

AccelReg: Accelerator for Linear Regression

CSEE 4840: Embedded Systems

Pranav Asuri

pa2708@columbia.edu

Varsha Keshava Prasad

vk2550@columbia.edu

Venkat Suprabath Bitra

vsb2127@columbia.edu

Doreen Sisanalli

ds4371@columbia.edu

Background

"fingerprints"

for

peaks



Motivation

- Instrument makers need on-board hardware for direct quantitative analysis, avoiding external PCs
- Portable spectrometers gather data fast, requires quick, real-time parameters for point-of-care applications
- Scientists prefer automated tools, typically avoiding to work directly with code for ease of use



Develop an FPGA linear regression module for portable Raman spectrometers, enabling

rapid, on-the-fly, user-friendly chemical concentration analysis

4-Bit Integer Quantization for Linear Regression

- Quantized 4-bit integers reduce FPGA resource usage and power
- Training with representative dataset ensures

robust quantized model performance

- Representative Dataset: reflects statistical properties and variations of the original full precision data.
- 4-bit quantization yields results closely matching the original full-precision model



Simplification for Linear Regression

$$n \text{ observations } (x_i, y_i) \quad y = w_0 + w_1 x \qquad S_1 = n \qquad (\text{count of obs})$$
$$y_i = \begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \qquad S_2 = \sum_{i=1}^n x_i \qquad (\text{sum of } x_i)$$
$$S_3 = \sum_{i=1}^n y_i \qquad (\text{sum of } y_i)$$
$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} n & \sum_{i=1}^n x_i^2 \\ \sum_{i=1}^n x_i^2 \end{bmatrix} \quad \mathbf{X}^T \mathbf{y} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i^2 \end{cases} \qquad (\text{sum of } x_i^2)$$
$$(\mathbf{X}^T \mathbf{X})^{-1} = \frac{1}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \begin{bmatrix} \sum_{i=1}^n x_i^2 \\ -\sum_{i=1}^n x_i \end{bmatrix} \qquad S_5 = \sum_{i=1}^n x_i y_i \qquad (\text{sum of } x_i y_i)$$

 $\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \frac{1}{S_1 S_4 - S_2^2} \begin{bmatrix} S_4 S_3 - S_2 S_5 \\ S_1 S_5 - S_2 S_3 \end{bmatrix}$

(count of observations)

- Use accumulators and multipliers in int4.
- Convert to float at the end for precision.

LR Accelerator Component (Block Diagram)

- 16 parallel pipelines accumulate data
- Adder sums pipeline outputs

- Processing unit computes products
- Register interface controls data flow



LR Accelerator Component (Accumulator Block)



LR Accelerator Component (Weight Computation Block)



Timing Diagram for LR Accelerator Component

◎ • ☞ 묘 ☜ 를 ≵ ☜ ጫ ♀ ≏ ◎ • ₩ 문 ◈ थ ≔ 梁 隊	5 🕇 🖛 🔿	IF _ 100 # \$ IL	F 📗 🛠 - +€ - 📴 Search: 💽 💌 🏨 🎆 🖉 🧠 🧠 🧏 🤽 🔂
<u>í</u>	Msgs		
	1'd0	\overline{m}	ատակատակատակատակատակությո
7/ID_IF_acc_512_8D/reset	1.00		
4 //b Ir acc 512 8h/write	1'd0		
✓ /b ir acc 512 8b/read	1'd0		
✓ /tb Ir acc 512 8b/chipselect	1'd0		
	10'dx	- anatanaahaaataaantaanatanaahaaa - a	
🖬 🔶 /tb_lr_acc_512_8b/readdata	32'd2480		1 X0 X7 X10752 X32 X240 X224 X2480
▪-	10'd31		
tb_lr_acc_512_8b/write_mem/x_val	4'd15		
/tb_lr_acc_512_8b/write_mem/y_val	4'd14		
<pre>/tb_ir_acc_512_8b/read_result/addr</pre>	10'd7		
//tb_lr_acc_512_8b/read_result/result_data	32'd224		
+	32'0224		
//////////////////////////////////////	1'd0		
$\frac{1}{2}$ / $\frac{1}{2}$ / $\frac{1}{2}$ / $\frac{1}{2}$ / $\frac{1}{2}$ / $\frac{1}{2}$	32'dy		
4 /tb Ir acc 512 8b/dut/write	1'd0		
↓ /tb Ir acc 512 8b/dut/read	1'd0		
↓ /tb Ir acc 512 8b/dut/chipselect	1'd0		
• 4 /tb_lr_acc_512_8b/dut/address	10'dx		
🖬 🔩 /tb_lr_acc_512_8b/dut/readdata	32'd2480		1 X0 X7 X10752 X32 X240 X224 X2480
/tb_lr_acc_512_8b/dut/master_done	1'd1		
tb_lr_acc_512_8b/dut/mem_bank	{4'd0 4'd0} {4'd	-{(0 0) {0 0) {0 0}, {0	D D} {O D} {
Itb_Ir_acc_512_8b/dut/pc_status_bank	{9'd511 4'd15 1		<u>X X X X {511 X{511 15 1 30 28 450 420} {510 14 1 28 24 392 336} {509 13 1 26 20 338 260} {508 1</u>
/tb_lr_acc_512_8b/dut/go	1'd0		
√/tb_lr_acc_512_8b/dut/pc	9'd0		
/t0_IF_acc_512_8b/dut/done	1'd0		
<pre>//b_ir_acc_512_8b/dut/local_s_1</pre>	18'd32		
$\mathbf{F}_{\mathbf{A}}$ //b Ir acc 512 8b/dut//dobal s 1	18'd32		[32
$\mathbf{F}_{\mathbf{v}} = \frac{1}{2} / \frac{1}{2} $ (b) $\mathbf{F}_{\mathbf{v}} = \frac{1}{2} $ (b) $\mathbf{F}_{\mathbf{v}} = \frac{1}{2} $ (c) \mathbf{F}	18'd240		1240
The second secon	18'd224		224
•	18'd2480		2480
🖅 🤣 /tb_lr_acc_512_8b/dut/global_s_5	18'd2016		2016
	32'd21760		<u>0 (21760</u>
	32'd71680		71680
	32'd10752		0 10752
/tb_lr_acc_512_8b/dut/start	16'd0	- (0) (65535) (655535) (655	
/tb_lr_acc_512_8b/dut/local_done	16'd0	0	
+ γ /tb_lr_acc_512_8b/dut/dut_num	4'dx		
$7/10$ [_acc_512_80/00/master_start	1 00 0/d511		
4/m in acc 512 8b/dut/alobal start adders	1/41		
$\mathbf{F}_{\mathbf{A}}$ //b. Ir. acc. 512. 8b/dut/local. global_s 1	18'd32		(32
$\mathbf{F}_{\mathbf{A}}$ /tb ir acc 512 8b/dut/local global s 2	18'd240		1240
The second secon	18'd224		224
	18'd2480		2480
+ / /tb_ir_acc_512_8b/dut/local_global_s_5	18'd2016		2016
	36'd79360		{0 X 79360
🖬 🧇 /tb_ir_acc_512_8b/dut/local_s2_squared	36'd57600	(0	(\$7600
■ 🧇 /tb_lr_acc_512_8b/dut/local_s3_times_s4	36'd555520		1555520
tb_lr_acc_512_8b/dut/local_s2_times_s5	36'd483840		483840
/tb_ir_acc_512_8b/dut/local_s1_times_s5	36'd64512		(0) (64512
/tb_ir_acc_512_8b/dut/local_s2_times_s3	36'd53760		153760 V 788890
/tb_lr_acc_512_8b/dut/s1_times_s4	36'd79360		
// // // // /////////////////////	36/05/600		157600
	36 0555520		422240
	36'464512		
Ab_n_acc_512_8b/dut/s2_times_s3	36'd53760		
	0000000		,

Accumulator Pipeline (Block Diagram)



Accumulator Pipeline (Analysis)

Fitter Summary

πιπιβ Απαιγέςτη (Επιαλ	[imin	g Ana	lyzer	(Fmax
------------------------	--------------	-------	-------	-------

Fitter Summary	
< <filter>></filter>	
Fitter Status	Successful - Tue May 13 02:15:41 2025
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	lr_acc_8b_wrapper
Top-level Entity Name	lr_acc_8b_wrapper
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	74 / 32,070 (< 1 %)
Total registers	193
Total pins	76 / 457 (17 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total RAM Blocks	0/397(0%)
Total DSP Blocks	2 / 87 (2 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0/6(0%)
Total DLLs	0/4(0%)



Timing Diagram

_ 🖒		Msgs														
	/tb Ir acc 8b/clk	1'd0														
	/tb_lr_acc_8b/reset	1'd0														
- 4	/tb_lr_acc_8b/start	1'd0														
.	/tb_lr_acc_8b/x	4'd3	(0		3	6	8	 11	, in the second s	3						
.	/tb_lr_acc_8b/y	4'd4	(0		4	7	8	 12	ľ	4						
•	/tb_lr_acc_8b/sum_x	32'd31	(0								3	ļ 9	17	28	31	
•	/tb_lr_acc_8b/sum_y	32'd35	(0								4	11	19	31	35	
•	/tb_lr_acc_8b/sum_x_squared	18'd239	(0								9	45	109	230	239	
.	/tb_lr_acc_8b/sum_xy	18'd262	(0								12	54	118	250	262	
- 4	/tb_lr_acc_8b/done	1'd0														

16 Number 18 Bit Adder Block



18 Bit Adder Block (Analysis)

Timing Analyzer (Fmax)

able of Contents	T Ø	Slow 1100mV 85	C Model Fmax Summ	nary	
📰 Flow Summary	^	< <filter>></filter>			
= Flow Settings		Fmax	Restricted Fmax	Clock Name	Note
📰 Flow Non-Default Global Se	ettings	1 377.5 MHz	377.5 MHz	clk	
📰 Flow Elapsed Time					
📰 Flow OS Summary					
📄 Flow Log					
📙 Analysis & Synthesis					
Fitter					
 Flow Messages 					
 Flow Suppressed Messages 					
Assembler					
🗖 📂 Timing Analyzer					
E Summary					
📰 Parallel Compilation					
📰 SDC File List					
📰 Clocks					
👻 📂 Slow 1100mV 85C Mod	el				
🎫 Fmax Summary					
📄 Timing Closure Reco	mmer				
📰 Setup Summary					
📰 Hold Summary					
📄 Recovery Summary					
📄 Removal Summary					
📰 Minimum Pulse Wid	th Sun				
📄 Metastability Summa	ary				
🕨 📙 Slow 1100mV 0C Mode	L				
🕨 📒 Fast 1100mV 85C Mode	et 👘				
🕨 📒 Fast 1100mV 0C Model					
📰 Multicorner Timing Ana	ysis S	This papel repor	s EMAX for every close	k in the design	regardless
🕨 📒 Advanced I/O Timing	1	that the duty cvc	le (in terms of a perce	ntage) is maintai	ined. Altera
Clock Transfers	*	, -, -		<u>.</u>	

|--|

🛳 .	Msgs				
/tb comb adder 18b/clk	1'd0				
/tb comb adder 18b/reset	1'd0				
/tb_comb_adder_18b/adder_start	1'd0				
+ /tb_comb_adder_18b/result	22'd240	0			240
H > /tb_comb_adder_18b/expected_sum	22'd240		240		
+ > /tb_comb_adder_18b/captured_result	22'd240				240
🛨 🔷 /tb_comb_adder_18b/i	32'd16	16			
/tb_comb_adder_18b/dut/clk	1'd0				
/tb_comb_adder_18b/dut/reset	1'd0				
/tb_comb_adder_18b/dut/adder_start	1'd0				
	18'd0	0			
Horizont - Horizont - Horizontal - Horizo	18'd2	0	12		
+ + /tb_comb_adder_18b/dut/op2	18'd4	0	4		
	18'd6	0	<u> </u>		
* /tb_comb_adder_18b/dut/op4	18'd8	0	<u> 18</u>		
	18'd10	0	10		
🕀 🖈 /tb_comb_adder_18b/dut/op6	18'd12	0	<u>, 12</u>		
+ /tb_comb_adder_18b/dut/op7	18'd14	0	<u>, 14</u>		
tb_comb_adder_18b/dut/op8	18'd16	0	<u>16</u>		
tb_comb_adder_18b/dut/op9	18'd18	0	<u> 18</u>		
tb_comb_adder_18b/dut/opA	18'd20	0	120		
+ /tb_comb_adder_18b/dut/opB	18'd22	0	122		
/tb_comb_adder_18b/dut/opC	18'd24	0	124		
+ /tb_comb_adder_18b/dut/opD	18'd26	0	126		
+ /tb_comb_adder_18b/dut/opE	18'd28	0	128		
+ /tb_comb_adder_18b/dut/opF	18'd30	0	130		
tb_comb_adder_18b/dut/result	18'0240	0	V -		1240
+ /tb_comb_adder_18b/dut/op01	18:02	0	12		
+ /tb_comb_adder_18b/dut/op23	18/010	0	<u>10</u>		
+ VID_comb_adder_18b/dut/op45	18018	0	118		
the comb_adder_18b/dut/op80	10'd24		120		
the comb_adder_16b/dut/op.69	10'034	0	¥42		
th comb_adder_18b/dut/opCD	10'450	0	¥50		
+ /th comb_adder_18b/dut/opEE	18'd58	0	158		
+ + th comb adder 18b/dut/local on01	19'd2	6	12		
+ + /tb_comb_adder_18b/dut/local_op23	18'd10	0	10		
+ + /tb comb adder 18b/dut/local op45	18'd18	0	¥ 18		
+ th comb adder 18b/dut/local op67	18'd26	0	126		
+ + /tb comb adder 18b/dut/local op89	18'd34	0	134		
+ > /tb comb adder 18b/dut/local opAB	18'd42	0	42		
+ > /tb comb adder 18b/dut/local opCD	18'd50	0	(50		
+ 🔶 /tb_comb_adder_18b/dut/local_opEF	18'd58	0	(58		
+ /tb_comb_adder_18b/dut/op0123	18'd12	0	12		
+ 🔶 /tb_comb_adder_18b/dut/op4567	18'd44	0	(44		
+ /tb_comb_adder_18b/dut/op89AB	18'd76	0	76		
Itb_comb_adder_18b/dut/opCDEF	18'd108	0	(10	3	
<u>+</u> /tb_comb_adder_18b/dut/local_op0123	18'd12	0		12	
How the comb_adder_18b/dut/local_op4567	18'd44	0		<u> 44</u>	
How the second secon	18'd76	0		176	
* /tb_comb_adder_18b/dut/local_opCDEF	18'd108	0		<u> 108</u>	
Itb_comb_adder_18b/dut/op01234567	18'd56	0		<u> 156</u>	
/tb_comb_adder_18b/dut/op89ABCDEF	18'd184	0		<u>, 184</u>	
	18'd56	0		<u> 156 </u>	
	18'd184	0		Ϊ18-	4
Now	85000.00				
NOW	00000 ps	os		50000 ps	

LR Accelerator Component with Avalon MM Agent Interface



Address Encoding for Read Instructions

a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	
0	0	0	0	0	0	0	0	0	0	Master Done (inform completion of operation)
0	0	0	0	0	0	0	0	0	1	N_0 (Used for w_0 computation)
0	0	0	0	0	0	0	0	1	0	N_1 (Used for w_1 computation)
0	0	0	0	0	0	0	0	1	1	D (Used for both w_0 and w_1)
0	0	0	0	0	0	0	1	0	0	S_1 (Provided for users who want all outputs)
0	0	0	0	0	0	0	1	0	1	S_2 (Provided for users who want all outputs)
0	0	0	0	0	0	0	1	1	0	S_3 (Provided for users who want all outputs)
0	0	0	0	0	0	0	1	1	1	S_4 (Provided for users who want all outputs)
0	0	0	0	0	0	1	0	0	0	S_5 (Provided for users who want all outputs)

Address Encoding for Write Instructions

a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	
0	0	0	0	0	0	0	0	0	0	Write to Address 0 in Memory $(a_{8:0} = 1)$
0	0	0	0	0	0	0	0	0	1	Write to Address 1 in Memory $(a_{8:0} = 1)$
0	0	0	0	0	0	0	0	1	0	Write to Address 2 in Memory $(a_{8:0} = 2)$
:	÷	÷	÷	÷	÷	÷	÷	÷	÷	
0	1	1	1	1	1	1	1	1	1	Write to Address 511 in Memory $(a_{8:0} = 511)$
1	0	0	0	0	0	0	0	0	0	Write Reset
1	y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0	Write Go with Count of Data y

Memory Map for Read Instructions

Memory Map for Write Instructions

Offset (Hex)	\mathbf{Bit}	Description
0	31:1	Unused
0	0	Master Done
1	31:0	Signed N_0
2	31:0	Signed N_1
3	31:0	Signed D
4	31:0	Unsigned S_1
5	31:0	Unsigned S_2
6	31:0	Unsigned S_3
7	31:0	Unsigned S_4
8	31:0	Unsigned S_5

Offset (Hex)	Bit	Description
	31:8	Unused
$0-1\mathrm{FF}$	7:4	y
	3:0	x
$200 - 2 \mathrm{FF}$	31:0	Unused

Signals and Interfaces

Component Type Xi Block Symbol Xi Fi About Signals Name Address • avalon slave 0 Avalon Memory Mapp • address • chipselect [1] chipselect • read [1] read • read [1] read • read [1] read • write [1] write • writedata [32] writedata • write [1] write • writedata [32] writedata < <add signal="">> • dock Clock Input • ck [1] clk • reset [1] reset <<add signal="">> <<add signal="">> <<<add signal="">> <<<add signal="">> <<<add signal="">> <<<add signal="">> <<<add signal="">> <<<>><<add signal="">> <<<>><<add signal="">> <<<add signal="">> <<<>><<add signal="">></add></add></add></add></add></add></add></add></add></add></add></add>	Tiles Parameters Signals & Interfaces Name: avalon_slave_0 Type: Avalon Memory Mapped Slave Associated Clock: clock Associated Reset: reset Assignments: Edit	 		avalon_slave_0 writedata[310] write chipselect address[90] read	writedata write chipselect address read	
	Block Diagram avalon_slave_0 writedata[310] write chipselect address[90] read read readdata null		WORDS ▼ clock	WORDS		null
Massagas 8	<u>j - 1</u>	111				
Info: No errors or warnings.						

Platform Designer System

🔀 System Contents 🖇 Address Map 🖇 Interconnect Requirements 🖇										
	System: soc_system Path: clk_0									
+	Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
	~		⊟ clk_0	Clock Source		_				
	X		clk_in clk_in_reset clk clk_reset □ □ hps_0 h2f_user1_clock memory hps_io h2f_reset h2f_axi_clock h2f_axi_clock h2f_axi_clock f2h_axi_clock f2h_axi_clock f2h_axi_clock f2h_axi_clock h2f_lw_axi_master □ Ir_acc_0 clock reset avalon_slave_0	Clock Input Reset Input Clock Output Reset Output Arria V/Cyclone V Hard Proce Clock Output Conduit Reset Output Clock Input AXI Master Clock Input AXI Slave Clock Input AXI Master Ir_acc Clock Input Reset Input Avalon Memory Mapped Slave	clk reset Double-click to Double-click to hps_ddr3 hps Double-click to Double-click to Double-click to Double-click to Double-click to Double-click to Double-click to Double-click to	exported clk_0 hps_0_h2 clk_0 [h2f_axi clk_0 [f2h_axi clk_0 [h2f_lw_a clk_0 [clock] [clock]		0x0000_0fff		
					III					•
	과 카 🔽 🔽 Current filter:									
A≣ Messages 🕴 – 🗗 🗖										
	Туре	Path				Messag	je			
የ 🔇		2 Info Messages								
	Image: soc_system.hps_0 HPS Main PLL counter settings: n = 0 m = 73									
	Image: soc_system.hps_0 HPS peripherial PLL counter settings: n = 0 m = 39									

Software Code

start	=	<pre>clock();</pre>

vla.go = 1; vla.address = 0;

set_lr_data(&vla);

```
for (int i = 0; i < n; i++) {
    d->data = data[i];
    vla.data = *d;
    vla.address = i;
    vla.go = 0;
```

set_lr_data(&vla);

end = clock();

<pre>start = clock();</pre>
<pre>vla.go = 1;</pre>
vla.address = n;
<pre>set_lr_data(&vla);</pre>
<pre>while (1) { read_lr_data(&obj);</pre>
<pre>if (obj.master_done == 1) break;</pre>
usleep(1); }
<pre>end = clock();</pre>

start = clock();

double v	w0	=	<pre>(double)obj.n0 /</pre>	<pre>(double)obj.d;</pre>
double v	w1	=	<pre>(double)obj.n1 /</pre>	<pre>(double)obj.d;</pre>

end = clock();

<pre>fprintf(stderr,</pre>	"Device finished processing\n")
<pre>fprintf(stderr,</pre>	"Results:\n");
<pre>fprintf(stderr,</pre>	"d: %d\n", obj.d);
<pre>fprintf(stderr,</pre>	"n0: %d\n", obj.n0);
<pre>fprintf(stderr,</pre>	"n1: %d\n", obj.n1);
<pre>fprintf(stderr,</pre>	"s1: %d\n", obj.s1);
<pre>fprintf(stderr,</pre>	"s2: %d\n", obj.s2);
<pre>fprintf(stderr,</pre>	"s3: %d\n", obj.s3);
<pre>fprintf(stderr,</pre>	"s4: %d\n", obj.s4);
<pre>fprintf(stderr,</pre>	"s5: %d\n", obj.s5);
<pre>fprintf(stderr,</pre>	"Weights:\n");
<pre>fprintf(stderr,</pre>	"w0: %f\n", w0);
<pre>fprintf(stderr,</pre>	"w1: %f\n", w1);

Software Code

```
void set_lr_data(const lr_acc_arg_t *d)
{
    if (ioctl(lr_acc_fd, LR_ACC_WRITE_DATA, d))
    {
      fprintf(stderr, "ioctl(LR_ACC_SET_DATA) failed");
      return;
    }
```

void read_lr_data(lr_acc_read_data_t *d)

if (ioctl(lr_acc_fd, LR_ACC_READ_DATA, d))
{

fprintf(stderr, "ioctl(LR_ACC_GET_DATA) failed");
return;

static void read_data(lr_acc_read_data_t *data)

int a = ioread32(dev.virtbase + 0); int b = ioread32(dev.virtbase + 4); int c = ioread32(dev.virtbase + 8); int d = ioread32(dev.virtbase + 12); int e = ioread32(dev.virtbase + 16); int f = ioread32(dev.virtbase + 20); int g = ioread32(dev.virtbase + 24); int h = ioread32(dev.virtbase + 28); int i = ioread32(dev.virtbase + 32);

data->master_done = a; data->d = b; data->n0 = c; data->n1 = d; data->s1 = e; data->s2 = f; data->s3 = g; data->s4 = h; data->s5 = i;

Software Code

static void write_data(lr_acc_arg_t *data)

```
if (data->go) {
    iowrite32((u32)1, dev.virtbase + 4 * ((1 << 9) + data->address));
} else {
    iowrite32((u32)data->data.data, dev.virtbase + 4 * data->address);
}
```

typedef struct {
 char data;
} lr_acc_data_t;

{

typedef struct {

lr_acc_data_t data; int address; char go;

} lr_acc_arg_t;

typedef struct {
 int master_done;
 int d, n0, n1, s1, s2, s3, s4, s5;
} lr_acc_read_data_t;

Validation of Accelerator



FILIVIELTICS							
Slope	:	-0.560855					
Intercept	:	9.455583					
R2 Score	:	0.951513					
MSE	:	0.323244					
Fit Metrics							
Slope	:	-0.525380					
ntercept	:	9.183580					
R2 Score	•	0.948967					
MSE	:	0.340218					

Param.		C Code (Local)	FGPA Accelerator
S1	:	432	432
S2	:	3440	3440
S3	:	2160	2160
S4	:	37294	37294
S5	:	11998	11998
NO	:	39281920	39281920
N1	:	-2247264	-2247264
D	:	4277408	4277408
W0	:	9.183580	9.183580
W1	•	-0.525380	-0.525380

The various intermediate as well as final output generated by the accelerator match the ones generated by the baseline C code.

Runtime Analysis of Accelerator

- > Runtime in Local Machine [Intel i9] (in Python): $154\mu s$
- > Runtime in FPGA (in C): $28.9\mu s$
- Runtime with Accelerator:
 - > Sending Data: $340 \mu s$
 - > Processing and Reading Data: $9\mu s$
 - > Calculating Weights (Floating Operation): $2\mu s$

◙ - ☞ 🖬 ॐ ቆ ໍ ዜ 🖻 🛍 ቧ 🗠 ◎ - М 🗄 🛛 🕸 🕮 🚑 🕅	🕒 🛧 👄 🕸 mittebere tebe tat 1996 tat 1995 tat 1996 tet 1997 b 1997 1996 tet 1997 b 1997 ann 1986 tet 1997 b
<u>.</u>	Msgs
/tb_lr_acc_512_8b/clk	1'd0
<pre> //tb_lr_acc_512_8b/reset </pre>	1'd0
▪- 🤣 /tb_lr_acc_512_8b/writedata	32'dx
/tb_lr_acc_512_8b/write	
/tb_ir_acc_512_8D/read	
$\mathbf{F}_{\mathbf{A}}$ // b Ir acc 512 8b/address	
■	32'd2480 - 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
+ / /tb_lr_acc_512_8b/write_mem/addr	10'd31
🖬 🧇 /tb_lr_acc_512_8b/write_mem/x_val	4'd15
	4'd14
//tb_lr_acc_512_8b/read_result/addr	
/tb_lr_acc_512_8b/read_result/result_data	
+ 7D_IF_acc_512_8D/#UDIK#1/2162/4#101/fead_Val	
$\frac{1}{2}$ / $\frac{1}{2}$ / $\frac{1}{2}$ - $\frac{1}$	1/10
$= \frac{1}{\sqrt{2}}$ /tb Ir acc 512 8b/dut/writedata	32'dx
//tb_lr_acc_512_8b/dut/write	1'd0
<pre>//b_lr_acc_512_8b/dut/read</pre>	
<pre> /tb_lr_acc_512_8b/dut/chipselect</pre>	
🗈 🖆 /tb_lr_acc_512_8b/dut/address	10'dx
/tb_lr_acc_512_8b/dut/readdata	32'd2480
/tb_lr_acc_512_8b/dut/master_done	
	{4 d0 4 d0} {4 d
4/th in acc 512 8b/dut/go	1/10
■/tb Ir acc 512 8b/dut/pc	9'd0
//////////////////////////////////////	1'd0
/tb_lr_acc_512_8b/dut/done_trigger	1'd0
	18'd32
/tb_lr_acc_512_8b/dut/global_s_1	
+	
$\frac{1}{2}$ / $\frac{1}{2}$	
$\mathbf{F}_{\mathbf{a}} = \frac{1}{2} / 10 \text{ Jr}_{\mathbf{a}} = \frac{1}{2} - \frac$	
	32'd21760
#- / tb_lr_acc_512_8b/dut/s3s4_minus_s2s5	32'd71680
/tb_Ir_acc_512_8b/dut/s1s5_minus_s2s3	32'd10752 1 051 1 051
★ ✓ /tb_lr_acc_512_8b/dut/start	16'd0 <u>1</u>
/tb_lr_acc_512_8b/dut/local_done	
+ γ /tb_ir_acc_512_8b/dut/dut_num	140X
/tb_r_acc_512_8b/dut/addr	9/d511
/tb ir acc 512 8b/dut/global start adders	1'd1
₽-♦//tb_lr_acc_512_8b/dut/local_global_s_1	18'd32
🖬 🥠 /tb_lr_acc_512_8b/dut/local_global_s_2	18'd240
tb_lr_acc_512_8b/dut/local_global_s_3	18'd224
//tb_lr_acc_512_8b/dut/local_global_s_4	18'd2480
Yub_Ir_acc_512_8b/dut/local_global_s_5	
Provide the second structure stru	
$\mathbf{F}_{\mathbf{A}}$ /tb ir acc 512 8b/dut/local s3 times s4	36'd555520
It in acc 512 8b/dut/local s2 times s5	36'd483840
	36'd64512
rtb_lr_acc_512_8b/dut/local_s2_times_s3	36'd53760
tb_lr_acc_512_8b/dut/s1_times_s4	36'd79360
/tb_lr_acc_512_8b/dut/s2_squared	36'd57600 (m
+ 7/tb_lr_acc_512_8b/dut/s3_times_s4	36/0555520
	36/d64512
$\mathbf{F}_{\mathbf{A}}$ /tb in acc 512 8b/dut/s2 times s3	36'd53760

