



Development of DNN algorithm on Versal AI-engine

Zhao-Zhi Liu

Institute of Frontier and Interdisciplinary Science, Shandong University

Belle II TRG parallel

2025/2/24





1.Introduction DNN in CDC track reconstruction

2.Al engine structure

3.Baseline version of the DNN deployment

4. The second version with the vectorize optimization

5. The third version with the location optimization



DNN for tack reconstruction



CDC raw hits

Built **Track Segment** (a set of CDC wires) in every super layer

 Each Track Segment(TS) has a set of α, t and Φ
 Variables in all 9 super layers input into the deep neuro-network



Input Drift time t_{drift} , wires relative location ϕ_{rel} , Crossing angle α

Output track vertex z_0 , track θ and signal/background classifier output (Q) 3



Design flow







Step1: Generate hardware platform in vivado(.xsa file)

Step2: Neural network deployment in AIE(.bin file)





Al engine

Al Engine A Al Engine-F	Array, 6 x 4 A PL/NoC Inte	I Engine Til rface Tiles	es and corr	esponding	
Al Engine	Al Engine	Al Engine	Al Engine	Al Engine	Al Engine
Tile	Tile	Tile	Tile	Tile	Tile
Al Engine	AI Engine	Al Engine	Al Engine	Al Engine	Al Engine
Tile	Tile	Tile	Tile	Tile	Tile
Al Engine	Al Engine	AI Engine	Al Engine	AI Engine	Al Engine
Tile	Tile	Tile	Tile	Tile	Tile
Al Engine	Al Engine	AI Engine	Al Engine	AI Engine	Al Engine
Tile	Tile	Tile	Tile	Tile	Tile
Configuration	PL	NoC	NoC	PL	PL
Interface	Interface	Interface	Interface	Interface	Interface
Tile	Tile	Tile	Tile	Tile	Tile
_					X20818-04

AIE array





VCK190 has an AIE array of 400 tiles. Each tile has an AIE which has vector unit and scalar unit for vector algorithm and scalar algorithm.



Baseline development on VCK190



1.One kernel represent for one hidden layer and its input dense layer and mapped inside one single tile.

2. The in/output of the kernels are buffer types, so one buffer between two kernels.

3.All kernels inside one graph.





6



♀ 咄 ↔ ▦ € ☉	2 63 🔿 🕅	≫ ⊠ → [⟨	▶ 🛨 🛨 +	[← →[–Г × ☴ Compare					7
									86.160 us	
NAME		VALUE	0.000000 us		20.000000 us	40.000000 us	60.000000 us	80.00000	us	100.000
> Tile(24,0)	Layer1	_main	•			_mai	n			
>Tile(24,1)										
>Interface Tile(24)										
> Tile(25,0)	Layer3	_main	_main			_ma	in			
> Tile(25,1)	Layer4	_main			m	ain		•	_main	
> Tile(25,2)	Layer2	_main	_main						_main	
> Tile(26,0)	Layer5	_main	-			_main			_main	
>Interface Tile(26)										

*Non-blue modules represent operational status

	Layer1 (leakyrelu)	Layer2 (softmax)	Layer3 (no act.)	Layer4 (leakyrelu)	Layer5 (tanh)	total
Latency	~12us	~66us	~1.5us	~5.5us	~0.9us	~86us

*Buffer port used, which need time to ready before usage.



Simple analyze on the layer 1



Vector algorithm cost about 2.1us

Scalar spends about 10us

Layer one has 71 inputs and 27 outputs

In dense layer, 71*27=1917 times multiplications .(vector part) In act. function, 27 times compare and multiplications. (scalar part)





Vectorize optimization comparison



Before opt.



After opt.

*outIter1++ = act_leakyrelu(sum[i]);

inline float act_leakyrelu(float x) {

if (x < 0) {return 0.01*x;}
else {return x;}</pre>

aie::vector<float,32>sum2 = aie::mul(sum,0.01f); aie::vector<float,32>act_out = aie::select(sum, sum2, aie::ge(sum, 0.0f));





		Layer 1 (leakyrelu)	Layer 2 (softmax)	Layer 3 (no act.)	Layer 4 (leakyrelu)	Layer 5 (tanh)	total
Before opt.	Latency	~12us	~66us	~1.5us	~5.5us	~0.9us	~86us
After opt.	Latency	~2.1us	~1.3us	~1.5us	~0.9us	~0.2us	~5us

Abandon all the scalar algorithm, all use vector algorithm instead.





Total 555 clk cycles one instance. Clk period 10ns. So the latency in ila is 5.55us.





Make the mac operation from **sequential** to **parallel** by disassembling the mac operation into **three aie tiles** using **location constraint**.













1.one graph represents one layer.
 2.three mac kernels form the dense layer.
 3.one act. kernel forming the hidden layer.



Location optimization

Latency	Layer 1 (leakyrelu)	Layer 2 (softmax)	Layer 3 (no act.)	Layer 4 (leakyrelu)	Layer 5 (tanh)	total
No opt.	~12us	~66us	~1.5us	~5.5us	~0.9us	~86us
Vectorize opt.	~2.1us	~1.3us	~1.5us	~0.9us	~0.2us	~5us
Location opt.	~479ns	~931ns	~327ns	~404ns	~100ns	~2.1us

Note:

① Due to the location constraint, some links between kernel ports have a longer way to move which brings **more delay in data moving**.

So the total latency is a bit larger than kernel combined latency.





Total **305 clk cycles** one instance. Clk period **10ns**. So the latency in ila is **3.05us**.



1 In **baseline** version, **vector algorithm** is used in **dense layer** while **scalar algorithm** is used in **act. function**.

And total latency in simulation is about **86us**. In this part, it finds out that vector algorithm is fast.

② In the version 2 with optimization, act. function uses vector algorithm for replacement. And total latency in simulation is about 5us. Ila shows the latency is about 5.5us.

③ In the version 3 with location optimization, one layer is divided into multiple ai-engines instead of one to improve the parallel execution to lower the latency. Simulation shows the latency is about 2.1us. Ila shows the latency is about 3.05us.







BACK UP



Layer1 latency

										1,600.800 ns	;		
NAME		VALUE	60	0.000 ns	1800.000 n	s 1,000.000	1,121.600 ns ns 1,200.000) ns 1,400.00	00 ns	1,600.000 ns	1,800.000	ns 12,000.000	ns 2,200.000
> Tile(19,0)													
> Tile(20,0)	mac	_main			_main_init	_main	kernell1(adf::i	o_b			_main		
> Tile(20,1)	mac	_main			_main_init	_main	kernel12(adf:::	.o_b			_main		
> Tile(20,2)													
> Interface Tile(20)													
> Tile(21,0)	mac	_main			_main_init	_main	kernel13(adf:::	.o_b			_main		
> Tile(21,1)	act	_main			_main_init		_main	ke	erne			_main	
> Tile(21,2)													
> Tile(21,3)		_main			_main_init			_main		kernel2	21(adf:	_ma	in
> Tile(21,4)													
>Interface Tile(21)													
> Tile(22,0)		_main			_main_init			_main		kernel3	31(adf:	_ma:	in
> Tile(22,1)													
> Tile(22,2)		_main			_main_init			_main		kernel2	23(adf:	_ma	in
> Tile(22,3)		_main			_main_init			_main		kerrel2	22(adf:	_ma	in
> Tile(23,0)		_main			_main_init			_main		kernel	32(adf:	_ma	in
> Tile(23,1)		_main			_main_init			_main		kernel:	33(adf:	_ma	in
> Tile(23,2)		_main		_	_main_init			_ma	in			kerne so	oftmax(adf::
> Tile(23,3)													
> Tile(23,4)									4				
> Tile(24,0)			600.000		100 000 00	200 000 00	0.000 pc	200,000 pc	100.00	479.200 ns		800 000 pc	1 000 000 pc
> Tile(24,1)		_main	-000.000	115 - 2		-200.000 IIS		200.000 IIS				000.000 115	1,000.000 iis



Layer2 latency

													2,597	.600 ns					
NAME		VALUE		1, L,600.0	,666.400 n: 000 ns 1	s ,800.	000 ns 2,000	0.000 ns 2,1	200.0	000 ns 2,	,400.	000 ns	2,60	0.000 ns 12,8	00.000 ns	3,000).000 ns	3,200	.000
> Tile(20,2)																			
> Interface Tile(20)																		
> Tile(21,0)	mac	_main								_main									
> Tile(21,1)		_main	kerne								_mai	n							
> Tile(21,2)																			
> Tile(21,3)		_main	_main	ke	ernel21(ad	f:						_	ain						
> Tile(21,4)																			
> Interface Tile(21))																		
> Tile(22,0)		_main	_main	k	ernel31(ad	lf:						_	nain						
> Tile(22,1)																			
> Tile(22,2)	mac	_main	_main	k	ernel23(ad	f:						_	nain						
> Tile(22,3)	mac	_main	_main	k	ernel22(ad	lf:						_	nain						
> Tile(23,0)		_main	_main	k	ernel32(ad	lf:						_	nain						
> Tile(23,1)		_main	_main	k	ernel33(ad	lf:						_	nain						
> Tile(23,2)	act	_main			main		kerne	. softmax(ad	f::i	o_buffer <fl< td=""><td>loat,</td><td>ad</td><td></td><td></td><td>_ma:</td><td>in</td><td></td><td></td><td></td></fl<>	loat,	ad			_ma:	in			
> Tile(23,3)									_										
> Tile(23,4)									+										
> Tile(24,0)						_			_										
> Tile(24,1)		_main		_	main		••						ma	in					
> Tile(24,2)		_main					_mair	1								nain			
> Tile(24,3)		_main	200 000 00		000 05	2	00 000 pc	ain	64	00 000 pc		an 198 n	931.2	200 ns nel41 a	df:	0.05	main		1 680
> Tile(24,4)			-200.000 hs					400.000 HS	1 00		8			1,000.000 //S	1,200.00		1,400.000		1,000



					1 655 400			1,993	<mark>.600 ns</mark>											
NAME		VALUE		1,600	.000 ns	1,800	.000 ns	2,000	0.000 ns	2,200	0.000 ns	2,400	0.000 ns 12	,600.00	0 ns 2,	,800.000 n	s 3,00	0.000 ns	3,200.00	00
> Tile(20,2)																				
> Interface Tile(20))																			
> Tile(21,0)		_main									_ma	in								
> Tile(21,1)		_main	kerne									ma	in							
> Tile(21,2)																				
> Tile(21,3)		_main	_main	1	kernel21(a	df:	•						_ma:	in						
> Tile(21,4)																				
>Interface Tile(21))																			
> Tile(22,0)	mac	_main	_main		kernel31(a	adf:	•						_ma.	in						
> Tile(22,1)																				
> Tile(22,2)		_main	_main		kernel23(a	df:							_ma.	in						
> Tile(22,3)		_main	_main		kernel22(a	adf:	•						_ma.	in						
> Tile(23,0)	mac	_main	main		kernel32(a	adf:.							_ma	in						
> Tile(23,1)	mac	_main	main		kernel33(a	adf:.	•						_ma	in						
> Tile(23,2)		kernel2(adf::io_buffe			_main		ker	ne	. softma	ax(adf:	:io_buffer	<floa< td=""><td>:, ad</td><td></td><td></td><td>_"</td><td>nain</td><td></td><td></td><td></td></floa<>	:, ad			_"	nain			
> Tile(23,3)																				
> Tile(23,4)																				
> Tile(24,0)																				
> Tile(24,1)	act	_main			_main									_main						
> Tile(24,2)		_main						_mair						••			_main			
> Tile(24,3)		_main	-200.000 n	s	0.000 ns		200 000 n	327.2	400.000	ns	600.000 ns	5	800.000 ns	1.0	kernel41(00.000 ns	adf:	000 ns	main	ns 1.6	600
> Tile(24,4)					111111			111		1.1					11111		11111			1.1



Layer4 latency

								3,106.400 ns			
NAME		VALUE	l ²	2,200.000 ns 2,400.	000 ns 2,600.	2,702.400 ns 000 ns 2,800.00	0 ns 3,000.0	0 ns 3,200.0	00 ns 3,400.0	990 ns 3,600.	0 <u>0</u> 0 ns
> Tile(22,2)		main					_main				
> Tile(22,3)							_main				-
> Tile(23,0)							_main				-
> Tile(23,1)		main					main				
> Tile(23,2)		 main	softmax(adf::io_buffer <float,< td=""><td>ad</td><td></td><td></td><td>_main</td><td></td><td></td><td>-</td></float,<>	ad			_main			-
> Tile(23,3)											
> Tile(23,4)											4
> Tile(24,0)											4
> Tile(24,1)		_main					_main				
> Tile(24,2)		main		_main				_ma:	.n		
> Tile(24,3)	mac	_main		_main		kernel41(adf:			_main		-
> Tile(24,4)	mao										
> Tile(25,0)		_main			_main			simp		_main	
> Tile(25,1)	act	_main			_main		kerne			main	
> Tile(25,2)	mac	_main		_main		kernel43(adf:	•		_main		
> Tile(25,3)	mac	_main		_main		kernel42(adf:			_main		
> Tile(25,4)											
> Tile(25,5)											4
> Interface Tile	:(25)										
> Tile(26,0)								484 888 pc			
> Tile(26,2)			-600.000	ns -400.000 ns	-200.000 ns	0.000 ns	200.000	400.000 ns	.0.000 ns	800.000 ns	1,000
$\Sigma Tile(26.4)$										L	1111









Level one trigger

- CDC, ECL: main triggers for tracks and clusters
- KLM: trigger muon
- TOP: event timing
- GRL: matching of sub-triggers
- GDL: final trigger decision

