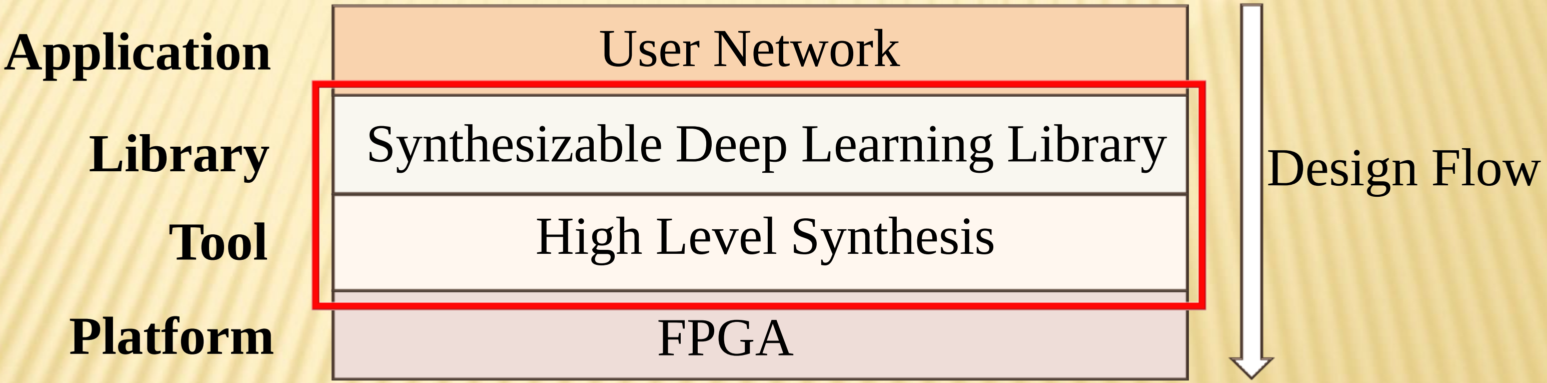
可综合深度学习库

金济芳 复旦大学
jfjin14@fudan.edu.cn
2016/11/22

设计流程



Application — User Network

Library — Synthesizable Deep Learning Library

Tool — High Level Synthesis

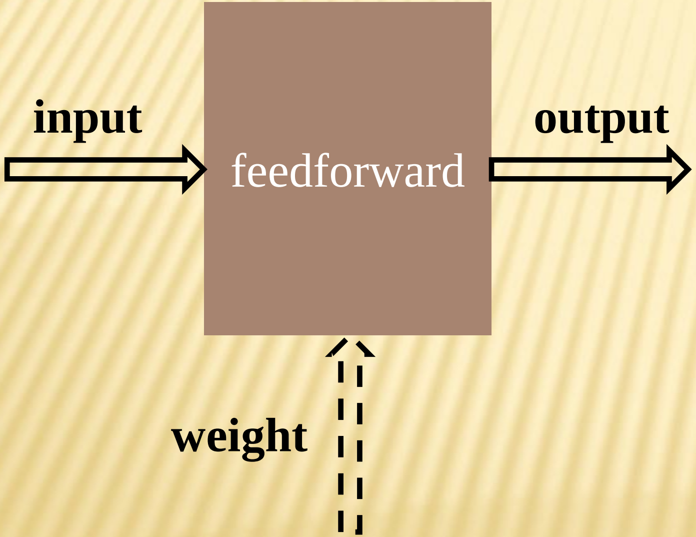
Platform — FPGA

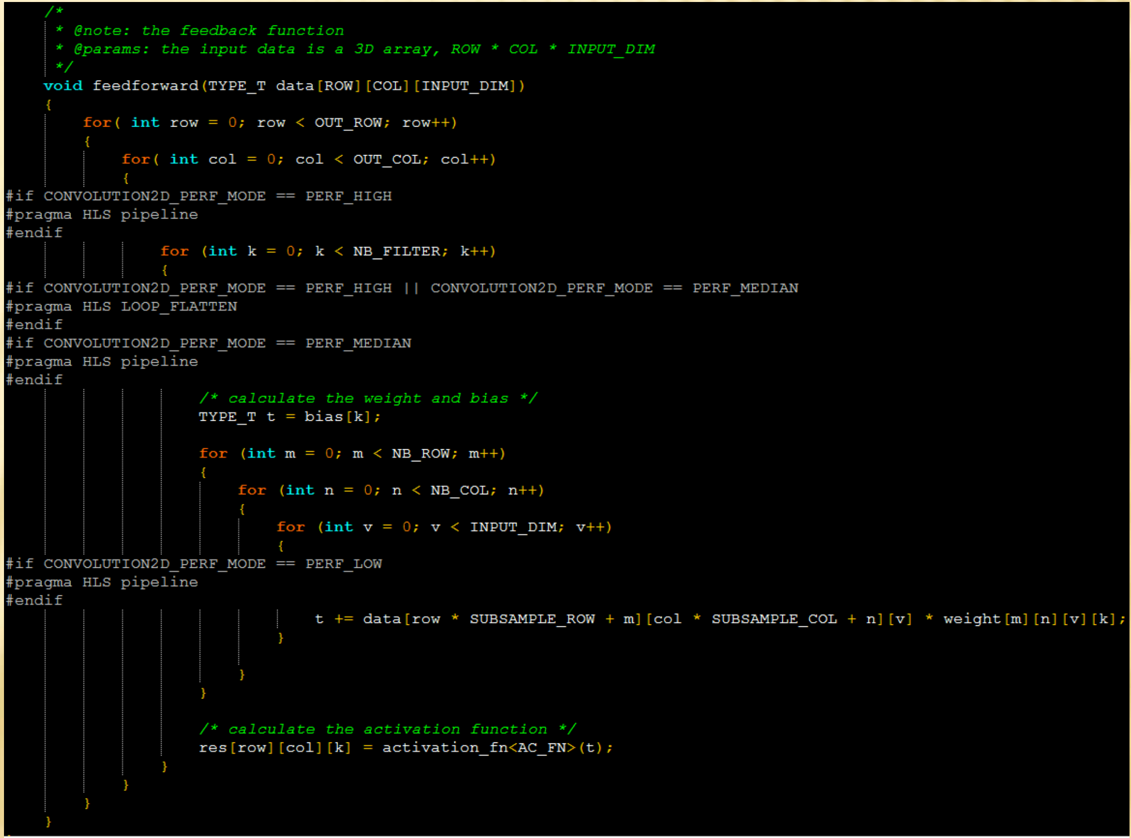
Design Flow

概述

- Deep Network加速器
  - 支持推断，不支持训练
- 基于HLS和C++模板
  - 编程灵活性、功能可配置性、结构可定制性
- 丰富的组件
  - 全连接、激活、卷积、循环、Pooling、Embedding等核心组件
- 完善的优化指令
  - 循环、高维数组、缓存等优化
- 兼容开源Keras

模块化设计





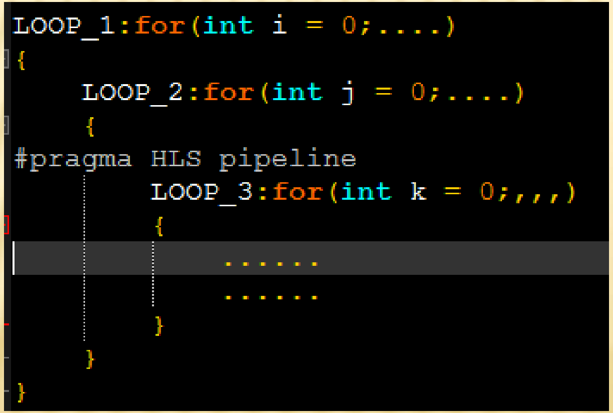
```
    /*
     * @note: the feedback function
     * @params: the input data is a 3D array, ROW * COL * INPUT_DIM
     */
    void feedforward(TYPE_T data[ROW][COL][INPUT_DIM])
    {
        for( int row = 0; row < OUT_ROW; row++)
        {
            for( int col = 0; col < OUT_COL; col++)
            {
#if CONVOLUTION2D_PERF_MODE == PERF_HIGH
#pragma HLS pipeline
#endif
                for (int k = 0; k < NB_FILTER; k++)
                {
#if CONVOLUTION2D_PERF_MODE == PERF_HIGH || CONVOLUTION2D_PERF_MODE == PERF_MEDIAN
#pragma HLS LOOP_FLATTEN
#endif
#if CONVOLUTION2D_PERF_MODE == PERF_MEDIAN
#pragma HLS pipeline
#endif
                    /* calculate the weight and bias */
                    TYPE_T t = bias[k];

                    for (int m = 0; m < NB_ROW; m++)
                    {
                        for (int n = 0; n < NB_COL; n++)
                        {
                            for (int v = 0; v < INPUT_DIM; v++)
                            {
#if CONVOLUTION2D_PERF_MODE == PERF_LOW
#pragma HLS pipeline
#endif
                                t += data[row * SUBSAMPLE_ROW + m][col * SUBSAMPLE_COL + n][v] * weight[m][n][v][k];
                            }

                        }
                    }

                    /* calculate the activation function */
                    res[row][col][k] = activation_fn<AC_FN>(t);
                }
            }
        }
    }
```

性能瓶颈

- 多层循环
  - 二维卷积层包含6层循环
- 高维数组
  - 二维卷积层Weight是4维数组
- 超大参数
  - FPGA存储资源有限

多层循环优化

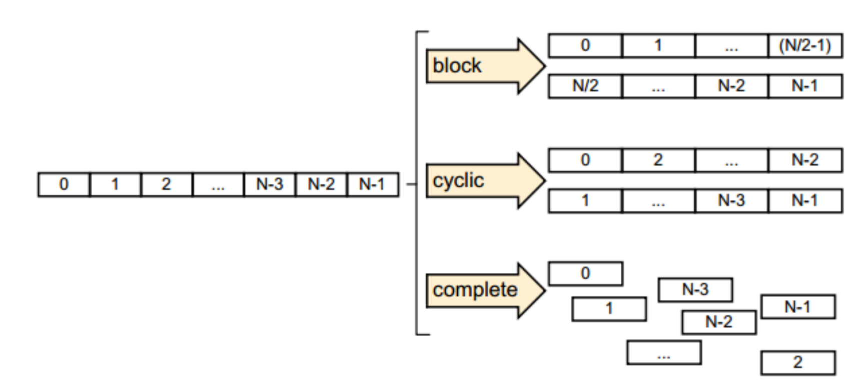
- Pipeline
  - #pragma HLS pipeline
  - 越外层，性能越好，代价越高；越运用到内层，性能越差，代价越低

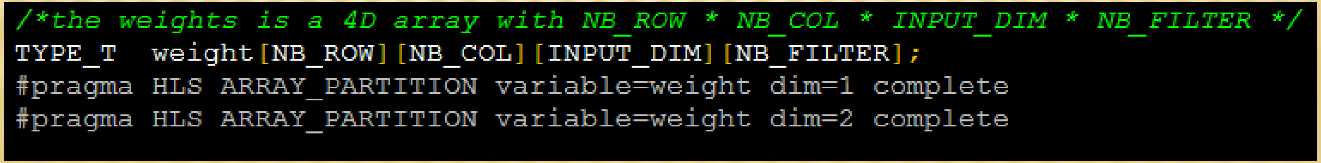


```
LOOP_1:for(int i = 0;....)
{
    LOOP_2:for(int j = 0;....)
    {
#pragma HLS pipeline
        LOOP_3:for(int k = 0;,,,)
        {
            ......
            ......
        }
    }
}
```

高维数组优化

- Array Partition
  - #pragma HLS ARRAY_PARTITION
  - BRAM资源换带宽

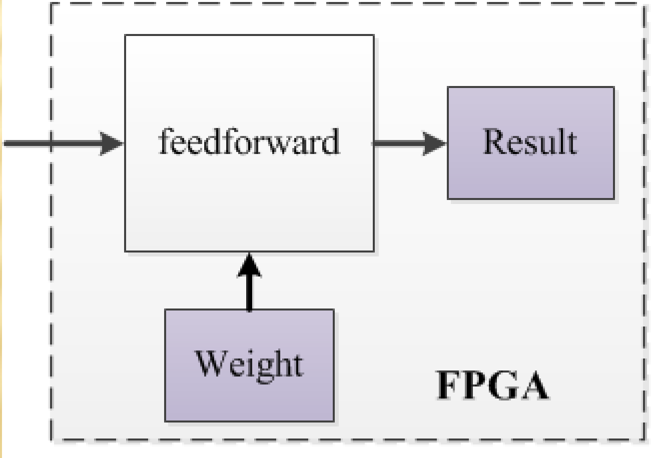




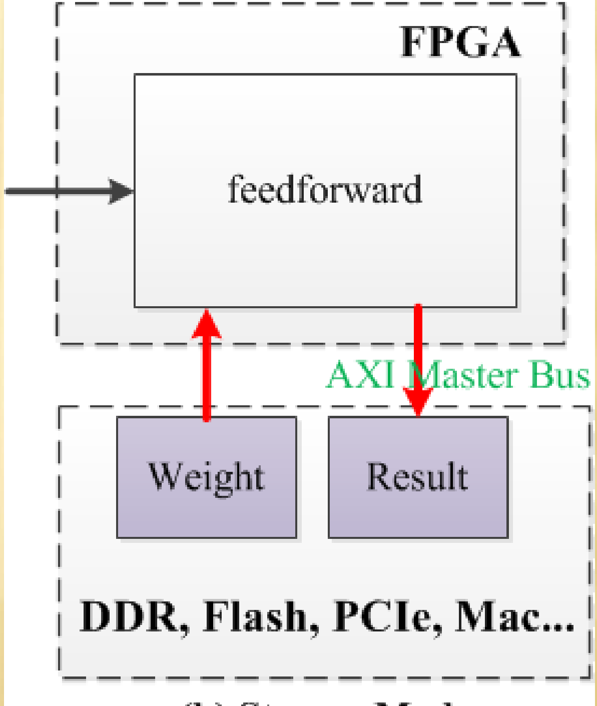
```
/*the weights is a 4D array with NB_ROW * NB_COL * INPUT_DIM * NB_FILTER */
TYPE_T  weight[NB_ROW][NB_COL][INPUT_DIM][NB_FILTER];
#pragma HLS ARRAY_PARTITION variable=weight dim=1 complete
#pragma HLS ARRAY_PARTITION variable=weight dim=2 complete
```

超大参数

- 存储模型
  - Local & Stream Mode



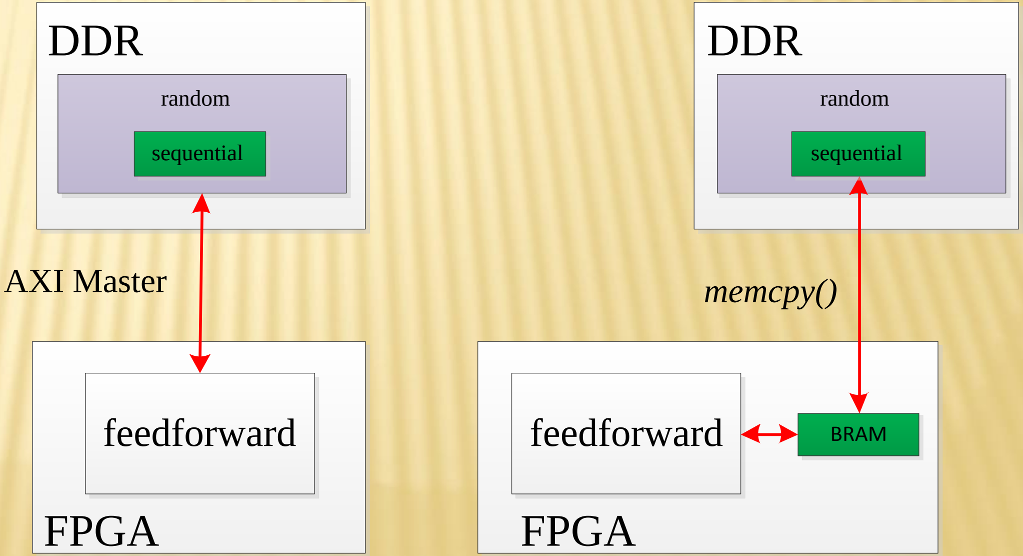
(a) Local Mode



(b) Stream Mode

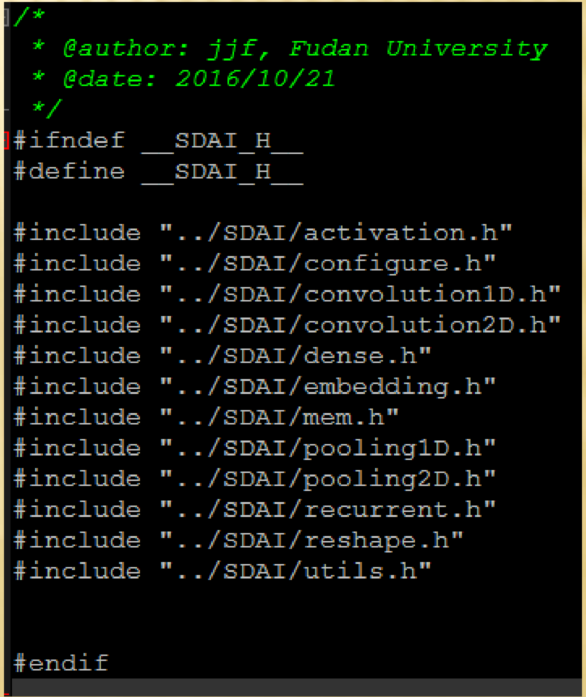
STREAM MODE优化

- AXI Master随机访问极大限制了性能。
- 优化方法
  - OPT_MEM
  - OPT_BUFFER



适用范围

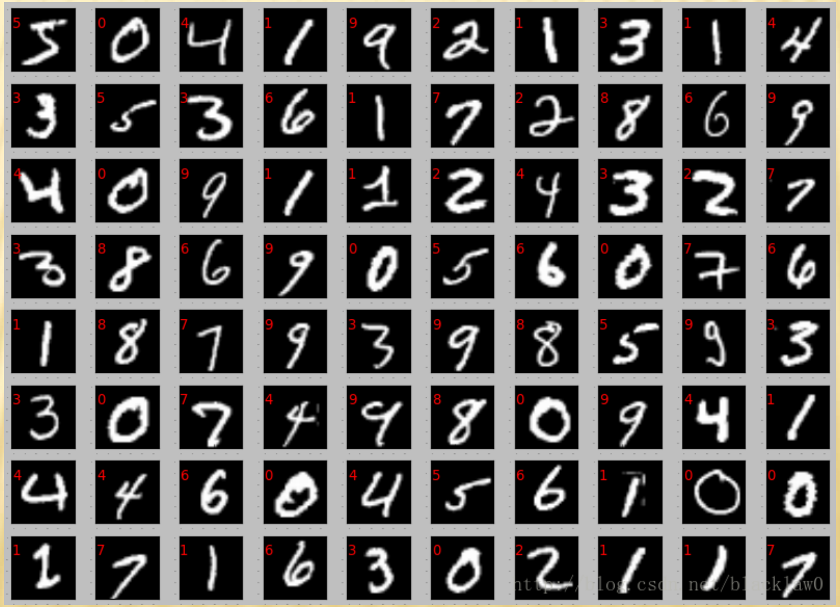
- 常用MLP, CNN, RNN等Deep Network
  - 丰富的组件
  - Local和Stream模式
  - 可配置的优化指令
  - 支持HLS和SDSoC



```
/*
 * @author: jjf, Fudan University
 * @date: 2016/10/21
 */
#ifndef __SDAI_H__
#define __SDAI_H__

#include "../SDAI/activation.h"
#include "../SDAI/configure.h"
#include "../SDAI/convolution1D.h"
#include "../SDAI/convolution2D.h"
#include "../SDAI/dense.h"
#include "../SDAI/embedding.h"
#include "../SDAI/mem.h"
#include "../SDAI/pooling1D.h"
#include "../SDAI/pooling2D.h"
#include "../SDAI/recurrent.h"
#include "../SDAI/reshape.h"
#include "../SDAI/utils.h"


#endif
```

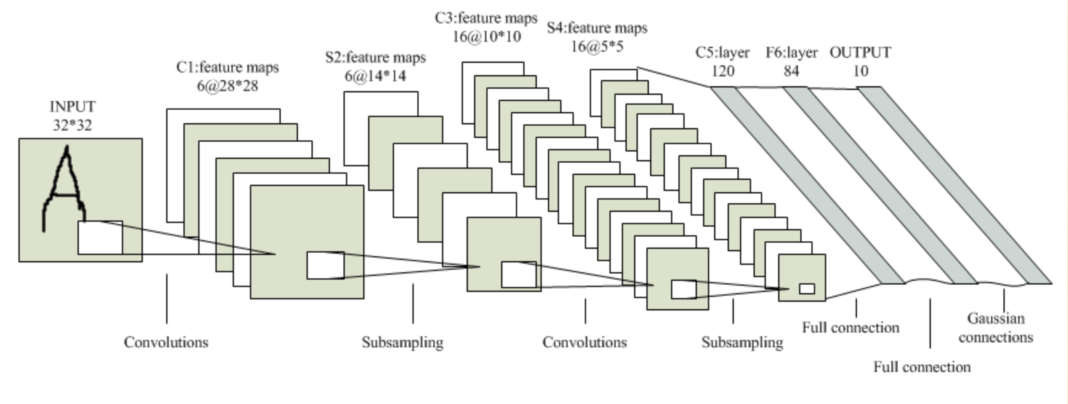
MNIST手写字识别

- 28x28图像分类



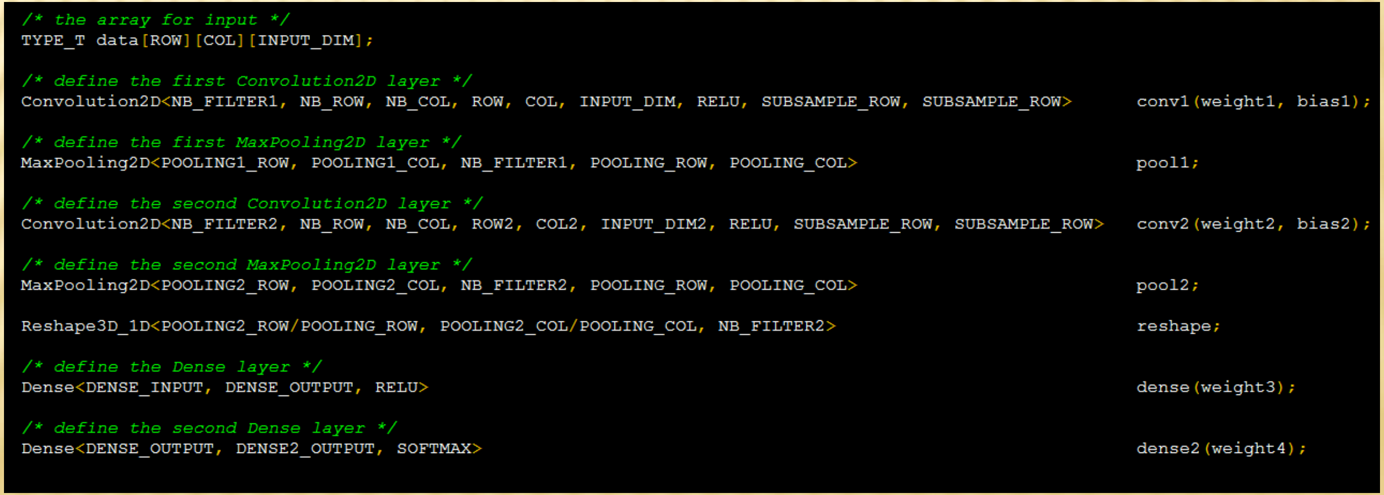
MNIST手写字识别

- LeNet5



MNIST手写字识别-LENET5

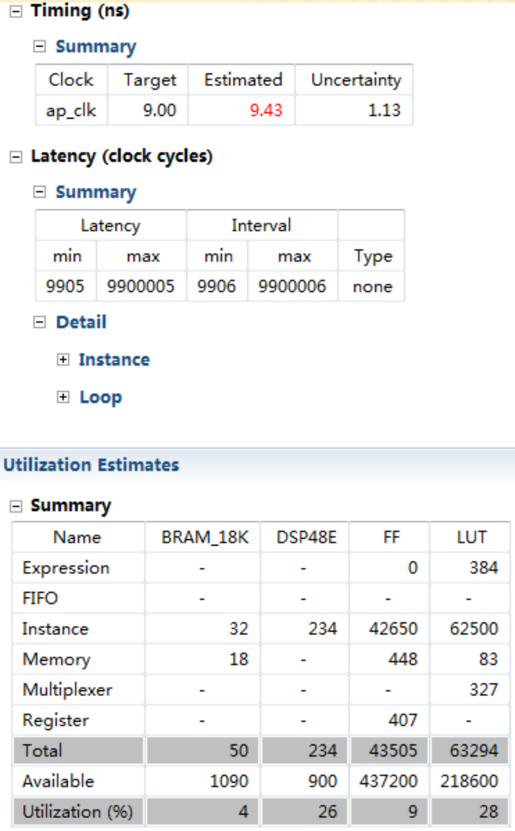
- 6层网络、5360个参数、Float精度、97.4%准确率



```
/* the array for input */
TYPE_T data[ROW][COL][INPUT_DIM];

/* define the first Convolution2D layer */
Convolution2D<NB_FILTER1, NB_ROW, NB_COL, ROW, COL, INPUT_DIM, RELU, SUBSAMPLE_ROW, SUBSAMPLE_ROW>      conv1(weight1, bias1);

/* define the first MaxPooling2D layer */
MaxPooling2D<POOLING1_ROW, POOLING1_COL, NB_FILTER1, POOLING_ROW, POOLING_COL>                          pool1;

/* define the second Convolution2D layer */
Convolution2D<NB_FILTER2, NB_ROW, NB_COL, ROW2, COL2, INPUT_DIM2, RELU, SUBSAMPLE_ROW, SUBSAMPLE_ROW>   conv2(weight2, bias2);

/* define the second MaxPooling2D layer */
MaxPooling2D<POOLING2_ROW, POOLING2_COL, NB_FILTER2, POOLING_ROW, POOLING_COL>                          pool2;

Reshape3D_1D<POOLING2_ROW/POOLING_ROW, POOLING2_COL/POOLING_COL, NB_FILTER2>                              reshape;

/* define the Dense layer */
Dense<DENSE_INPUT, DENSE_OUTPUT, RELU>                                                                  dense(weight3);

/* define the second Dense layer */
Dense<DENSE_OUTPUT, DENSE2_OUTPUT, SOFTMAX>                                                             dense2(weight4);
```

MNIST手写字识别-LENET5

- LeNet_v2
  - Local Mode
- LeNet_Stream_v3
  - Mixed Mode



⊟ Timing (ns)

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	9.00	9.43	1.13

⊟ Latency (clock cycles)

⊟ Summary

Latency		Interval		
min	max	min	max	Type
9905	9900005	9906	9900006	none

⊟ Detail

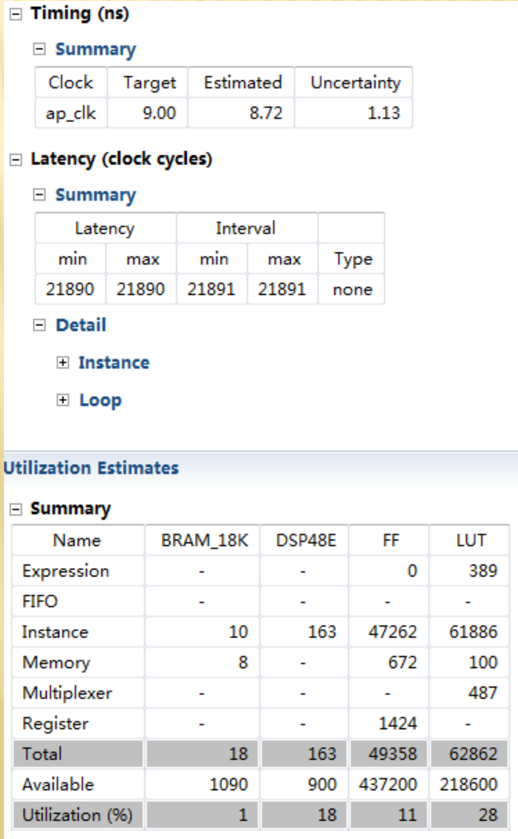
⊞ Instance

⊞ Loop

Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	384
FIFO	-	-	-	-
Instance	32	234	42650	62500
Memory	18	-	448	83
Multiplexer	-	-	-	327
Register	-	-	407	-
Total	50	234	43505	63294
Available	1090	900	437200	218600
Utilization (%)	4	26	9	28



⊟ Timing (ns)

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	9.00	8.72	1.13

⊟ Latency (clock cycles)

⊟ Summary

Latency		Interval		
min	max	min	max	Type
21890	21890	21891	21891	none

⊟ Detail

⊞ Instance

⊞ Loop

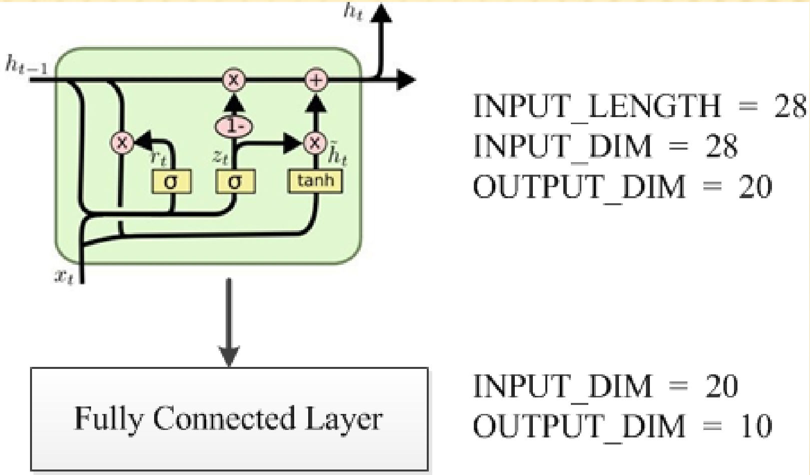
Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	389
FIFO	-	-	-	-
Instance	10	163	47262	61886
Memory	8	-	672	100
Multiplexer	-	-	-	487
Register	-	-	1424	-
Total	18	163	49358	62862
Available	1090	900	437200	218600
Utilization (%)	1	18	11	28

MNIST手写字识别-LSTM

- 2层网络、4130个参数、Float精度、94.2%准确率

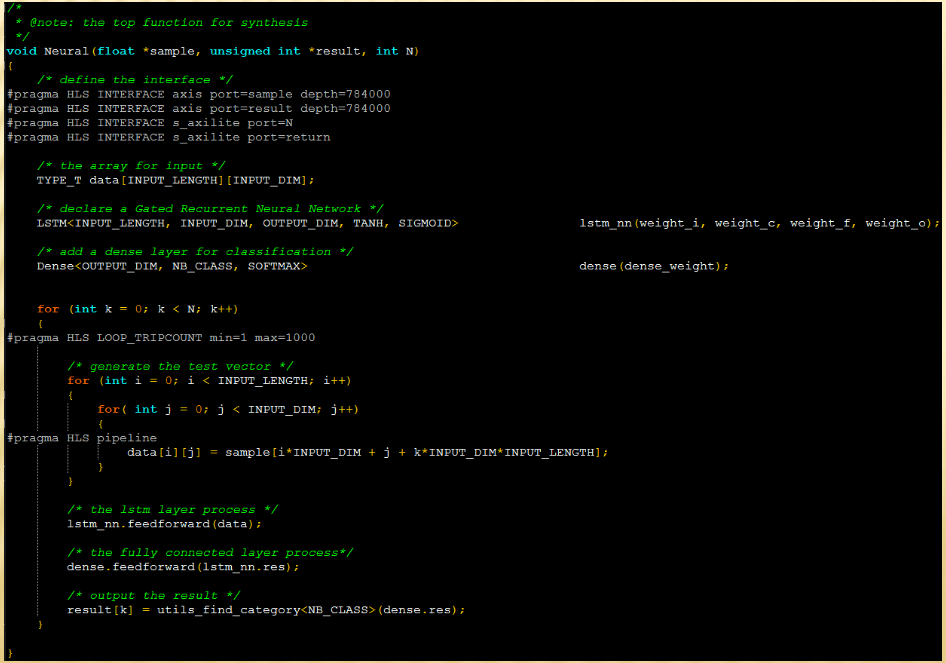


INPUT_LENGTH = 28
INPUT_DIM = 28
OUTPUT_DIM = 20

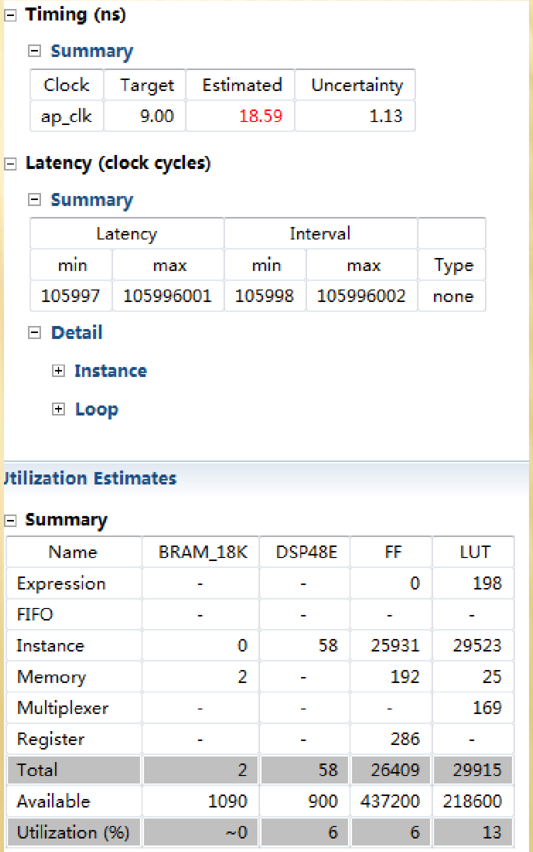
INPUT_DIM = 20
OUTPUT_DIM = 10

MNIST手写字识别-LSTM

- LSTM_v2



```
/*
 * @note: the top function for synthesis
 */
void Neural(float *sample, unsigned int *result, int N)
{
    /* define the interface */
#pragma HLS INTERFACE axis port=sample depth=784000
#pragma HLS INTERFACE axis port=result depth=784000
#pragma HLS INTERFACE s_axilite port=N
#pragma HLS INTERFACE s_axilite port=return

    /* the array for input */
    TYPE_T data[INPUT_LENGTH][INPUT_DIM];

    /* declare a Gated Recurrent Neural Network */
    LSTM<INPUT_LENGTH, INPUT_DIM, OUTPUT_DIM, TANH, SIGMOID>                lstm_nn(weight_i, weight_c, weight_f, weight_o);

    /* add a dense layer for classification */
    Dense<OUTPUT_DIM, NB_CLASS, SOFTMAX>                                    dense(dense_weight);


    for (int k = 0; k < N; k++)
    {
#pragma HLS LOOP_TRIPCOUNT min=1 max=1000

        /* generate the test vector */
        for (int i = 0; i < INPUT_LENGTH; i++)
        {
            for( int j = 0; j < INPUT_DIM; j++)
            {
#pragma HLS pipeline
                data[i][j] = sample[i*INPUT_DIM + j + k*INPUT_DIM*INPUT_LENGTH];
            }
        }

        /* the lstm layer process */
        lstm_nn.feedforward(data);

        /* the fully connected layer process*/
        dense.feedforward(lstm_nn.res);

        /* output the result */
        result[k] = utils_find_category<NB_CLASS>(dense.res);
    }

}
```



⊟ **Timing (ns)**

⊟ **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	9.00	18.59	1.13

⊟ **Latency (clock cycles)**

⊟ **Summary**

Latency		Interval		
min	max	min	max	Type
105997	105996001	105998	105996002	none

⊟ **Detail**

⊞ **Instance**

⊞ **Loop**

Utilization Estimates

⊟ **Summary**

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	198
FIFO	-	-	-	-
Instance	0	58	25931	29523
Memory	2	-	192	25
Multiplexer	-	-	-	169
Register	-	-	286	-
Total	2	58	26409	29915
Available	1090	900	437200	218600
Utilization (%)	~0	6	6	13

THANK YOU!

Q&A