ANDANTE

AI FOR NEW DEVICES AND TECHNOLOGIES AT THE EDGE

Simulation framework for energy and latency in multi-core neuromorphic architectures

Stefano Traferro (IMEC-NL)

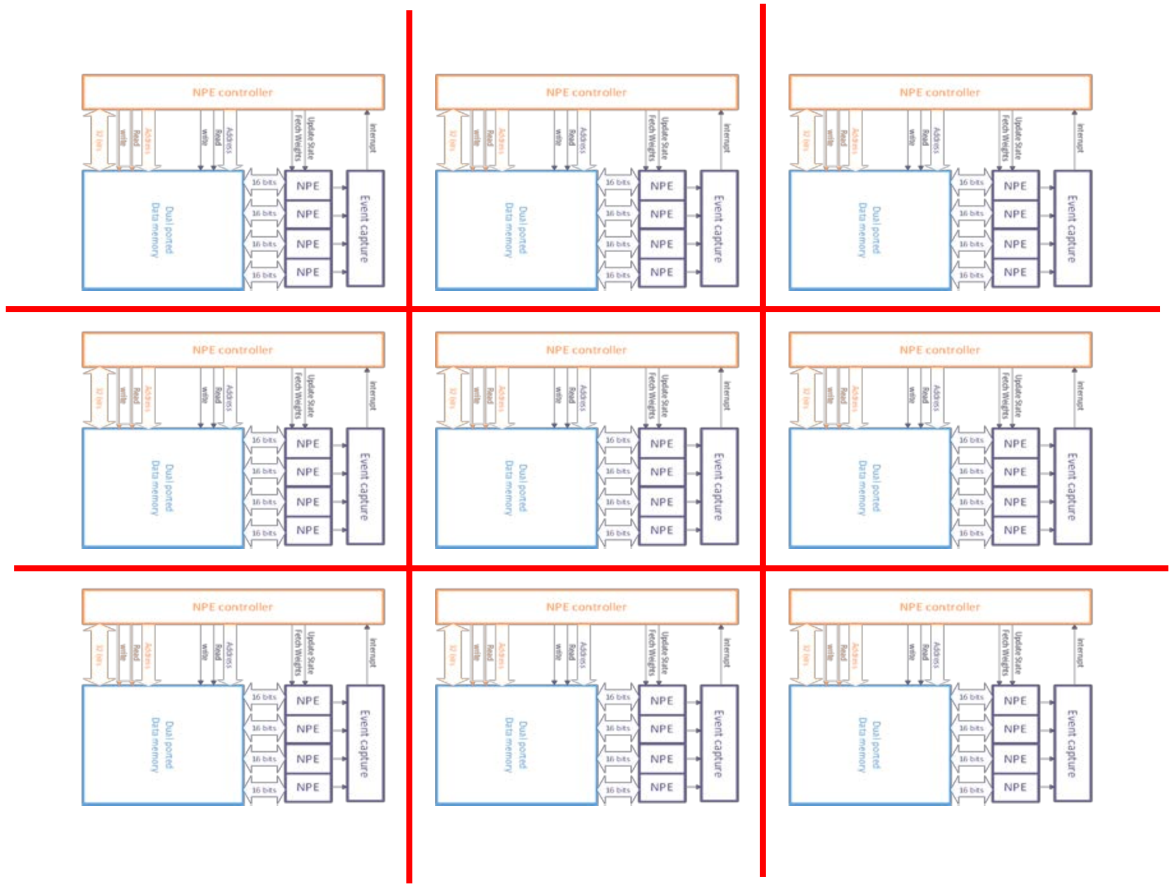
CONTENTS

SENSim - High level simulator for SENeCA, Imec-NL's large scale neuromorphic platform

INTRODUCTION TO THE SENeCA ARCHITECTURE

Scalable Energy efficient Neuromorphic Computing Architecture

- RISC-V NPE controller
  - Programmable neuron model in C.
  - Supports for online learning.
  - Flexible and efficient use of resources.
- Optimized **event-driven core I/O**
  - Reduced interrupt overhead.
- Optimized interconnect
  - Low latency **Multicasting**
  - Scalable
    - Tiling.
    - Chiplet.
    - Using off-chip memory.



PURPOSE OF THE SIMULATOR

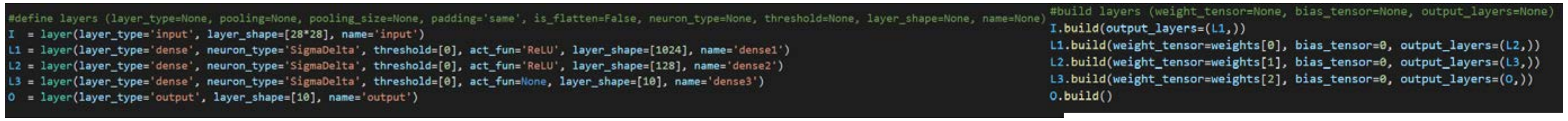
SENSim: SENeCA Simulator

- Purpose
  - Architectural exploration
    - Interconnect, Asynchronous/Synchronous execution, Flow-control, Neuron models, …
    - Benchmarking of emerging memory technologies, e.g. NVM, HBM, …
  - Emulation of the scaled-up platform before silicon realization
    - Not possible in single FPGA.
    - Too slow in low level simulation.
  - Abstract the hardware for
    - Early application optimization.
    - Iterative close loop mapping optimization.
- Provides
  - Relative power estimation (memory access, operations, …).
  - Processing latency.
  - Status of the cores, event-queues, idle times, …

STRUCTURE OF THE SIMULATOR

Hardware aware SNN simulator

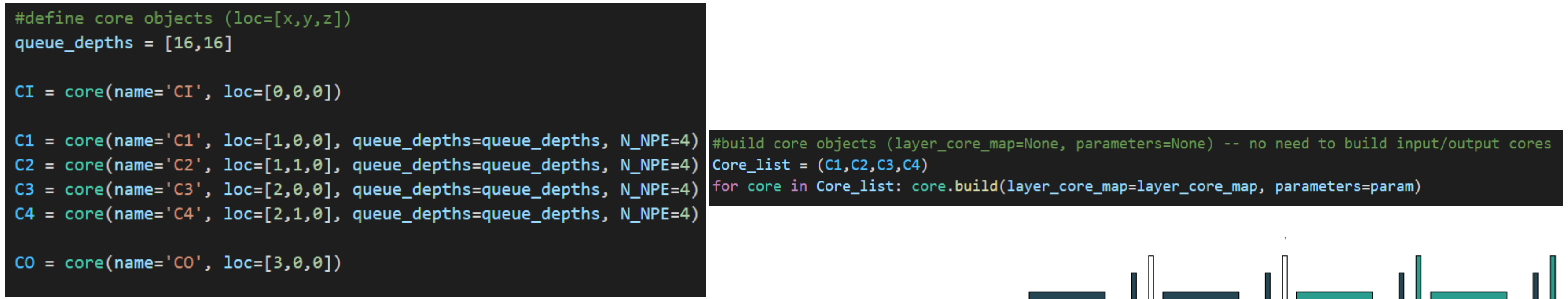
Definition of the SNN layers and connectivity



```
#define layers (layer_type=None, pooling=None, pooling_size=None, padding='same', is_flatten=False, neuron_type=None, threshold=None, layer_shape=None, name=None)
I  = layer(layer_type='input', layer_shape=[28*28], name='input')
L1 = layer(layer_type='dense', neuron_type='SigmaDelta', threshold=[0], act_fun='ReLU', layer_shape=[1024], name='dense1')
L2 = layer(layer_type='dense', neuron_type='SigmaDelta', threshold=[0], act_fun='ReLU', layer_shape=[128], name='dense2')
L3 = layer(layer_type='dense', neuron_type='SigmaDelta', threshold=[0], act_fun=None, layer_shape=[10], name='dense3')
O  = layer(layer_type='output', layer_shape=[10], name='output')
```

```
#build layers (weight_tensor=None, bias_tensor=None, output_layers=None)
I.build(output_layers=(L1,))
L1.build(weight_tensor=weights[0], bias_tensor=0, output_layers=(L2,))
L2.build(weight_tensor=weights[1], bias_tensor=0, output_layers=(L3,))
L3.build(weight_tensor=weights[2], bias_tensor=0, output_layers=(O,))
O.build()
```

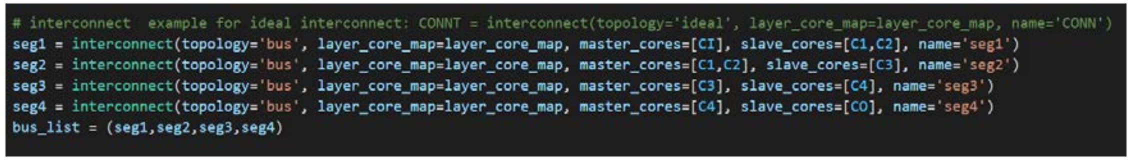
Definition of the processing cores



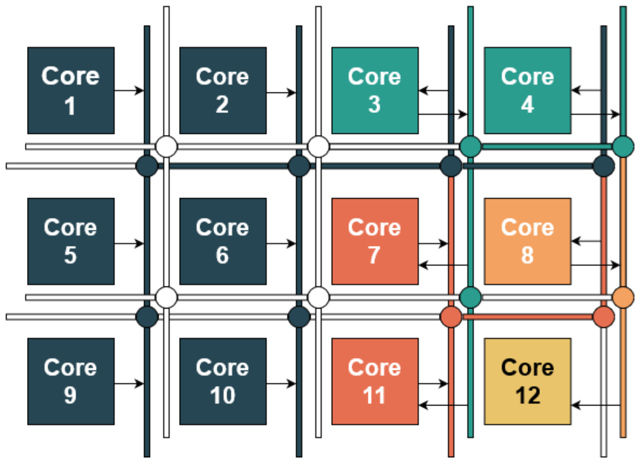
```
#define core objects (loc=[x,y,z])
queue_depths = [16,16]

CI = core(name='CI', loc=[0,0,0])

C1 = core(name='C1', loc=[1,0,0], queue_depths=queue_depths, N_NPE=4)
C2 = core(name='C2', loc=[1,1,0], queue_depths=queue_depths, N_NPE=4)
C3 = core(name='C3', loc=[2,0,0], queue_depths=queue_depths, N_NPE=4)
C4 = core(name='C4', loc=[2,1,0], queue_depths=queue_depths, N_NPE=4)

CO = core(name='CO', loc=[3,0,0])
```

```
#build core objects (layer_core_map=None, parameters=None) -- no need to build input/output cores
Core_list = (C1,C2,C3,C4)
for core in Core_list: core.build(layer_core_map=layer_core_map, parameters=param)
```

Definition of the Interconnect

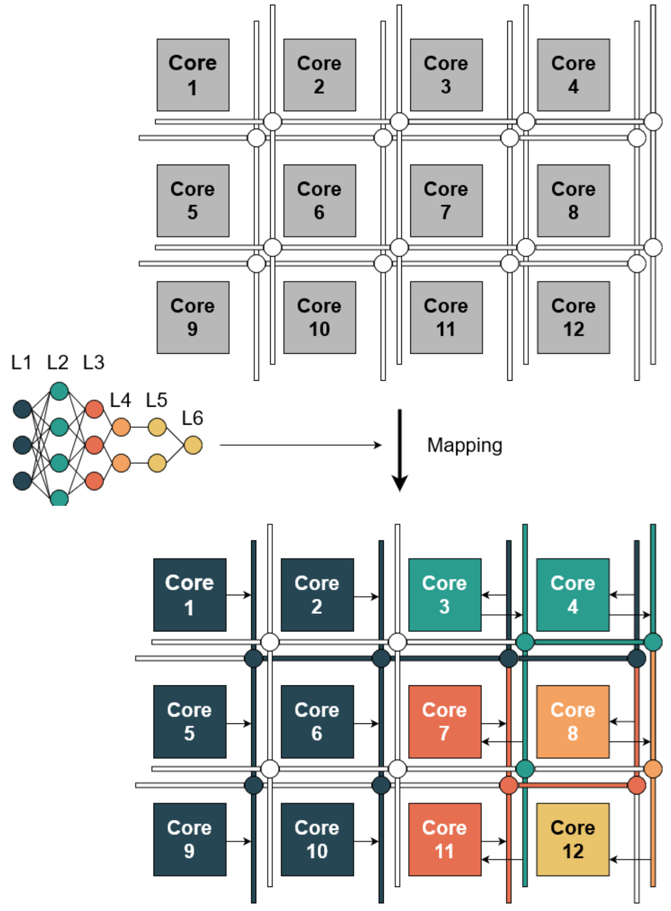


```
# interconnect  example for ideal interconnect: CONNT = interconnect(topology='ideal', layer_core_map=layer_core_map, name='CONN')
seg1 = interconnect(topology='bus', layer_core_map=layer_core_map, master_cores=[CI], slave_cores=[C1,C2], name='seg1')
seg2 = interconnect(topology='bus', layer_core_map=layer_core_map, master_cores=[C1,C2], slave_cores=[C3], name='seg2')
seg3 = interconnect(topology='bus', layer_core_map=layer_core_map, master_cores=[C3], slave_cores=[C4], name='seg3')
seg4 = interconnect(topology='bus', layer_core_map=layer_core_map, master_cores=[C4], slave_cores=[CO], name='seg4')
bus_list = (seg1,seg2,seg3,seg4)
```

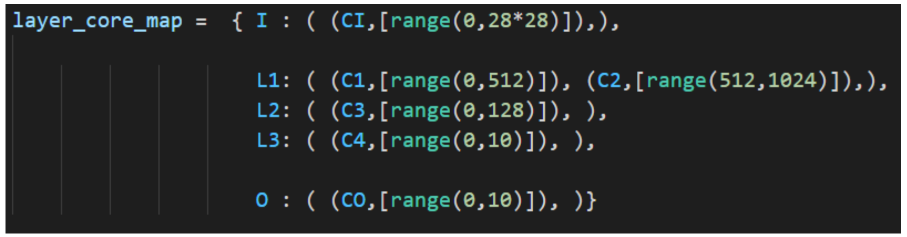


STRUCTURE OF THE SIMULATOR

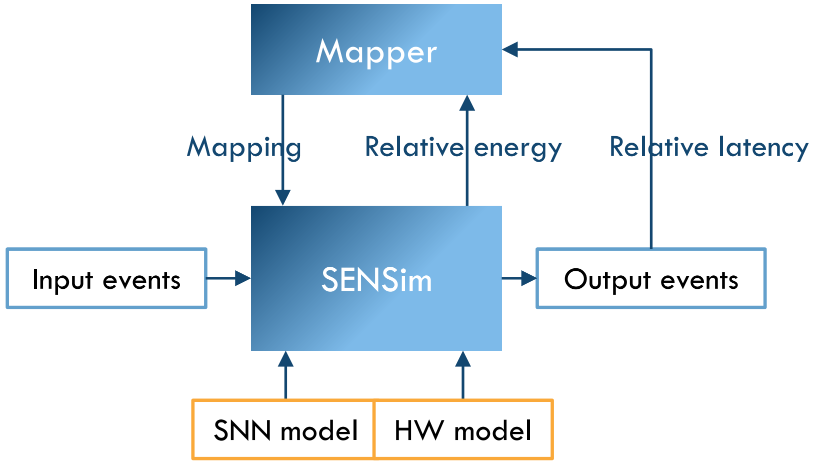
Mapping layers to cores



Mapping from layers to cores is an input to SENSim

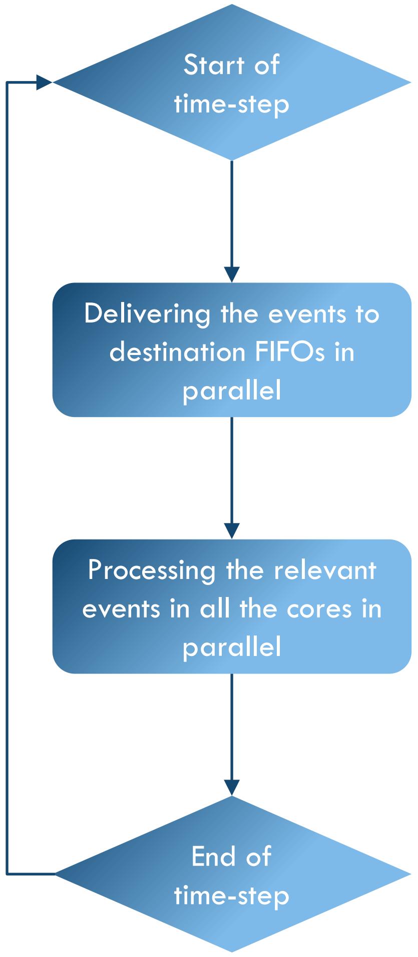


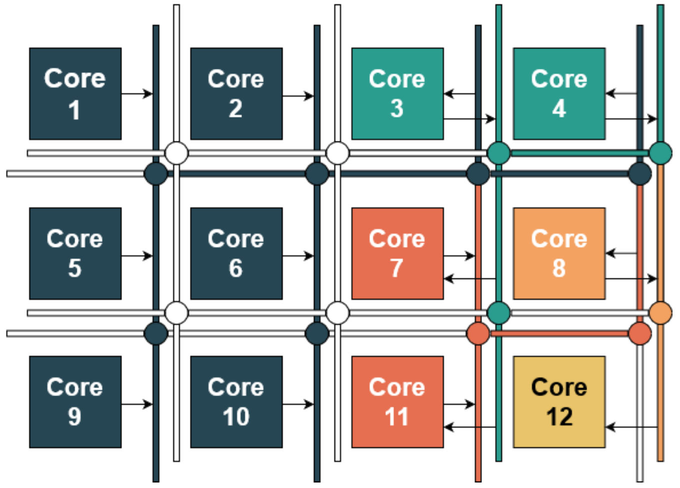
```
layer_core_map =  { I : ( (CI,[range(0,28*28)]),),

                    L1: ( (C1,[range(0,512)]), (C2,[range(512,1024)]),),
                    L2: ( (C3,[range(0,128)]), ),
                    L3: ( (C4,[range(0,10)]), ),

                    O : ( (CO,[range(0,10)]), )}
```

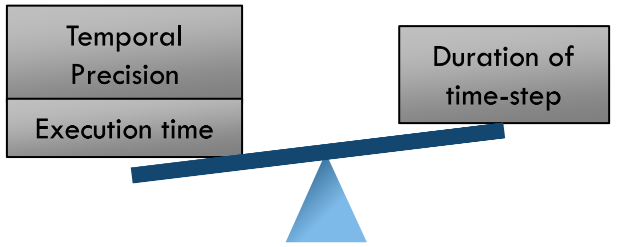


SIMULATION MODEL AND PROCESSING

- Hybrid event-based and time-step driven
- Pure event-driven simulators are very slow (not scalable)
- Parallel execution model by using time-step & events
  - Each core has its own input event queue and output event queue.
  - At each the time step, the interconnect delivers the generated events in the previous time-step to the destination input queues.
  - At each time step, all cores can process their input events in parallel.
- Time-step limits the temporal resolution
  - In the middle of time-step, the dependencies between cores are ignored.
  - It is user responsibility to trade off the smaller time-step with higher execution time.
- Hybrid step-event execution:
  - Parallelization is possible.
  - The temporal resolution is limited to the time-step.
  - A small time-step results in a closer to fully event-driven execution -> it will be slower.


Start of time-step
Delivering the events to destination FIFOs in parallel
Processing the relevant events in all the cores in parallel
End of time-step

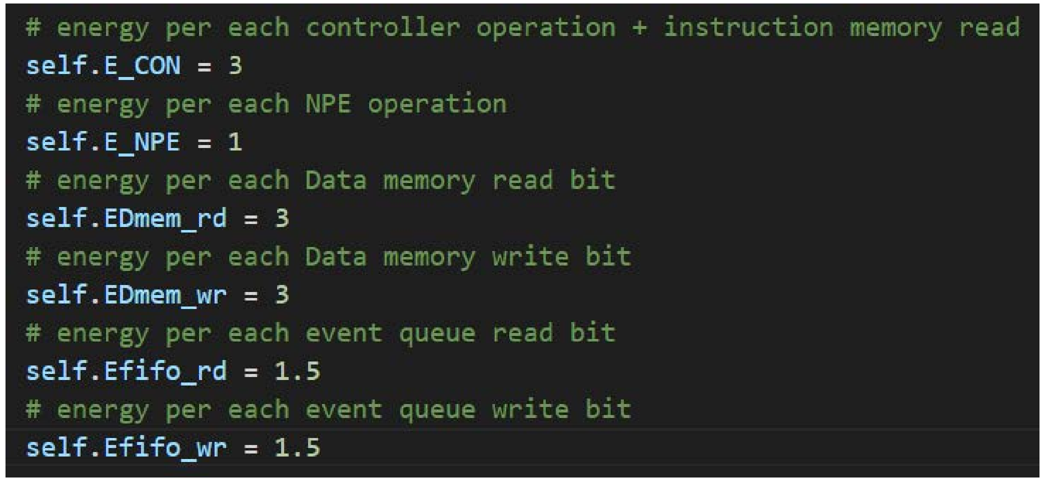




PARAMETERS OF THE SIMULATOR

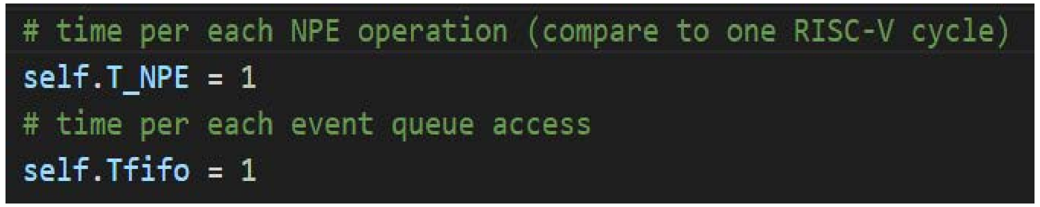
ANDANTE

- Relative energy consumptions
  - Compared to an ALU operation



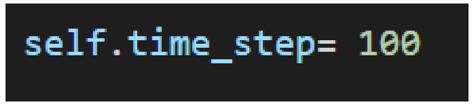
```
# energy per each controller operation + instruction memory read
self.E_CON = 3
# energy per each NPE operation
self.E_NPE = 1
# energy per each Data memory read bit
self.EDmem_rd = 3
# energy per each Data memory write bit
self.EDmem_wr = 3
# energy per each event queue read bit
self.Efifo_rd = 1.5
# energy per each event queue write bit
self.Efifo_wr = 1.5
```

- Relative time consumption
  - Compared to RISC-V clock period



```
# time per each NPE operation (compare to one RISC-V cycle)
self.T_NPE = 1
# time per each event queue access
self.Tfifo = 1
```

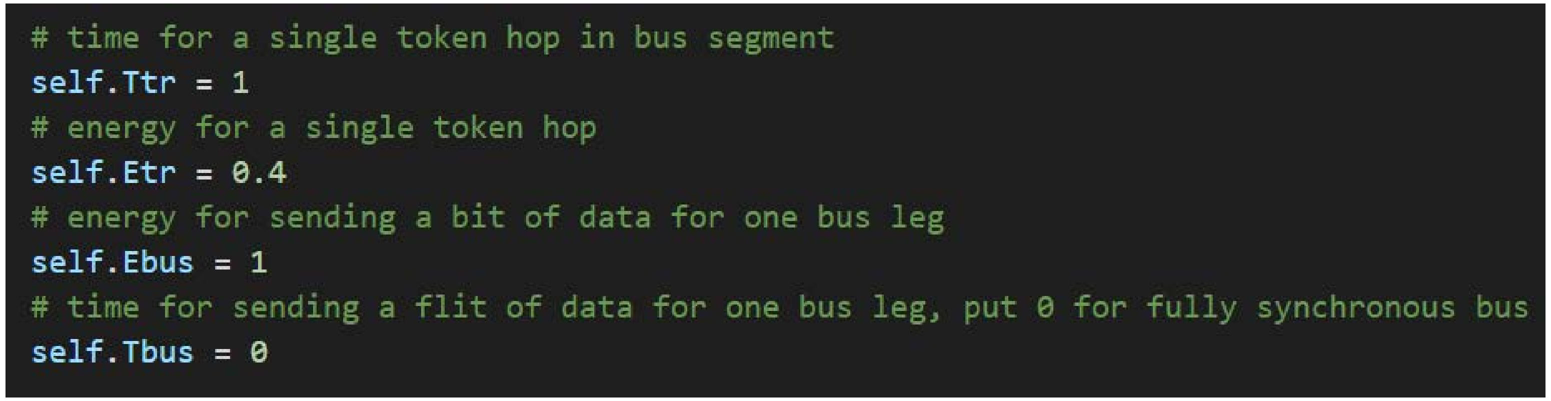
- Time-step duration
  - Unit: one RISC-V clock period



```
self.time_step= 100
```

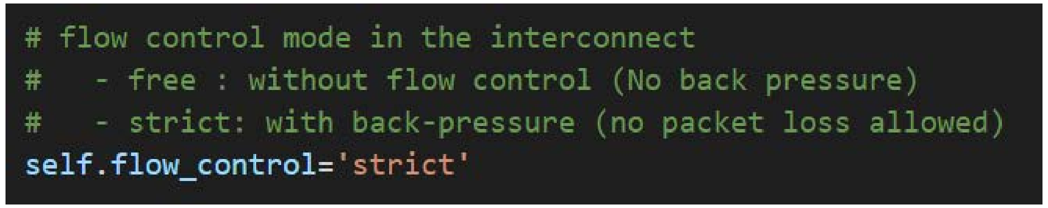
PARAMETERS OF THE SIMULATOR

- Relative energy/time for the interconnect



```
# time for a single token hop in bus segment
self.Ttr = 1
# energy for a single token hop
self.Etr = 0.4
# energy for sending a bit of data for one bus leg
self.Ebus = 1
# time for sending a flit of data for one bus leg, put 0 for fully synchronous bus
self.Tbus = 0
```

- Type of flow control

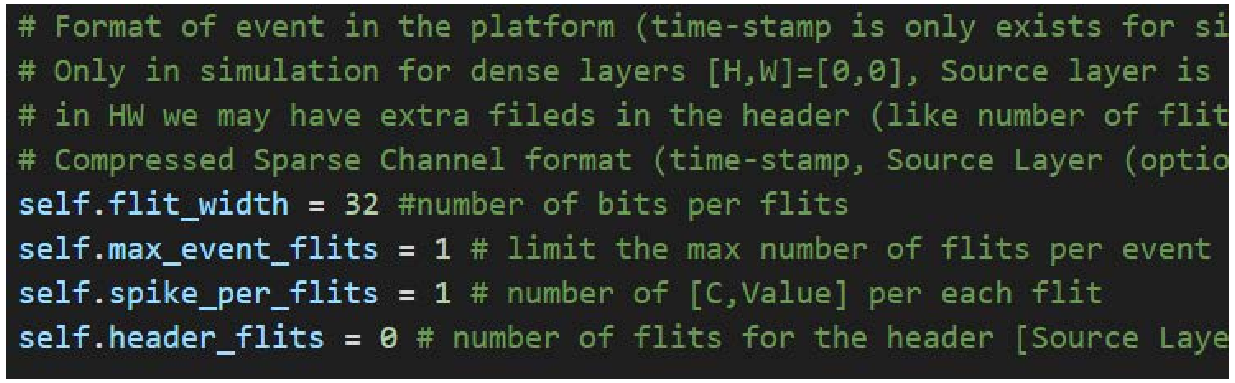


```
# flow control mode in the interconnect
#   - free : without flow control (No back pressure)
#   - strict: with back-pressure (no packet loss allowed)
self.flow_control='strict'
```

PARAMETERS OF THE SIMULATOR

ANDANTE

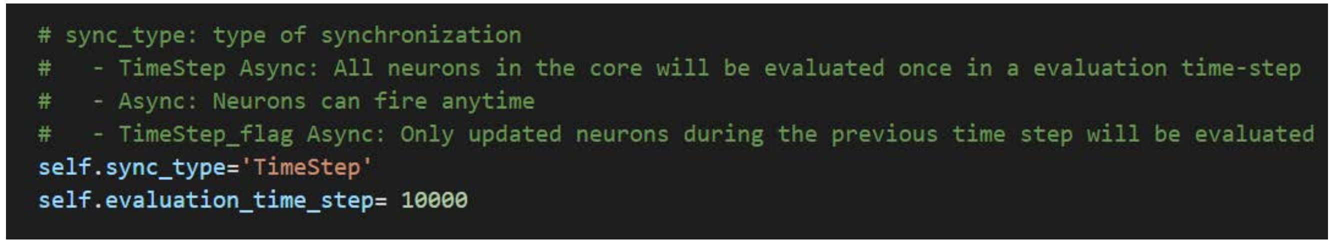
- Event format
  - Coordinate list
  - Compressed Sparse Channel https://en.wikipedia.org/wiki/Sparse_matrix



```
# Format of event in the platform (time-stamp is only exists for si
# Only in simulation for dense layers [H,W]=[0,0], Source layer is
# in HW we may have extra fileds in the header (like number of flit
# Compressed Sparse Channel format (time-stamp, Source Layer (optio
self.flit_width = 32 #number of bits per flits
self.max_event_flits = 1 # limit the max number of flits per event
self.spike_per_flits = 1 # number of [C,Value] per each flit
self.header_flits = 0 # number of flits for the header [Source Laye
```

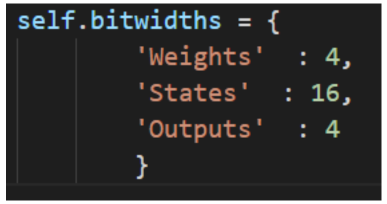
- Neuron evaluation

  Synchronization type



```
# sync_type: type of synchronization
#   - TimeStep Async: All neurons in the core will be evaluated once in a evaluation time-step
#   - Async: Neurons can fire anytime
#   - TimeStep_flag Async: Only updated neurons during the previous time step will be evaluated
self.sync_type='TimeStep'
self.evaluation_time_step= 10000
```

- Bit width of the variables



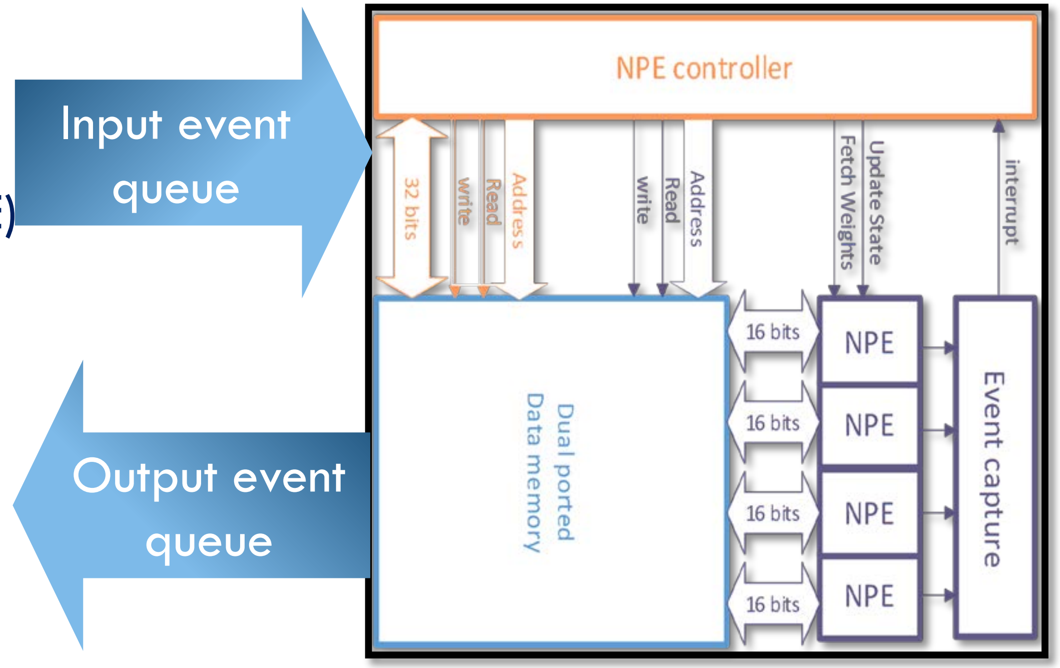
```
self.bitwidths = {
        'Weights'  : 4,
        'States'  : 16,
        'Outputs'  : 4
        }
```

PARAMETER OF THE SIMULATOR

Neuron Processing Cores

- **Parameters:**
  - Size of the input/output event queues
  - Number of Neural Processing Elements (NPE)
  - Geometrical location in the chip (3D)
- **Attributes:**
  - Queue occupancy
  - Accumulated Energy consumption
  - Accumulated idle times
  - Total event loss

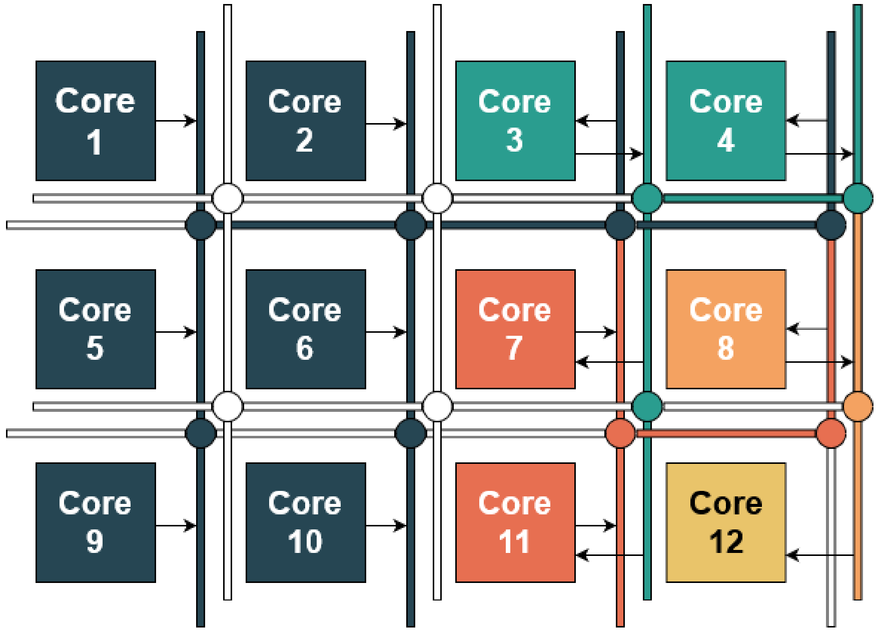
Event-based processor



PARAMETER OF THE SIMULATOR

Interconnect

- **Parameters:**
  - Topology (ideal, PS-NOC, Segmented BUS)
- **Attributes:**
  - Accumulated energy consumption

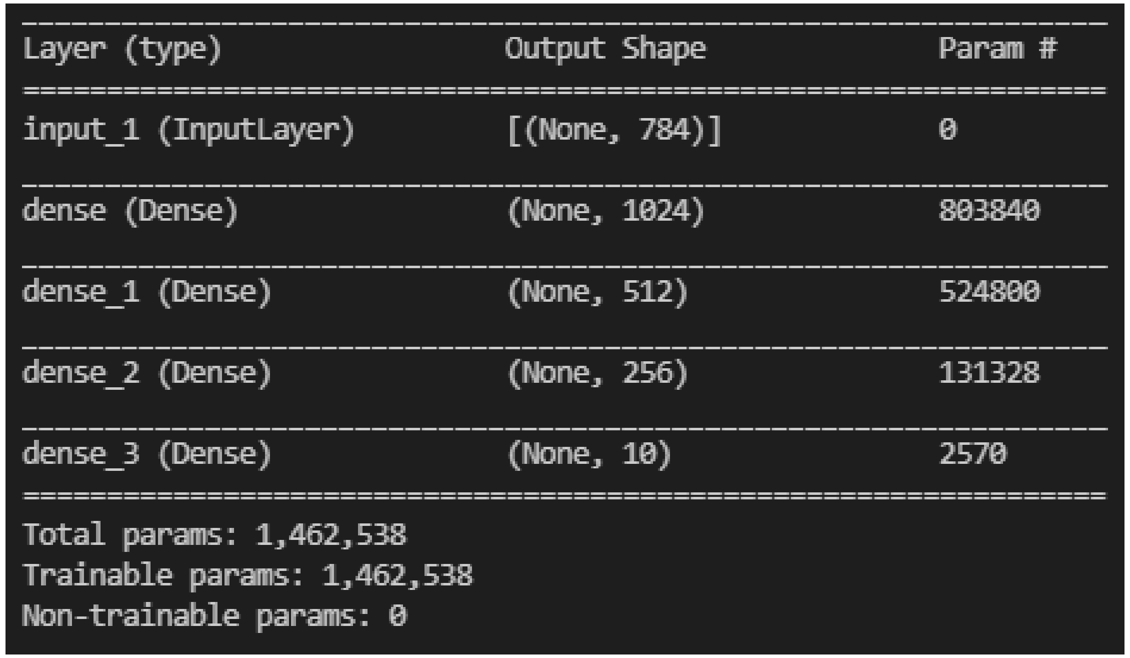


EXPERIMENTAL RESULTS

4 layer fully-connected SNN trained for MNIST dataset: Network model.

28x28 → 1024FC → 512FC → 256FC → 10FC

Accuracy: 0.9873



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense (Dense)                (None, 1024)              803840
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,462,538
Trainable params: 1,462,538
Non-trainable params: 0
```

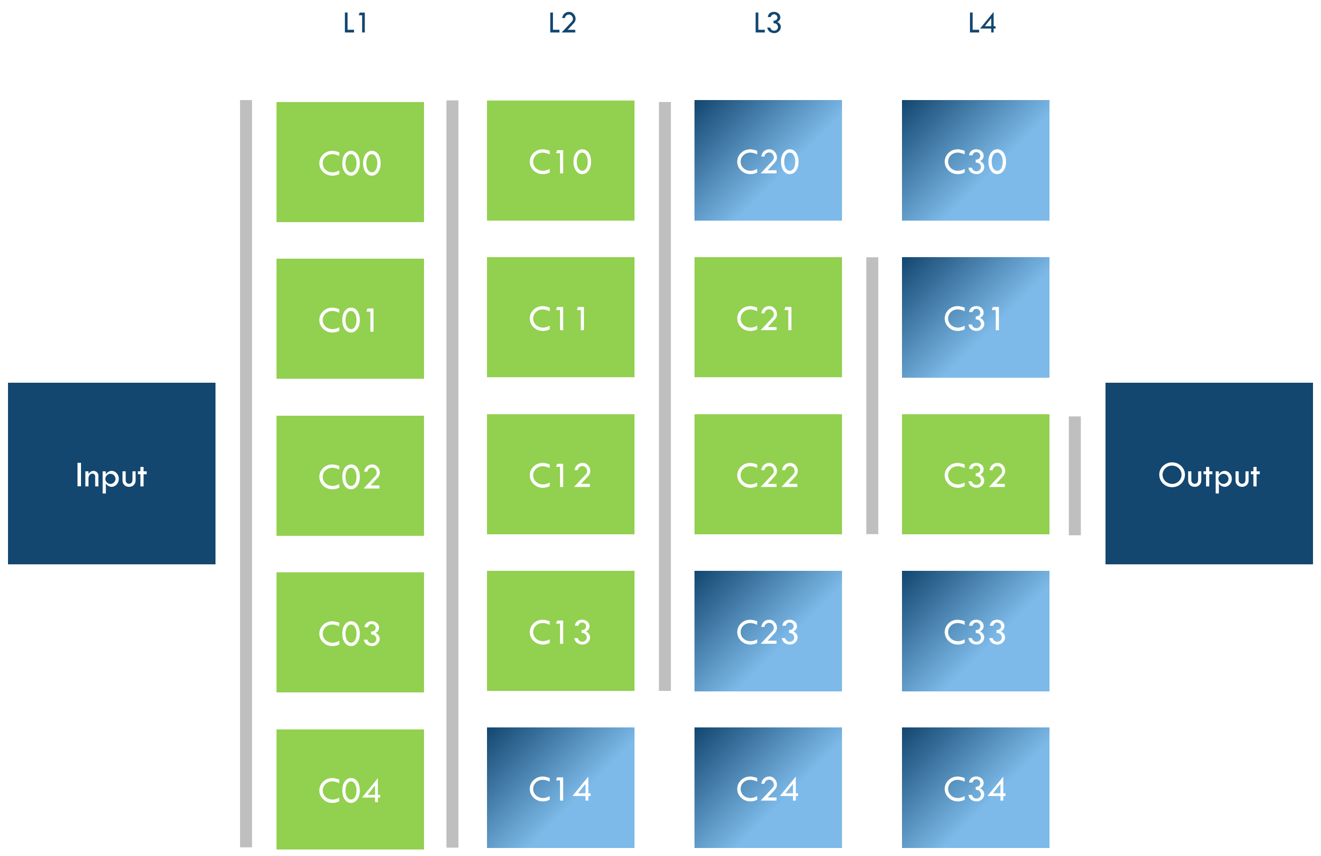
DNN converted to SNN using DeltaDNN conversion method.

1.8k neurons

1.462M parameters

EXPERIMENTAL RESULTS

4 layer fully-connected SNN trained for MNIST dataset: Core allocation.



- L1 (5 cores)
  - 1024 neurons
  - 784k synapses
- L2 (4 cores)
  - 512 neurons
  - 512k synapses
- L3 (2 core)
  - 256 neurons
  - 128k synapses
- L4 (1 core)
  - 10 neurons
  - 2.5k synapses

EXPERIMENTAL RESULTS

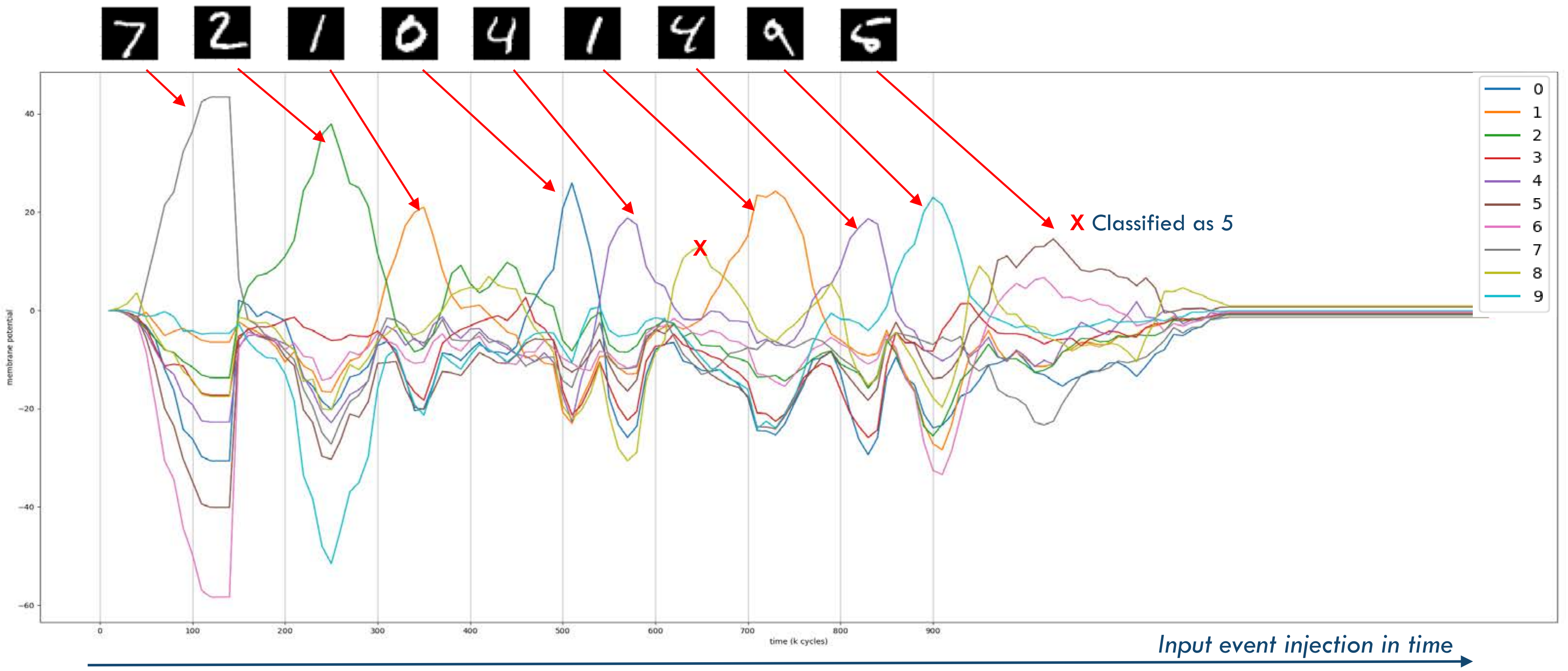
- Input
  - Convert first 9 test digits of MNIST to delta events.
  - 100 cycles rest time between events.
  - 100k cycles rest time between frames.

- Default parameters:
  - Time step = 100 cycles.
  - Flow Control = Strict (no packet loss allowed).
  - Single flit event with size of 32b.
  - Neurons evaluate every 10,000 cycles.
  - Weights=4b, States =16b, Outputs =1b.
  - Input/output queue depth = 16.
  - Number of NPEs per core = 4.

Start (k cycles)	0	100	200	300	400	500	600	700	800
End (k cycles)	11	124	218	321	424	518	618	721	824

EXPERIMENTAL RESULTS

- Output neurons' cumulative sum of the output events.

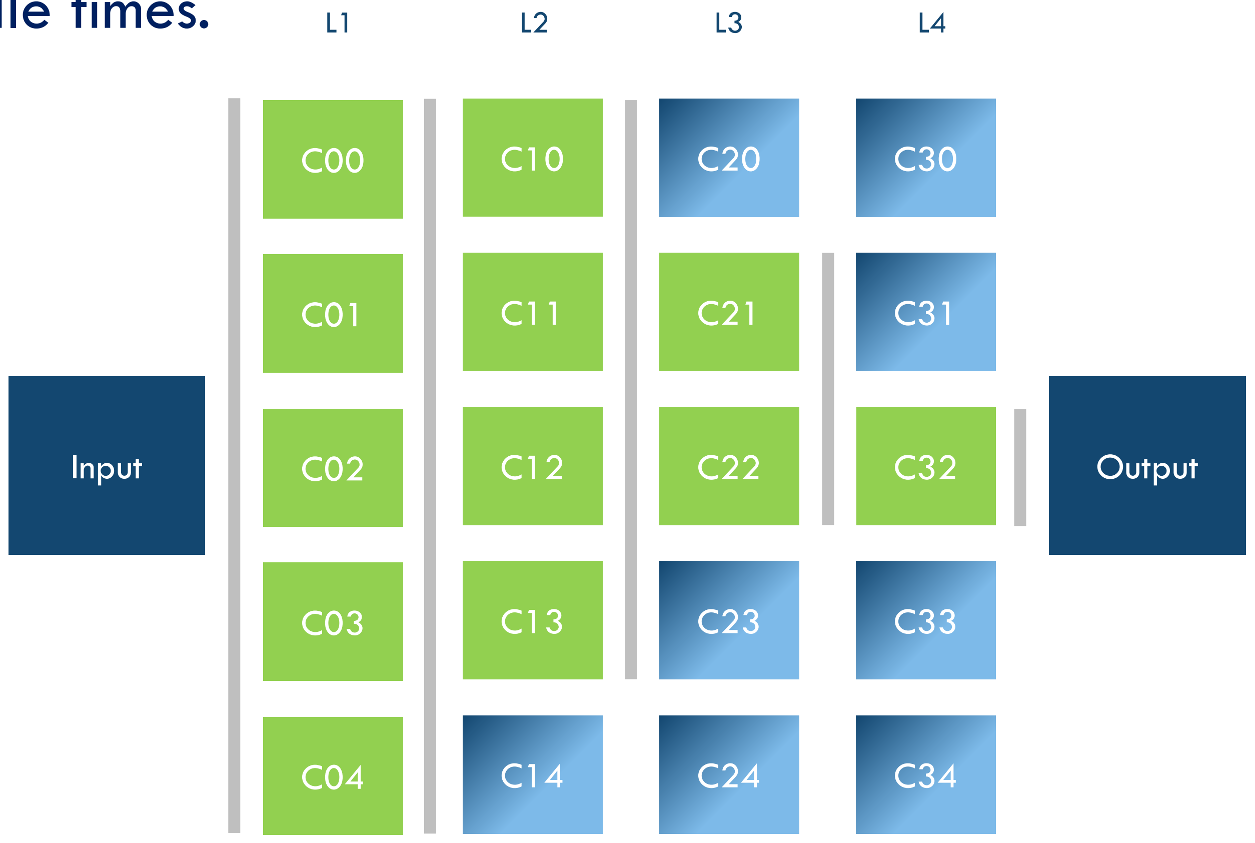


EXPERIMENTAL RESULTS

Packet loss: No packet is lost due to strict flow control.

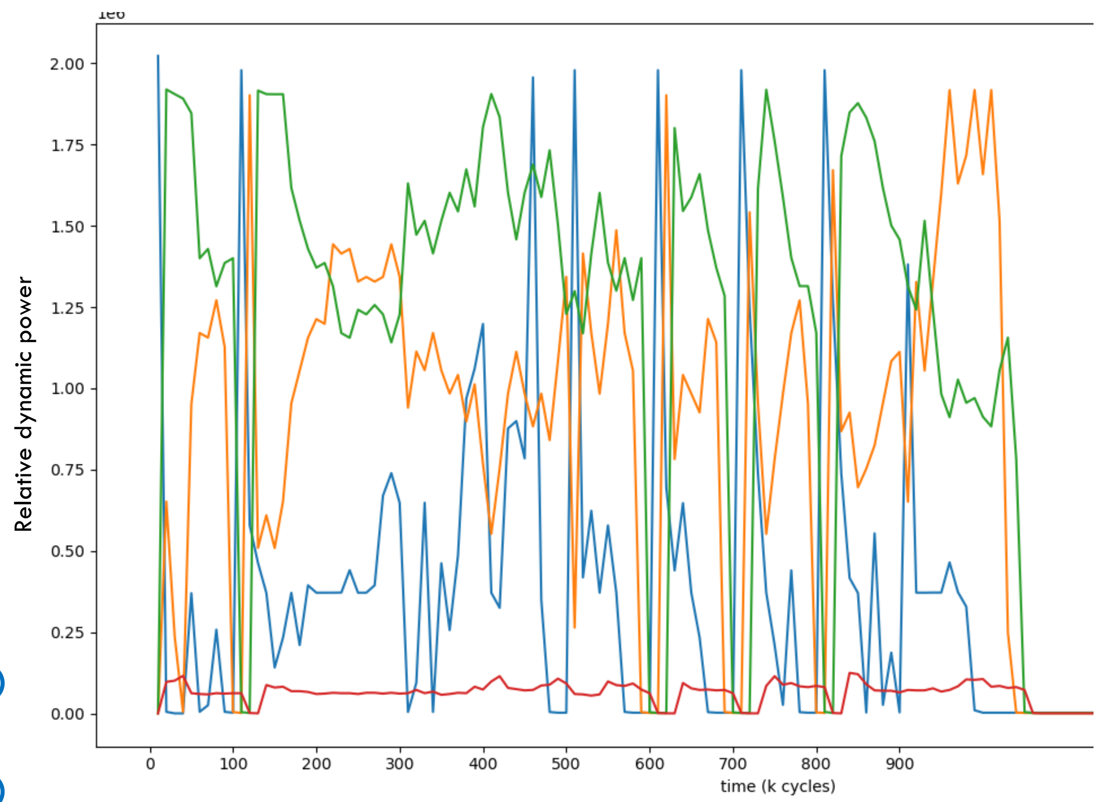
... but it results in push-back → Idle times.

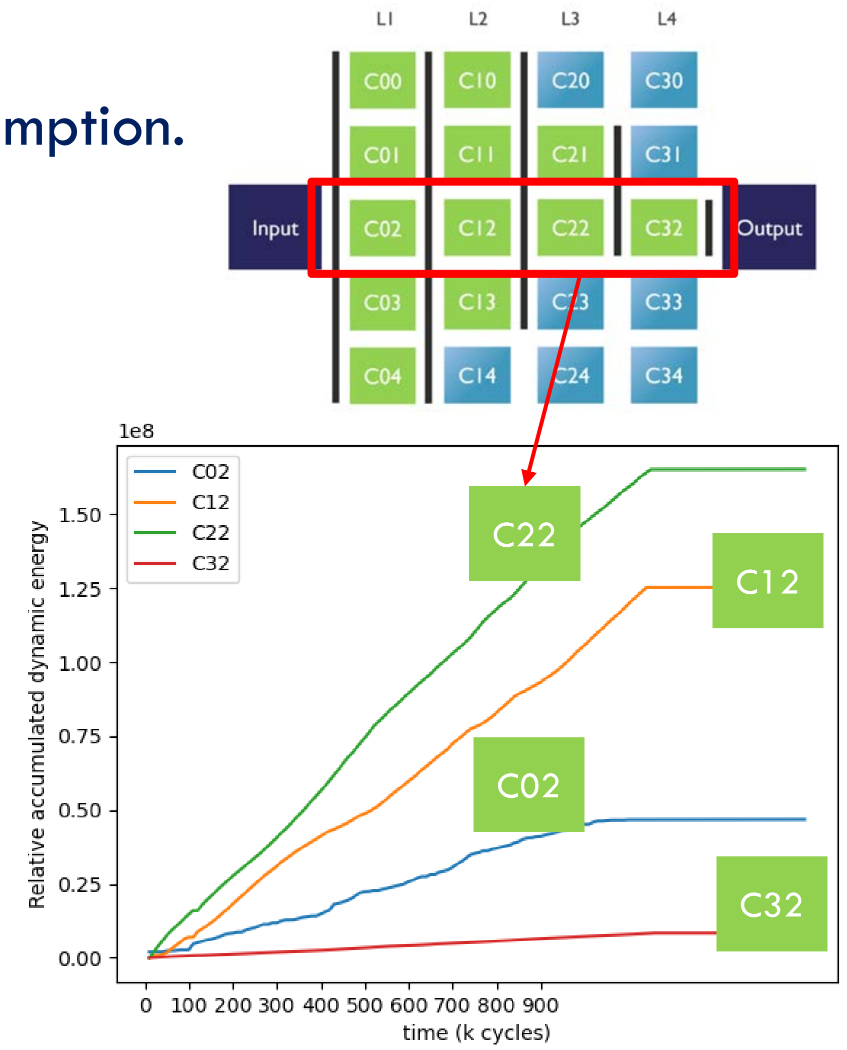
C00	packet loss:	0 ,	idle time(k cycles):	1229
C01	packet loss:	0 ,	idle time(k cycles):	1229
C02	packet loss:	0 ,	idle time(k cycles):	1229
C03	packet loss:	0 ,	idle time(k cycles):	1230
C04	packet loss:	0 ,	idle time(k cycles):	1234
C10	packet loss:	0 ,	idle time(k cycles):	854
C11	packet loss:	0 ,	idle time(k cycles):	855
C12	packet loss:	0 ,	idle time(k cycles):	855
C13	packet loss:	0 ,	idle time(k cycles):	854
C21	packet loss:	0 ,	idle time(k cycles):	584
C22	packet loss:	0 ,	idle time(k cycles):	585
C32	packet loss:	0 ,	idle time(k cycles):	1391



EXPERIMENTAL RESULTS

Cores' status: Relative dynamic energy consumption.

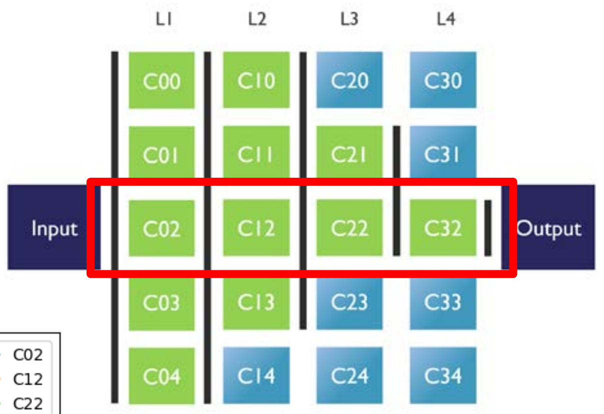


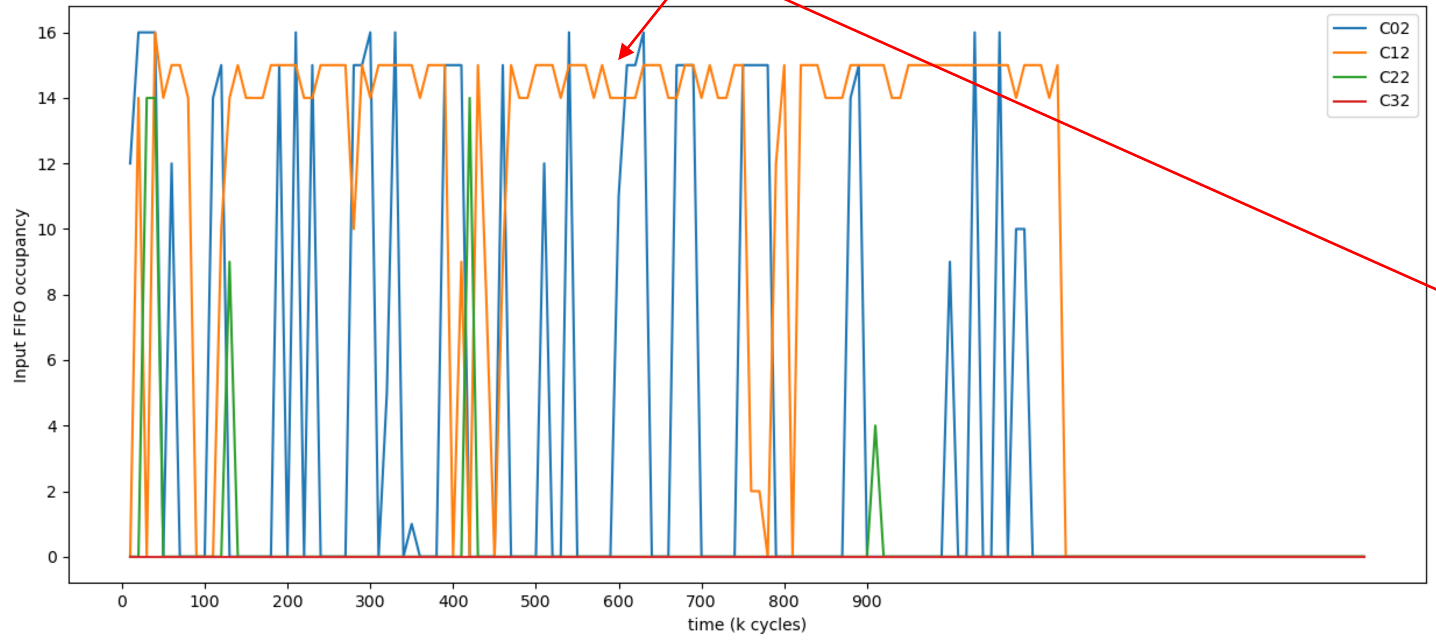


EXPERIMENTAL RESULTS

Cores' status: Input FIFO occupancy [FIFO depth=16]

It looks like L2 is a bottleneck: the FIFO for C12 is most of the times full

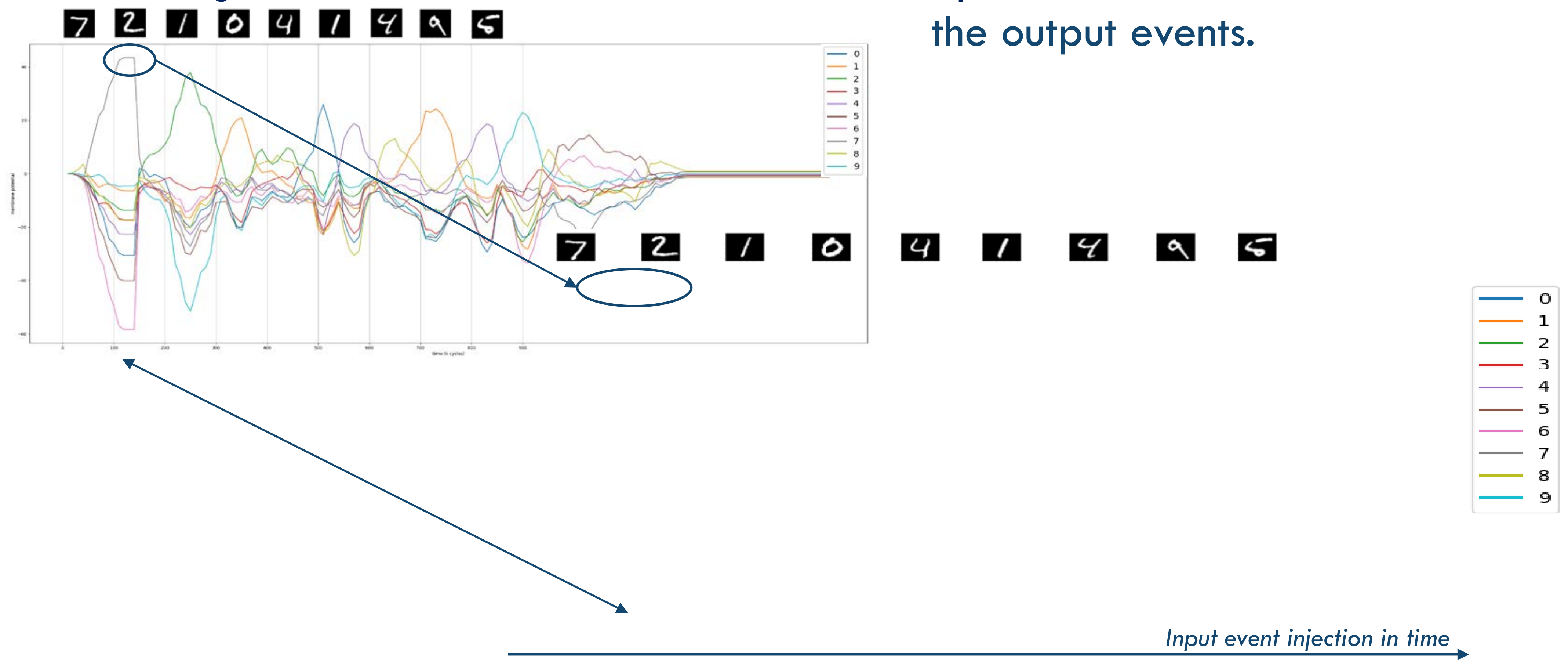




Average input FIFO occupancy:
C02 → 3.7
C12 → 9.7
C22 → 0.4
C32 → 0

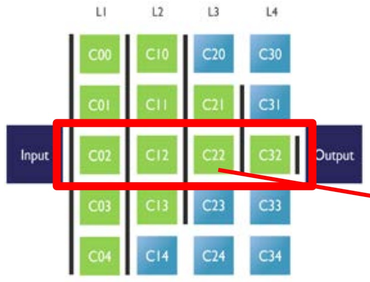
EXPERIMENTAL RESULTS

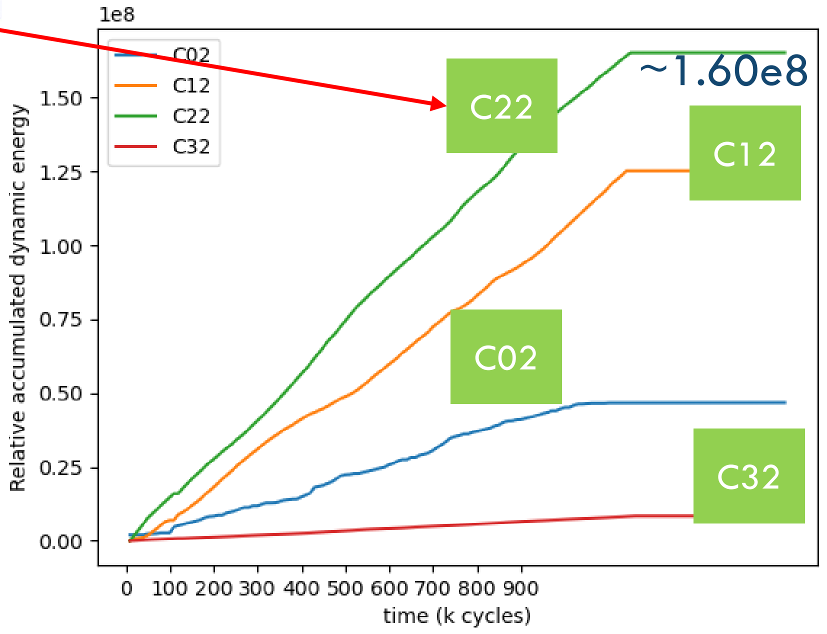
Increasing the number of NPEs from 4 to 16: Output neurons' cumulative sum of the output events.

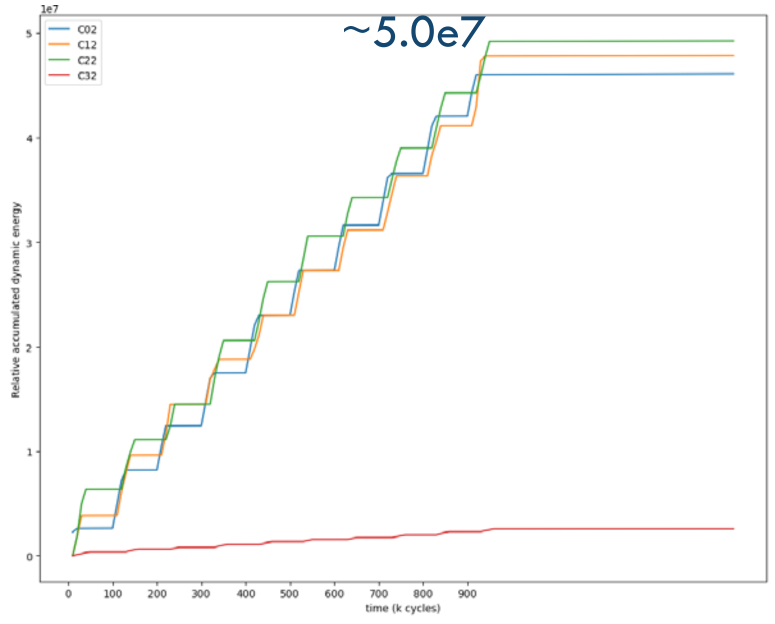


EXPERIMENTAL RESULTS

Increasing the number of NPEs from 4 to 16: Relative dynamic energy. consumption

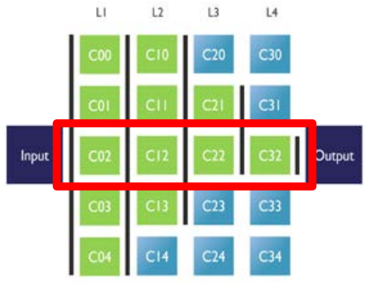




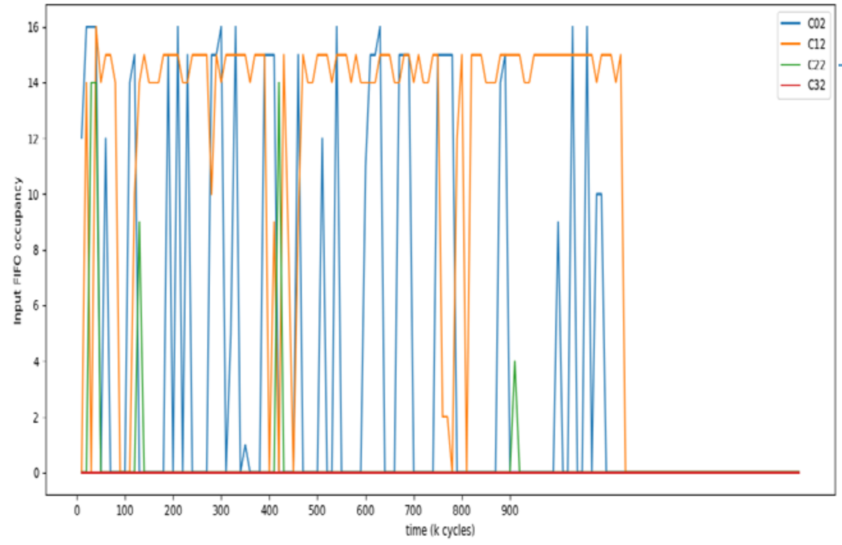


EXPERIMENTAL RESULTS

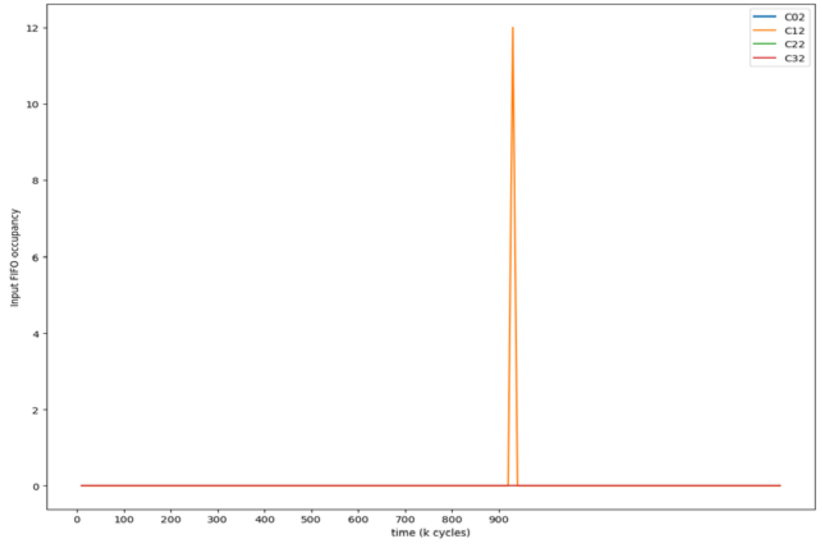
Increasing the number of NPEs from 4 to 16: Input FIFO occupancy [FIFO depth=16]



Average input FIFO occupancy:
C02 → 3.7
C12 → 9.7
C22 → 0.4
C32 → 0

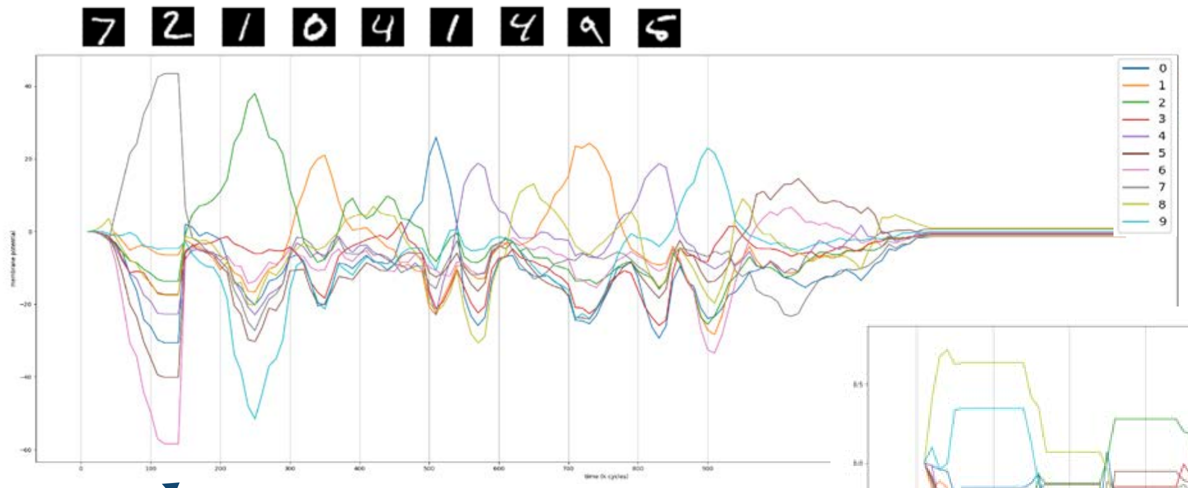


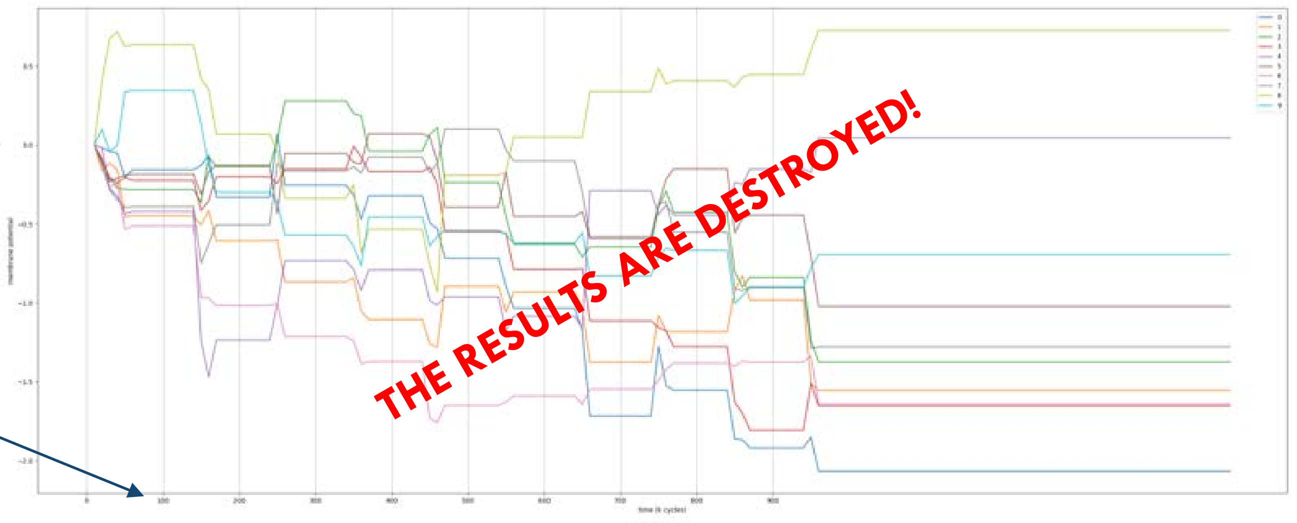
Average input FIFO occupancy:
C02 → 0
C12 → 0.1
C22 → 0
C32 → 0



EXPERIMENTAL RESULTS

Relaxed flow control (packet drop allowed), #NPE=16: Output neurons' cumulative sum of the output events.



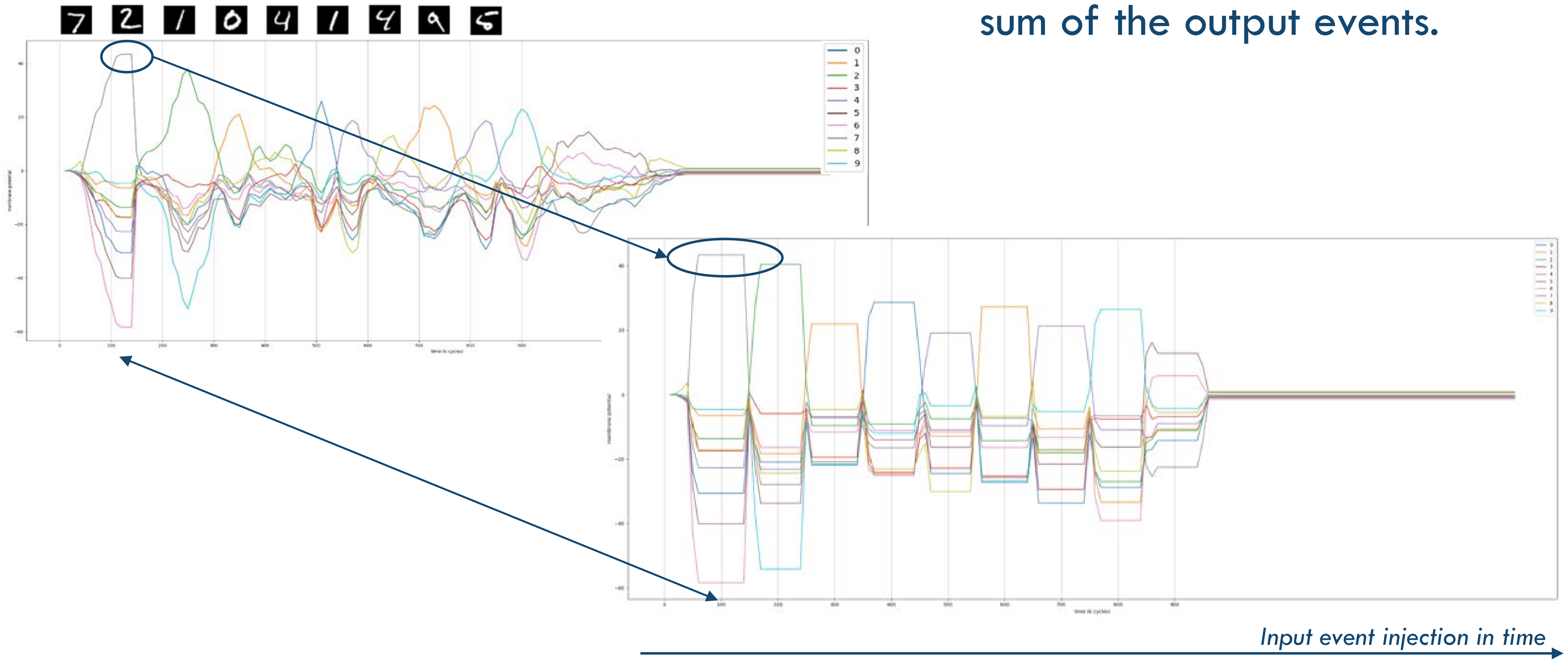


THE RESULTS ARE DESTROYED!

Input event injection in time

EXPERIMENTAL RESULTS

Increased the input FIFO depth from 16 to 1024: Output neurons' cumulative sum of the output events.



CONCLUSIONS

- Simulation framework for energy and latency estimation in multi-core neuromorphic architectures.
- Highly parametrizable.
- Allow wide architecture and resource allocation exploration.

- Next steps
  - Improve accuracy integrating characterization data from actual designs.
  - Improve runtime to handle larger NN models.
  - Integrate the simulator with a resource mapper to get optimal mappings.

ANDANTE

THANK YOU

Next presentation: *Kay Bierzynski,* “Consideration of the real world in use case-based benchmarking”