HLS for Neural Classification Model

Group Number: 5
Group Members:
Akshay Bhosale - 234101006
Ritik Koshta - 234101044
Rumit Gore - 236101045
Prarbdh Tiwari - 234101038

1. **Description:**

Task of Model	Number of layers	Type of Layers	Other Details
To train, and evaluate a neural network model for binary classification using a synthetic dataset. The model's architecture consists of several dense layers with ReLU activation functions, and it is trained using the Adam optimizer and binary crossentropy loss.	5	4 dense layers with ReLU activation functions and 1 dense sigmoid layer as output layer.	Hidden Neurons in the layers: 1st layer: 128 2nd layer: 64 3rd layer: 32 4th layer: 16 5th layer: 1

2. **Changes made to make keras2c generated files synthesizable and a brief description of the change made.**

    a. It generated 3 files sample.c, sample.h, sample_test_suite.c
    b. For C Simulation

- All missing functions are included after repetitive iterations (Initially we included total 6 files later we bring the required functions in sample.c file itself).

```
#include "./include/k2c_include.h"
#include "./include/k2c_tensor_include.h"
#include "./include/k2c_core_layers.h"
#include "./include/k2c_recurrent_layers.h"
#include "./include/k2c_helper_functions.h"
#include "./include/k2c_convolution_layers.h"
#include "./include/k2c_activations.h"
```

- All redefinition of "size_t" data type variables errors resolved from all files by declaring the variable one only in the function at top.
- All Segmentation errors are also removed by declaring required array globally.
- All other small errors of missing parenthesis and missing ";" are resolved.
- Returning non-zero value error also resolved by changing the return statement from "1" to "0" in test bench file's main function.

```
printf("Max absolute error for 10 tests: %e \n", maxerror);
agri_keras_terminate();
if (maxerror > 1e-05)
{
    return 0;
}
return 0;
```

- Simulation succeed after it.

c. For Synthesis Part

- All memset() functions changed using a loop in all files (took many multiple iterations).

```
    // memset(C, 0, outrows*outcols*sizeof(C[0]));
    size_t i, j;
    size_t total_ele = outcols*outrows;
for (i = 0; i < total_ele; i++) {
    C[i] = 0;
}
```

- All memcpy() functions changed using a loop in all files (took many multiple iterations).

```
    // memcpy(output->array, input->array, input->numel*sizeof(input->array[0]));
    size_t i;
for (i = 0; i < input->numel; i++) {
    output->array[i] = input->array[i];
}
```

- All function pointers errors are resolved (by calling function directly instead of passing a pointer).

```
// void k2c_relu_func(float x[], const size_t size) {

//     for (size_t i=0; i < size; ++i) {
//         if (x[i] <= 0.0f) {
//             x[i] = 0.0f;
//         }
//     }
// }
//k2c_activationType * k2c_relu = k2c_relu_func;
```

```
if(flag == 0){
    // k2c_relu_func(output->array,outsize);
    for (i=0; i < outsize; ++i) {
if( output_array[i] <= 0.0f) {
    output_array[i] = 0.0f;
}
```

- All double pointers errors are resolved.
- Structure is redefined (Later for resource optimization we didn’t use the structure. We manually created required size arrays for individual instance).

```c
#define K2C_MAX_NDIM 5


/**
 * tensor type for keras2c.
 */
struct k2c_tensor
{
    /** Pointer to array of tensor values flattened in row major order. */
    float  array[10000];

    /** Rank of the tensor (number of dimensions). */
    size_t ndim;

    /** Number of elements in the tensor. */
    size_t numel;

    /** Array, size of the tensor in each dimension. */
    size_t shape[K2C_MAX_NDIM];
};

typedef struct k2c_tensor k2c_tensor;
```

- Structure objects created manually instead of using functions (around 50 such occurrences).

```c
// k2c_tensor test1_dense_13_input_input = {&test1_dense_13_input_input_array[0],1,20,{20, 1, 1, 1, 1}};
k2c_tensor test1_dense_13_input_input;

for(i=0;i<20;i++)
{
    #pragma HLS UNROLL factor=4
    test1_dense_13_input_input.array[i]=test1_dense_13_input_input_array[i];
}
test1_dense_13_input_input.ndim=1;
test1_dense_13_input_input.numel= 20;
test1_dense_13_input_input.shape[0]=20;
test1_dense_13_input_input.shape[1]=1;
test1_dense_13_input_input.shape[2]=1;
test1_dense_13_input_input.shape[3]=1;
test1_dense_13_input_input.shape[4]=1;
```

```c
// k2c_tensor dense_13_output;
// k2c_tensor dense_13_output = {&dense_13_output_array[0],1,128,{128,  1,  1,  1,  1}};
size_t dense_13_output_dim = 1;
size_t dense_13_output_numel = 128;
size_t dense_13_output_shape[5] = {128, 1, 1, 1, 1};
```

- All dynamic memory allocation converted to static.
- Segmentation fault resolved here also (Array declared Globally).
- All function definition’s also changed.

```c
// k2c_dense(&dense_13_output,dense_13_input,&dense_13_kernel,
//  &dense  float dense_13_output_array[128]
k2c_dense(dense_13_output_array,dense_13_output_dim,&dense_13_output_numel,dense_13_output_shape,
 dense_13_input_input_array, dense_13_input_input_dim, dense_13_input_input_numel, dense_13_input_input_shape,
 dense_13_kernel_array,dense_13_kernel_dim,&dense_13_kernel_numel,dense_13_kernel_shape,
 dense_13_bias_array,dense_13_bias_dim,&dense_13_bias_numel, dense_13_bias_shape , 0 ,dense_13_fwork);
```

- Synthesis succeed after it and all required reports generated.

d. C/RTL Simulation Performed after it and we get the Latency.

3. **Changes made to generate HLS4ML reports if a pragma is removed in this process. For each of the removed pragmas, a valid argument must be mentioned.**

   We have not removed any pragmas. But we have done some changes to optimise the code.

   a. We have removed ***#config_schedule -enable_dsp_full_reg=false*** from build_prj.tcl as it was throwing a warning and interrupting the flow.
   b. We faced problems while importing hls4ml and we had to downgrade the tensorflow version 14.1 to make hls4ml run.
   c. We also faced problems of system crashing again and again due to high memory requirements and less resources available on our system. We used TA's lab system to run our code for synthesis of HLS4ML.
   d. It took around 30 mins to run the synthesis on HLS4ML.
   e. We have used the FPGA board as: **xc7a200t-fbg676-2 (artix7)**
   f. The configuration of the board is:

BRAM_18K	**DSP48E**	**FF**	**LUT**	**URAM**
730	740	269200	134600	0

4. **In a markdown cell of jupyter notebook mention all the issues that are faced(dependencies and versions) and solutions to resolve.**
   The dependencies and libraries installed to run and generate the overall HLS4ML reports are saved in the markdown cell of jupyter notebook. The same are presented here:

   - pip3 install hls4ml[profiling]
   - pip3 install tensorflow==2.15.1
   - pip3 install torch torchvision --index-url https://download.pytorch.org/whl/cpu
   - pip3 install --upgrade pip==22.3
   - pip3 install tensorrt

- pip3 install tf_keras
- pip3 install pydot
- pip3 install pydotplus
- pip3 install graphviz
- !export CUDA_VISIBLE_DEVICES=0

5. **Optimizations: For each optimization applied (pragma), justify why it has been used.**
   - **#pragma HLS unroll**
     We are using it for unrolling the loop by required factor, which is reducing latency at the cost of resource.
   - **#pragma HLS array_partition**
     First we have cyclic partitioned with block factor 8 in dimension 1 of the arrays A and B then we have performed unrolling of the loops, which helps in reduction of latency and also we have applied pipeline II=1 in the innermost loop which helps in further reduction in latency.
   - **#pragma HLS PIPELINE**
     In the code we have applied pipeline II=1 in the inner most loop which helps in reduction of latency.

```
INFO: [SCHED 204-61] Pipelining loop 'Loop 1'.
INFO: [SCHED 204-61] Pipelining result : Target II = 1, Final II = 1, Depth = 29.
```

6. **Results:**
   a. **Latency and area overhead table for Baseline (Unoptimized).**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	402
FIFO	-	-	-	-
Instance	0	334	57821	42478
Memory	427	-	64	16
Multiplexer	-	-	-	1655
Register	-	-	531	-
Total	427	334	58416	44551
Available	730	740	269200	129000
Utilization (%)	58	45	21	34

Cosimulation Report for 'sample0'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	176149	176149	176149	NA	NA	NA

Export the report(.html) using the Export Wizard

b. Latency and area overhead table for Optimised (if there are multiple versions like various tradeoffs, give all of them).

- **Resource Optimized**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	26
FIFO	-	-	-	-
Instance	78	269	46095	41946
Memory	4	-	64	8
Multiplexer	-	-	-	242
Register	-	-	18	-
Total	82	269	46177	42222
Available	730	740	269200	129000
Utilization (%)	11	36	17	32

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	80839	80839	80839	NA	NA	NA

- **Latency Optimization (Try 1)**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	26
FIFO	-	-	-	-
Instance	78	282	47810	44793
Memory	6	-	64	8
Multiplexer	-	-	-	269
Register	-	-	18	-
Total	84	282	47892	45096
Available	730	740	269200	129000
Utilization (%)	11	38	17	34

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	45679	45679	45679	NA	NA	NA

- **Latency Optimization (Try 2)**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	26
FIFO	-	-	-	-
Instance	94	309	69842	96194
Memory	2	-	1536	88
Multiplexer	-	-	-	1061
Register	-	-	18	-
Total	96	309	71396	97369
Available	730	740	269200	129000
Utilization (%)	13	41	26	75

Cosimulation Report for 'sample'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	60288	60288	60288	NA	NA	NA

Export the report(.html) using the Export Wizard

- **Latency Optimization (Try 3)**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	26
FIFO	-	-	-	-
Instance	94	317	82285	90722
Memory	2	-	1536	88
Multiplexer	-	-	-	1061
Register	-	-	18	-
Total	96	317	83839	91897
Available	730	740	269200	129000
Utilization (%)	13	42	31	71

Cosimulation Report for 'sample'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	12624	12624	12624	NA	NA	NA

Export the report(.html) using the Export Wizard

- **Latency Optimization (Final)**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	26
FIFO	-	-	-	-
Instance	70	343	58977	52153
Memory	2	-	1536	88
Multiplexer	-	-	-	1061
Register	-	-	18	-
Total	72	343	60531	53328
Available	730	740	269200	129000
Utilization (%)	9	46	22	41

Cosimulation Report for 'sample'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	1187	1187	1187	1188	1188	1188

Export the report(.html) using the Export Wizard

c. HLS4ML generated Latency and area overhead table.

```
+ Latency:
    * Summary:
    +---------+---------+----------+----------+-----+-----+----------+
    | Latency (cycles) |  Latency (absolute)  | Interval | Pipeline |
    |   min   |   max   |    min   |    max   | min | max |   Type   |
    +---------+---------+----------+----------+-----+-----+----------+
    |       47|       51| 0.235 us | 0.255 us |    4|    4| dataflow |
    +---------+---------+----------+----------+-----+-----+----------+
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+--------+--------+-----+
|       Name      | BRAM_18K| DSP48E|   FF   |   LUT  | URAM|
+-----------------+---------+-------+--------+--------+-----+
|DSP              |        -|      -|       -|       -|    -|
|Expression       |        -|      -|       0|    1434|    -|
|FIFO             |        0|      -|    1765|    9002|    -|
|Instance         |      798|   1796|  140222|  131682|    -|
|Memory           |        -|      -|       -|       -|    -|
|Multiplexer      |        -|      -|       -|    3168|    -|
|Register         |        -|      -|     352|       -|    -|
+-----------------+---------+-------+--------+--------+-----+
|Total            |      798|   1796|  142339|  145286|    0|
+-----------------+---------+-------+--------+--------+-----+
|Available        |      730|    740|  269200|  134600|    0|
+-----------------+---------+-------+--------+--------+-----+
|Utilization (%)  |      109|    242|      52|     107|    0|
+-----------------+---------+-------+--------+--------+-----+
```

d. **Finally, a comparison report of both Optimised and HLS4ML generated reports.**

In Vivado, we have significantly optimized the BRAM, registers, LUTs, flip-flops, and DSPs compared to HLS4ML. This optimization has led to improved resource utilization. However, as a result of these enhancements, there is a slightly higher latency, which we are actively addressing to further enhance overall performance.

In Vivado we have used array partitioning, piplining and unrolling to achieve these results.

In our each try we were trying to bring the results of Vivado and HLS4ML closer and closer to each other. We had tried many ways and mentioned only the ones which worked the best among the many. As we saw an increase in resources we got a lesser latency and vice versa as expected. We have tried to reach the most optimal and balanced result in Vivado.

And hence overall we reached better utilization of resources in Vivado than HLS4ML.

The same can be seen in the table below, where the last row is for HLS4ML.

Provided the HLS results for each of the above scenarios in the below table.

Design	LUT	FF	DSP	BRAM	Latency (min/max)	Clock Period
Artix 7	44551	58416	334	427	176149/ 176149	5
Artix 7	42222	46177	269	82	80839/ 80839	10

Artix 7	45096	47892	282	84	45679/ 45679	10
Artix 7	97369	71396	309	96	60288/ 60288	5
Artix 7	91897	83839	317	96	12624/ 12624	5
Artix 7	53328	60531	343	72	1187/1188	5
Artix 7	145286	142339	1796	798	47/51	5

7. **All the modified keras2c generated files and HLS4ML jupyter note-book must be uploaded in a zip format.**
   Done and Submitted.