A general form of HDL-based Neural Network in a digital ASIC

Yun-Tsung Lai

KEK IPNS

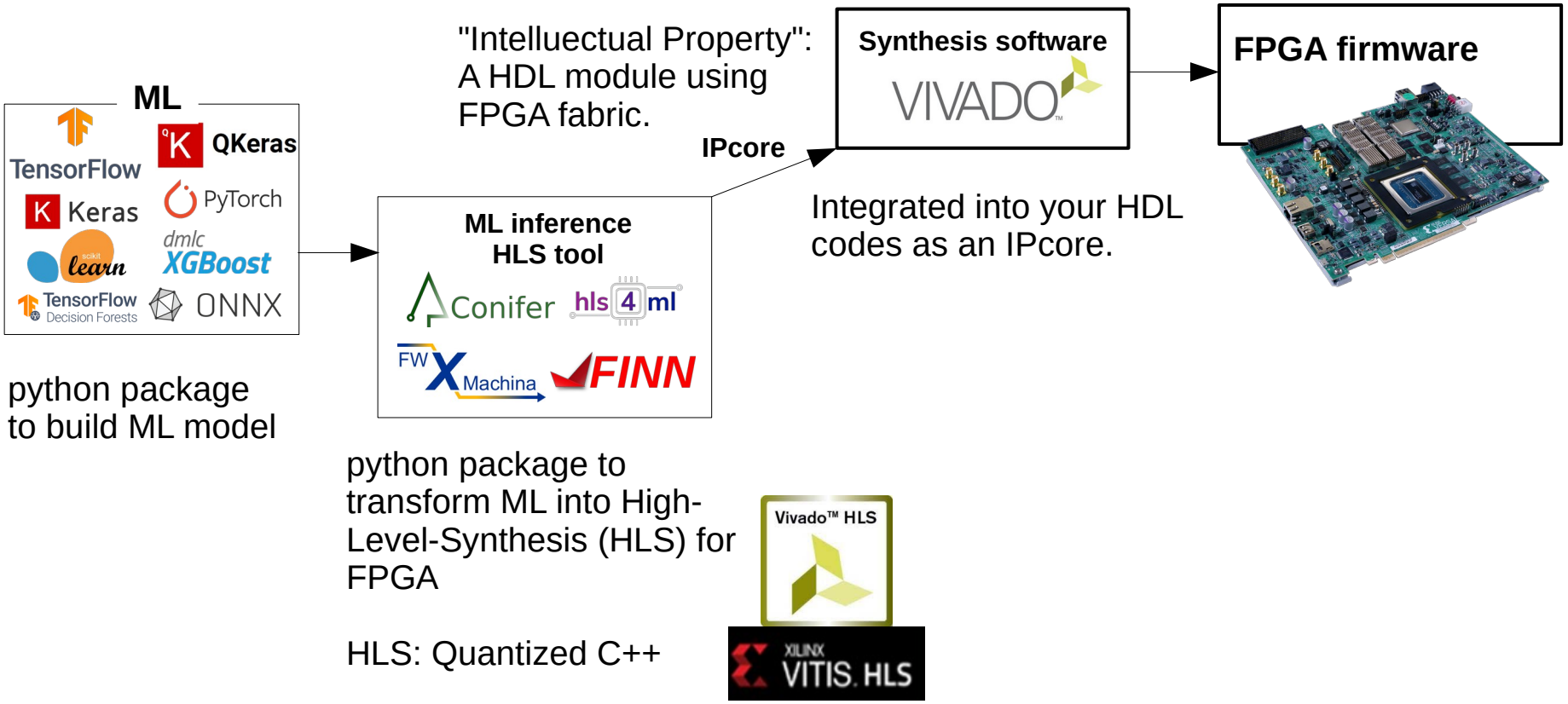
ytlai@post.kek.jp

ML4FE workshop @ University of Hawaii at Manoa

19th May, 2025

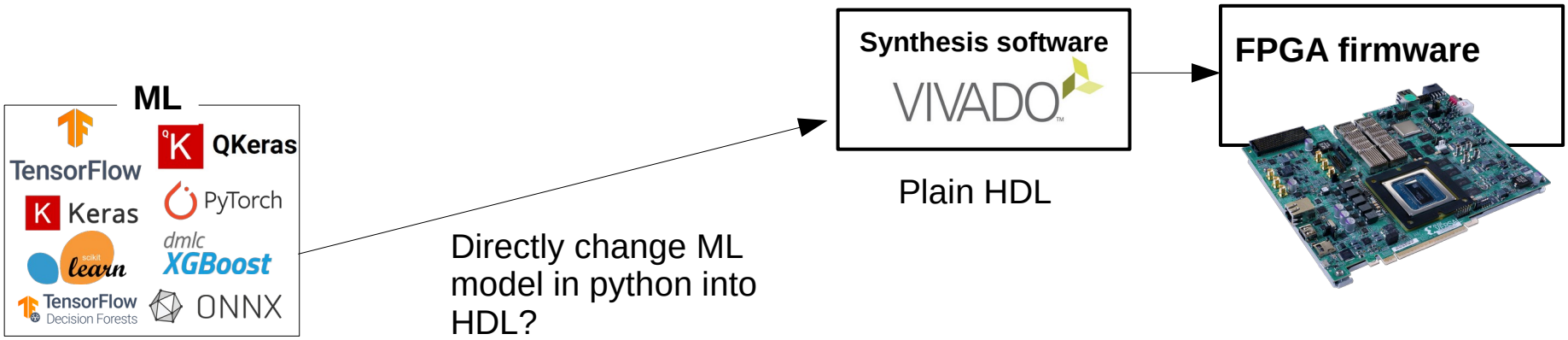
ML in FPGA based on HLS

- ML inference at FPGA based on HLS has been widely utilized in the filed of experimental HEP.
- Quantized C++ → HLS → IPcore in FPGA
- A kind of black box with FPGA fabric involved: DSP, memory, etc.
  - Not exactly a pure HDL module.



ML in FPGA based on HDL

- Can we directly represent a ML model in **plain HDL**?
  - Yes, if we understand the math of the ML model first.

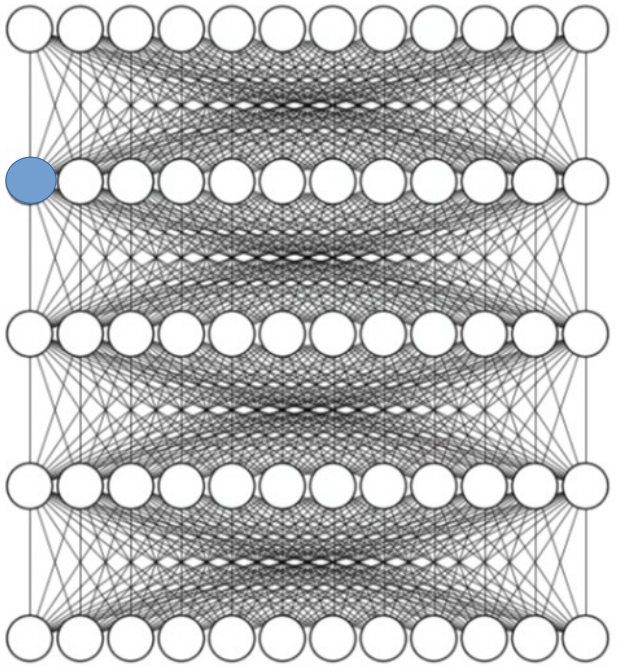


HDL-based NN

- In each neuron, a basic form is:
  - output = relu($\Sigma_i(w_i \times input_i) + b$)
  - "a x b + c" is easy for HDL.
  - relu is also easy for HDL.

- For the activation function, exponential or the other special functions are difficult to be implemented by using HDL only.
  - LUT/BRAM is needed.
  - Linear output is enough for most of the purpose (separation or regression).

- A general form of NN can be written with HDL.
  - CNN is also possible: convolutional layer and pooling layer are not difficult for HDL.

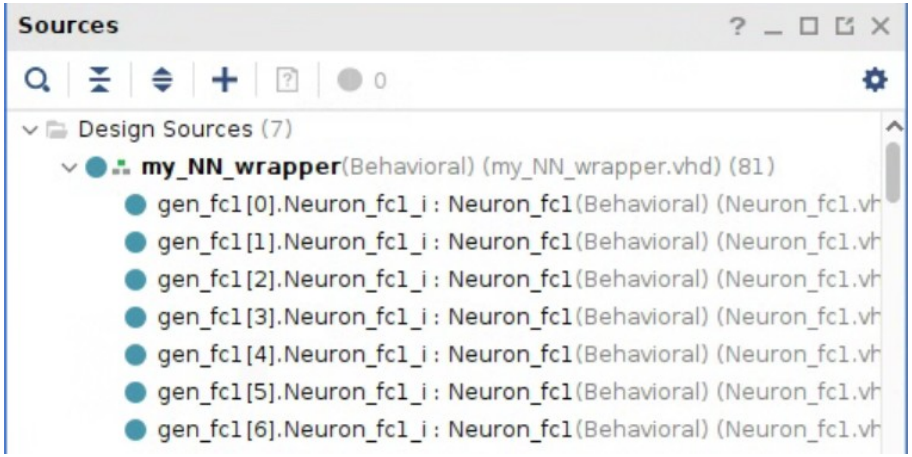


- Easy for HDL:
  - Addition
  - Subtration
  - Multiplication
  - If-else

- Difficult for HDL:
  - Special function: exp(), sin(), cos(), ...
    - Using BRAM is OK, but more latency.
  - Division
    - It is OK for dividor of 2^n.

HDL module design

- 1 neuron = 1 HDL code = 1 register
- Everything is parameteried: model shape, precision, weights and biases.

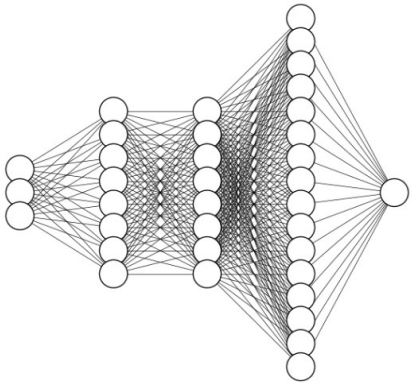



```
package library_NN is

  constant N_bit_input : integer := 16;
  constant I_bit_input : integer := 8;
  constant F_bit_input : integer := N_bit_input - I_bit_input;

  constant N_bit : integer := 16;
  constant I_bit : integer := 8;
  constant F_bit : integer := N_bit - I_bit;

  constant N_neuron_input : integer := 3;
  constant N_neuron_fc1 : integer := 16;
  constant N_neuron_fc2 : integer := 32;
  constant N_neuron_fc3 : integer := 32;
  constant N_neuron_output : integer := 1;
```

```
constant weight_fc1 : array_weight_fc1 := (
  0 => (
    0 => X"0000", -- 0.0
    1 => X"0000", -- 0.0
    2 => X"FFE8" -- -0.09375
    ),
  1 => (
    0 => X"0000", -- 0.0
    1 => X"FF4E", -- -0.6953125
    2 => X"0078" -- 0.46875
    ),
  2 => (
    0 => X"FF3C", -- -0.765625
    1 => X"008C", -- 0.546875
    2 => X"0048" -- 0.28125
    ),
  3 => (
    0 => X"FFDC", -- -0.140625
    1 => X"FFB4", -- -0.296875
    2 => X"00A6" -- 0.6484375
    ),
  4 => (
    0 => X"FF58", -- -0.65625
    1 => X"FFF8", -- -0.03125
    2 => X"0000" -- 0.0
    ),
```

Comparison to hls4ml

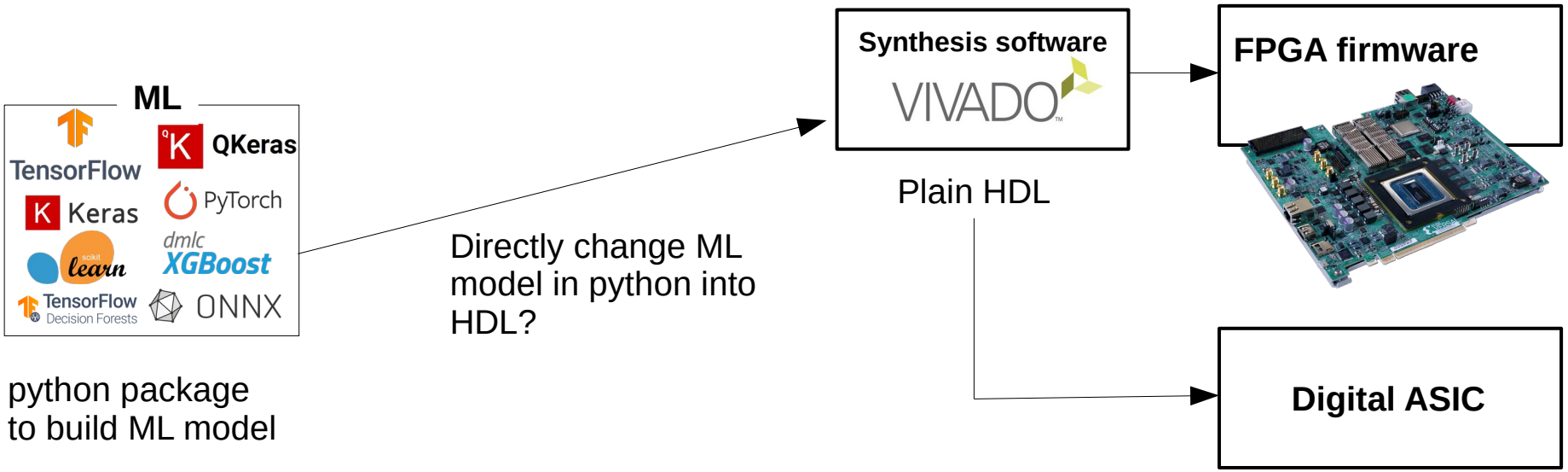
- We implement the same NN model in the Nexys Video card.
  - For hls4ml, sigmoid is used at the output layer, but the HDL-based version is not.
  - In the HDL-based design, each layer is doing relu($\Sigma_i(w_i \times input_i) + b$) in a register, so 1 clock-cycle is consumed.



	hls4ml	HDL-based NN
LUT	35278	41097
FF	52620	1276
DSP	289	0
Latency (clock-cycle)	30	4

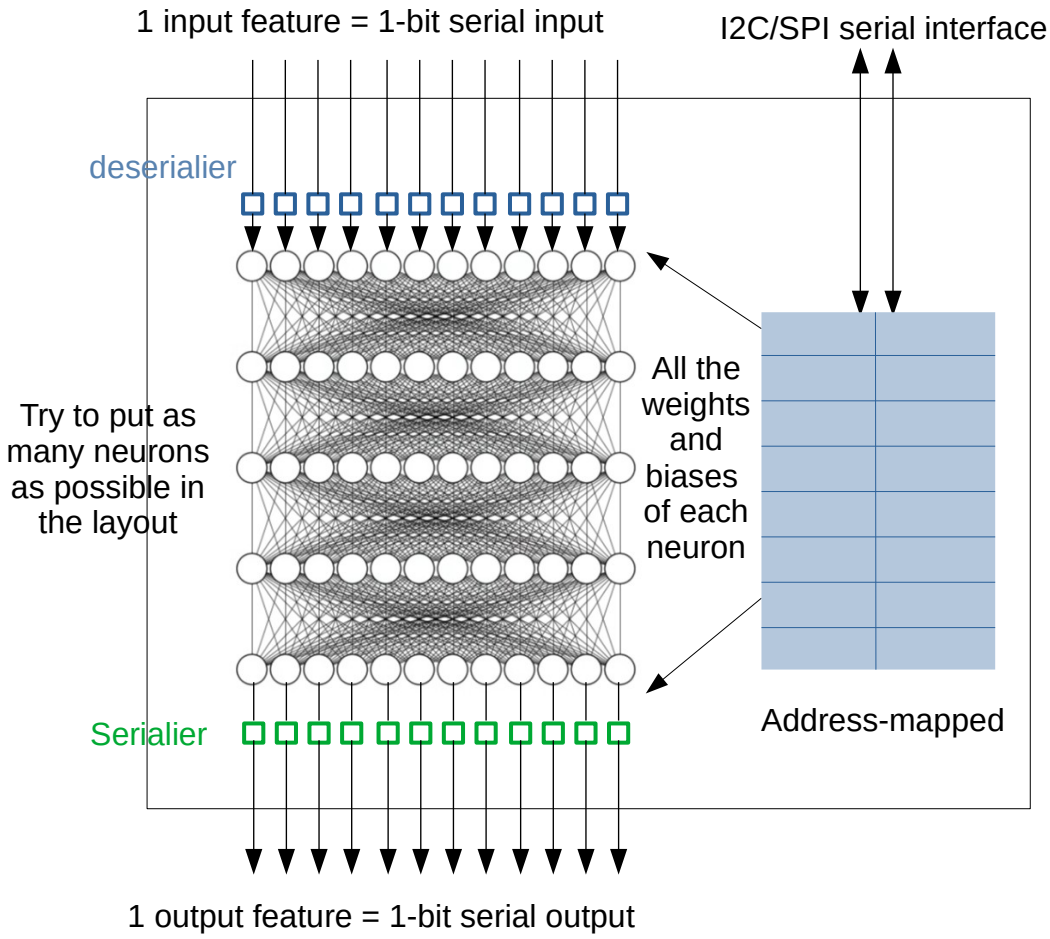
Further application: digital ASIC

- Therefore, with some conditions, make a NN with plain HDL is doable.
- **If everything is written by plain HDL, it can be made into an ASIC.**
- The approaches with higher level are convenient. Although the lower level design at RTL or even transistor level are relatively technically difficult, it has better flexibility on the hardware design and faster processing.



Plan for ASIC design

- The shape of NN to be put in the ASIC depends on area in the layout.
- In addition, if all of the wrights and biases can be updated via a serial interface and address-mapped design, we can make it a **programmable NN in ASIC**.



- Plan:
  - HDL design is almost ready.
  - Started layout design for ASIC.
  - Aiming for submission in the end of 2025 with TSMC 22nm.
  - Circuit board design: Using FPGA for serial I/O and control.
- We are expecting that a small-scale NN in ASIC (outside of FPGA) up to O(GHz) processing could be helpful for FEE or trigger in real-time DAQ.