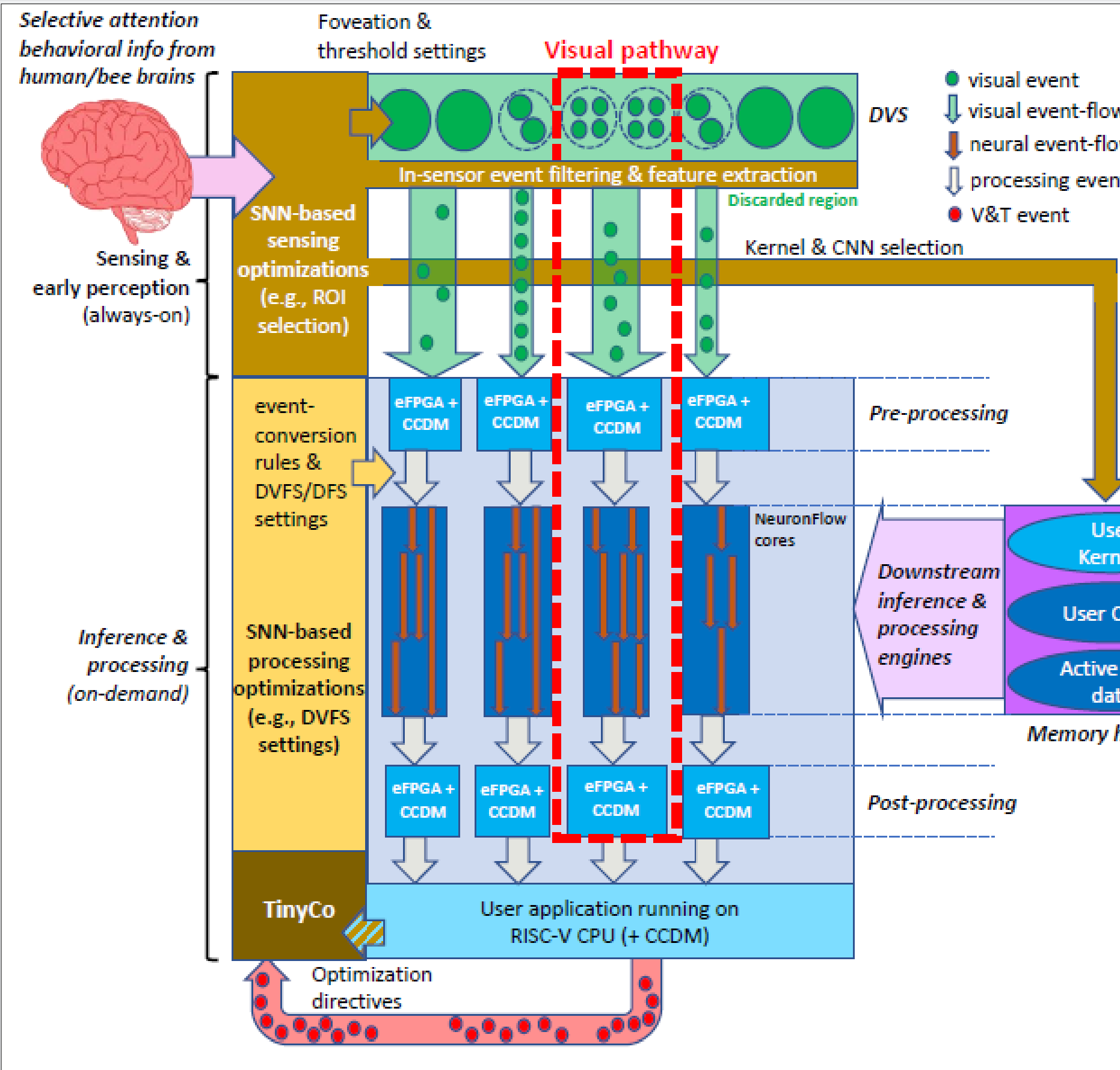
salcaide@bsc.es

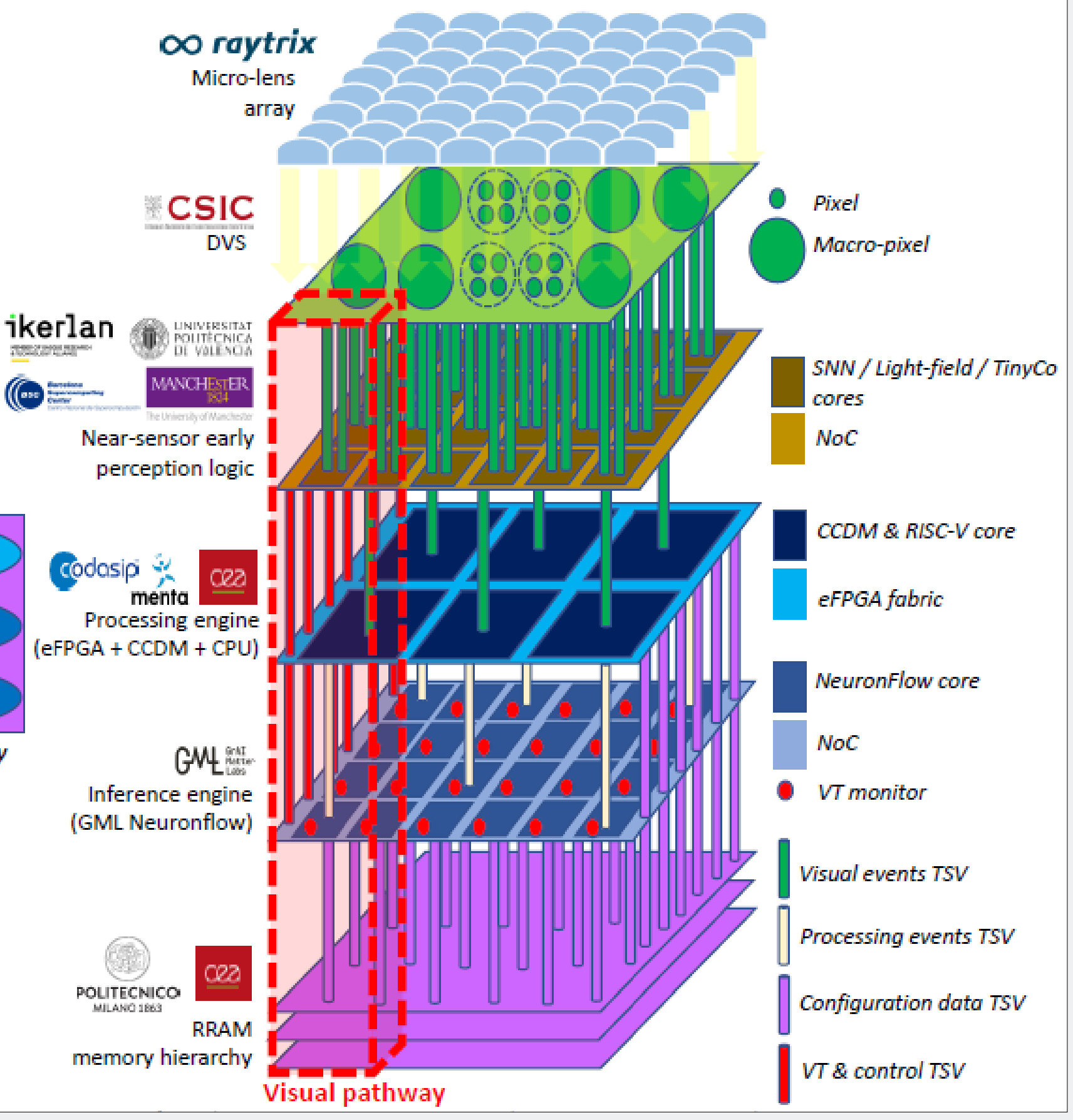
Omar Alymlahi[1], Juan Carlos Rodriguez[1], Marcel Sarraseca[1], Francisco Fuentes[1], Feng Chang[1], Ilham Lasfar[1], Marc Grau[1,2], Francisco J.Cazorla[1], Ramon Canal[1,2], Jaume Abella[1]

[1]Barcelona Supercomputing Center (BSC)
Barcelona, Spain

[2]Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain

NimbleAI system & Motivation

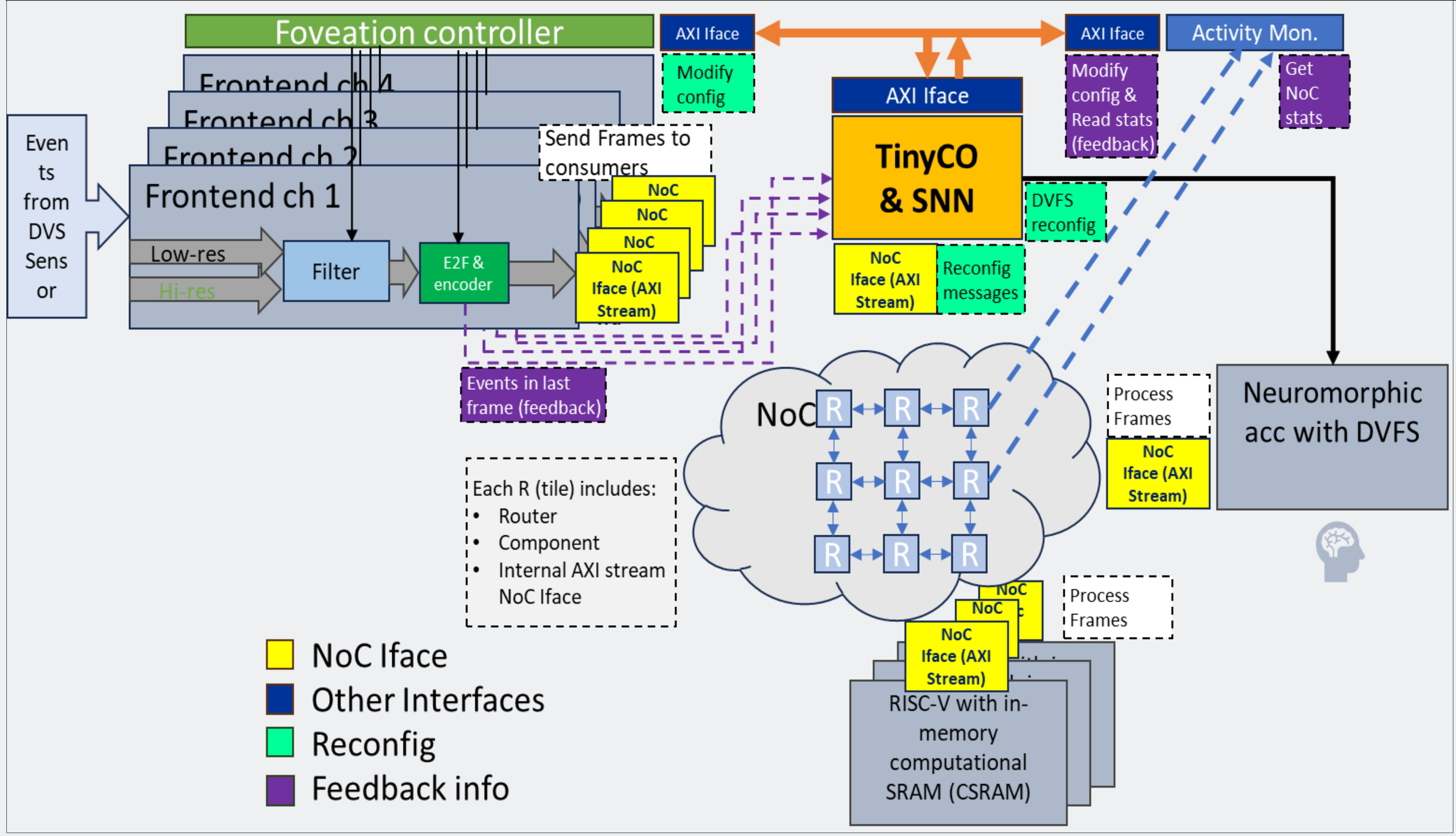

Selective attention behavioral info from human/bee brains
Foveation & threshold settings
Visual pathway
DVS
visual event
visual event-flow
neural event-flow
processing event-flows
V&T event
In-sensor event filtering & feature extraction
Discarded region
Kernel & CNN selection
Sensing & early perception (always-on)
SNN-based sensing optimizations (e.g., ROI selection)
event-conversion rules & DVFS/DFS settings
eFPGA + CCDM
Pre-processing
NeuronFlow cores
Downstream inference & processing engines
User Kernels
User CNNs
Active CNN data
Memory hierarchy
Inference & processing (on-demand)
SNN-based processing optimizations (e.g., DVFS settings)
Post-processing
TinyCo
User application running on RISC-V CPU (+ CCDM)
Optimization directives



- Designing a 3D integrated sensing-processing neuromorphic chip
- NimbleAI system:
  1) Data is captured using a foveated **DVS** (Dynamic Vision Sensor) as events
  2) Events are filtered and accumulated as frames based on their **Visual Path**
  3) The NoC distributes the frames to a back-end processing engine:
     - A **neuromorphic** accelerator with DVFS (Dynamic Voltage and Frequency Scaling)
     - Dedicated RISC-V CPUs coupled with an **in-memory computing SRAM**
  4) Results computed are sent to the host CPU
- Main challenge is the platform's adaptability to **optimize energy efficiency** based on the application characterization and the platform's **runtime status**.
- This is achieved by integrating a tiny controller, named **SafeTCo** (Safety Controller) or **TinyCo** (Tiny Controller)

NimbleAI platform orchestration

- SafeTCo is a 32-bit RISC-V Core written in Verilog with separated data and instruction memories
- Data memory contains a region reserved for the **system configuration**
- **SafeTCo operation**
  1. Host CPU initialize the system configuration and allows the user to modify it
  2. SafeTCo is also connected to multiple system points to receive runtime feedback from the system
  3. During runtime SafeTCo can adapt the system with modifications on different components to improve energy efficiency



Feedback interfaces

SafeTCo takes optimizations decisions based on thresholds (set by the user) and current status of the platform. Currently, we have two sources of system feedback:

SafeSU: As a monitoring unit, the SafeTCo will read the statistics recollected from the NoC by reading the internal counters of the SafeSU. SafeSU can also trigger interruptions on the core if needed.

Interface used: AXI4

Number of events per frame: As a method to retrieve information of the input data, the E2F will send this information that can be used to modify the DVFS or change the backend processing unit.

Interface used: Custom signals

Runtime reconfiguration Interfaces

Certain NimbleAI components can be reconfigured during runtime:

NoC reconfiguration system: NimbleAI components are organized in tiles.

NoC routers allow runtime reconfigurations including: QoS rules, activation/deactivation of virtual channels, …

Achieved by sending ***reconfiguration messages***.

Interface used: AXI-Stream with custom header message

Foveation Controller: Front-end configuration of the platform is centralized in one component. Reconfigurations include:

- Size of the Regions Of Interest (ROI)
- Event filtering tunning
- Frame rate creation
- Target consumer for each frame

Interface used: AXI4

SafeSU: Activity monitor tracking the NoC traffic, to provide feedback and detect safety/security threats. Requires initial configuration: counters, thresholds, interruptions.

SafeTCo will retrieve data periodically accessing the internal counters using the SafeSU AXI subordinate interface.

Interface used: AXI4

Neuromorphic accelerator DVFS: By changing its frequency or voltage, there is potential to gain energy efficiency (e.g., processing time vs frame rate)

Interface used: Custom interface

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon Europe Programme under the NimbleAI Project (www.nimbleai.eu/), grant agreement num. 101070679. This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant PID2019-107255GB-C21/AEI/10.13039/501100011033. Authors thank the support given to the Research Group SSAS (Code: 2021 SGR 00637) by the Research and University Department of the Generalitat de Catalunya.