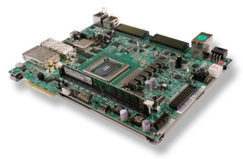
Designing a Gated Recurrent Unit in the Versal AI Engines

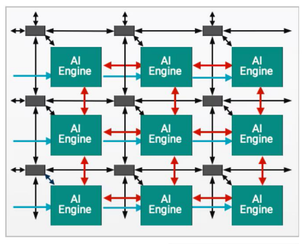
Sapkas Michail, RL4AA, Mar 30 – Apr 2, 2026, Liverpool

University of Padova – INFN Padova

The AMD VCK190 Versal AI Engine (AIE)

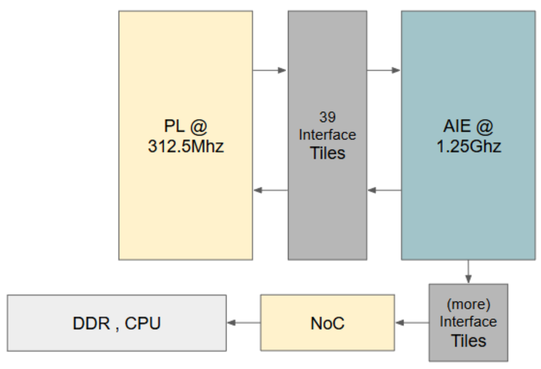
- A 50 by 8 **Tiled Array** of **400 Vectorized Processors**
- Distinct FP32 MAC pipelines
- Each vectorized unit has each own local memory accessed by the Processing System
- Multiple Data Paths inside the Ai Engine:
  - Cascade Streams
  - Shared Memory Banks
  - Stream Interconnect
  - Packet Streams
- 39 PL (128b) to AIE (32b @ 1.25Ghz) Interface Tiles
- Graph - Kernel Programming model



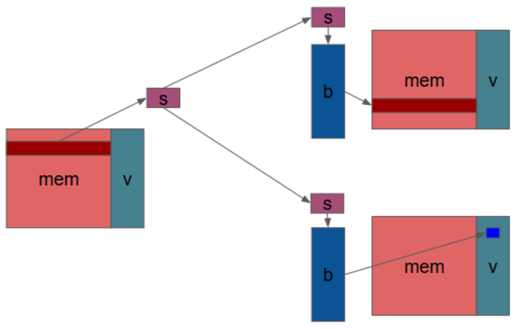


AIE Connectivity and Data Transfers

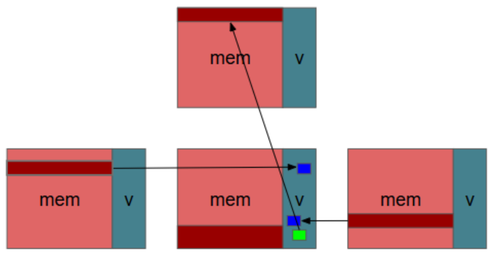
- Interface Tiles move data between the AIE – FPGA and peripherals – CPU



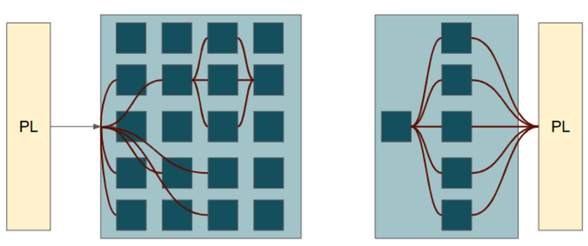
- AIE Tiles communicate via Streams



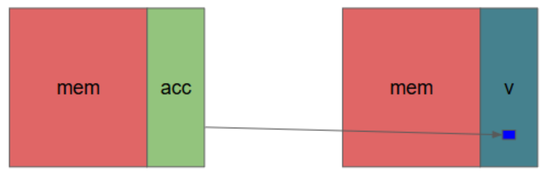
- via Memory Sharing mechanism



- AIE Tiles communicate via Packets



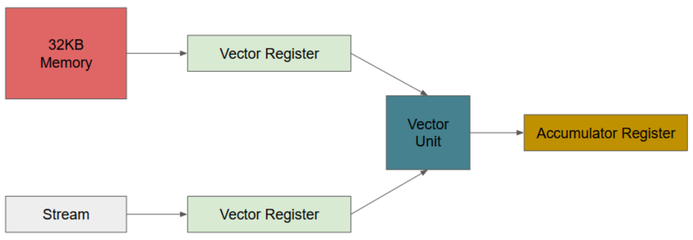
- via Cascade Streams



- Topology matters!

Vectorized Operations

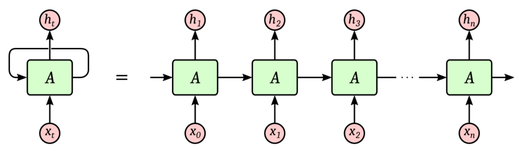
- Vectorized registers can operate on 256b of Data
- Multiply Accumulate Pipeline



The Gated Recurrent Unit (GRU)

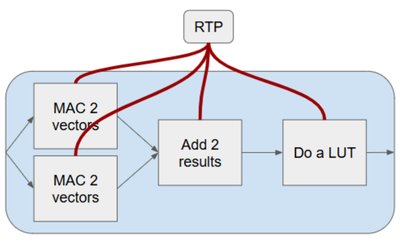
- Belongs to the family of Recurrent Neural Networks, specialized models for timeseries data
- Makes using of the "Gating" Mechanism and the Hidden State Recurrency which imposes Data Flow in the AIE

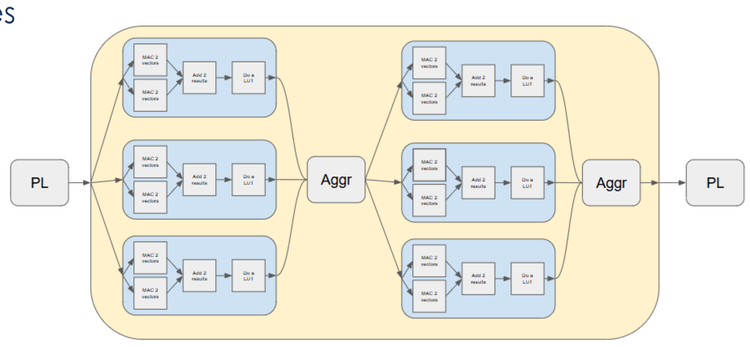
$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$



Graph Programming

- Dataflow graphs define the topology and compute of the AIE Tiles
- Hierarchical graph structures
- Run Time Parameters





Objective

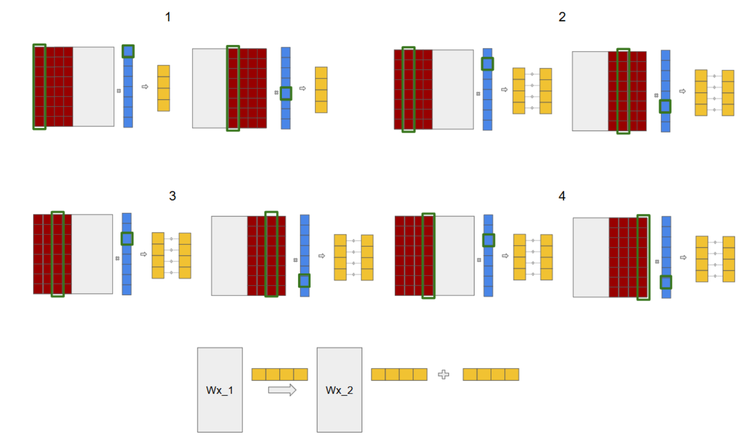
- To be used as **Real Time Controllers** trained continuously in a **Reinforcement Learning** framework
- For the first time, AIE gives us the possibility to deploy **Large** and **Unquantized (FP32) RNN** models with **microsecond latency**

Implementing the Matrix Vector Multiplication

- It's not trivial which Data Path to use inside the AIE nor which Matrix-Vector method to use. The one affects the other.
- Inside the Vectorized Processor we are using the **Multiply Accumulate (MAC)** operation, which has a latency of 8 clock cycles for FP32

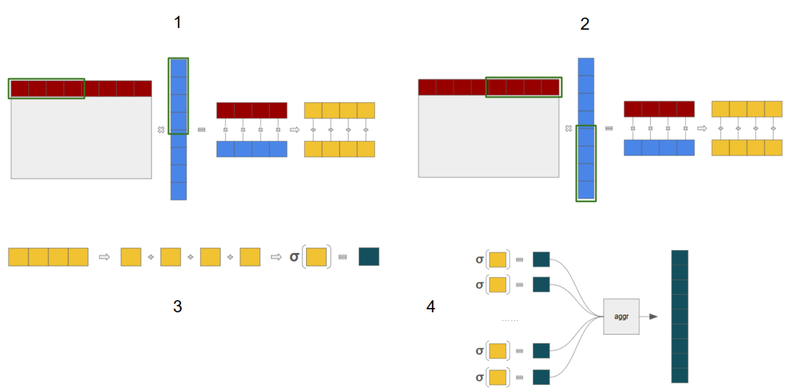
Method 1: Column-Element-Cascade

- This method respects the Interface tiles throughput by using the vector elements partially: We MAC elements of the vector with columns of the matrix



Method 2: Row-Vector-Reduce-Stream

- This method breaks pipelining to take advantage as many of the cores as possible exploiting the ability of the AIE to broadcast the same signal via the AXI4 Interconnect



Data Aggregation and Activation Functions

- We explore two possibilities for data aggregation **using only Method 2:**
- **Packet Split aggregation** and **sorting** inside the AIE using specialized kernels
- Or **aggregating**, **sorting** and **applying activation function LUTs in the PL,** using a **32-input HLS kernel connected to 32 interface tiles**. This data path can be reused for more than 32 Hidden state model by adding the same latency:

INTERFACE TILES HIDDEN DIM	AIE GRU TILES USED	AIE HYBRID TILES USED	PL kernel				AIE AGGR TILE LAT
			II	DEPTH	FF	LUTS	
20	184	181	6	7 cc = 22.4 ns	5124	9250	229.6 ns
24	220	217	6	7 cc = 22.4 ns	6066	11834	273.6 ns
28	256	253	8	9 cc = 28.8 ns	7057	15663	319.2 ns
32	292	289	8	9 cc = 28.8 ns	8003	19198	363.2 ns

Results

- Results were presented at **TWEPP 2025** and **published** in proceedings