WTM2101 存算编译工具链介绍

知存科技

软件整体框架—软件架构

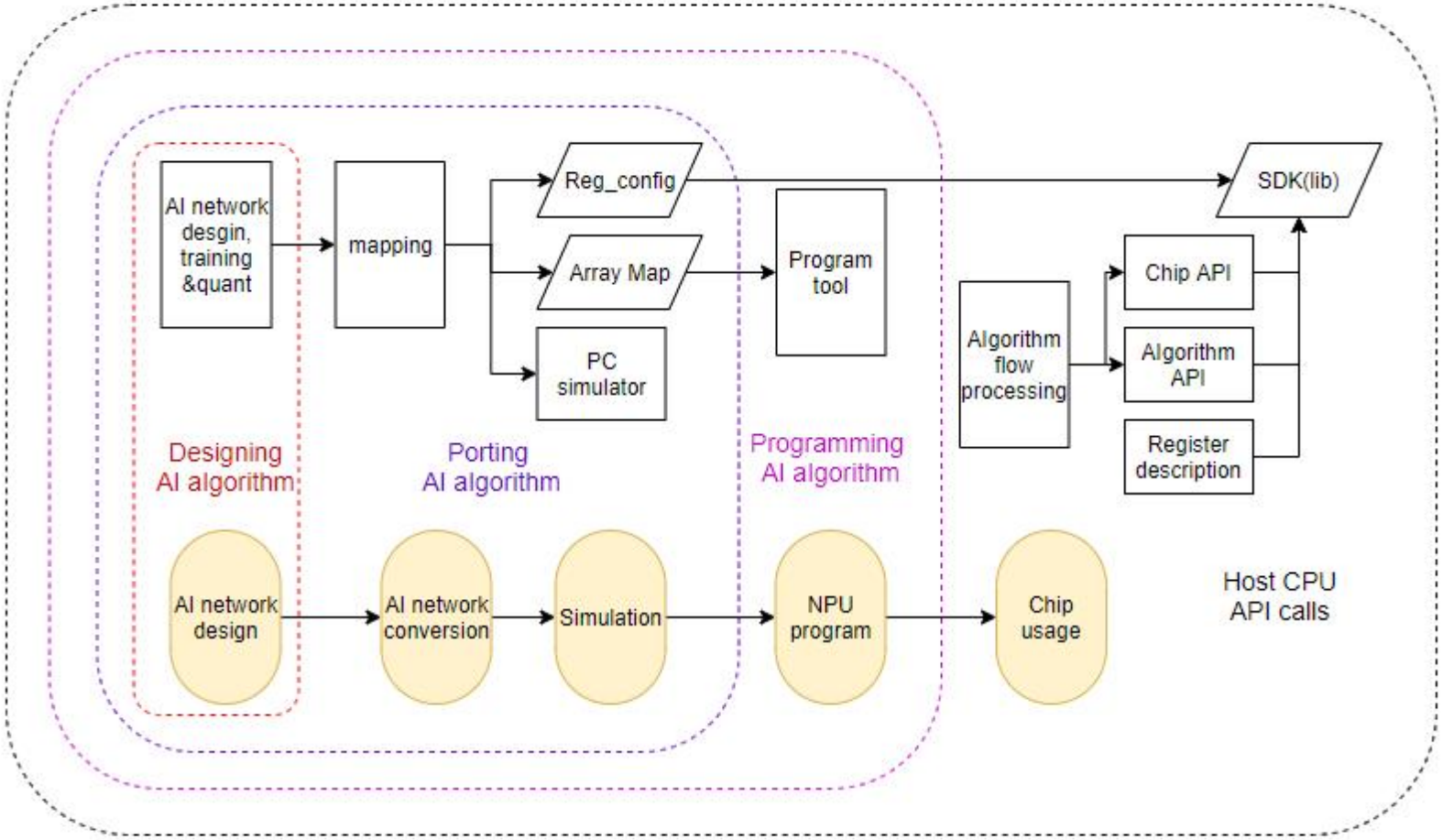


图3. 软件架构图

WTM2101 AI ToolChain

witin_mapper是知存科技自研的用于神经网络映射的编译软件栈，可以将量化后的神经网络模型映射到WTM2101 MPU加速器上，是一种包括RiscV和MPU的完整解决方案，可以完成算子和图级别的转换和优化，将预训练权重编排到存算阵列中，并针对网络结构和算子给出存算优化方案，同时将不适合MPU运算的算子调度到CPU上运算，实现整网的调度，让神经网络开发人员高效快捷的将训练好的算法运行在WTM2101芯片上，极大缩短模型移植的开发周期并提高算法开发的效率。

获取witim_mapper：

1、直接安装

2、docker容器

工具链整体框架—docker容器安装

1) 获取镜像：

```
docker pull witin/toolchain:v001.000.034
```

2) 查看镜像：

```
docker images
```

3) 创建容器：

```
docker run -it --name=witin_toolchain witin/toolchain:v001.000.034
```

(witin_toolchain为创建容器名称，witin/toolchain:v001.000.034为witin_mapper镜像，创建后会默认进入workspace文件夹下)

4) 运行测试：

进入workspace下的witin_mapper文件夹，并验证工具链安装是否正常。

```
cd witin_mapper
sh auto_test.sh
```

正常执行，即可验证工具链容器安装OK.

工具链整体框架—docker容器安装

5) 退出容器：

```
exit
```

6) 重新进入容器：

```
docker start witin_toolchain
docker attach witin_toolchain
```

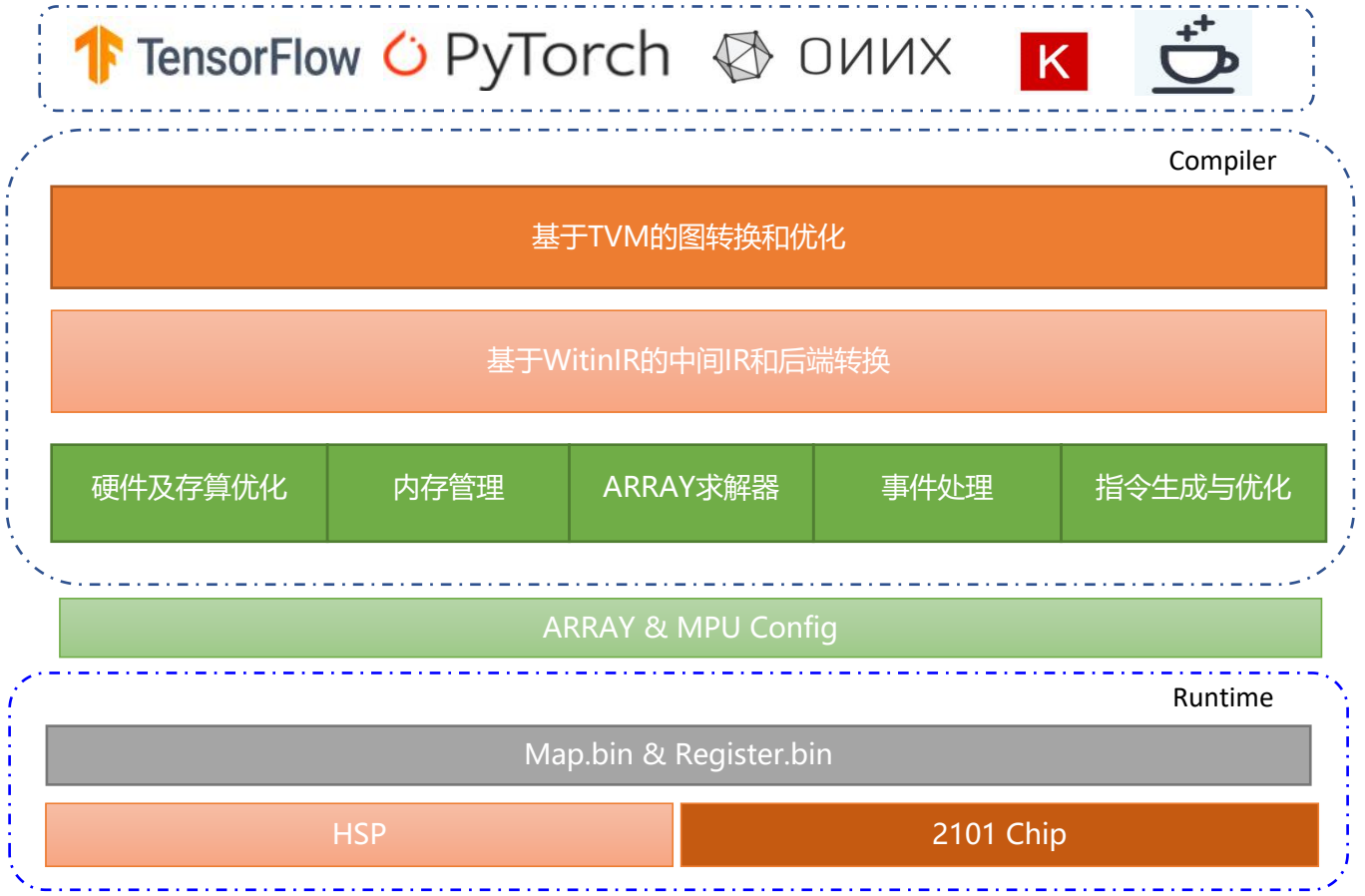
7) 将文件（或文件夹）上传至容器：

```
docker cp 本地文件路径 witin_toolchain:/workspace/witin_mapper(即本地容器名称:/+容器内路径)
```

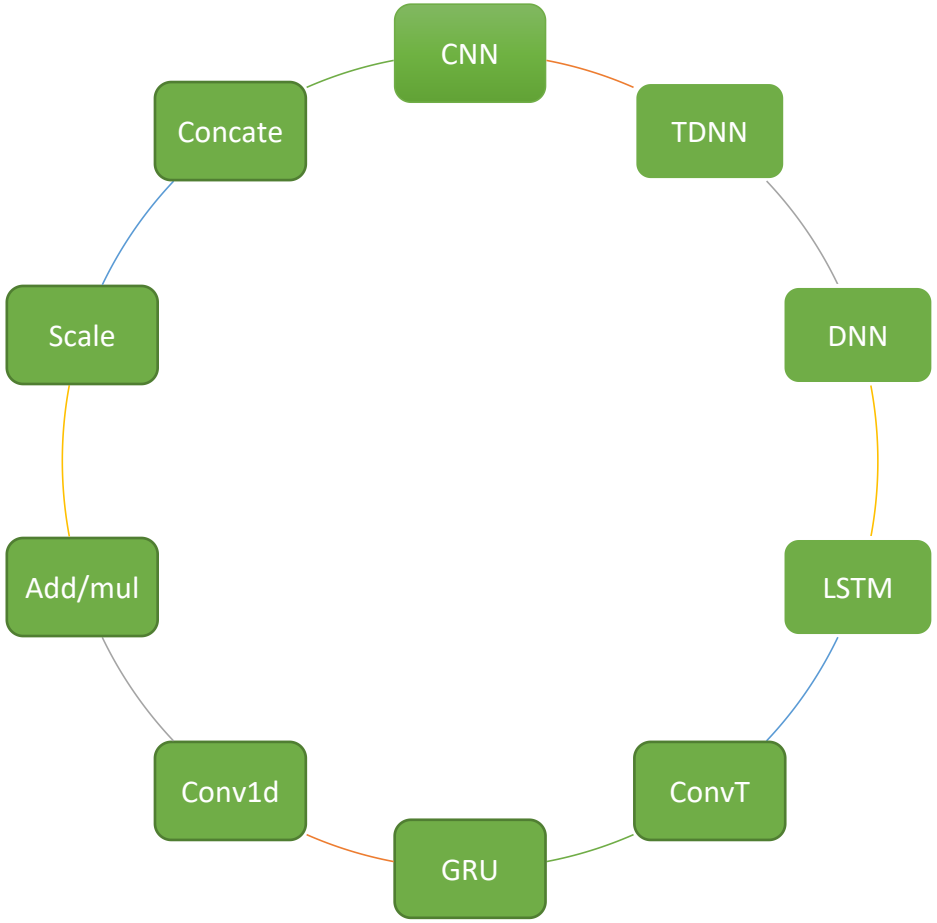
8) 将容器文件（或文件夹）下载至本地：

```
docker cp witin_toolchain:/workspace/witin_mapper(即本地容器名称:/+容器内路径) 本地文件路径
```

WTM2101软件栈-witin_mapper



工具链整体框架—支持算子



支持　优化中　未支持

图5. 工具链支持网络

工具链整体框架—图结构优化

算子融合

- Conv+Scale+Activation
- Conv/Gemm+Scalar

算子转换

- Reshape+Gemm --> Conv+Reshape
- Add --> EltwiseAdd/Scalar
- Mul --> EltwiseMul/Scalar
- Scale --> Scalar

卷积维度转换

- [N,C,H,W] ---> [N,W,H,C]

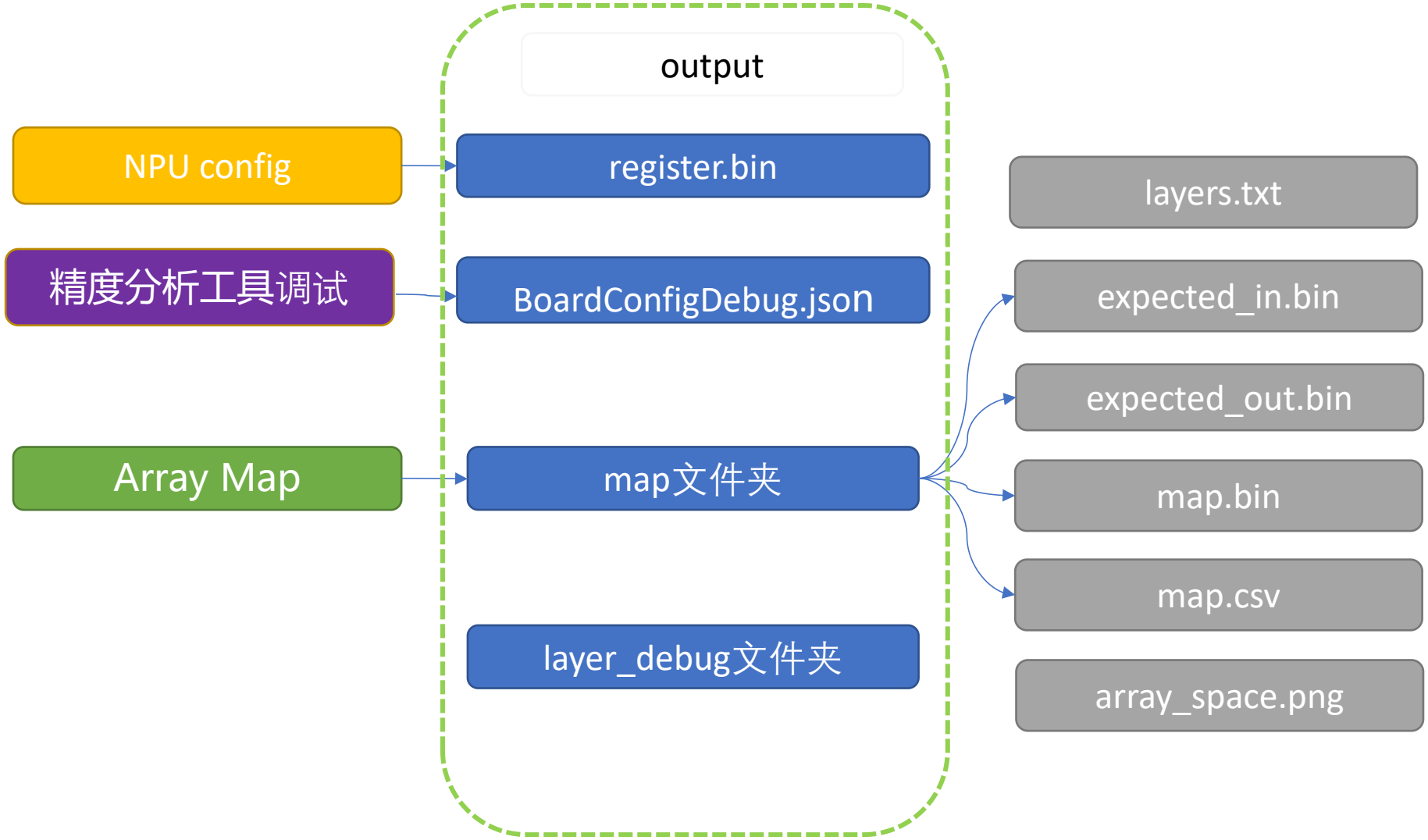
工具链使用—前端模型导入


```
mod, params = witin_frontend.frontend.from_tensorflow(graph_def,  shape_dict)
```

```
with witin.transform.PassContext(opt_level=3):
    npu_graph, mod, params = witin_frontend.build_module.build(mod, target=target,
    target_host=target_host, params=params, input_data=input_dt ,
    chip="BB04P1")
```

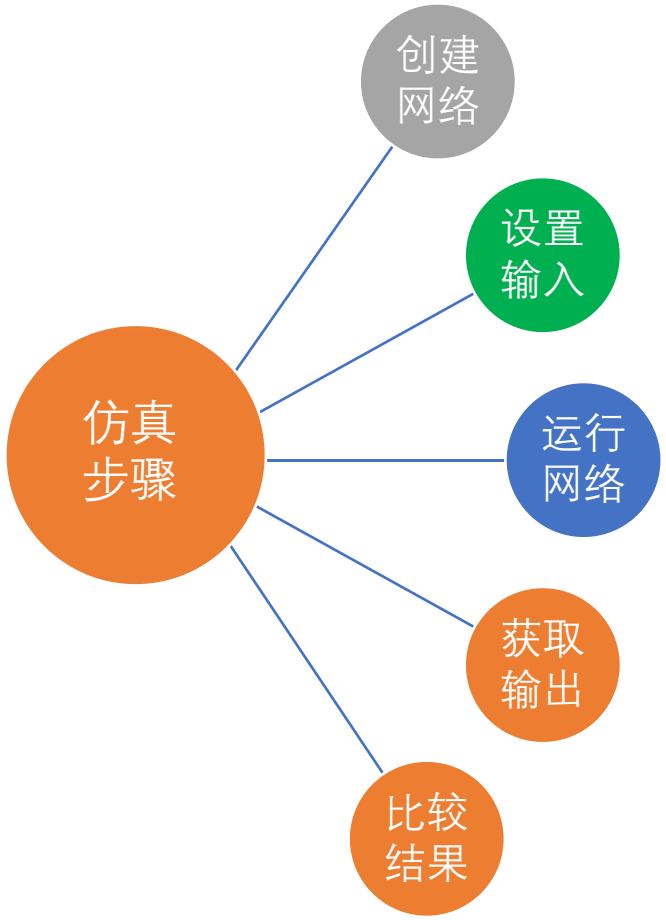
witin阵列Array校准需要用到的输入数据,且需要大于100帧。

工具链整体框架——文件生成



工具链输出文件结构

工具链整体框架——工具链仿真



工具链仿真步骤

- m=npu_graph_runtime.create(npu_graph)
- m.set_input（‘in1’, wintin.nd.array(data))
- m.run()
- m.get_output(0)

➢ npu功能仿真器，模拟npu的相关计算，c++接口。

- Enter json mode!
- func simulation result is equal to cmodel result!

工具链整体框架——前端模型导入

□ 多网络

```
mod_left, param_left = witin_frontend.frontend.from_onnx(graph_left, shape_dict_left)
mod_right, param_right = witin_frontend.frontend.from_onnx(graph_right, shape_dict_right)
mul_mods = []; mul_params = [] mul_input_datas = []
mul_mods.append(mod_left);    mul_mods.append(mod_right)
mul_params.append(param_left); mul_params.append(param_right)
mul_input_datas.append(input_left); mul_input_datas.append(input_right);
```

```
with witin.transform.PassContext(opt_level=3):
    _, _, _, npu_graph = witin_frontend.build_module.build(mul_mods, target='npu',
    target_host='npu', params=mul_params,  input_data=mul_input_datas,
    chip="BB04P1",
    output_dir=build_dir,
    optimize_method_config='opt_config.protobuf')
```

工具链整体框架——工具链特殊配置环境变量

□ Array分配

a） export ARRAY_ALLOC_RESERVED_COLUMN=N跳过前N列的Array分配

默认N=128,表示避开前128列，也可以通过python/witin/__init__.py中

os.environ['ARRAY_ALLOC_RESERVED_COLUMN'] = ‘N'进行设置

b） export ARRAY_ALLOC_TIMEOUT_LIMITS = Time_out

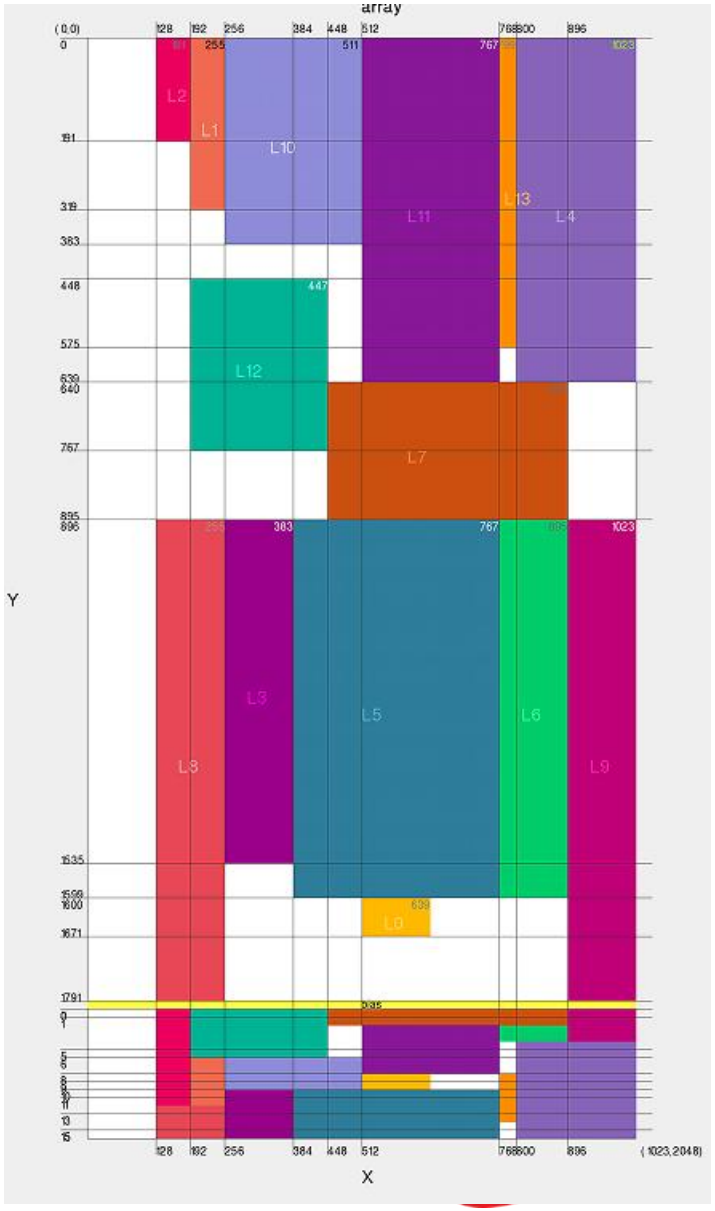
默认的搜索超时时间通常在10s~100s之间

□ 开启FIFO

a） export WITIN_FIFO_EN = 1，设置使用硬件fifo

b）也可以通过python/witin/__init__.py中

os.environ[‘WITIN_FIFO_EN’] = ‘1’ 进行设置



工具链整体框架——工具链特殊配置环境变量

□ 设置校准数据数量

a) export WITIN_EXPECTED_NUM=N，会保存N帧的校准数据，至少N=100,也可以通过 python/witin/__init__.py中 os.environ[‘WITIN_EXPECTED_NUM’] = N设置

工具链整体框架——工具链特殊配置环境变量

□ 其他设置

a) os.environ[‘DMLC_LOG_DEBUG’] = ‘N’

N可取0、1、2，数字越大，log等级更详细：0最粗糙，1居中，2最详细

b) os.environ[‘SAVE_LOG_MESSAGE’] = ‘0’，0表示不保存log，1表示保存log


WITiNMEM
知存科技