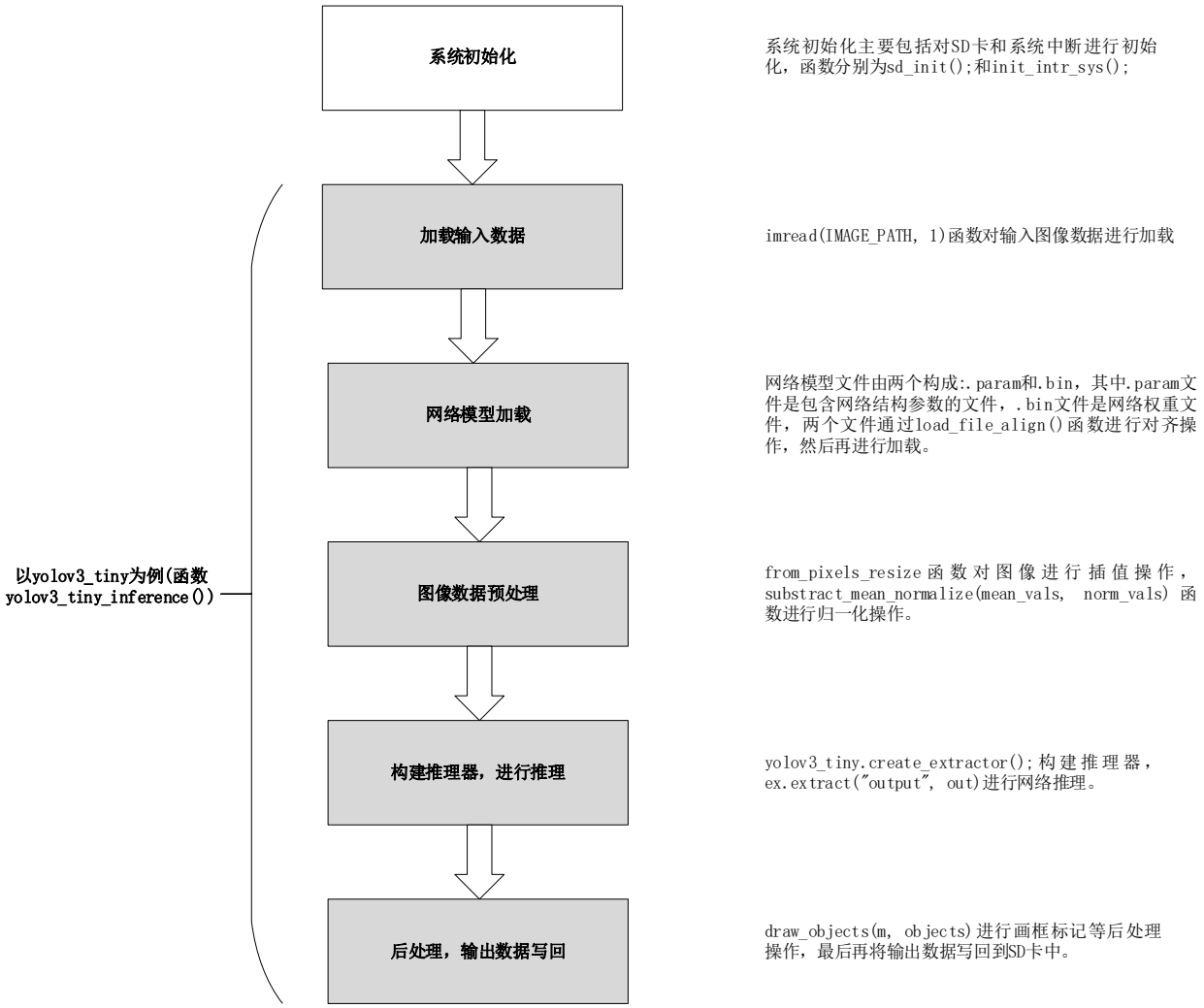
一. 系统工作流程（yolov3_tiny 网络为例）



上面的图示展示了 yolov3_tiny 网络在系统中的完整运行流程，其他网络的流程与其基本相同，所以这里再不进行赘述。

二.算子的寄存器配置

上面介绍的.param 文件是包含网络结构参数的文件，对于完整网络的推理，就是根据.param 文件按照 ncnn 中所定义的层为单位进行顺序解析，配置寄存器，调用 NPU 进行推理。NPU 支持的单个算子主要包括通用卷积、深度卷积、全连接、最大值池化、均值池化、常用的激活函数和 ALU 操作等，下面以池化为例子，对寄存器的配置过程进行介绍。

在介绍算子寄存器配置之前需要对工程的目录结构进行一个简要介绍。

src/

├── config.h

├── hw

```
├──── layer
├──── lscript.ld
├──── main.cpp
├──── ncnn
├──── net
├──── README.txt
├──── testLayer
├──── testOp
└──── utils
```

1. **config.h:**存放了整个系统工程全部的宏定义，重要的宏定义将在后文进行介绍；
2. **hw：存放了与硬件相关的函数文件，包括系统的初始化函数，以及底层算子推理的寄存器配置函数；**
3. **layer：**存放了 ncnn 的 layer 定义代码；
4. **main.cpp：**存放了系统工程的主函数；
5. **net：**存放了网络推理函数；
6. **testLayer：**存放了单个算子的仿真测试文件，包括卷积、池化、激活函数、ALU 等，每种算子都包含了大量仿真 case；
7. **utils：**存放了系统工程中调用的一些常用函数；

对于池化算子的寄存器列表和功能介绍可以详见 **ncnnAccel RegMap.pdf** 文档，这里介绍具体的配置过程。NPU 所有寄存器的地址定义全部在 **hw/accel_params.h** 文件中，**accel_params.h** 文件中定义了寄存器基地址 ACCEL_REG_BASEADDR 为 0x80020000，每个寄存器以 4 个字节为单位，根据 **ncnnAccel RegMap.pdf** 文档中寄存器的索引 ID 可知，池化算子的寄存器地址为 ACCEL_REG_BASEADDR＋40*4 至 ACCEL_REG_BASEADDR＋49*4，如下所示，

```
1.  //POOL
2.  #define ACCEL_REG_POOL_CTRL          ACCEL_REG_BASEADDR + 40*4
3.  #define ACCEL_REG_POOL_SHAPE_IC      ACCEL_REG_BASEADDR + 41*4
4.  #define ACCEL_REG_POOL_SHAPE_IWH     ACCEL_REG_BASEADDR + 42*4
5.  #define ACCEL_REG_POOL_SHAPE_ICSTEP ACCEL_REG_BASEADDR + 43*4
6.  #define ACCEL_REG_POOL_SHAPE_OC      ACCEL_REG_BASEADDR + 44*4
7.  #define ACCEL_REG_POOL_SHAPE_OWH     ACCEL_REG_BASEADDR + 45*4
8.  #define ACCEL_REG_POOL_SHAPE_OCSTEP ACCEL_REG_BASEADDR + 46*4
9.  #define ACCEL_REG_POOL_IFM_ADDR      ACCEL_REG_BASEADDR + 47*4
10. #define ACCEL_REG_POOL_OFM_ADDR      ACCEL_REG_BASEADDR + 48*4
11. #define ACCEL_REG_POOL_PAD_VALUE     ACCEL_REG_BASEADDR + 49*4
```

下面介绍池化算子的寄存器配置宏定义，在 **hw/accel_params.h** 文件中由下面十个宏定义构成，完成对上述十个寄存器的配置，其中函数 Xil_Out32(ACCEL_REG_POOL_CTRL, x)的功能为往指定地址的寄存器中写入指定的数据，这个函数是 Xilinx 的库函数，平台移植需要进行替换；

```
/*****************************POOL*******************************/
#define POOL_RESET                  Xil_Out32(ACCEL_REG_POOL_CTRL, 0)
#define POOL_CTRL_SET(en,op,kernel_w,kernel_h,stride_w,stride_h,src_sel,pad_bottom,pad_top,pad_right,pad_left,pad_mode) \
                                    Xil_Out32(ACCEL_REG_POOL_CTRL, en | op<<1 | kernel_w<<3 | kernel_h<<5 | stride_w<<7 | stride_h<<9 | src_sel<<11 | pad_bottom<<16 | pad_top<<18 | pad_right<<20 | pad_left<<22 | pad_mode<<24)
#define POOL_IFM_C_SET(c)           Xil_Out32(ACCEL_REG_POOL_SHAPE_IC, c)
#define POOL_IFM_WH_SET(w,h)        Xil_Out32(ACCEL_REG_POOL_SHAPE_IWH, w<<16 | h)
#define POOL_IFM_CSTEP_SET(x)       Xil_Out32(ACCEL_REG_POOL_SHAPE_ICSTEP, x)
#define POOL_OFM_C_SET(c)           Xil_Out32(ACCEL_REG_POOL_SHAPE_OC, c)
#define POOL_OFM_WH_SET(w,h)        Xil_Out32(ACCEL_REG_POOL_SHAPE_OWH, w<<16 | h)
#define POOL_OFM_CSTEP_SET(x)       Xil_Out32(ACCEL_REG_POOL_SHAPE_OCSTEP, x)
#define POOL_IFM_ADDR_SET(x)        Xil_Out32(ACCEL_REG_POOL_IFM_ADDR, x)
#define POOL_OFM_ADDR_SET(x)        Xil_Out32(ACCEL_REG_POOL_OFM_ADDR, x)
#define POOL_PAD_VALUE_SET(x)       Xil_Out32(ACCEL_REG_POOL_PAD_VALUE, x)
```

在 **hw/hw_pool.cpp** 中函数 forward(ncnn::Mat &ifm, ncnn::Mat &ofm)就对上述的十个宏进行调用，代码解析详见注释。不同的 case 情况会传入相应的值完成寄存器的配置，对于单个池化算子测试的顶层 case 可以详见 **testLayer/test_layer_pool.cpp** 文件中的函数 void test_layer_pool_batch()。

```
void hw_pool::forward(ncnn::Mat &ifm, ncnn::Mat &ofm) {
    POOL_RESET;    //寄存器复位
    Xil_DCacheFlush();  //刷 cache
    int* src0_addr0 = ifm;
    int* dst_addr0 = ofm;
    POOL_IFM_ADDR_SET((uint64_t)src0_addr0);  //配置输入特征图基地址
    POOL_OFM_ADDR_SET((uint64_t)dst_addr0);  //配置输出特征图基地址
    POOL_IFM_WH_SET(ifm.w,ifm.h);  //配置输入特征图的 w 和 h
    POOL_IFM_C_SET(ifm.c);  //配置输入特征图的 c
    POOL_IFM_CSTEP_SET(ifm.cstep);  //配置输入特征图的 cstep
    POOL_OFM_WH_SET(ofm.w,ofm.h);  //配置输出特征图的 w 和 h
    POOL_OFM_C_SET(ofm.c);  //配置输出特征图的 c
    POOL_OFM_CSTEP_SET(ofm.cstep);  //配置输出特征图的 cstep
    POOL_PAD_VALUE_SET(*((uint32_t*)&pad_value)); //配置 padding 的常数值
```

```
15.     POOL_CTRL_SET(1, pool_type,kernel_w,kernel_h,stride_w,stride_h,1,pad_bottom,pad_top
    ,pad_right,pad_left,pad_mode); //配置池化的使能、类型、kernel 的 wh，stride 的 wh，padding
    的值、padding 的模式等
16.     wait_pool_done();//等待池化计算完成
17.     Xil_DCacheInvalidate();  //刷 cache
18.     POOL_RESET;  //池化寄存器复位
19. }
```

其他算子的寄存器配置过程与池化算子基本相同，这里也不再详细赘述。