# HLS for Recognizer Model

**Group Number:** 3

**Group members:**

| | |
|---|---|
| Shantanu Shrivastav | 234101063 |
| Shubham Mourya | 234101064 |
| Ayush Agarwal | 234101060 |
| Pradoom Varma | 234101036 |
| Tarun Kumar | 210101104 |

## Description of the model

| Task of model | Number of layers | Type of Layers | Other details |
|---|---|---|---|
| Face Detection & Recognition | 6 | Dense Layers | Image Detection |

## Changes made to keras2c

Converted recognizer.h5 to c files using python -m recognizer.h5  final

### Simulation and Synthesis

#### Linking Header Files

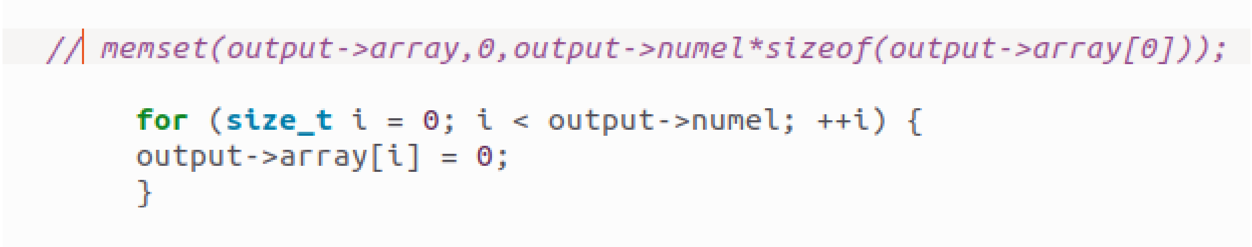
Header files present in keras2c were in .c format. Converted them into .h
Added path to these header files in “final.c” source file.

#### Re-declaration of variables

In the header file there was a re-declaration of the iterator in for loops. Variable renaming was done to remove this error.

### Removing STL

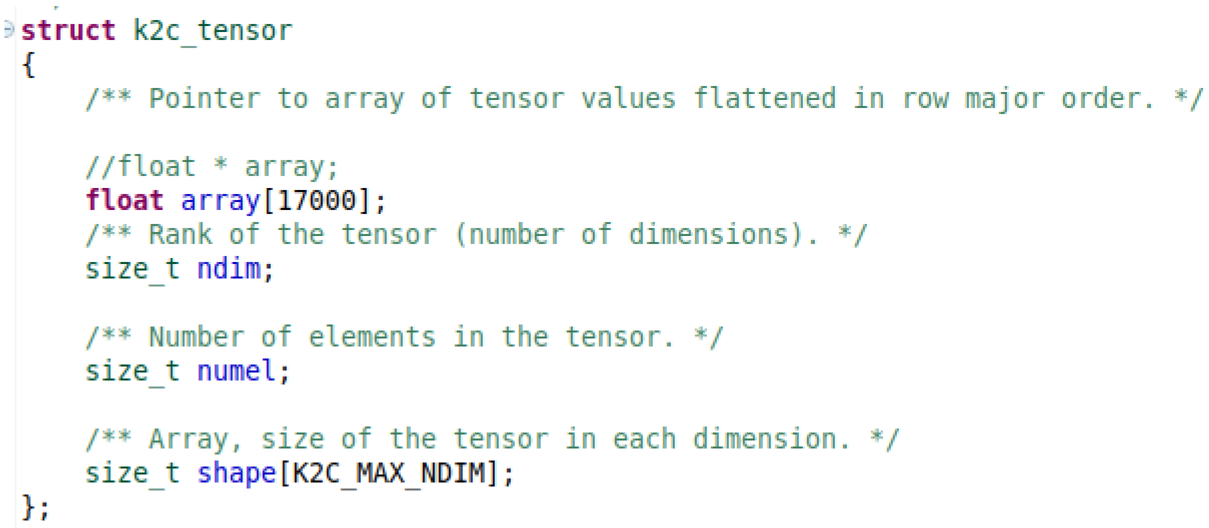
Replaced memset with for loop.



```
  // memset(output->array,0,output->numel*sizeof(output->array[0]));

      for (size_t i = 0; i < output->numel; ++i) {
      output->array[i] = 0;
      }
```

### Floating point Conversion

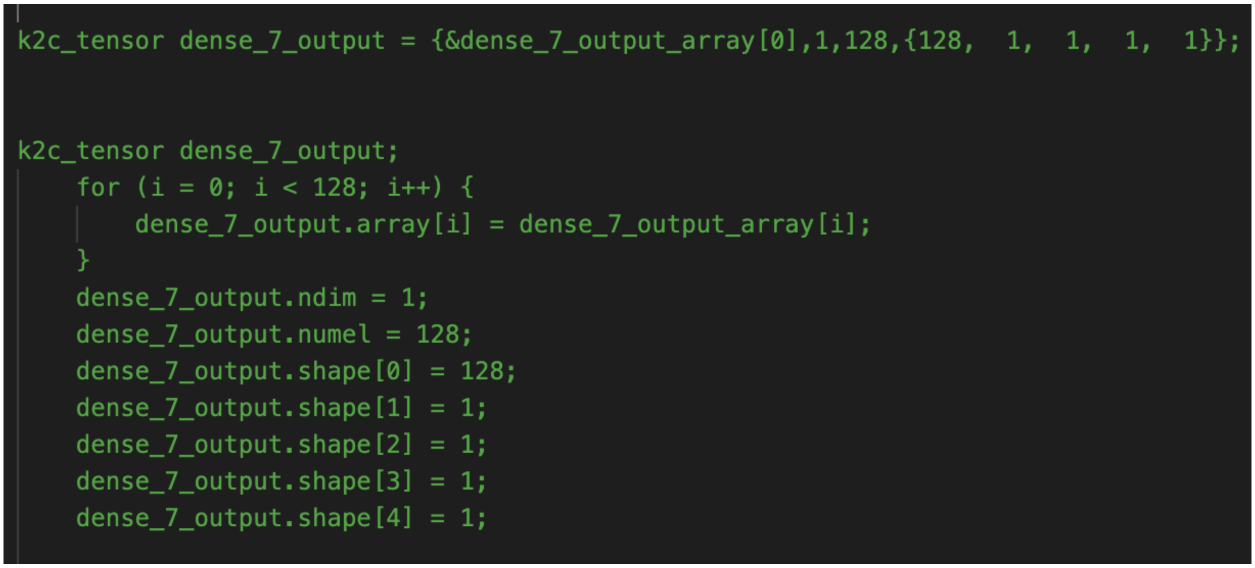
In the argument of main function getting pointer to pointer or global pointer error to resolve this changed float *array to float array[17000] in k2c_tensor_include.h, struct k2c tensor.



```
struct k2c_tensor
{
    /** Pointer to array of tensor values flattened in row major order. */

    //float * array;
    float array[17000];
    /** 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];
};
```

### k2c Tensor Expansion

Due to the above change we were getting error in all k2c tensor declared arrays that value of type "float *" cannot be used to initialize an entity of type "float" to resolve this declare all k2c_tensor array manually.



```
k2c_tensor dense_7_output = {&dense_7_output_array[0],1,128,{128,  1,  1,  1,  1}};


k2c_tensor dense_7_output;
    for (i = 0; i < 128; i++) {
        dense_7_output.array[i] = dense_7_output_array[i];
    }
    dense_7_output.ndim = 1;
    dense_7_output.numel = 128;
    dense_7_output.shape[0] = 128;
    dense_7_output.shape[1] = 1;
    dense_7_output.shape[2] = 1;
    dense_7_output.shape[3] = 1;
    dense_7_output.shape[4] = 1;
```

**Dense Layer**

- In k2c_dense function, k2c_relu argument was giving error 'k2c_relu' has an unsynthesizable type 'void (float*, i64)P*' so replaced it with it's actual value(2) similarly for k2c_softmax.

**No function body**

K2c_dense has no function body so declared it in final.c. Along with this declared function body of both activation functions(Relu and Softmax) inside K2c_dense function body.

**AbnormalTermination**

Segmentation fault: Resolved this by replacing all arrays with size 17000.
Since maximum array size in code was of 16512, so to avoid segmentation fault we declared arrays with a size of 17000.

**Cleaning**

Reduced number of modules to 2, removed many functions and wrote them inline because they were getting called just once, and removed unnecessary switch cases from activator whereas kept a separate module for k2c_dense because it was getting called many times.
In k2c_dense checked input.ndim of k2c_tensor input and removed functions under else statement to reduce unnecessary functions import.

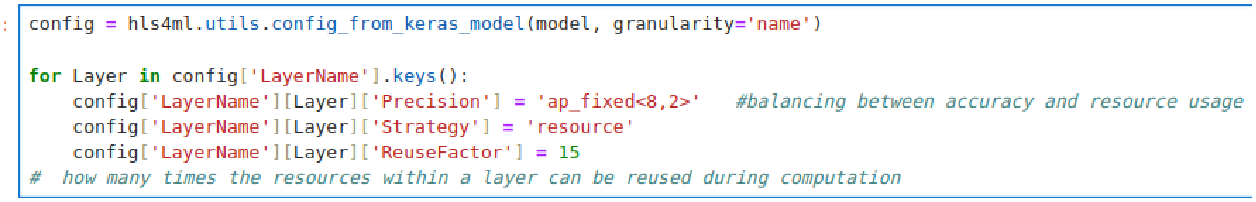
# Changes made to generate HLS4ML

**Pragmas Removed:** None

**Pragmas Modified:**

While unrolling the layers (Dense layers) of the model, unrolling was leading to large size code which our system due to limited resources was not able to execute.

Hence to overcome this issue following script was executed

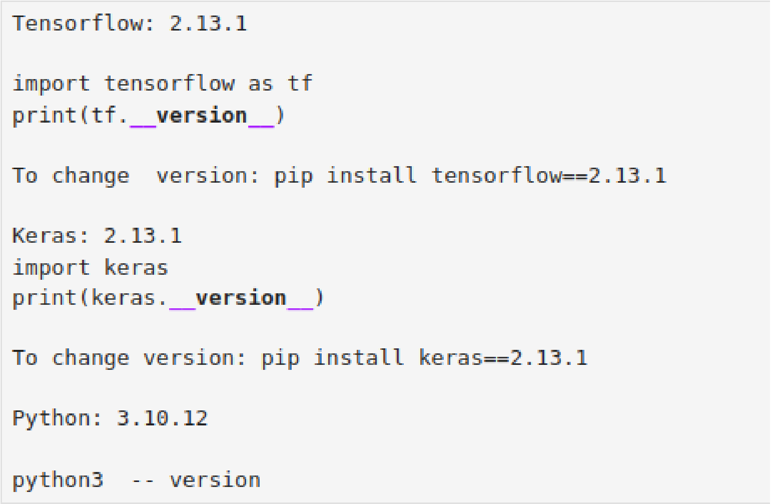


```
config = hls4ml.utils.config_from_keras_model(model, granularity='name')

for Layer in config['LayerName'].keys():
    config['LayerName'][Layer]['Precision'] = 'ap_fixed<8,2>'   #balancing between accuracy and resource usage
    config['LayerName'][Layer]['Strategy'] = 'resource'
    config['LayerName'][Layer]['ReuseFactor'] = 15
#  how many times the resources within a layer can be reused during computation
```

**Precision** : Setting between accuracy and resource usage
**Strategy**: Changed from Latency to Resource
**Reuse factor**: Number of times within a layer a resource can be reused set to 15

These changes were made only for layers. Apart from this no changes have been made.

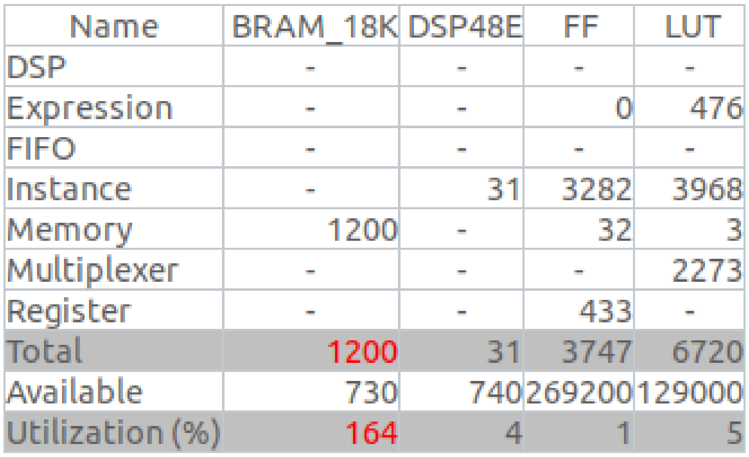
# Dependencies and Versions



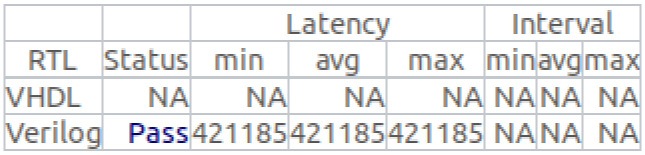
```
Tensorflow: 2.13.1

import tensorflow as tf
print(tf.__version__)

To change  version: pip install tensorflow==2.13.1

Keras: 2.13.1
import keras
print(keras.__version__)

To change version: pip install keras==2.13.1

Python: 3.10.12

python3  -- version
```

# Optimizations

### INITIAL LATENCY AND RESOURCE



| Name | BRAM_18K | DSP48E | FF | LUT |
|---|---|---|---|---|
| DSP | - | - | - | - |
| Expression | - | - | 0 | 476 |
| FIFO | - | - | - | - |
| Instance | - | 31 | 3282 | 3968 |
| Memory | 1200 | - | 32 | 3 |
| Multiplexer | - | - | - | 2273 |
| Register | - | - | 433 | - |
| Total | 1200 | 31 | 3747 | 6720 |
| Available | 730 | 740 | 269200 | 129000 |
| Utilization (%) | 164 | 4 | 1 | 5 |



| | | Latency | | | Interval | | |
|---|---|---|---|---|---|---|---|
| RTL | Status | min | avg | max | min | avg | max |
| VHDL | NA | NA | NA | NA | NA | NA | NA |
| Verilog | Pass | 421185 | 421185 | 421185 | NA | NA | NA |

1. **Static Arrays** - we have to reduce the latency we made all arrays static since our maximum array size was 16512 it did not affect much to the latency.

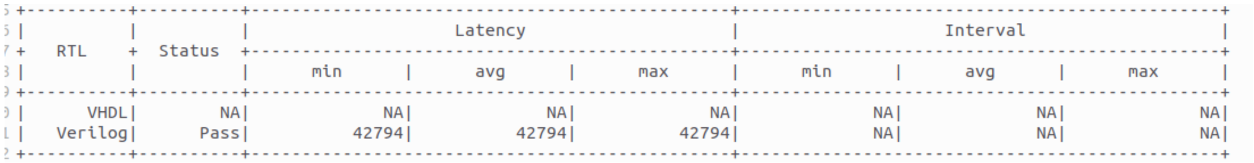
2. **Array partitioning** - to reduce latency during k2c_tensor arrays initialization we did cyclic partition with factors 16,8,4 and we tried multiple combination.

3. **Pipeline in matmul function** - we tried multiple combination of pipeline in matmul function but it didn't affect much to the latency.

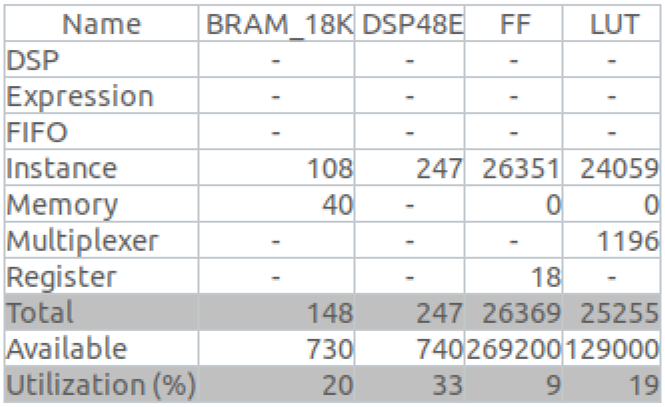
4. **Removing k2c_tensor arrays**- Since we are declaring k2c_tensor arrays of size 16512 everytime ,we removed k2c_tensor arrays and replaced it with original arrays it reduces resources utilization from 164%(BRAM) to 19%(BRAM)

5. **Matmul Function Optimization**- we applied array partitioning and pipeline in matmul function to improve parallelism but it increased hardware utilization latency reduced to 42000 from 421000.

### FINAL LATENCY AND RESOURCE

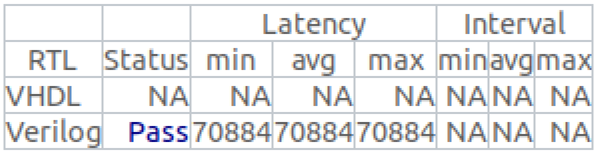


```
+----------+---------+-----------------------------------------+-----------------------------------------+
|          |         |                 Latency                 |                 Interval                |
+   RTL    +  Status +-----------------------------------------+-----------------------------------------+
|          |         |     min     |     avg     |     max     |     min     |     avg     |     max     |
+----------+---------+-----------------------------------------+-----------------------------------------+
|      VHDL|       NA|           NA|           NA|           NA|           NA|           NA|           NA|
|   Verilog|     Pass|        42794|        42794|        42794|           NA|           NA|           NA|
+----------+---------+-----------------------------------------+-----------------------------------------+
```



| Name | BRAM_18K | DSP48E | FF | LUT |
|---|---|---|---|---|
| DSP | - | - | - | - |
| Expression | - | - | - | - |
| FIFO | - | - | - | - |
| Instance | 108 | 247 | 26351 | 24059 |
| Memory | 40 | - | 0 | 0 |
| Multiplexer | - | - | - | 1196 |
| Register | - | - | 18 | - |
| Total | 148 | 247 | 26369 | 25255 |
| Available | 730 | 740 | 269200 | 129000 |
| Utilization (%) | 20 | 33 | 9 | 19 |

To make it comparable with HLS4ML which was running on clock 5ns we changed our clock to 5ns after which the latency became 70884.



|  |  | Latency |  |  | Interval |  |  |
|---|---|---|---|---|---|---|---|
| RTL | Status | min | avg | max | min | avg | max |
| VHDL | NA | NA | NA | NA | NA | NA | NA |
| Verilog | Pass | 70884 | 70884 | 70884 | NA | NA | NA |

# Results

| Design | LUT | FF | DSP | BRAM | Latency |  | Clock period(in ns) |
|---|---|---|---|---|---|---|---|
|  |  |  |  |  | Min | Max |  |
| Unoptimized | 6720 | 3747 | 31 | 1200 | 421185 | 421185 | 10 |
| Unoptimized | 7294 | 5539 | 31 | 1200 | 742654 | 742654 | 5 |
| Resource | 7675 | 5776 | 37 | 73 | 343424 | 343424 | 10 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Optimized | 25255 | 26369 | 247 | 148 | 42794 | 42794 | 10 |
| Optimized | 28275 | 36195 | 247 | 148 | 70884 | 70884 | 5 |
| HLS4ML | 468331 | 31851 | 1 | 448 | 36000 | 36000 | 5 |