



AccelReg: Accelerator for Linear Regression

CSEE 4840: Embedded Systems

Pranav Asuri

pa2708@columbia.edu

Venkat Suprabath Bitra

vsb2127@columbia.edu

Varsha Keshava Prasad

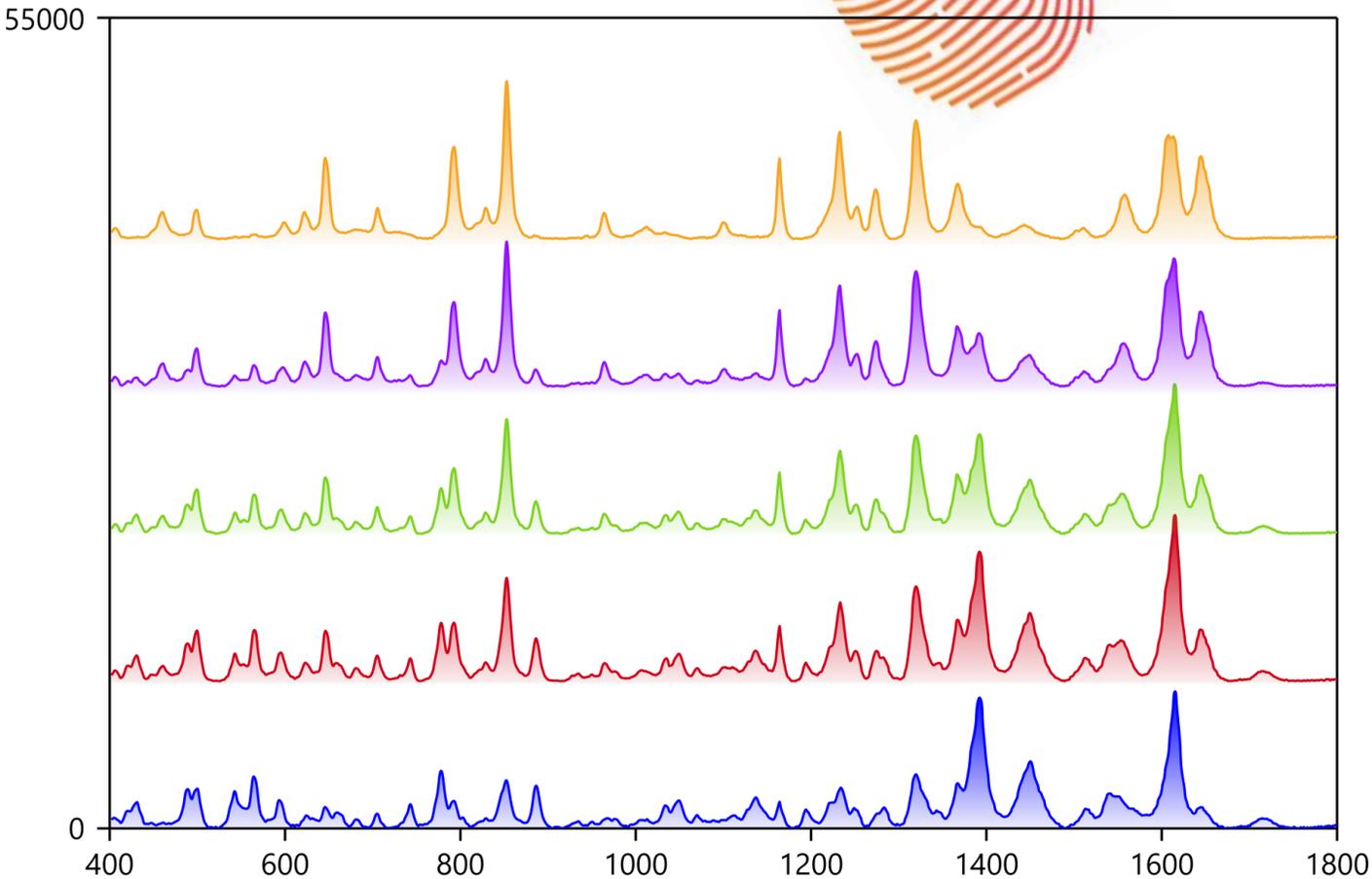
vk2550@columbia.edu

Doreen Sisanalli

ds4371@columbia.edu

Background

Molecular Fingerprint



- Unique spectral "fingerprints" for identifying chemical compounds and their compositions
- Concentration of components can be determined by analyzing the intensity ratios of their prominent Raman peaks
- Ratio of these characteristic peaks exhibits a near-linear relationship with the concentration

Motivation

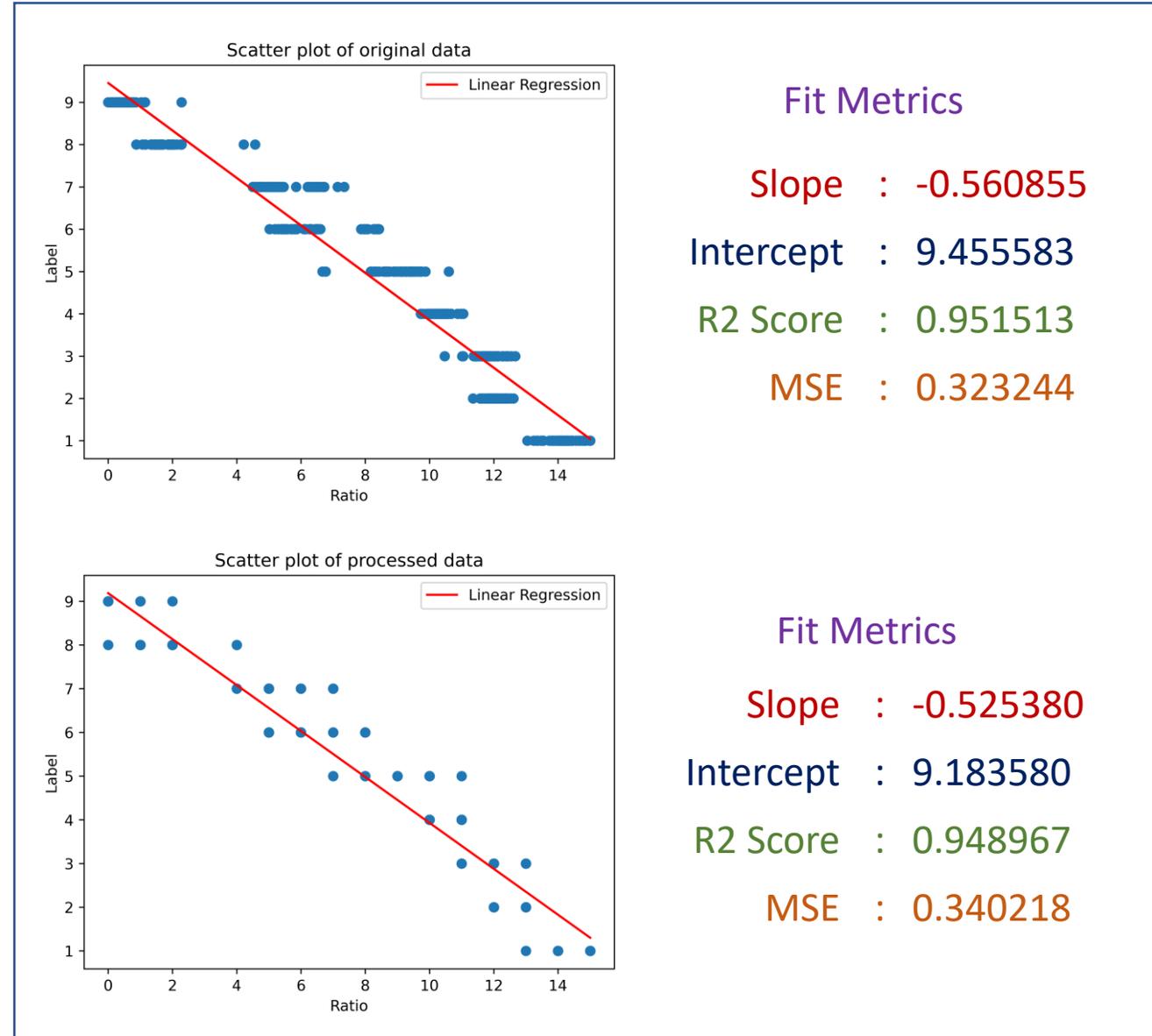
- Instrument makers need on-board hardware for direct quantitative analysis, avoiding external PCs
- Portable spectrometers gather data fast, requires quick, real-time parameters for point-of-care applications
- Scientists prefer automated tools, typically avoiding to work directly with code for ease 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 observations (x_i, y_i) $y = w_0 + w_1x$

$$y_i = [1 \quad x_i] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \quad \mathbf{X}^T \mathbf{y} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

$$(\mathbf{X}^T \mathbf{X})^{-1} = \frac{1}{n \sum x_i^2 - (\sum x_i)^2} \begin{bmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & n \end{bmatrix}$$

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

$S_1 = n$ (count of observations)

$S_2 = \sum_{i=1}^n x_i$ (sum of x_i)

$S_3 = \sum_{i=1}^n y_i$ (sum of y_i)

$S_4 = \sum_{i=1}^n x_i^2$ (sum of x_i^2)

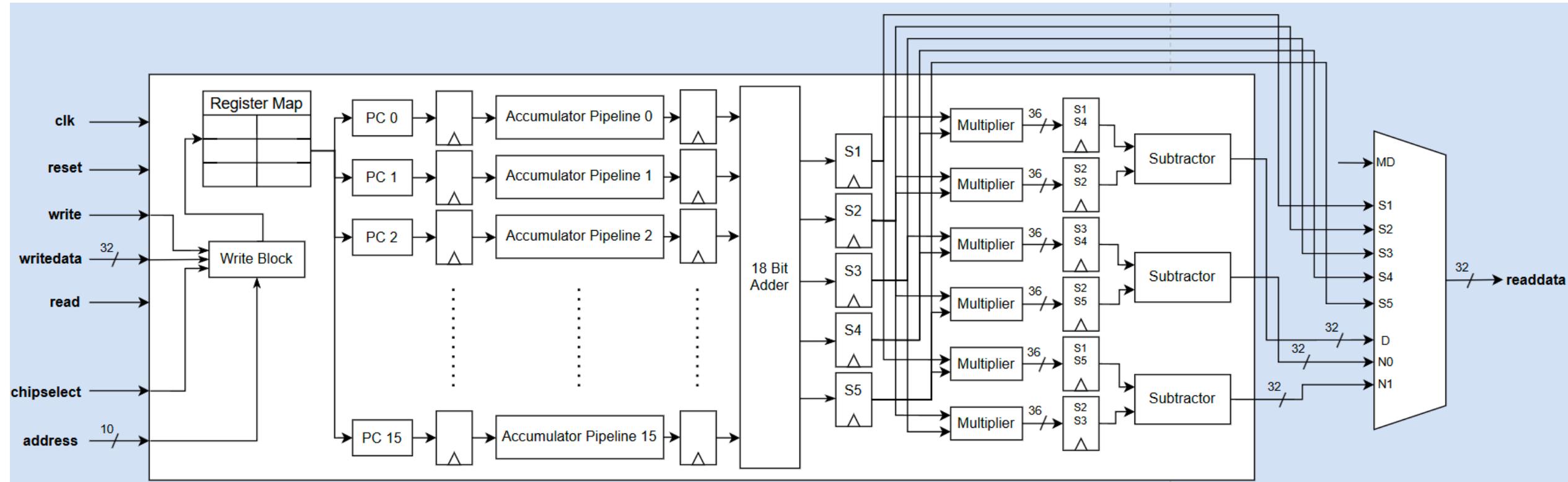
$S_5 = \sum_{i=1}^n x_i y_i$ (sum of $x_i y_i$)

➤ Use accumulators and multipliers in int4.

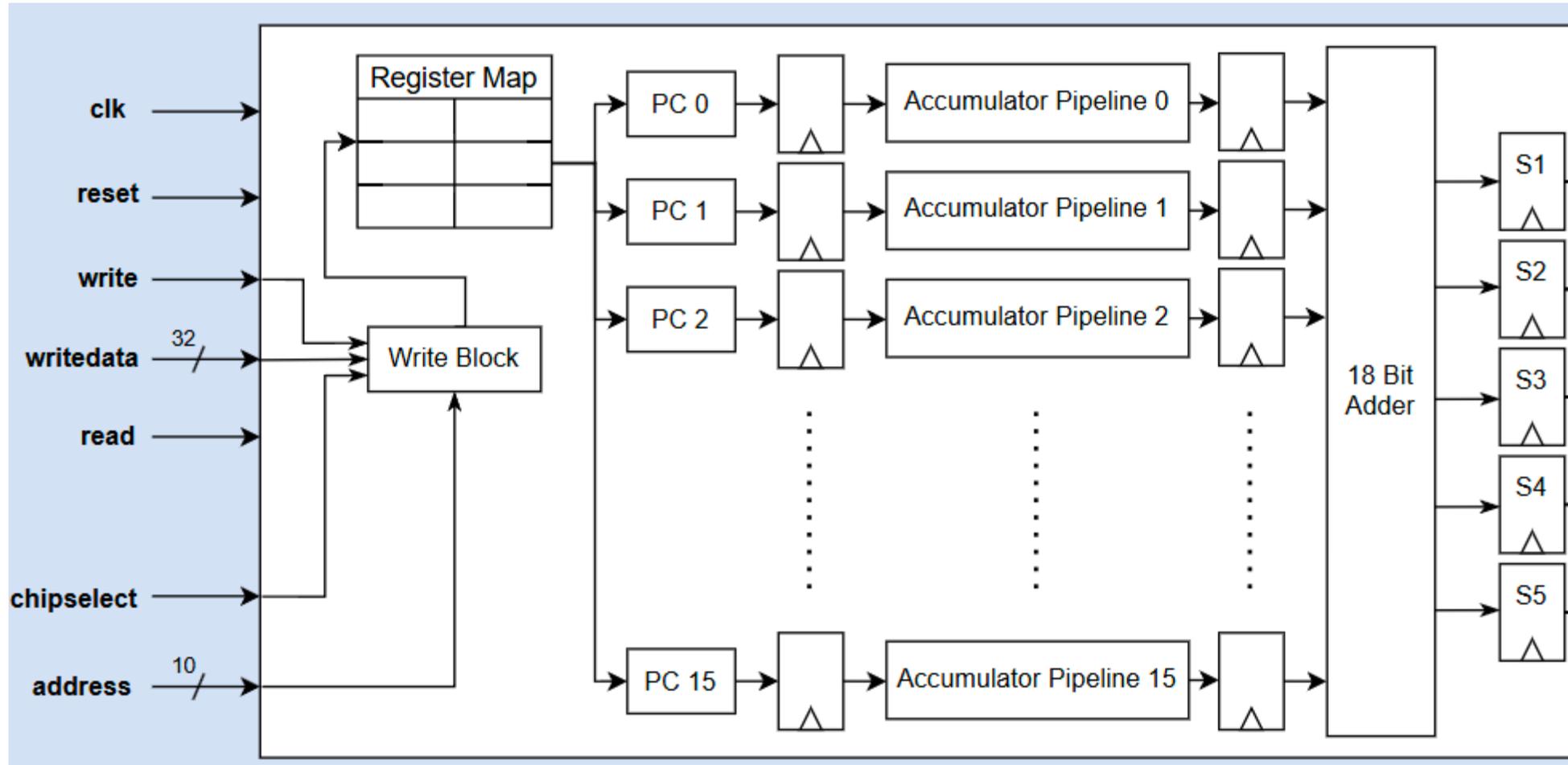
➤ Convert to float at the end for precision.

LR Accelerator Component (Block Diagram)

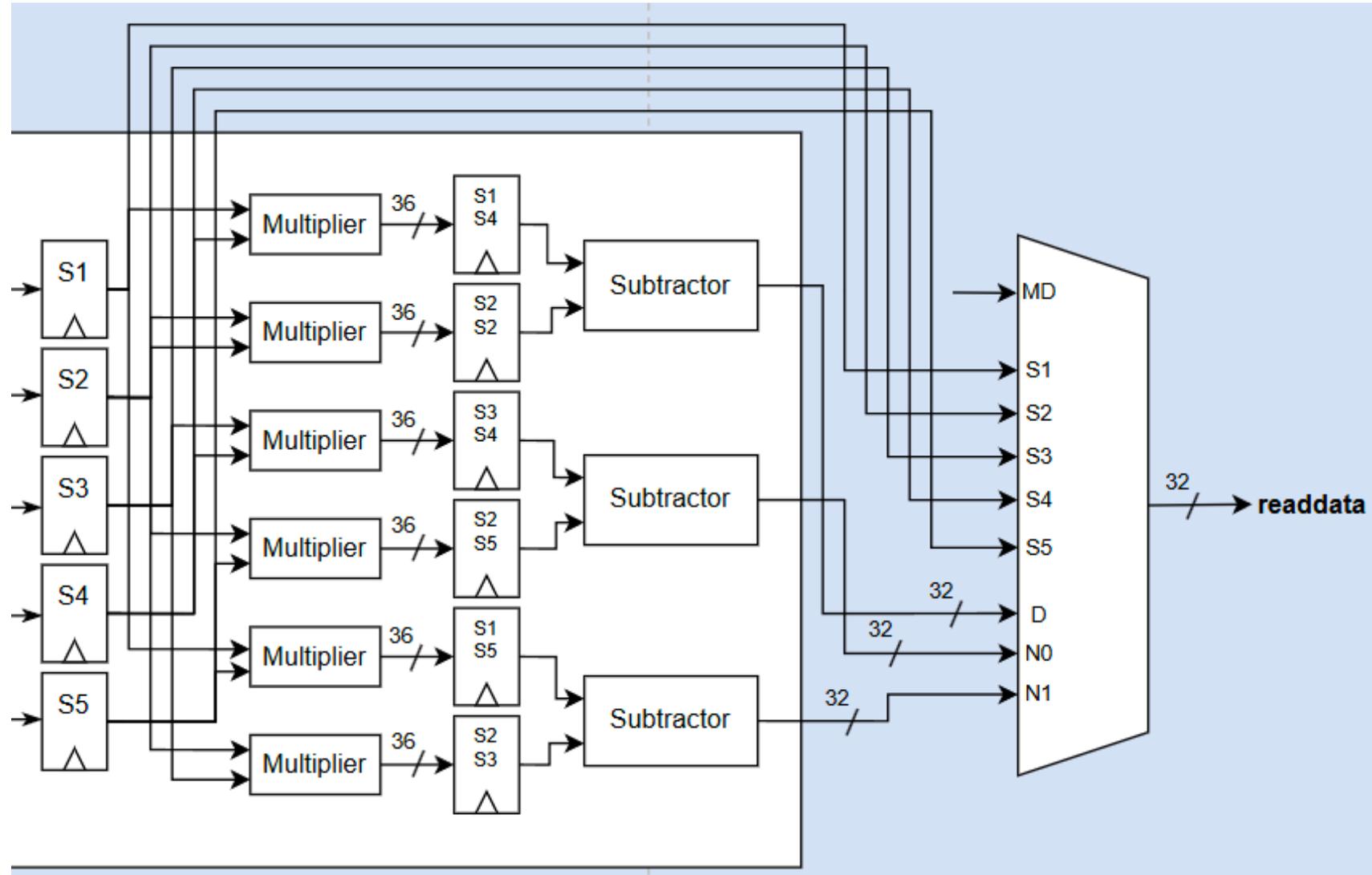
- 16 parallel pipelines accumulate data
- Processing unit computes products
- Adder sums pipeline outputs
- Register interface controls data flow



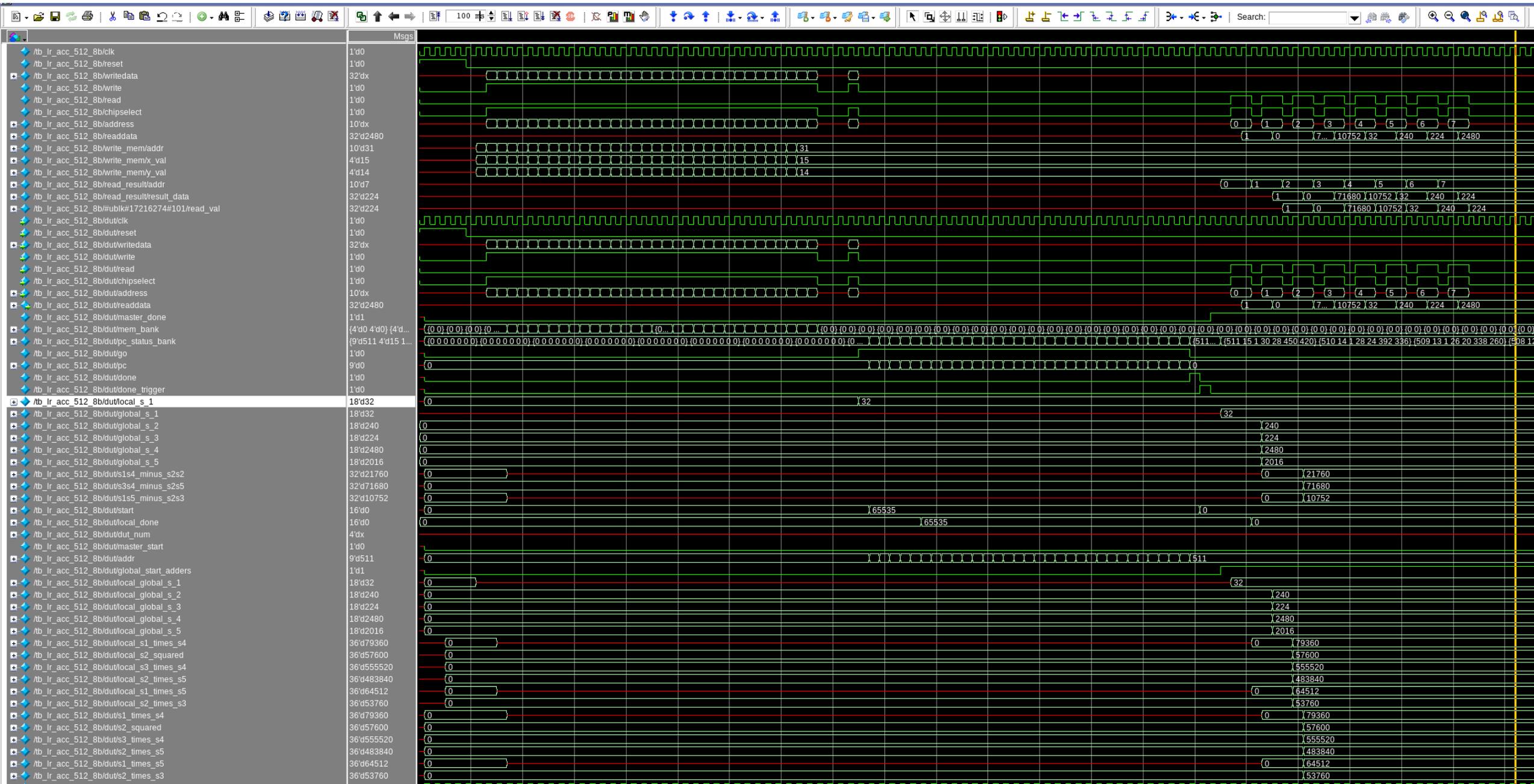
LR Accelerator Component (Accumulator Block)



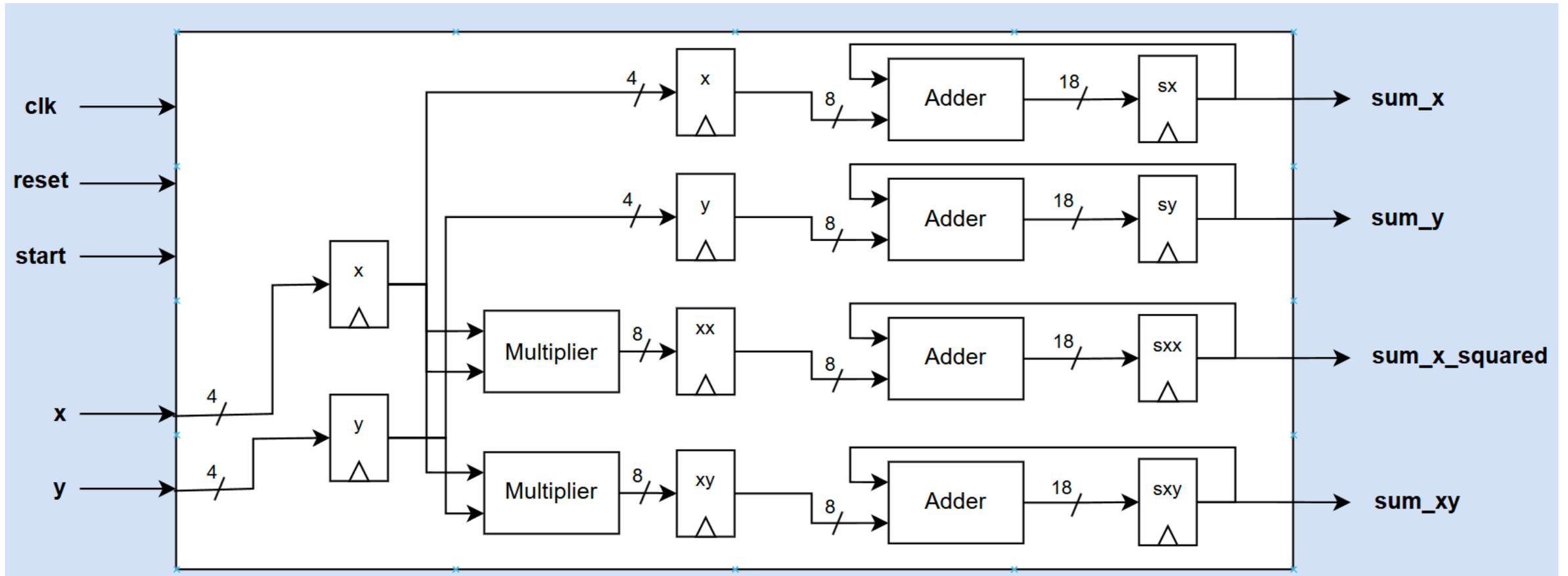
LR Accelerator Component (Weight Computation Block)



Timing Diagram for LR Accelerator Component



Accumulator Pipeline (Block Diagram)



Accumulator Pipeline (Analysis)

Fitter Summary

Fitter Summary	
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 Analyzer (Fmax)

	Fmax	Restricted Fmax	Clock Name	Note
1	335.23 MHz	310.08 MHz	clk	lim...in)

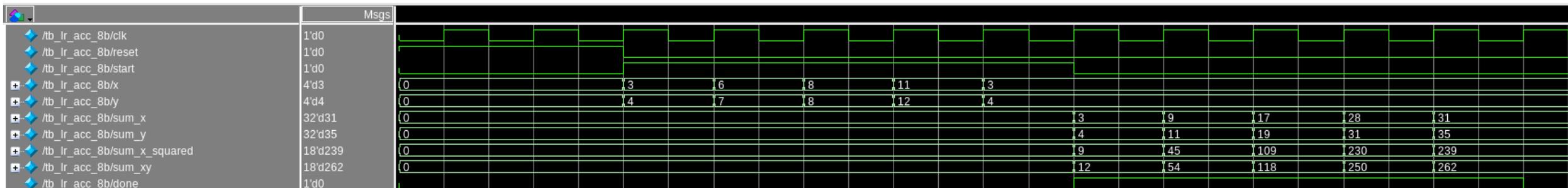
Tasks

Task	Time
Compile Design	00:00:45
Analysis & Synthesis	00:00:11
Fitter (Place & Route)	00:00:20
Assembler (Generate programming files)	00:00:07
Timing Analysis	00:00:07
EDA Netlist Writer	
Edit Settings	
Program Device (Open Programmer)	

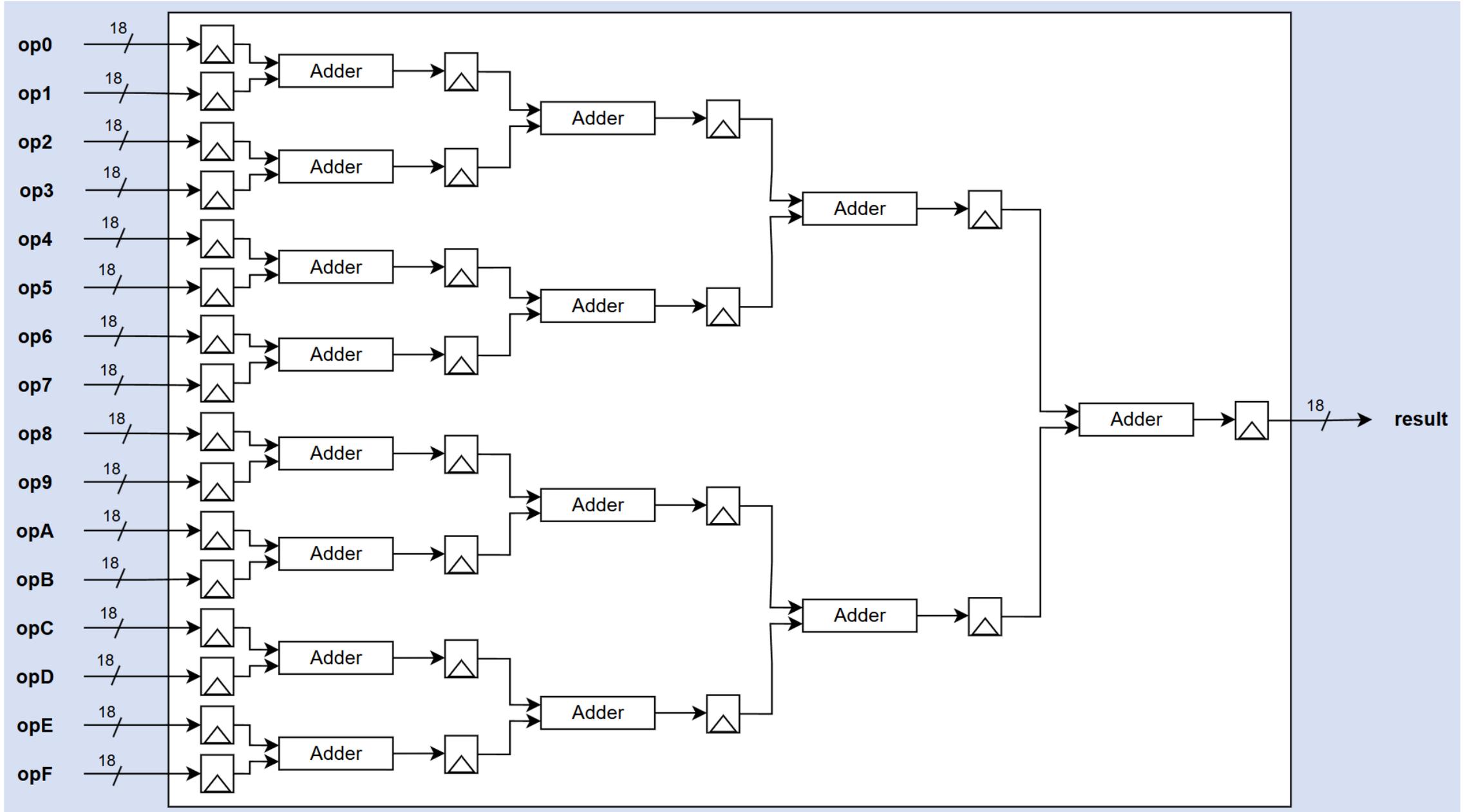
Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Pin-Out File
- Resource Section
- I/O Rules Section
- Device Options
- Operating Settings and Condi
- Estimated Delay Added for Hc
- Messages
- Suppressed Messages
- Flow Messages
- Flow Suppressed Messages
- Assembler
- Timing Analyzer
 - Summary
 - Parallel Compilation
 - SDC File List
 - Clocks
 - Slow 1100mV 85C Model
 - Fmax Summary
 - Timing Closure Recomm
 - Setup Summary
 - Hold Summary
 - Recovery Summary
 - Removal Summary
 - Minimum Pulse Width Sum
 - Metastability Summary
 - Slow 1100mV OC Model
 - Fast 1100mV 85C Model
 - Fast 1100mV OC Model

Timing Diagram



16 Number 18 Bit Adder Block



18 Bit Adder Block (Analysis)

Timing Analyzer (Fmax)

Compilation Report - comb_adder_18b_wrapper x comb_adder_18b sv x

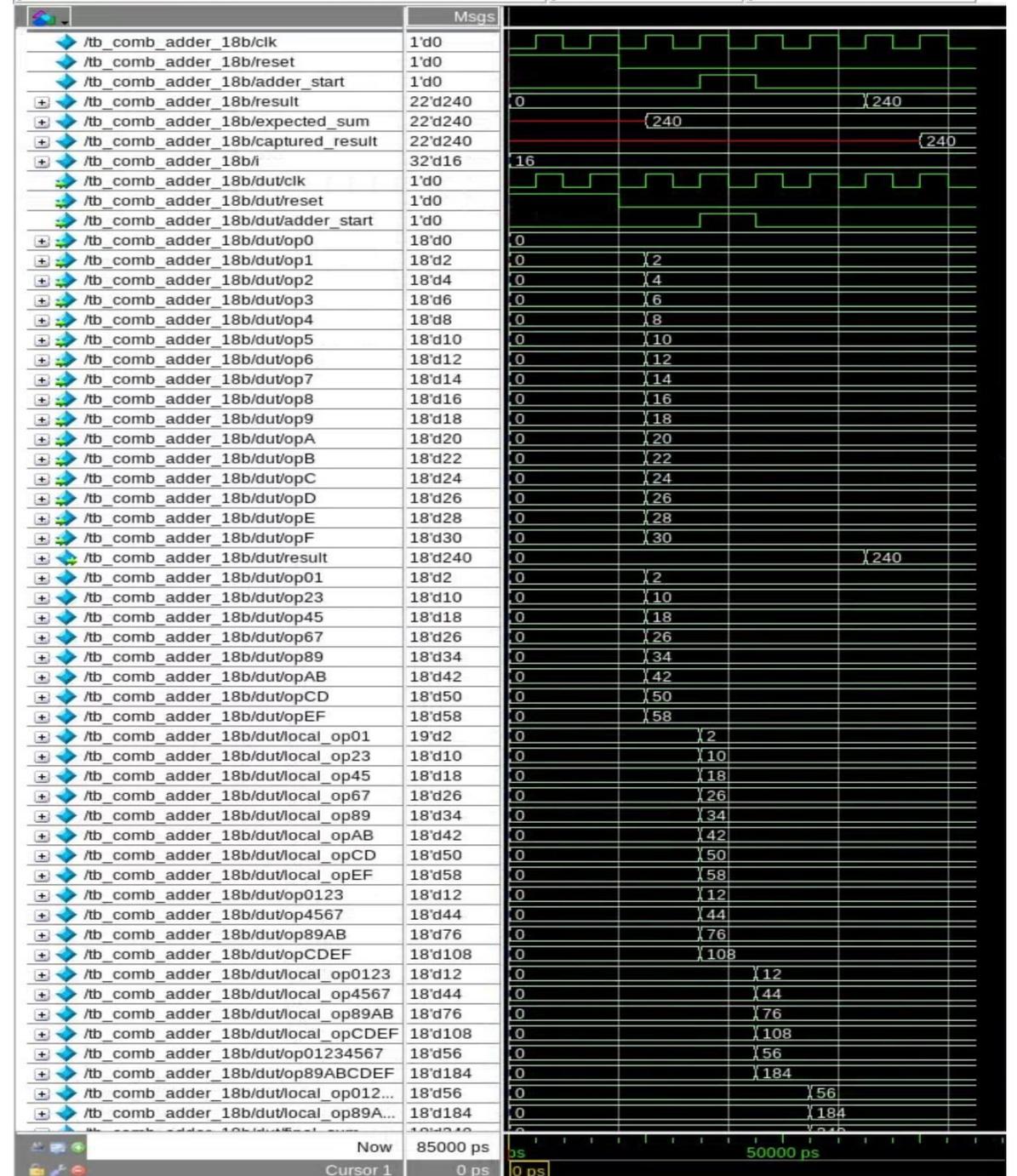
Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
 - Fitter
 - Flow Messages
 - Flow Suppressed Messages
 - Assembler
 - Timing Analyzer
 - Summary
 - Parallel Compilation
 - SDC File List
 - Clocks
 - Slow 1100mV 85C Model
 - Fmax Summary
 - Timing Closure Recommender
 - Setup Summary
 - Hold Summary
 - Recovery Summary
 - Removal Summary
 - Minimum Pulse Width Summary
 - Metastability Summary
 - Slow 1100mV OC Model
 - Fast 1100mV 85C Model
 - Fast 1100mV OC Model
 - Multicorner Timing Analysis Summary
 - Advanced I/O Timing
 - Clock Transfers

Slow 1100mV 85C Model Fmax Summary

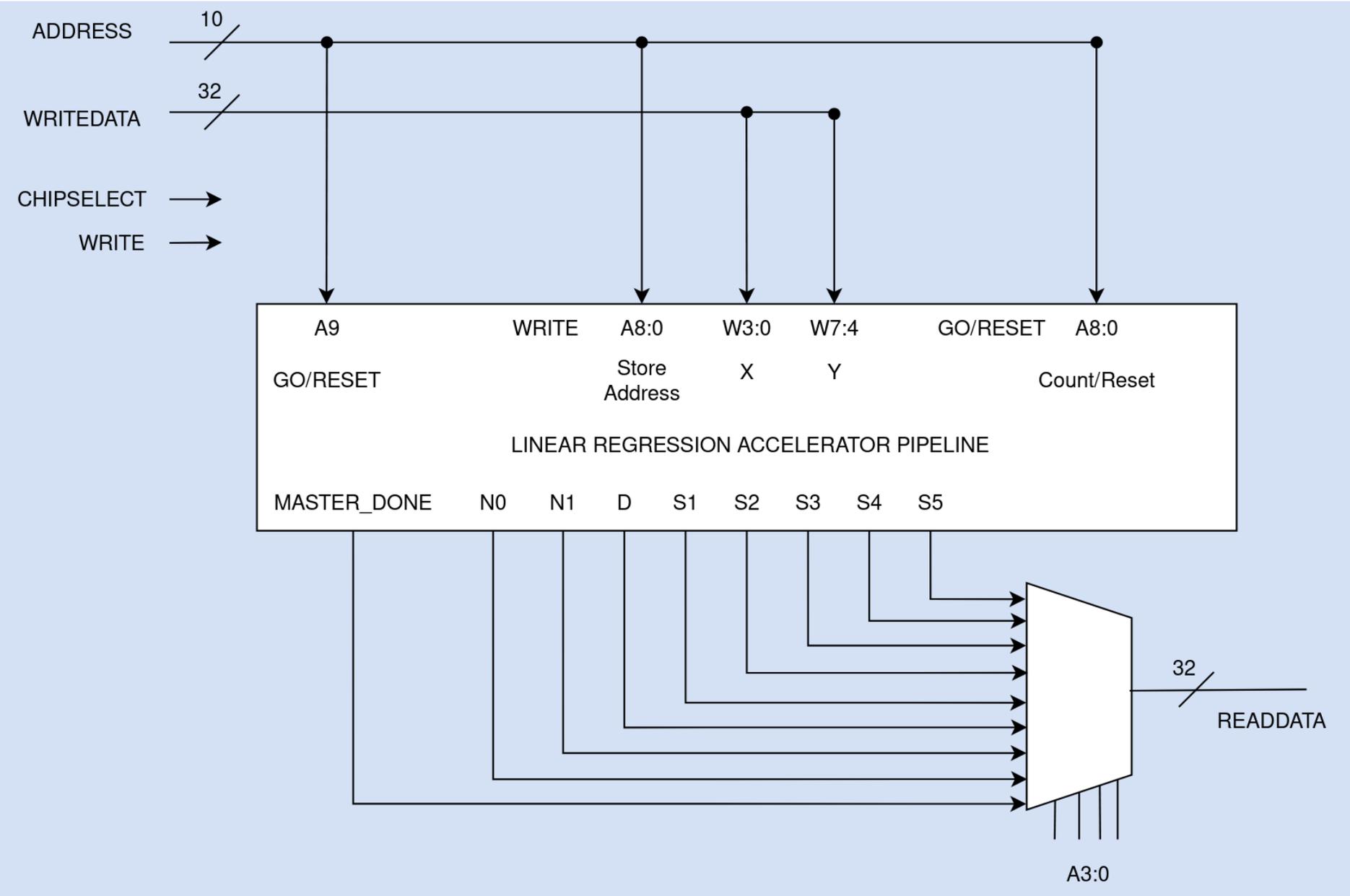
	Fmax	Restricted Fmax	Clock Name	Note
1	377.5 MHz	377.5 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of that the duty cycle (in terms of a percentage) is maintained. Altera



Timing Diagram

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 ($a_{8:0} = 2$)
\vdots										
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

Offset (Hex)	Bit	Description
0	31:1	Unused
	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

Memory Map for Write Instructions

Offset (Hex)	Bit	Description
0 – 1FF	31:8	Unused
	7:4	y
	3:0	x
200 – 2FF	31:0	Unused

Signals and Interfaces

Component Type Block Symbol Files Parameters **Signals & Interfaces**

▶ About Signals

Name

- ▶ **avalon_slave_0** *Avalon Memory Mapped*
 - ▶ address [10] *address*
 - ▶ chipselect [1] *chipselect*
 - ▶ read [1] *read*
 - ▶ readdata [32] *readdata*
 - ▶ write [1] *write*
 - ▶ writedata [32] *writedata*
 - <<add signal>>
- ▶ **clock** *Clock Input*
 - ▶ clk [1] *clk*
- ▶ **reset** *Reset Input*
 - ▶ reset [1] *reset*
 - <<add signal>>
 - <<add interface>>

Name: Documentation

Type: Avalon Memory Mapped Slave

Associated Clock: clock

Associated Reset: reset

Assignments:

Block Diagram

Parameters

Address units: WORDS

Associated clock: clock

Associated reset: reset

Bits per symbol: 8

Burstcount units: WORDS

Explicit address span: 00000000000000000000

Timing

Setup: 0

Read wait: 1

Write wait: 0

Hold: 0

Timing units: Cycles

Pipelined Transfers

Read latency: 0

Maximum pending read transactions: 0

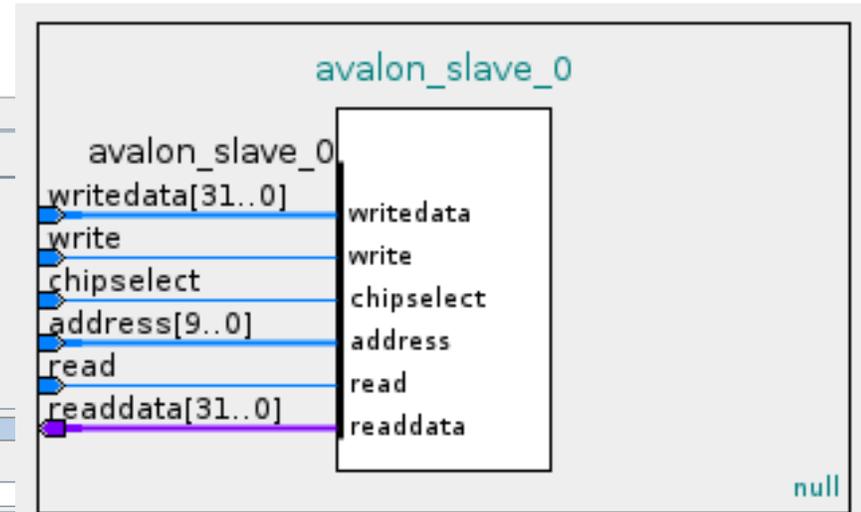
Maximum pending write transactions: 0

Burst on burst boundaries only

Linewrap bursts

Messages

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	
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> clk_0	Clock Source							
		clk_in	Clock Input	clk	<i>exported</i>					
		clk_in_reset	Reset Input	reset		clk_0				
		clk	Clock Output	<i>Double-click to</i>						
		clk_reset	Reset Output	<i>Double-click to</i>						
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> hps_0	Arria V/Cyclone V Hard Proce...							
		h2f_user1_clock	Clock Output	<i>Double-click to</i>		hps_0_h2...				
		memory	Conduit	hps_ddr3						
		hps_io	Conduit	hps						
		h2f_reset	Reset Output	<i>Double-click to</i>						
		h2f_axi_clock	Clock Input	<i>Double-click to</i>	clk_0					
		h2f_axi_master	AXI Master	<i>Double-click to</i>	[h2f_axi_...					
		f2h_axi_clock	Clock Input	<i>Double-click to</i>	clk_0					
	f2h_axi_slave	AXI Slave	<i>Double-click to</i>	[f2h_axi_...						
	h2f_lw_axi_clock	Clock Input	<i>Double-click to</i>	clk_0						
	h2f_lw_axi_master	AXI Master	<i>Double-click to</i>	[h2f_lw_a...						
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> lr_acc_0	lr_acc								
	clock	Clock Input	<i>Double-click to</i>	clk_0						
	reset	Reset Input	<i>Double-click to</i>	[clock]						
	avalon_slave_0	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clock]	<input checked="" type="checkbox"/> 0x0000_0000		0x0000_0fff			

Current filter:

Messages

Type	Path	Message
2 Info Messages		
soc_system.hps_0		HPS Main PLL counter settings: n = 0 m = 73
soc_system.hps_0		HPS peripheral PLL counter settings: n = 0 m = 39

Software Code

```
start = clock();

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();
```

```
start = clock();
vla.go = 1;
vla.address = n;
set_lr_data(&vla);

while (1)
{
    read_lr_data(&obj);

    if (obj.master_done == 1)
        break;

    usleep(1);
}

end = clock();
```

```
start = clock();

double w0 = (double)obj.n0 / (double)obj.d;
double w1 = (double)obj.n1 / (double)obj.d;

end = clock();

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

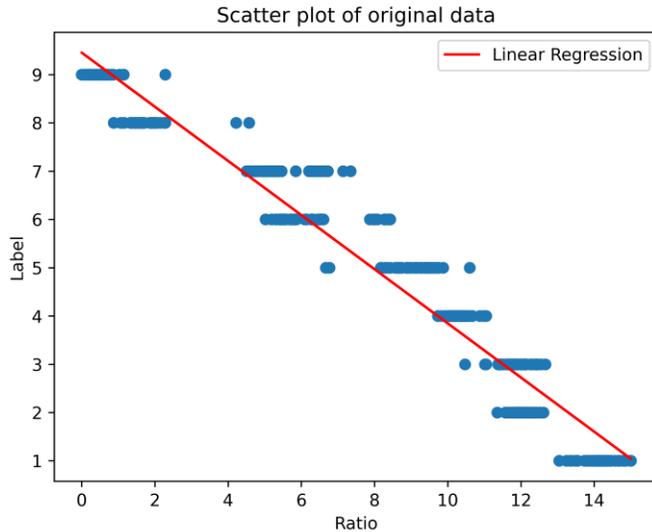
```
static void write_data(lr_acc_arg_t *data)
{
    if (data->go) {
        iowrite32((u32)1, dev.virtbase + 4 * ((1 << 9) + data->address));
    } else {
        iowrite32((u32)data->data.data, dev.virtbase + 4 * data->address);
    }
}
}
```

```
typedef struct {
    char data;
} lr_acc_data_t;
```

```
typedef struct {
    lr_acc_data_t data;
    int address;
    char go;
} lr_acc_arg_t;
```

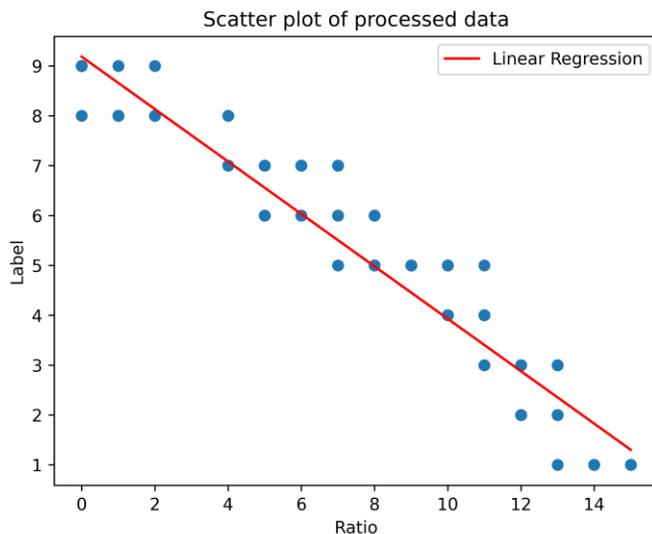
```
typedef struct {
    int master_done;
    int d, n0, n1, s1, s2, s3, s4, s5;
} lr_acc_read_data_t;
```

Validation of Accelerator



Fit Metrics

Slope : -0.560855
Intercept : 9.455583
R2 Score : 0.951513
MSE : 0.323244



Fit Metrics

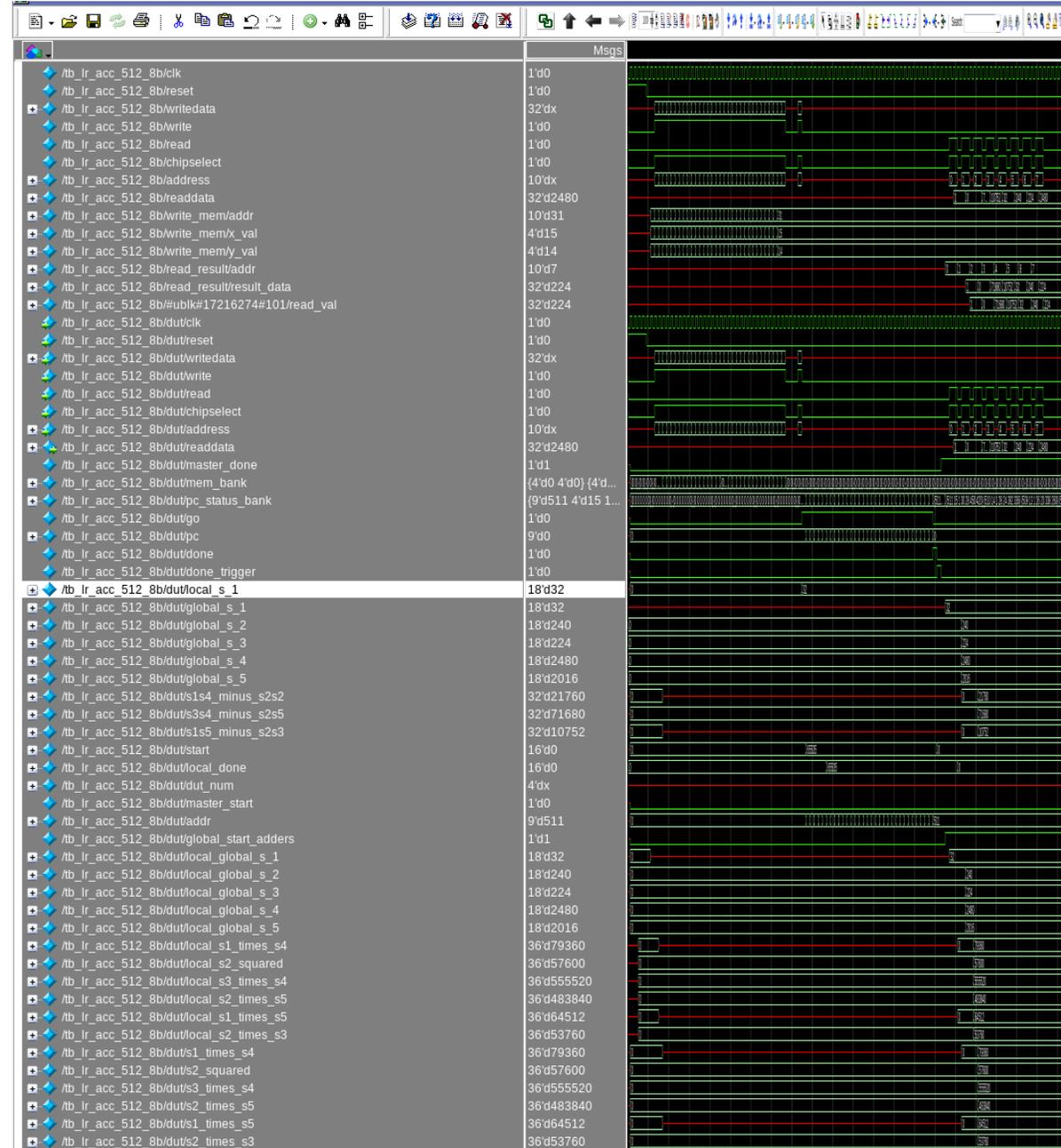
Slope : -0.525380
Intercept : 9.183580
R2 Score : 0.948967
MSE : 0.340218

Param.	C Code (Local)	FGPA Accelerator
S1	432	432
S2	3440	3440
S3	2160	2160
S4	37294	37294
S5	11998	11998
N0	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): $154\mu\text{s}$
- Runtime in FPGA (in C): $28.9\mu\text{s}$
- Runtime with Accelerator:
 - Sending Data: $340\mu\text{s}$
 - Processing and Reading Data: $9\mu\text{s}$
 - Calculating Weights (Floating Operation): $2\mu\text{s}$



THANKS