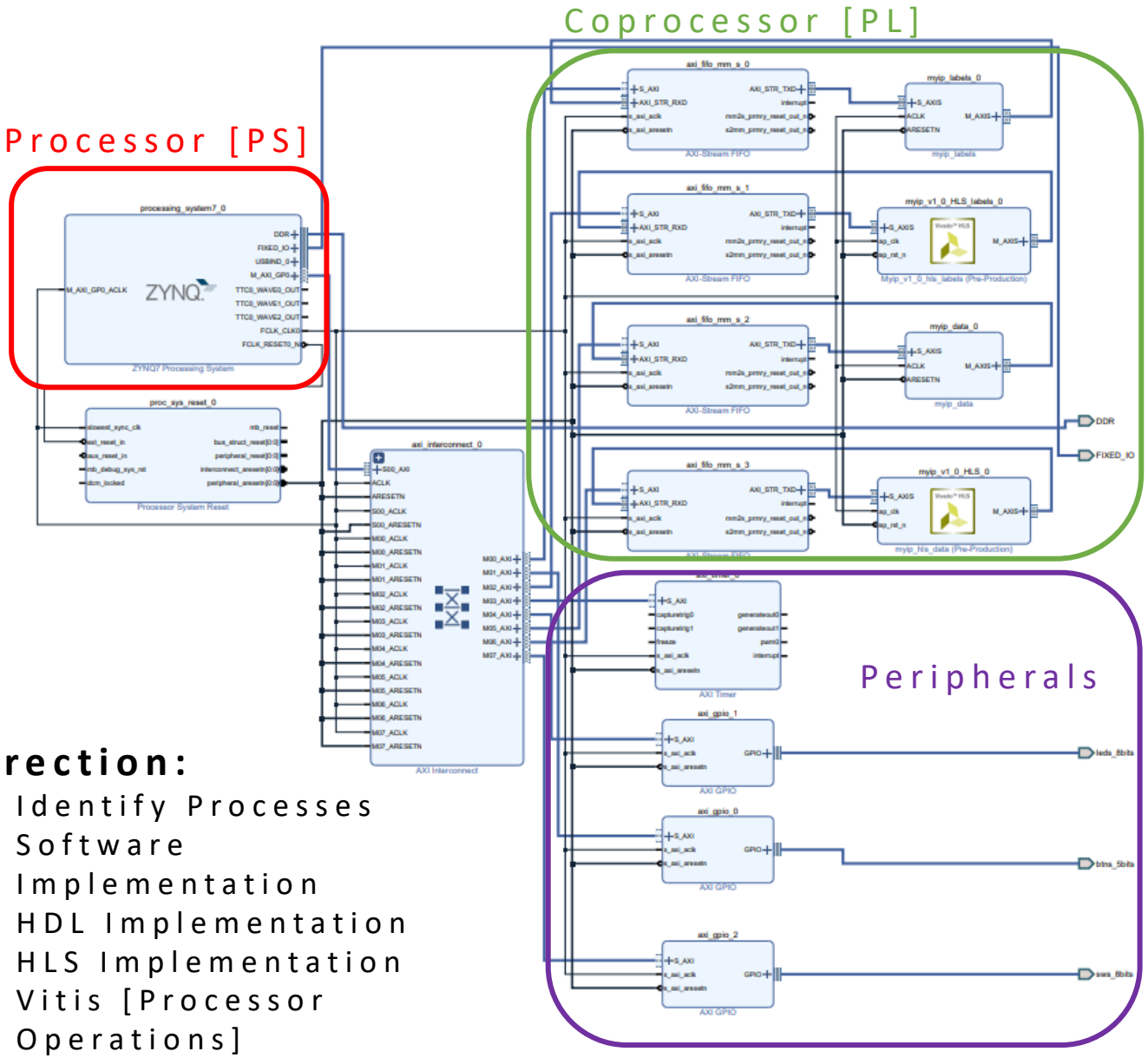
EE4218 MLP PROJECT OVERVIEW



Direction:

1. Identify Processes
2. Software Implementation
3. HDL Implementation
4. HLS Implementation
5. Vitis [Processor Operations]

Processor:

- Data Retrieval
- Software Implementation
- Data Sending To Coprocessors
- Data Classifying/Decoding
- Data Checking [Labels]
- Peripheral Interactions
- User Interface

Coprocessor(s):

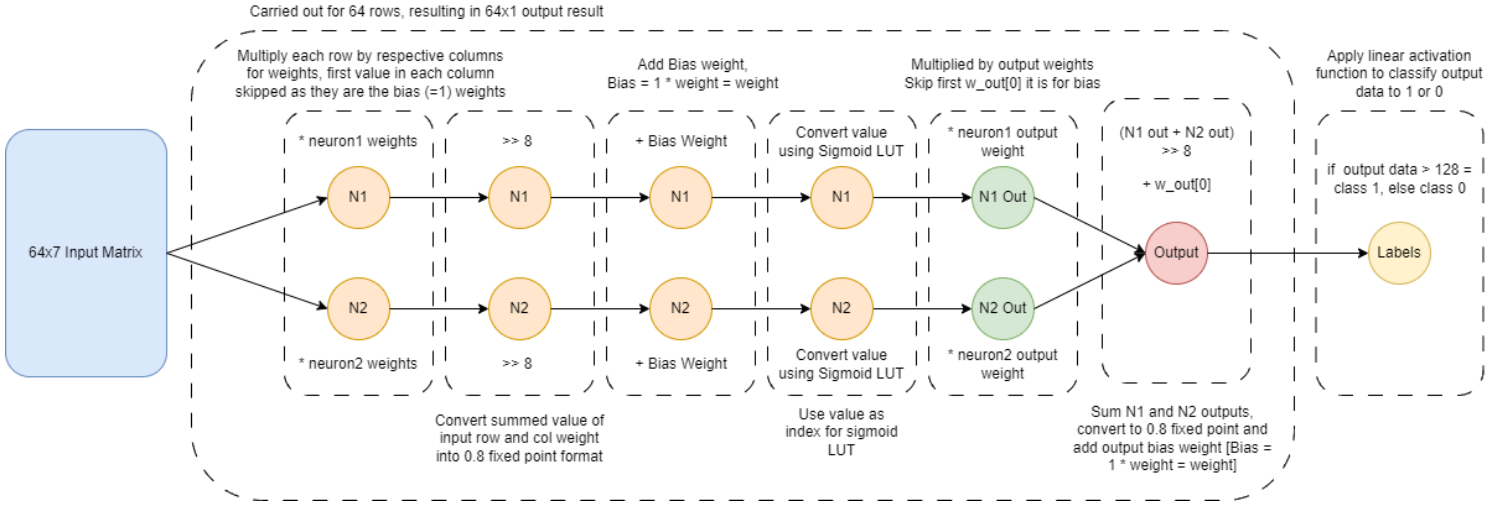
- 10ns / Clock Cycle
- Hardware Implementation
  - Data Retrieval From FIFO
  - HDL
  - HLS

Peripherals:

- Buttons [User Interfacing]
- LED [User Interfacing]
- Timer [Analysis Of Performance]

Software

Identified Process:



Software Implementation:

```
for (test_case_cnt=0 ; test_case_cnt < NUMBER_OF_TEST_VECTORS ; test_case_cnt++){

    for(word_cnt = 0; word_cnt < input_matrix_size; word_cnt++){
        neuron1_weight = test_input_memory[input_matrix_size + (neuron1_weight_index%14) + 2];  //index from 2, 4, ... , 14
        neuron2_weight = test_input_memory[input_matrix_size + (neuron2_weight_index%14) + 2];  //index from 3, 5, ... , 15

        neuron1_weight_index = neuron1_weight_index + 2;
        neuron2_weight_index = neuron2_weight_index + 2;

        neuron1 += test_input_memory[word_cnt]*neuron1_weight;
        neuron2 += test_input_memory[word_cnt]*neuron2_weight;


        if((neuron1_weight_index%14) == 0){
            neuron1 = (neuron1 >> 8) + test_input_memory[input_matrix_size];
            neuron2 = (neuron2 >> 8) + test_input_memory[input_matrix_size + 1];

            neuron1 = test_input_memory[input_matrix_size + hidden_layer_size + output_layer_size + neuron1];
            neuron2 = test_input_memory[input_matrix_size + hidden_layer_size + output_layer_size + neuron2];

            neuron1 = (neuron1 * test_input_memory[input_matrix_size + hidden_layer_size + 1]);
            neuron2 = (neuron2 * test_input_memory[input_matrix_size + hidden_layer_size + 2]);

            soft_result_memory[expected_result_index] = (((neuron1 + neuron2) >> 8) + (test_input_memory[input_matrix_size + hidden_layer_size]));

            neuron1 = 0;
            neuron2 = 0;
            expected_result_index++;
        }
    }
}
```

- Software Implementation executed within the processor
- Time: 68890ns
  - Direct access to data for computation [no FIFO overhead]
  - No direct control of resource used for computation

HDL Implementation

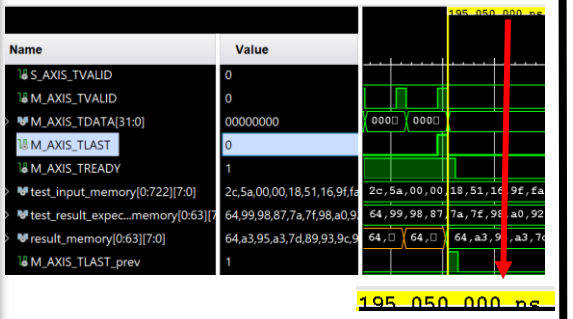
Design 1: Raw calculation

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	568	0	53200	1.07
LUT as Logic	536	0	53200	1.01
LUT as Memory	32	0	17400	0.18
LUT as Distributed RAM	32	0		
LUT as Shift Register	0	0		
Slice Registers	249	0	106400	0.23
Register as Flip Flop	249	0	106400	0.23
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

Simulation Time = 195 050 . 000 ns

Simulation cycles
= 195 050 ns / 100 ns = 1950.5 cycles

Time in Processor
= 1950.5 cycles * 10 ns = 19505 ns



Details / Improvements

1) **Processes only the raw data**
2) **Design 1 has smaller resource usage than Design 2**

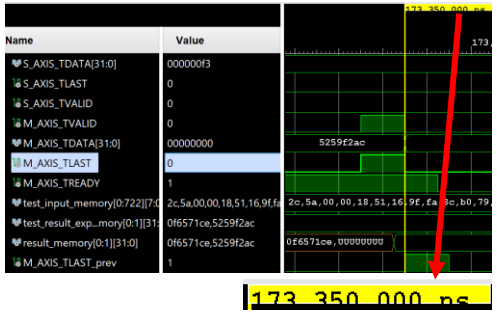
Design 2: With Labels Calculation

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	650	0	53200	1.22
LUT as Logic	594	0	53200	1.12
LUT as Memory	56	0	17400	0.32
LUT as Distributed RAM	56	0		
LUT as Shift Register	0	0		
Slice Registers	305	0	106400	0.29
Register as Flip Flop	305	0	106400	0.29
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

Simulation Time = 173 350 . 000 ns

Simulation cycles
= 173 350 ns / 100 ns = 1733.5 cycles

Time in Processor
= 1733.5 cycles * 10 ns = 17335 ns



Details / Improvements

1) **Calculated labels within coprocessor & output 64 words results into two 32-bits results**
2) **Design 2 reduced cycles at the expense of more resources`**

Overall Improvements:

1) **Read sigmoid LUTs in S_AXIS_TDATA**
   - Offers flexibilities of changing to other equation LUTs as compared to hardcoding
2) **Read X_data without bias (64 x 7) instead of (64 x 8)**
   - Reduced S_AXIS_READ_DATA time
3) **Bias is added separately for intermediate calculations**
   - Instead of calculating (64 x 8), we are calculating (64 x 7) which saves time
4) **Edited w_hid_RAM and sigmoid_RAM to read two address values from the same RAM at the same time.**
   - Enable parallel extraction of data in a single RAM
   - Save into 1 single RAM instead of separate RAMs
5) **Parallel Calculation of Neuron 1 & Neuron 2**
   - Neuron 1 & 2 calculated at the same time reducing calculation time
6) **Read addresses / enable signals / certain processes are stored combinationally to save multiple cycles & reduce registers**

HLS Implementation

Optimisations Done:

1. Pragma Pipeline II applied to all For loops except Input loops
2. Classifying the raw data within the coprocessor and encoding into 2, 32bits words, allows only 2 transmission through FIFO for to obtain labels in coprocessor

Pre-optimised

Performance Estimates

Timing — Time = 39940ns

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.313 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
3995	3995	39.950 us	39.950 us	3995	3995	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	821	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	2	-	80	26	0
Multiplexer	-	-	-	454	-
Register	-	-	348	-	-
Total	2	0	428	1301	0
Available	280	220	106400	53200	0
Utilization (%)	~0	0	~0	2	0

Cosimulation Report for 'myip_v1_0_HLS'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	Pass	3994	3994	3994	NA	NA	NA
Verilog	NA	NA	NA	NA	NA	NA	NA

Pragma

Optimised [Raw Data]

Performance Estimates

Timing — Time = 18940ns

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.585 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1895	1895	18.950 us	18.950 us	1895	1895	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	931	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	4	-	48	10	0
Multiplexer	-	-	-	556	-
Register	0	-	531	96	-
Total	4	0	579	1593	0
Available	280	220	106400	53200	0
Utilization (%)	1	0	~0	2	0

Cosimulation Report for 'myip_v1_0_HLS'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	Pass	1894	1894	1894	NA	NA	NA
Verilog	NA	NA	NA	NA	NA	NA	NA

Optimised [Encoded Labels]

Performance Estimates

Timing — Time = 18340ns

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.585 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1835	1835	18.350 us	18.350 us	1835	1835	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	1575	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	4	-	32	2	0
Multiplexer	-	-	-	615	-
Register	0	-	709	96	-
Total	4	0	741	2288	0
Available	280	220	106400	53200	0
Utilization (%)	1	0	~0	4	0

Cosimulation Report for 'myip_v1_0_HLS_labels'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	Pass	1834	1834	1834	NA	NA	NA
Verilog	NA	NA	NA	NA	NA	NA	NA

Only HLS pipeline II was used, as the loops are operationally intensive. No array partitioning was done as adding this pragma increased time and resource [unnecessary in our case as data is stored in multiple arrays for its own use]

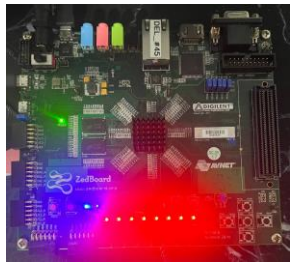
Vitis

Implementation:

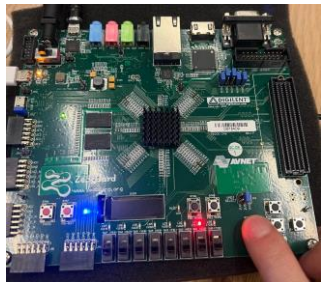
- Initialise FIFO, Timer and GPIOs
- Request for Files to be uploaded, all LED on till end of readFn()
- Enter calculations method while loop
    - Instructs user on what buttons and switch to press for soft, 2xHDL , 2xHLS and End Session [Led 0 – 2 lighted indicating readiness to use]
    - If Button pressed, Function call for selected implementation to run, respective LED (0, 1, 2) + LED(7) lights up till end of process
    - Values classified and presented/captured
    - If HDL and HLS Labels function: Decode the Labels then proceed to check for correctness against given labels
    - If HDL and HLS Labels and Data function: Take raw values, apply linear activation function to classify raw data, then proceed to check for correctness against given labels
    - 1s delay before system goes back to instructing and waiting
    - If SW7 Up: loops breaks, all LED blinks twice, end of session message is shown and session ends
- Timer set to record use of coprocessor and classifying data for each method

How it is displayed:

```
Please Upload Files For MLP
Input Matrix Uploaded
Hidden Layer Weights Uploaded
Output Layer Weights Uploaded
Sigmoid LUT Uploaded
Labels Uploaded

Press Button To Select:
Left Button for HDL Labels
Right Button for HLS Labels
Up Button for HDL Data
Down Button for HLS Data
SW7 to End Session

Starting Soft
Comparing data ...

Test Success

Co-Processor Results

Soft Data and Labels
Soft Data:100, Soft Label:0, Expected Label:0
.
.
Soft Data:125, Soft Label:0, Expected Label:0

Software matmul cycles = 6892, timing = 68920ns

Press Button To Select:
Left Button for HDL Labels
Right Button for HLS Labels
Up Button for HDL Data
Down Button for HLS Data
SW7 to End Session

Starting HDL LABELS
Comparing data ...

Test Success

Co-Processor Results

HDL Labels
HDL Label:0, Expected Label:0
```



Waiting for files to be uploaded



Left button pressed: Led 1 and HDL Labels run

Timing Analysis within Vitis:

```
Software matmul cycles = 6892, timing = 68920ns
HDL LABELS matmul cycles = 37771, timing = 377710ns
HLS LABELS matmul cycles = 37807, timing = 378070ns
HDL DATA and LABELS matmul cycles = 39412, timing = 394120ns
HLS DATA and LABELS matmul cycles = 39765, timing = 397650ns
```

Soft seems faster here because it directly accesses the data from processor, for the coprocessor, there is FIFO overhead that attributes the large timing difference. Looking at the different Hardware implementations, the output methods (encoding made positive impacts to the overall cycles

Overall Conclusion:

Soft: 68920ns
HDL Data: 19505ns, 249 FF, 568 LUT | HDL Labels: 17335ns, 305 FF, 650 LUT
HLS Data: 18940ns, 579 FF, 1593 LUT | HLS Labels: 18340ns, 741 FF, 2288 LUT

Looking solely at the computation and labelling process, without considering the exclusion of FIFO overhead for software, it is observed that software takes up a longer time. HLS despite taking 560ns less, it utilises ~47% more FFs and ~280% more LUTs compared to HDL when raw data was sent. However, with the use of the encoded labels, despite taking less time, HDL was superior being faster and resource efficient.