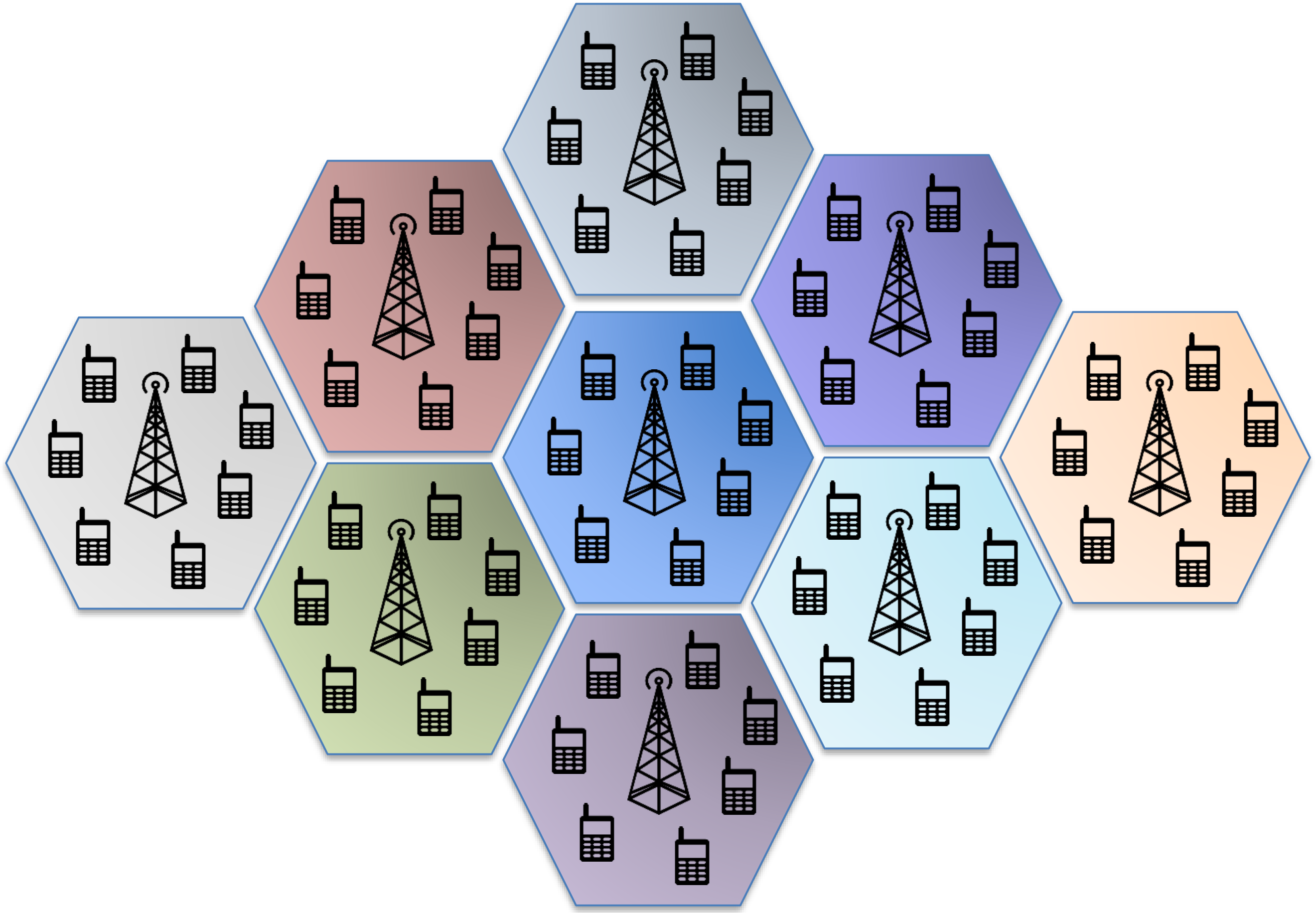
An Application Specific Vector Processor for CNN-based Massive MIMO Positioning

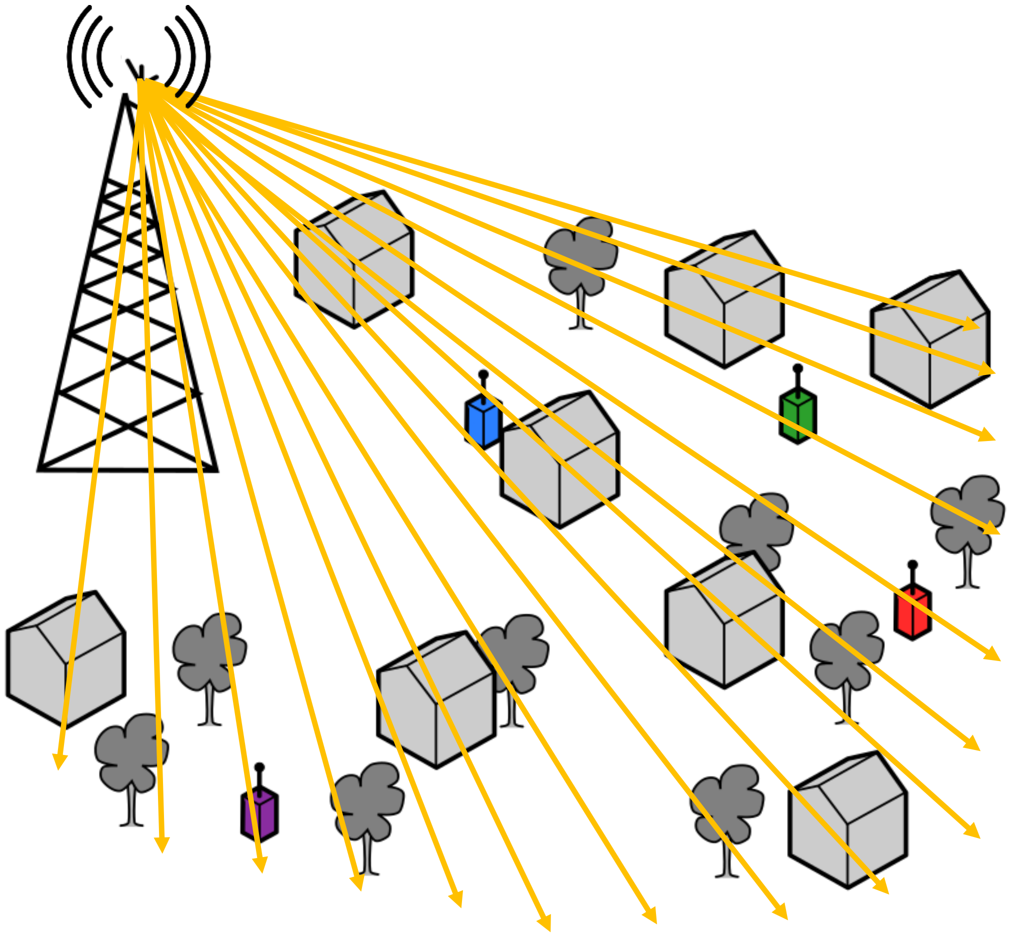
MOHAMMAD ATTARI, JESÚS RODRÍGUEZ SÁNCHEZ, LIANG LIU, AND STEFFEN MALKOWSKY
LUND UNIVERSITY, ASIP UNIVERSITY DAY, 2021

Cellular Wireless Networks

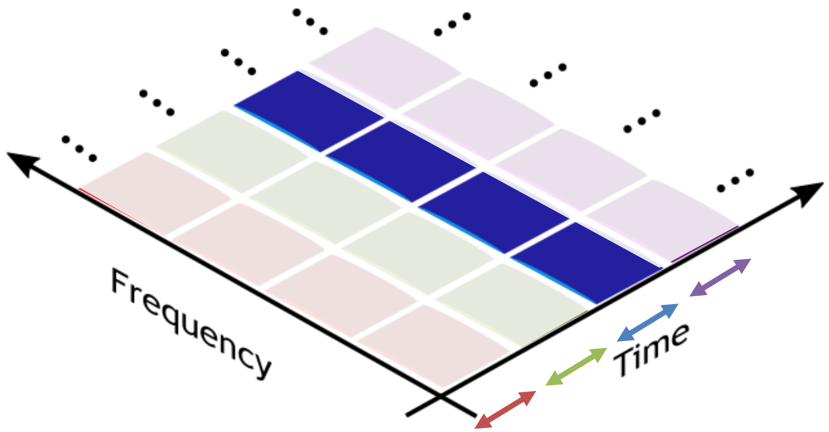


- Cells
- Base station (BS)
- User equipment (UE)
- Cellular network
- 5G & beyond
  - Communication
  - Positioning
  - Sensing

Traditional Time Division Multiple Access (TDMA)

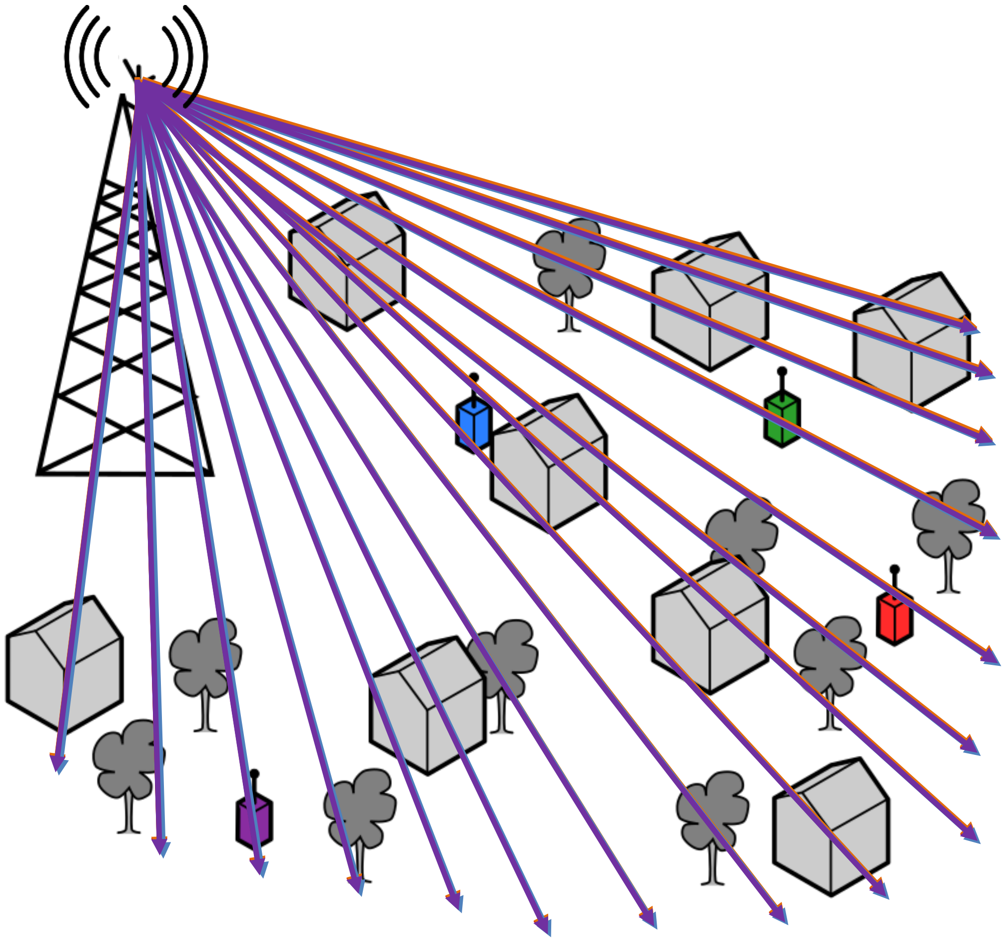


- Only one antenna
- Time multiplexing

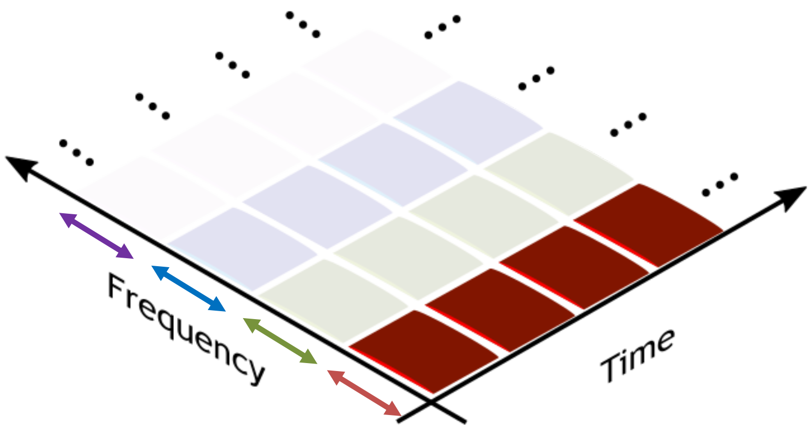


LUND UNIVERSITY

Traditional Frequency Division Multiple Access (FDMA)

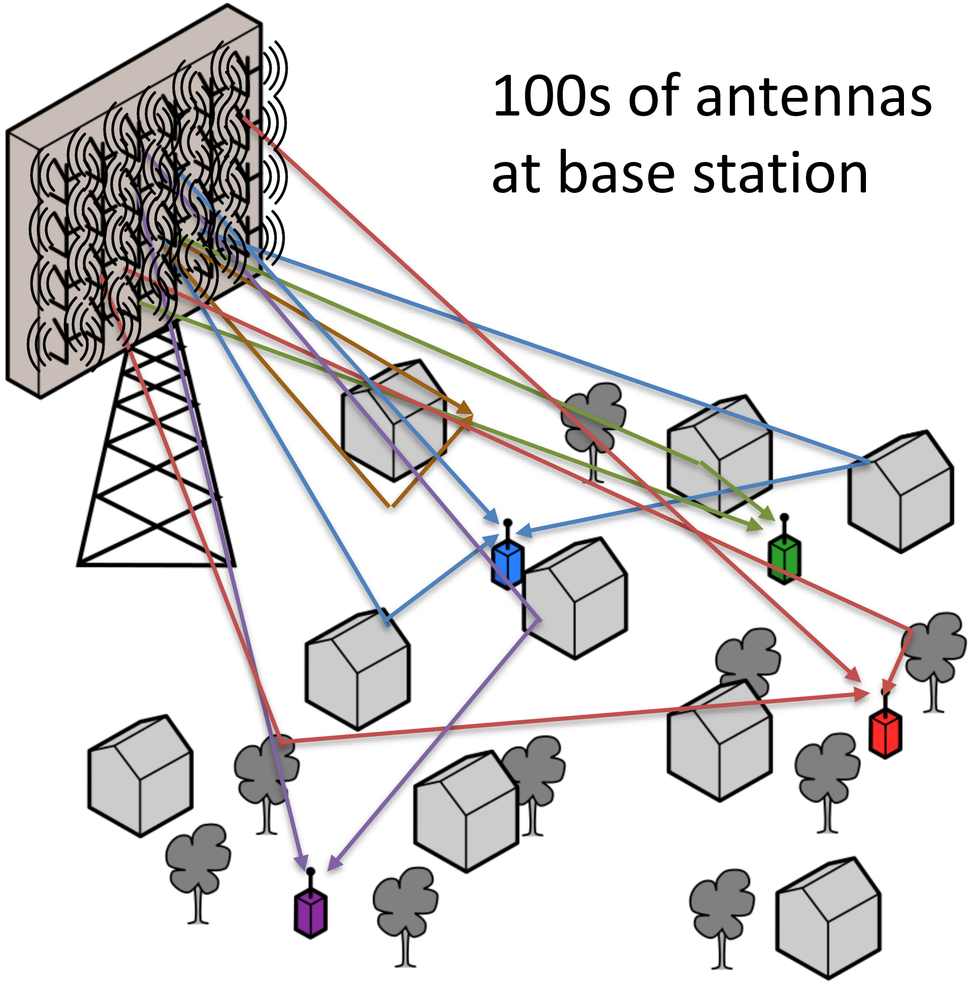


➢ Frequency multiplexing



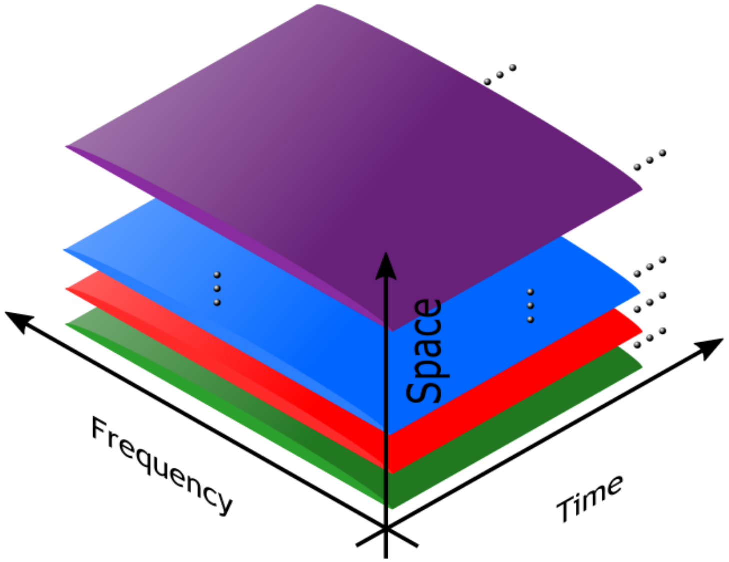
LUND UNIVERSITY

Massive MIMO Spatial Division Multiple Access (SDMA)

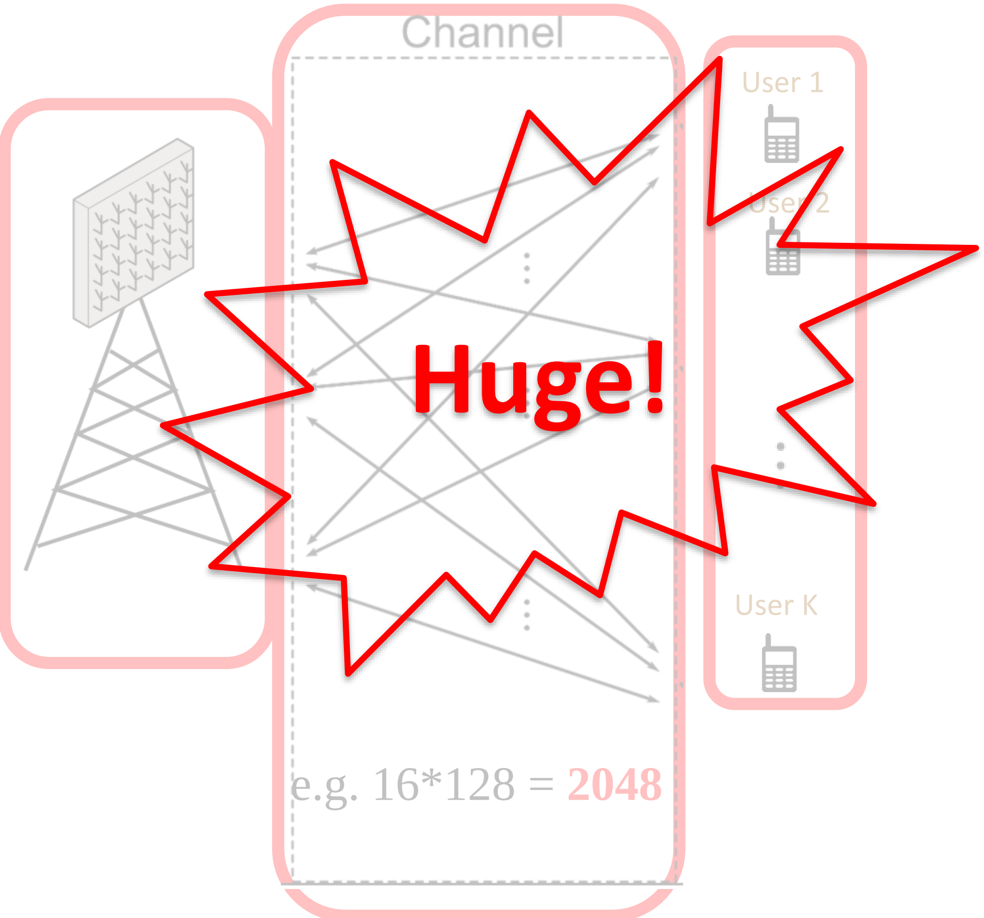


100s of antennas
at base station

- ➢ Spatial multiplexing
- ➢ High spectral efficiency



Massive MIMO: System Perspective



UEs (e.g. K=16)

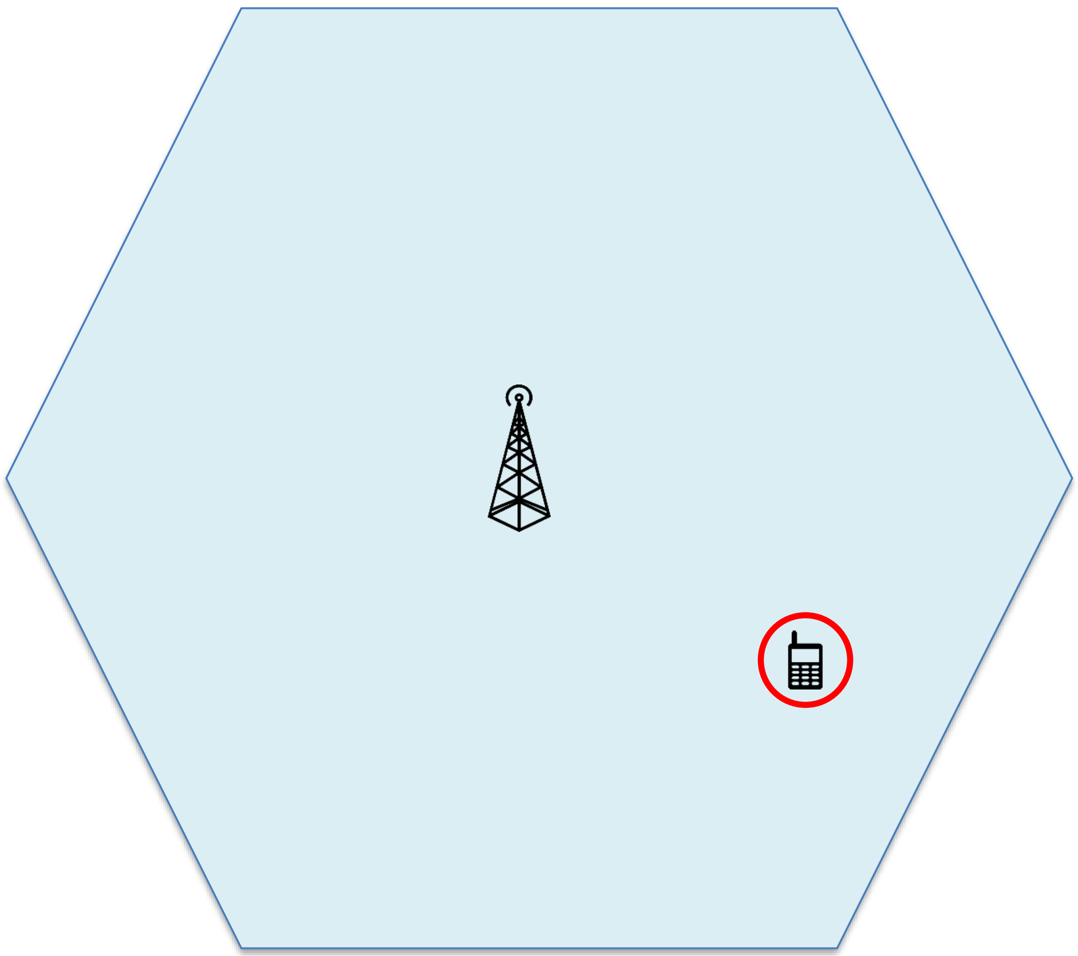
BS antennas (e.g. M=128)

Wireless channel (H)

$$\begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,M} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,M} \\ \vdots & \vdots & \ddots & \cdots \\ b_{K,1} & b_{K,2} & \cdots & b_{K,M} \end{bmatrix}_{KxM}$$

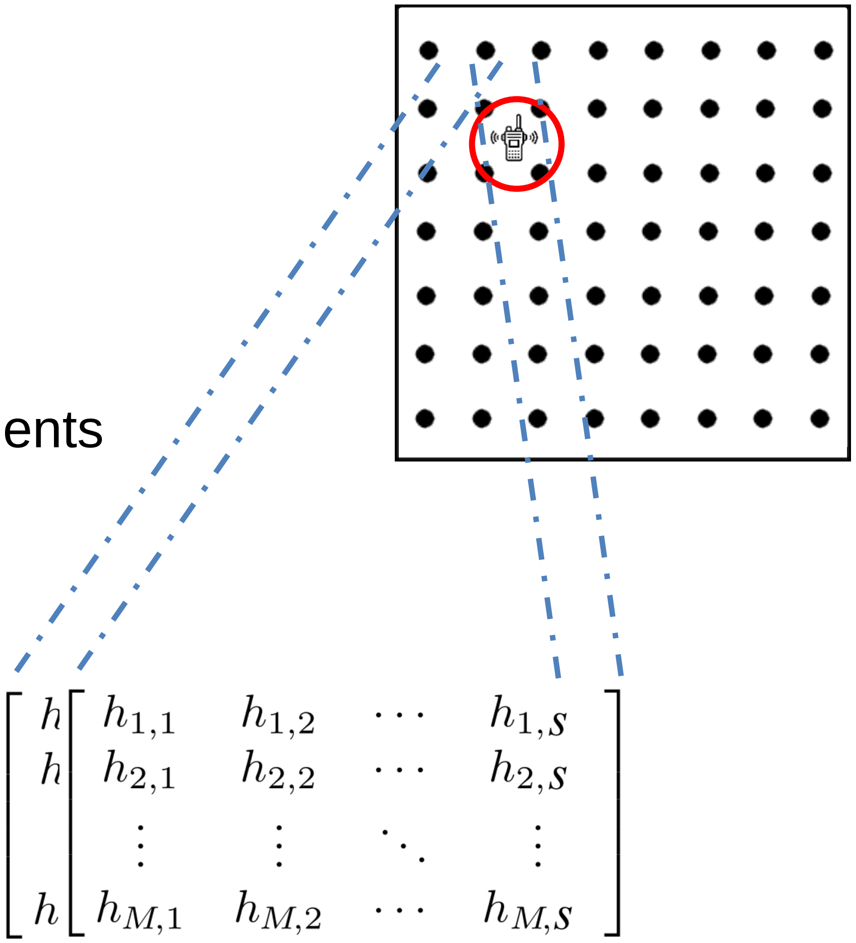
User Positioning

- **Find user's location**
- **Methods**
  - Proximity based
  - Angle based
  - Range based
  - Fingerprint based



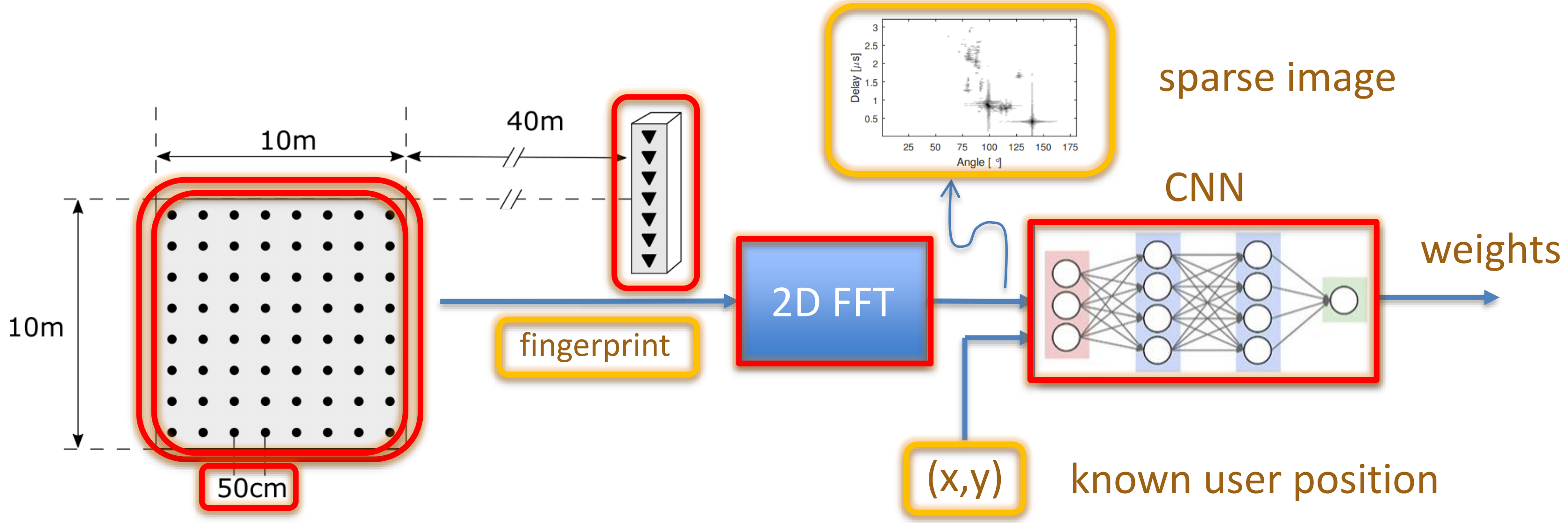
Fingerprint-based Positioning

- **Find terminals' position in the area**
  - Set of training samples (fingerprints)
    - Known positions measurements
  - Online measurements
- **Drawbacks**
  - Computationally heavy
  - Performance degradation in dynamic environments



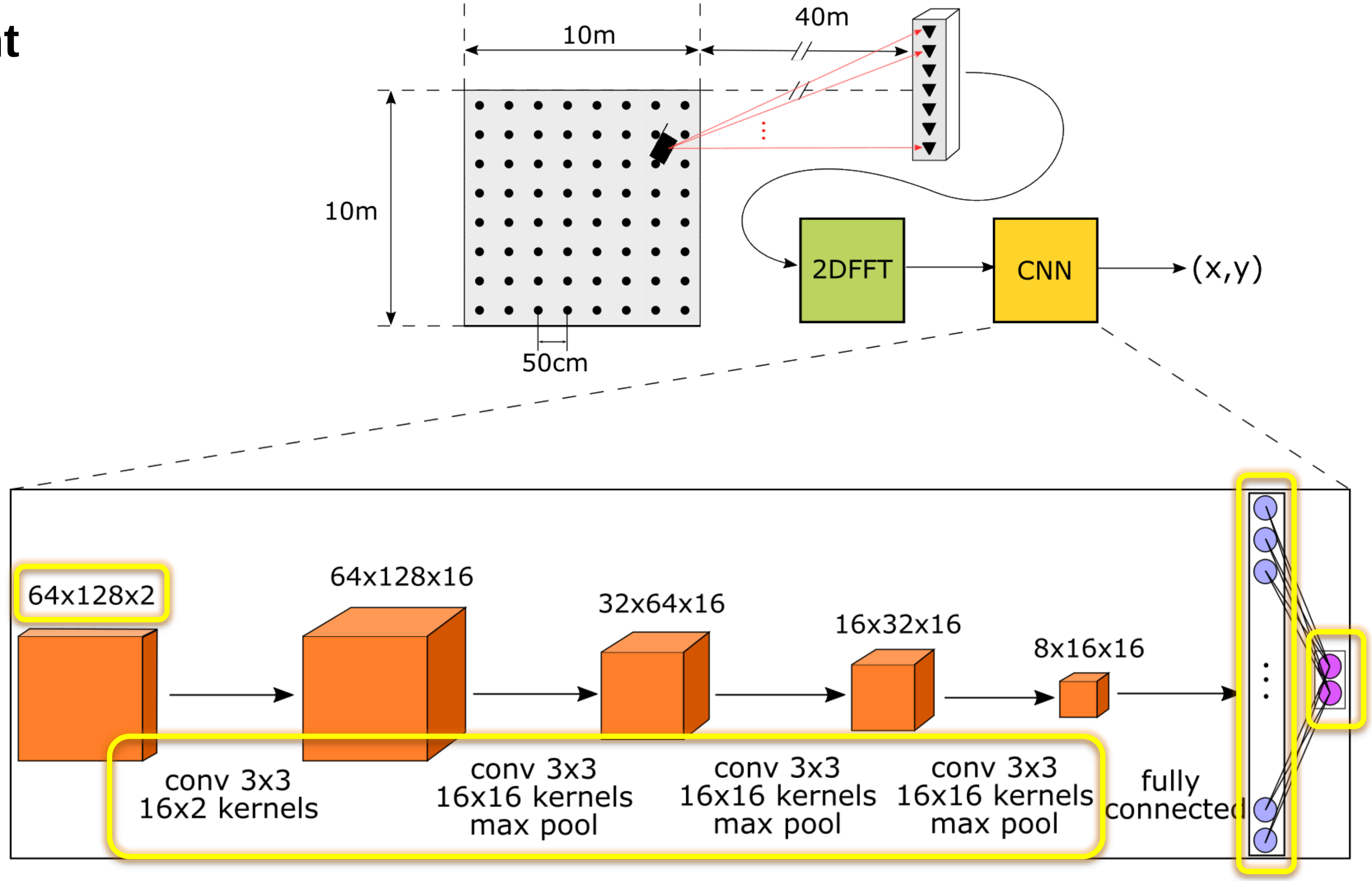
Fingerprint-based Positioning: Training

- **System setup**
  - Area: 10m x 10m
  - BS 40m away (antennas=128, subcarriers=64)
  - 441 locations, 50cm apart
  - Channel data from the COST 2100 channel model (fingerprint)
  - 2D FFT (from **antenna-frequency** to **angular-delay** domain)
  - Sparse image and (x,y) fed to CNN (obtain weights)

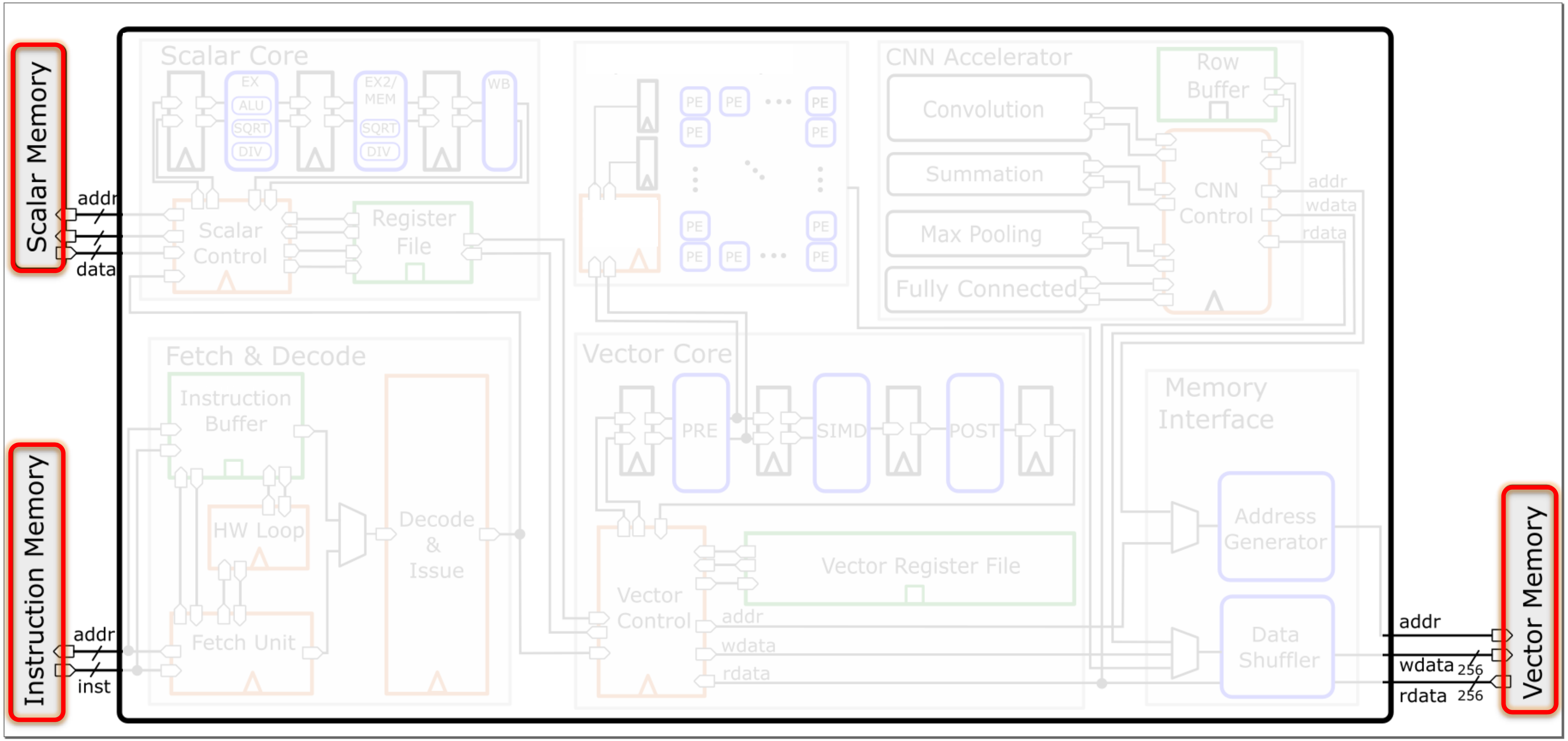


Fingerprint-based Positioning: Inference

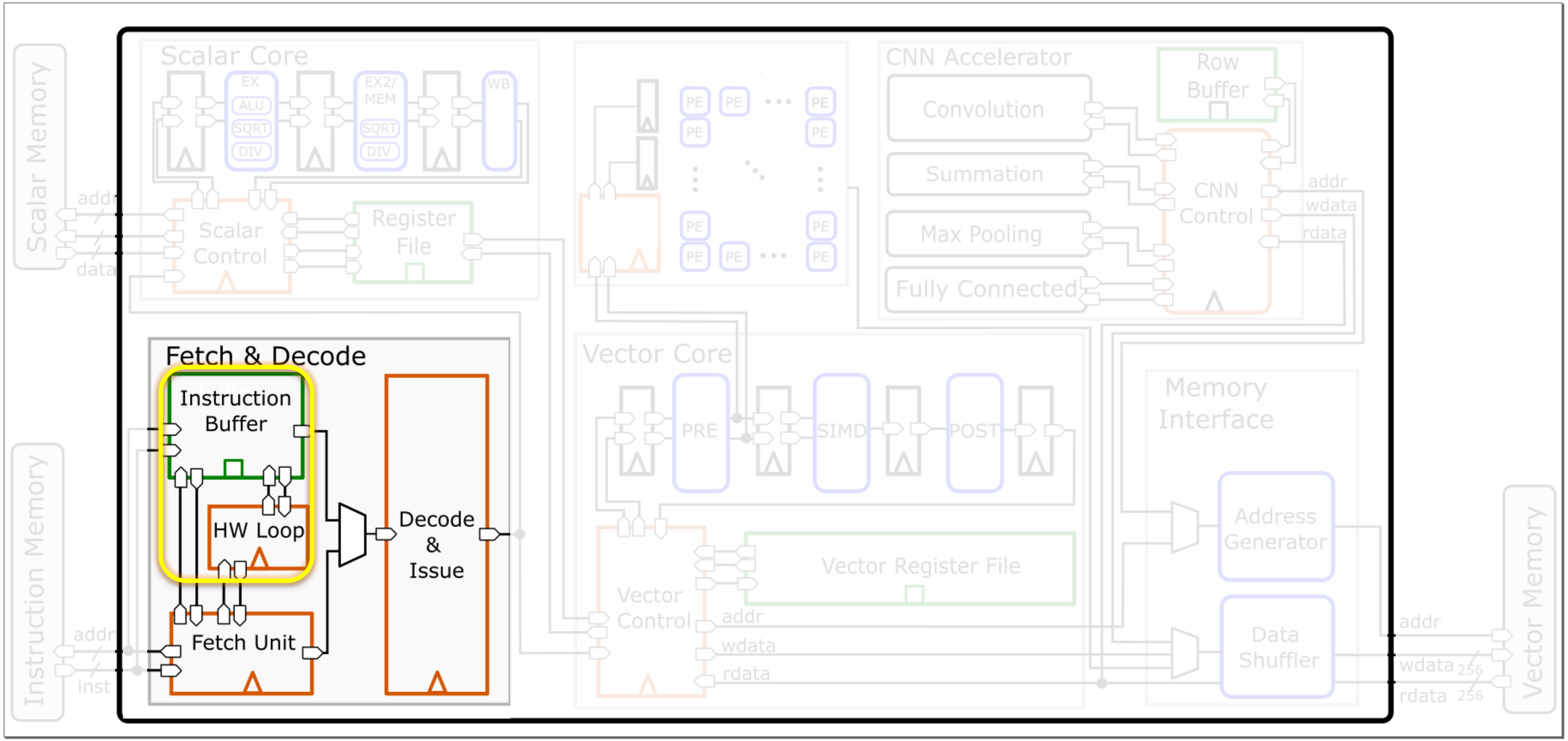
- **System setup**
- **Online measurement**
- **FFT & CNN**
  - Input: 64×128×2
  - 4 CONV layers
  - 3x3 kernels
  - 1 FC layer



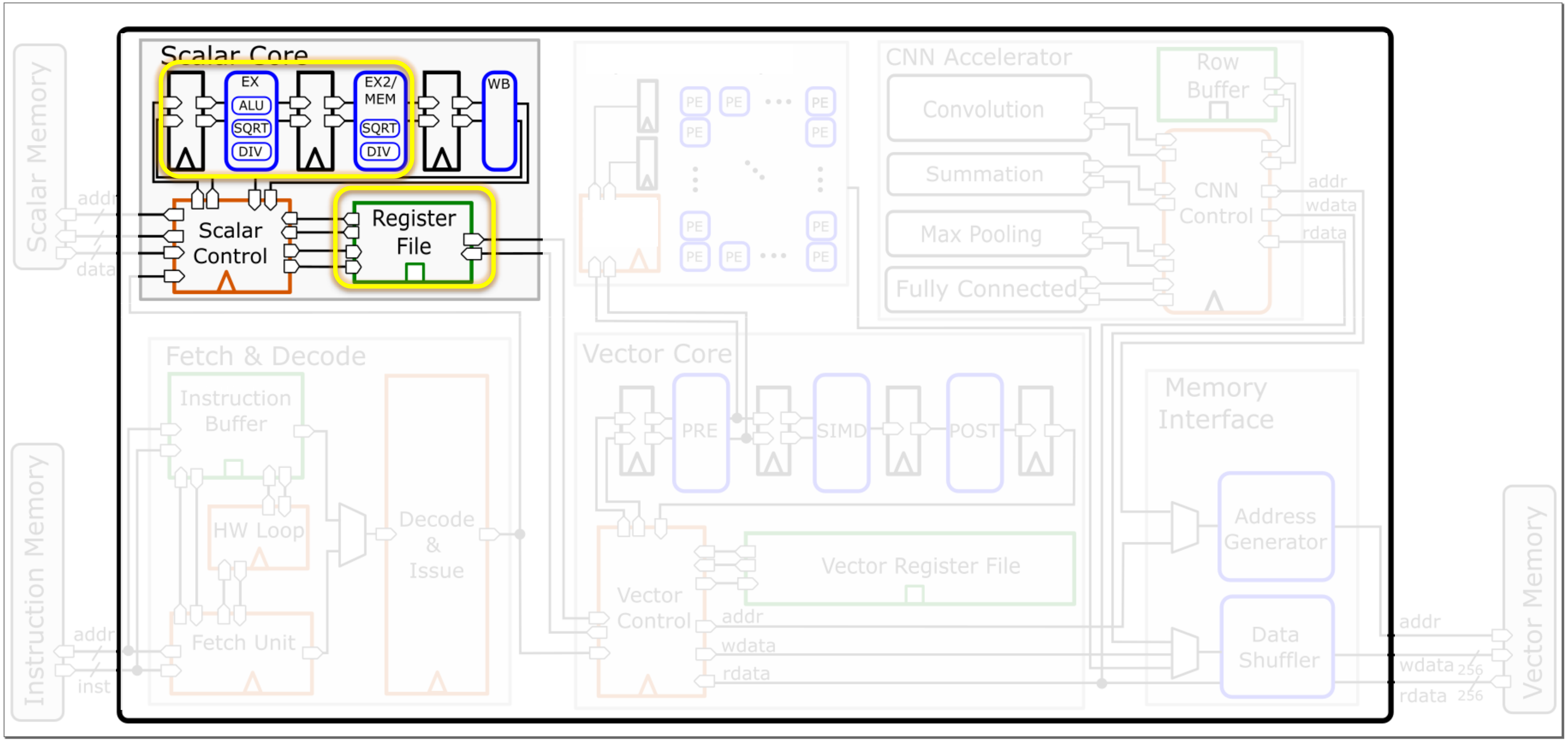
Bird's-eye View of the Processor



Fetch & Decode

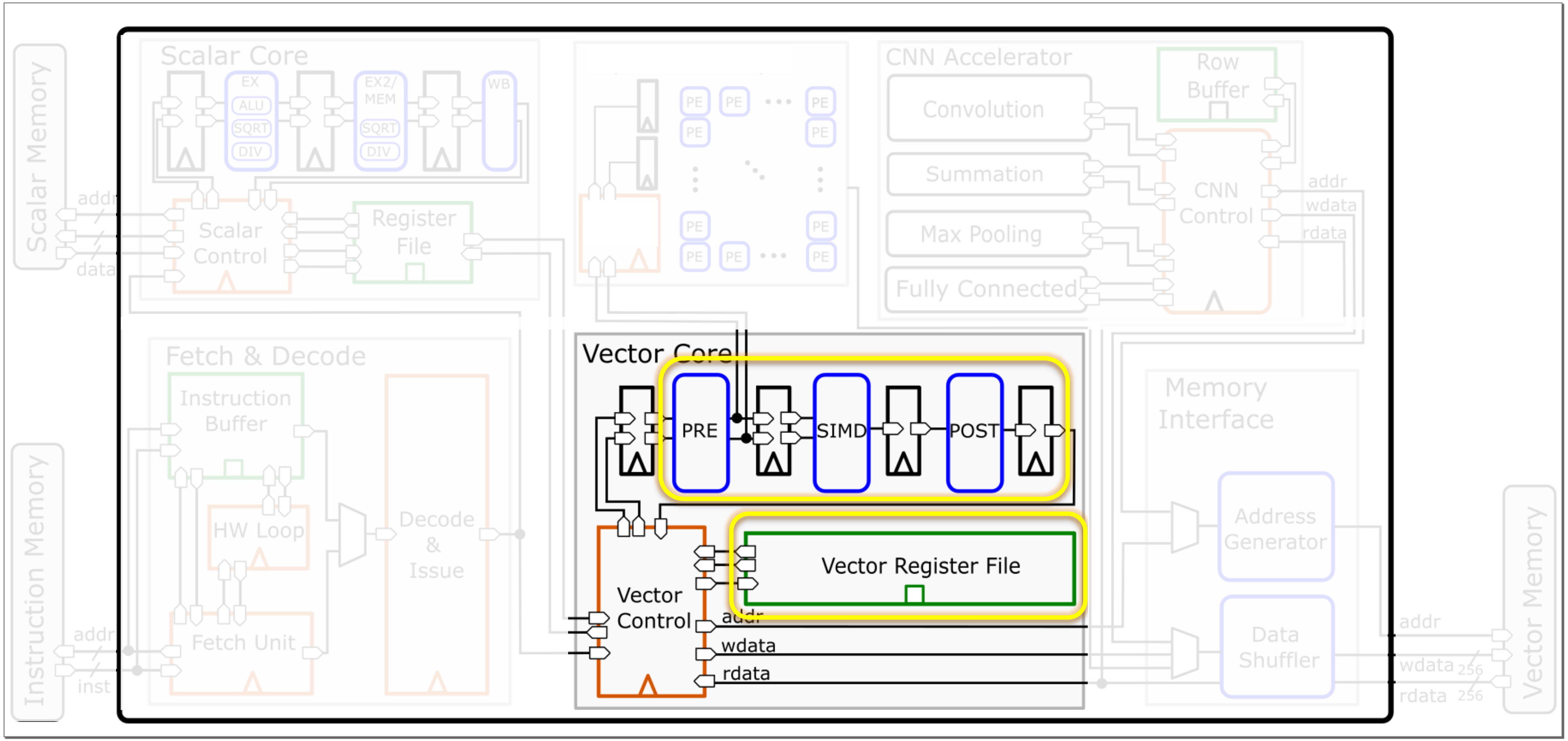


Scalar Core



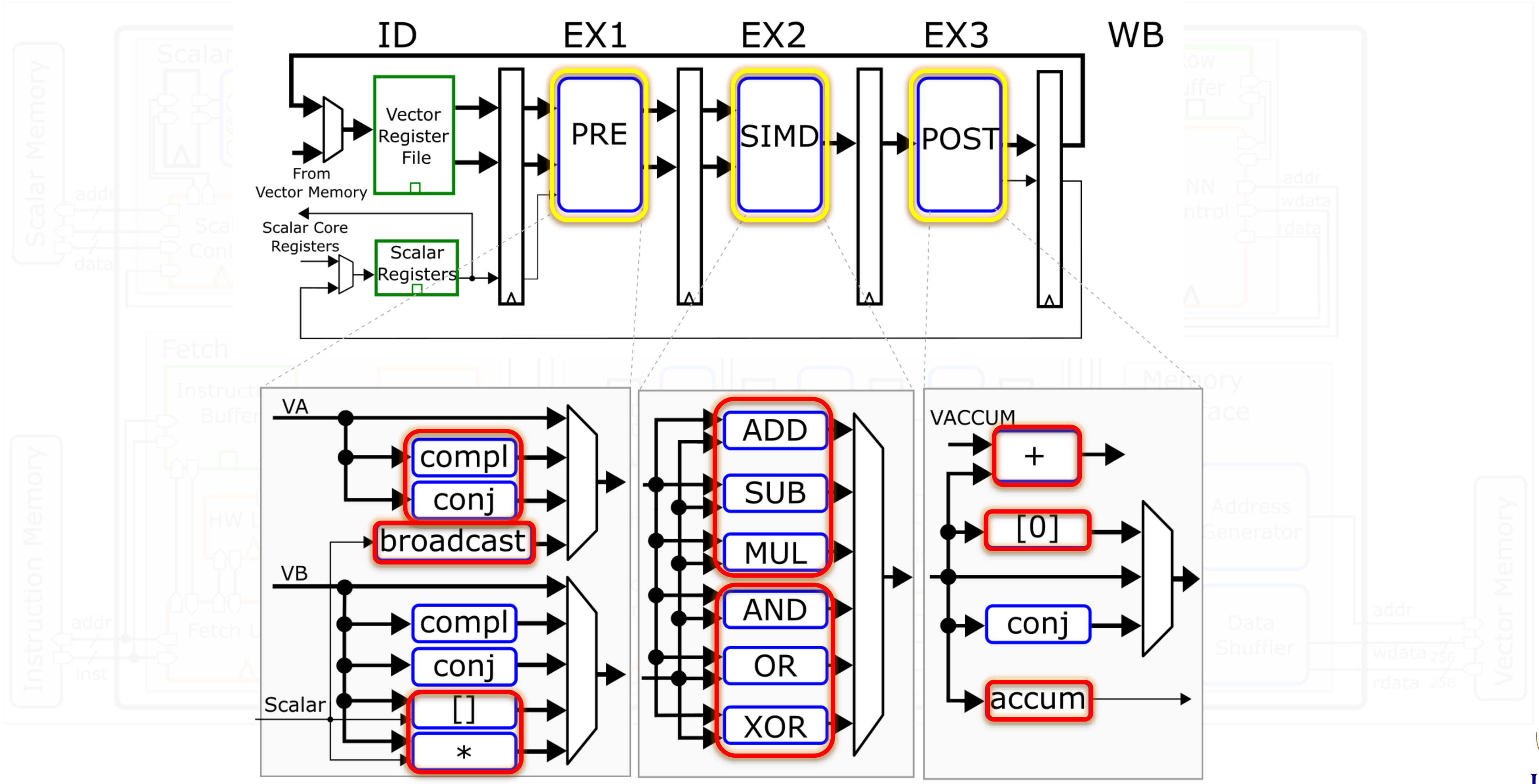
Scalar Core
EX
ALU
SQRT
DIV
EX2/
MEM
SQRT
DIV
WB
Scalar
Control
Register
File
Scalar Memory
addr
data
CNN Accelerator
Convolution
Summation
Max Pooling
Fully Connected
Row
Buffer
CNN
Control
addr
wdata
rdata
PE
Fetch & Decode
Instruction
Buffer
HW Loop
Fetch Unit
Decode
&
Issue
Instruction Memory
addr
inst
Vector Core
PRE
SIMD
POST
Vector Register File
Vector
Control
addr
wdata
rdata
Memory
Interface
Address
Generator
Data
Shuffler
addr
wdata 256
rdata 256
Vector Memory

Vector Core



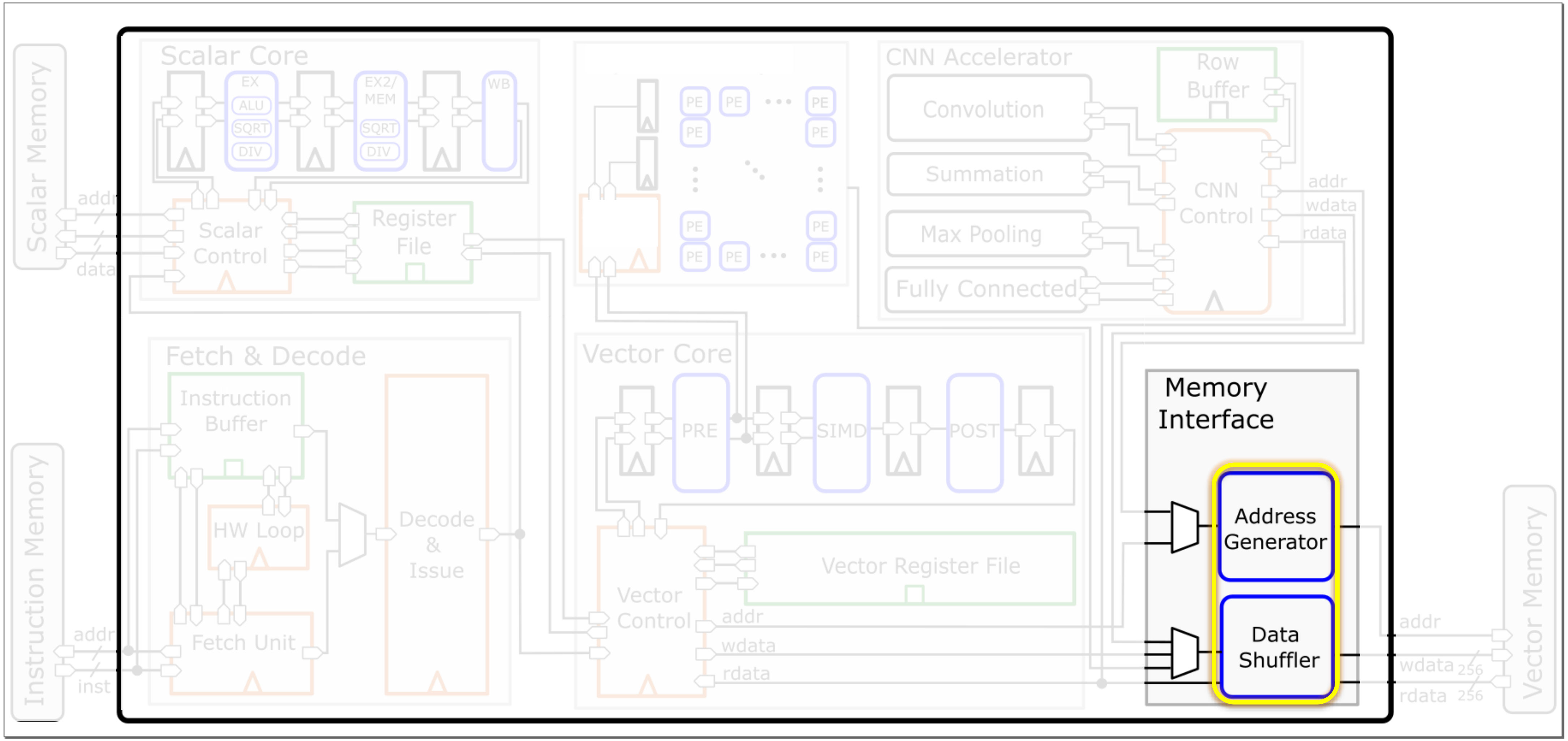
LUND UNIVERSITY

Vector Core

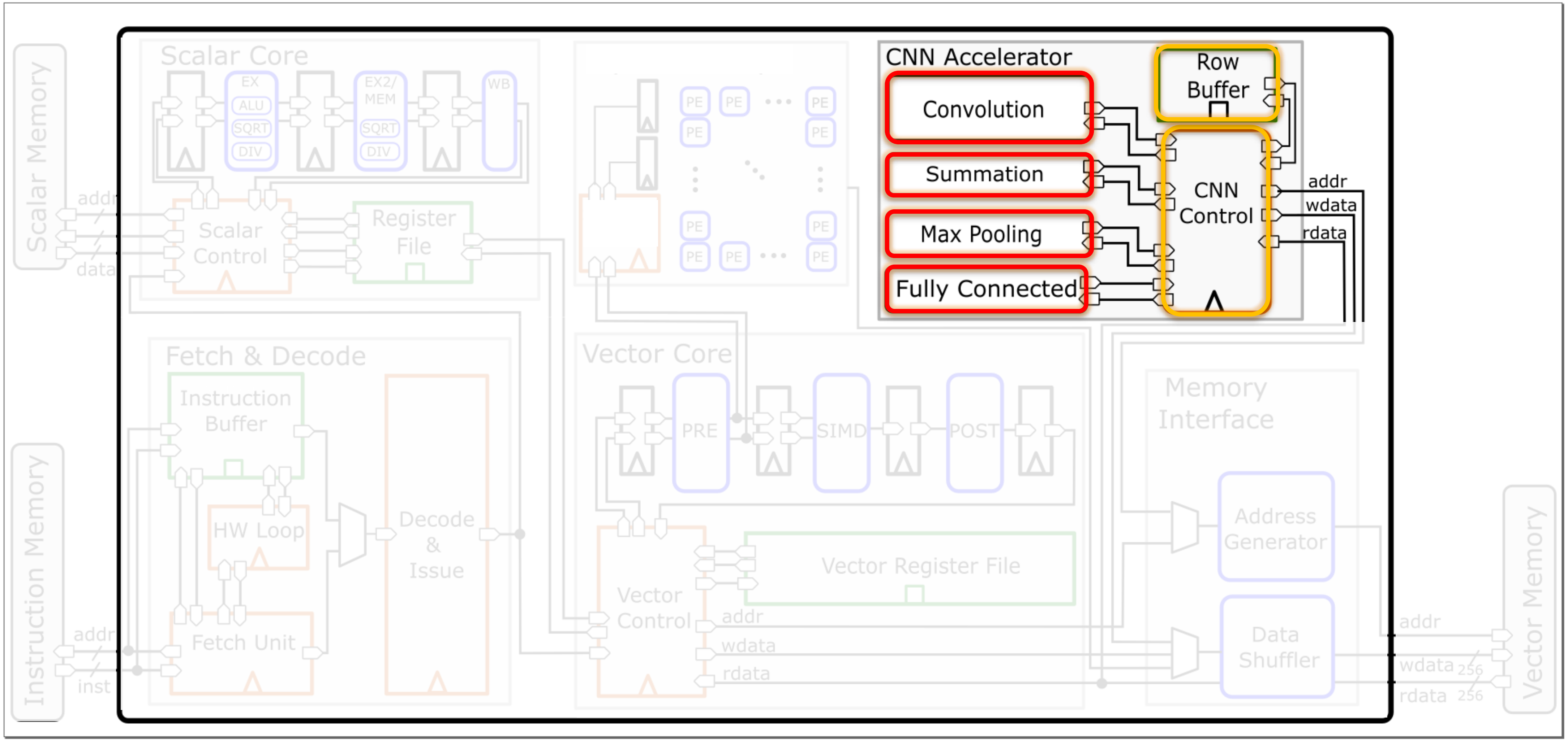


ID
EX1
EX2
EX3
WB
Vector Register File
From Vector Memory
Scalar Core Registers
Scalar Registers
PRE
SIMD
POST
VA
compl
conj
broadcast
VB
compl
conj
Scalar
[]
*
ADD
SUB
MUL
AND
OR
XOR
VACCUM
+
[0]
conj
accum

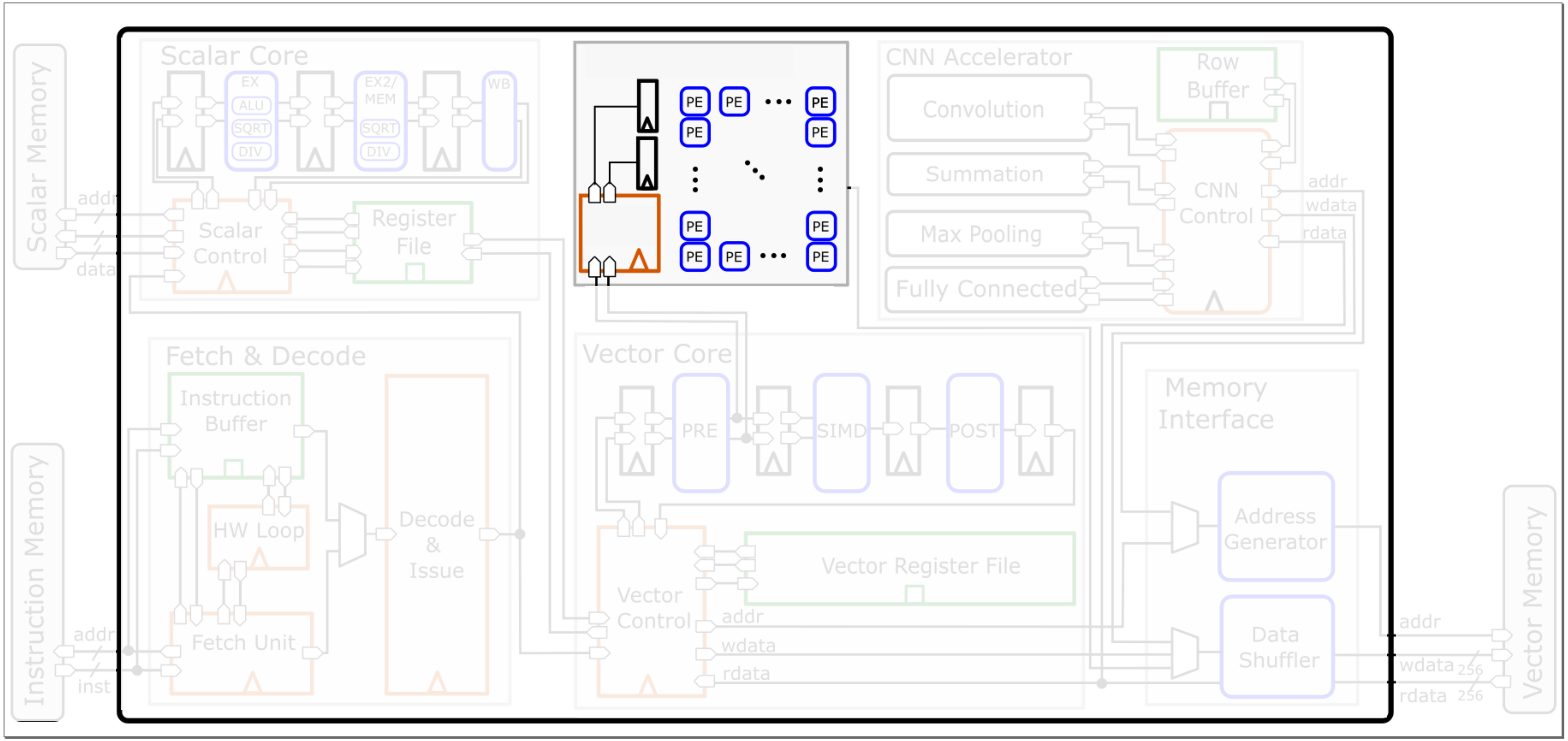
Parallel Memory



CNN Accelerator



Accelerator



Previous Work: In-pipeline Systolic Array

nML + PDG

Lund University

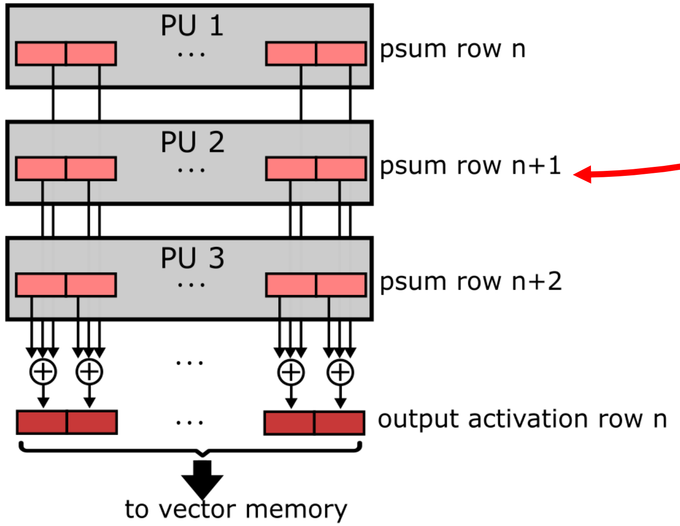
Systolic Array with IO Interfaces

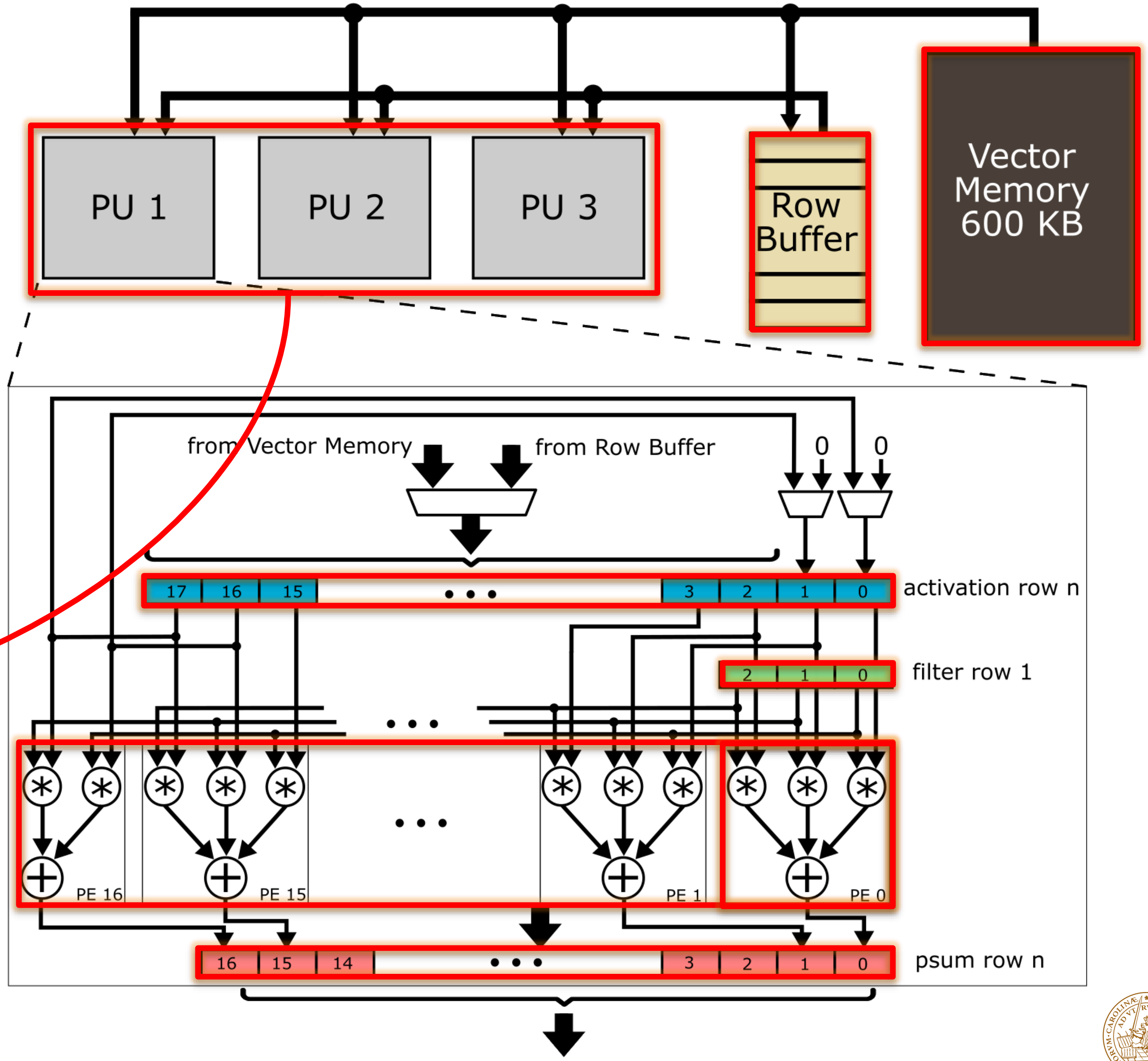
- **Developed inside ASIP Designer**
  - No RTL integration needed
- **~300 lines of code**
- **Compiler-friendly intrinsics**

```
io_interface systolic_handler()
{
    reg systolic_counter <uint5>;
    hw_init systolic_counter = 0;

    reg systolic_start <uint1>;

    reg reg_systolic_row_0 <v16w32>;
    reg reg_systolic_row_1 <v16w32>;
    reg reg_systolic_row_2 <v16w32>;
    reg reg_systolic_row_3 <v16w32>;

    // ...
    if (systolic_counter < 8)
    {
        reg_systolic_accum_00 = systolic_mac(reg_systolic_row_0[0], reg_systolic_col_0[0], reg_systolic_accum_00);
    }
    // ...
    if (systolic_counter == 16)
    {
        vm_vaddr_w0 = vAddr_r0 = systolic_out_address;
        VM[vm_vaddr_w0] = vm_write0 =
            reg_systolic_accum_07::reg_systolic_accum_06::reg_systolic_accum_05::reg_systolic_accum_04::
            reg_systolic_accum_03::reg_systolic_accum_02::reg_systolic_accum_01::reg_systolic_accum_00;
    }
    // ...
}
```

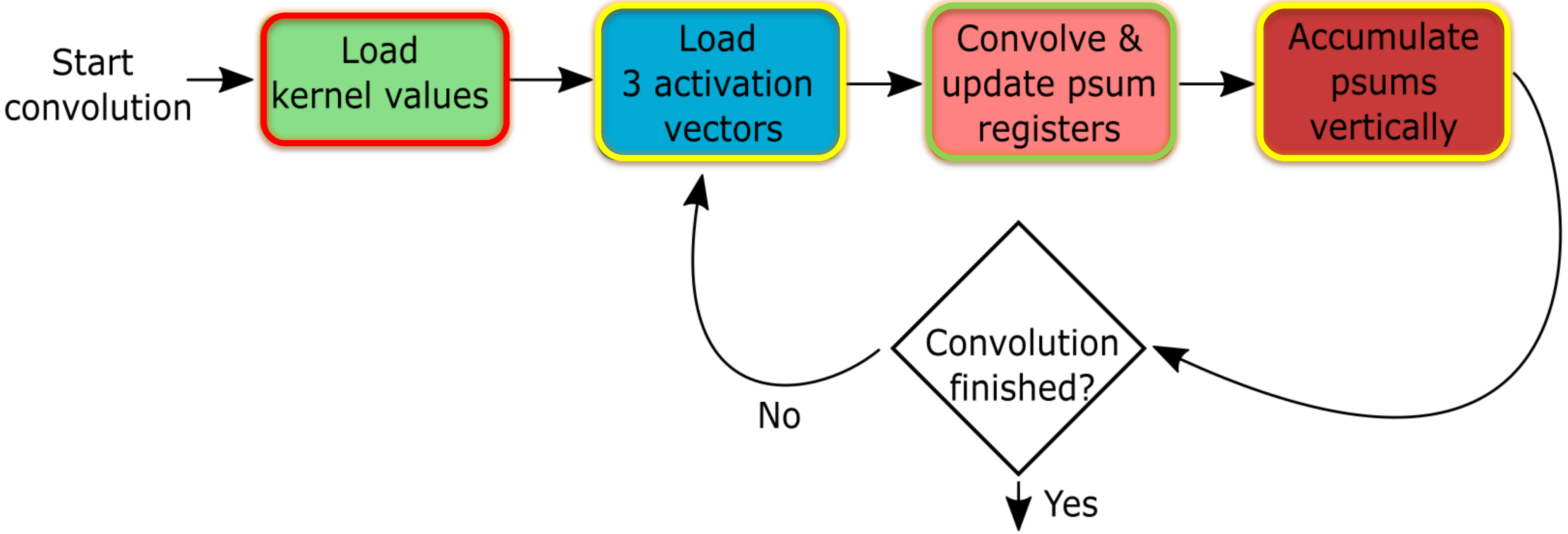
Convolution Engine

- **Vector memory (600 KB)**
- **Row buffer**
- **1D primitive units (PU)**
  - Activation row register
  - Filter row register
  - 17 processing engines (PE)
  - Partial sum (psum) row register





CNN Dataflow



LUND UNIVERSITY

CNN Configuration

Set up 1st layer

```
setup_cnn(
    CNN_NUM_FILTER_CHANNELS_1ST_LAYER,
    CNN_NUM_CONVOLUTIONS_1ST_LAYER,
    CNN_NUM_TOTAL_ROW_SECTIONS_1ST_LAYER,
    CNN_NUM_ROW_SECTIONS_2ND_LAYER,
    CNN_NUM_ROWS_2ND_LAYER,
    CNN_BUFFER_END_ADDR_2ND_LAYER
);
```

Set up 2nd layer

```
setup_cnn(
    CNN_NUM_FILTER_CHANNELS_2ND_LAYER,
    CNN_NUM_CONVOLUTIONS_2ND_LAYER,
    CNN_NUM_TOTAL_ROW_SECTIONS_2ND_LAYER,
    CNN_NUM_ROW_SECTIONS_3RD_LAYER,
    CNN_NUM_ROWS_3RD_LAYER,
    CNN_BUFFER_END_ADDR_3RD_LAYER
);
```

Set up 3rd layer

```
setup_cnn(
    CNN_NUM_FILTER_CHANNELS_3RD_LAYER,
    CNN_NUM_CONVOLUTIONS_3RD_LAYER,
    CNN_NUM_TOTAL_ROW_SECTIONS_3RD_LAYER,
    CNN_NUM_ROW_SECTIONS_4TH_LAYER,
    CNN_NUM_ROWS_4TH_LAYER,
    CNN_BUFFER_END_ADDR_4TH_LAYER
);
```

Set up 4th layer

```
setup_cnn(
    CNN_NUM_FILTER_CHANNELS_4TH_LAYER,
    CNN_NUM_CONVOLUTIONS_4TH_LAYER,
    CNN_NUM_TOTAL_ROW_SECTIONS_4TH_LAYER,
    CNN_NUM_ROW_SECTIONS_1ST_LAYER,
    CNN_NUM_ROWS_1ST_LAYER,
    CNN_BUFFER_END_ADDR_1ST_LAYER
);
```

Start the CNN

```
start_cnn();
```

```
#define CNN_NUM_VECTOR_ELEMENTS 16

#define CNN_NUM_ROWS_1ST_LAYER 64
#define CNN_NUM_ROWS_2ND_LAYER 64
#define CNN_NUM_ROWS_3RD_LAYER 32
#define CNN_NUM_ROWS_4TH_LAYER 16
#define CNN_NUM_COLS_1ST_LAYER 128
#define CNN_NUM_COLS_2ND_LAYER 128
#define CNN_NUM_COLS_3RD_LAYER 64
#define CNN_NUM_COLS_4TH_LAYER 32
#define CNN_NUM_ROW_SECTIONS_1ST_LAYER (CNN_NUM_COLS_1ST_LAYER / CNN_NUM_VECTOR_ELEMENTS)
#define CNN_NUM_ROW_SECTIONS_2ND_LAYER (CNN_NUM_COLS_2ND_LAYER / CNN_NUM_VECTOR_ELEMENTS)
#define CNN_NUM_ROW_SECTIONS_3RD_LAYER (CNN_NUM_COLS_3RD_LAYER / CNN_NUM_VECTOR_ELEMENTS)
#define CNN_NUM_ROW_SECTIONS_4TH_LAYER (CNN_NUM_COLS_4TH_LAYER / CNN_NUM_VECTOR_ELEMENTS)
// ...
```

Results and Conclusion

- **CNN-based positioning hardware for Massive MIMO**
- **Developed with ASIP Designer from Synopsys**
- **Synthesized in 22nm technology node**
- **Cell area of around 1.1 mm^2**
- **Power consumption of 55 mW (not including external DRAM)**
- **2.5M cycles to perform one localization**
- **3.1 milliseconds (with an 800 MHz clock)**
- **320 user positions per second**

Table 1: Cell area breakdown

Module	**Cell Area***	**Utilization**
Vector memory	782,766	70%
Program Memory	60,980	5%
Scalar Data Memory	23,069	2%
Systolic Array	94,279	8%
CNN Engine	73,529	6%
ASIP	92,126	8%
Total	1,126,749	100%

* in square micrometers

LUND
UNIVERSITY