



HiSVP

开发指南

文档版本 00B05

发布日期 2017-09-18

版权所有 © 深圳市海思半导体有限公司 2017。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前言

概述

本文档旨在帮助用户了解 SVP(Smart Vision Platform)的硬件特性、工具链及开发流程，以期达到快速上手和开发出充分利用 SVP 特性的智能方案。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3559A	V100ES



读者对象

本文档（本指南）主要适用于以下工程师：




- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。



符号	说明
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	版本	修改描述
2017-09-18	00B05	第五次临时版本发布 2.3、2.6.2 和 3.5.2 小节涉及修改
2017-07-20	00B04	第四次临时版本发布 3.3.3 小节涉及修改
2017-06-30	00B03	2.2 小节添加注意 新增 3.1.2 小节，修改 3.1.1 新增 3.3 小节 3.5.1.4 和 3.5.2 小节涉及修改 3.6.2.1 小节，修改图 3-43~图 3-46 3.6.3.1 小节，修改图 3-51~图 3-53 3.6.3 节，表 3-2 和表 3-3 涉及修改 3.6.9 小节涉及修改
2017-05-27	00B02	第二次临时版本发布 新增 1.5 小节 3.4.1.1、3.4.1.4、3.5.2.1、3.5.3.1、3.5.3.4、3.5.4.1、 3.5.5.1、3.5.11.2、3.5.11.3 小节涉及修改 3.5.3.2 小节，表 3-1 涉及修改 3.5.4.2 小节，表 3-2 涉及修改 3.5.5.2 小节，表 3-3 涉及修改
2017-04-10	00B01	第一次版本发布。



目 录

1 概述	1
1.1 SVP 简介	1
1.2 开发框架	1
1.3 硬件资源	2
1.4 软件开发	3
1.5 开发环境	3
1.6 相关文档	4
2 DSP 开发指南	5
2.1 开发工具介绍	5
2.2 Windows 环境下安装 DSP 工具链和配置核	5
2.3 修改 memmap 配置	8
2.4 查看 stack 使用	9
2.5 Linux 环境下安装 DSP 工具链和配置核	9
2.5.1 安装 DSP 工具链	9
2.5.2 安装 DSP 配置核	16
2.5.3 配置环境变量	18
2.6 开发流程	18
2.6.1 开发模式一	19
2.6.2 开发模式二	20
3 NNIE 开发指南	21
3.1 NNIE 介绍	21
3.1.1 NNIE 支持规格	21
3.1.2 NNIE 硬件资源利用率	23
3.2 工具链介绍	23
3.3 工具链安装	24
3.3.1 NNIE 工具链本体安装	24
3.3.2 Protobuf 安装	24
3.3.3 OpenCV 安装	25
3.4 开发流程	26
3.5 Prototxt 要求及模型下载	27



3.5.1 Prototxt 要求.....	27
3.5.2 网络扩展说明.....	48
3.5.3 模型下载	51
3.6 HiSVP_CNNIE8Bit 使用	53
3.6.1 创建工程	53
3.6.2 网络过滤器	57
3.6.3 编译器	62
3.6.4 仿真器	67
3.6.5 prototxt 编辑.....	70
3.6.6 终止	72
3.6.7 Dot 图	72
3.6.8 性能分析	74
3.6.9 目标检测	75
3.6.10 偏好设置	77
3.7 仿真库使用.....	80
3.8 FAQ.....	80
3.8.1 软件运行期间报错 “periodic workspace save failed”	80
3.8.2 软件运行崩溃，无法启动.....	80
3.8.3 工具栏没有显示.....	80



插图目录

图 1-1 SVP 开发框架	2
图 1-2 SVP 基于 OpenVX 的开发框架	3
图 2-1 System Overview	6
图 2-2 Find and Install a Configuration Build	7
图 2-3 查看安装好的配置核	8
图 2-4 创建的工程界面	8
图 2-5 查看栈信息操作界面	9
图 2-6 显示栈信息窗口	9
图 2-7 DSP 应用程序开发框图	19
图 3-1 NNIE 层间约束	21
图 3-2 HiSVP_CNNIE8Bit 集成 UI 目录	24
图 3-3 执行 tests.exe 结果	25
图 3-4 第三方开源库放置目录	25
图 3-5 NNIE 软件开发流程	26
图 3-6 HiSVP_CNNIE8Bit.exe 支持 prototxt 格式说明示意图（左边支持）	28
图 3-7 Faster RCNN 类型网络四段式执行	30
图 3-8 RFCN 类型网络两段式执行	31
图 3-9 Report1 修改前后对比	33
图 3-10 Report2 修改前后对比	33
图 3-11 Report3 修改前后对比	34
图 3-12 Report4 修改前后对比	34
图 3-13 Report5 修改前后对比	34
图 3-14 Split 修改前后对比	35
图 3-15 Report1 修改前后对比	35
图 3-16 Report2 修改前后对比	35



图 3-17 Report3 修改前后对比	36
图 3-18 Report4 修改前后对比	36
图 3-19 SSD 类型网络四段式执行	37
图 3-20 Report1 修改前后对比	38
图 3-21 Split1 修改前后对比	38
图 3-22 Report13 修改前后对比	39
图 3-23 Report12 修改前后对比	39
图 3-24 Report3 修改前后对比	40
图 3-25 Report2 修改前后对比	41
图 3-26 Report5 修改前后对比	42
图 3-27 Report4 修改前后对比	42
图 3-28 Report7 修改前后对比	43
图 3-29 Report6 修改前后对比	44
图 3-30 Report9 修改前后对比	45
图 3-31 Report8 修改前后对比	46
图 3-32 Report11 修改前后对比	47
图 3-33 Report10 修改前后对比	48
图 3-34 Faster RCNN、iFDT 类型网络扩展	49
图 3-35 SSD 类型网络扩展	50
图 3-36 RFCN 类型网络扩展	51
图 3-37 创建工程方式一	54
图 3-38 创建工程方式二	54
图 3-39 创建工程方式三	54
图 3-40 选择 CNNIE8Bit Project	55
图 3-41 设置工程名及工程路径	56
图 3-42 创建工程后的工程视图	56
图 3-43 check.cfg 文件两种打开方式	57
图 3-44 check.cfg 以 check Net Type Configuration Editor 模式打开	58
图 3-45 check.cfg 选择 faster RCNN 网络	58
图 3-46 check.cfg 选择 Text Editor 模式打开	58
图 3-47 Check 按钮	61
图 3-48 网络过滤器输出	61



图 3-49 网络 check 成功输出示意图 .	61
图 3-50 网络 check 失败示意图 .	62
图 3-51 compiler.cfg 以 Compiler Configuration Editor 模式打开 .	62
图 3-52 compiler.cfg 选择 faster RCNN.	63
图 3-53 compiler.cfg 选择 Text Editor 模式打开.	64
图 3-54 Compile 按钮	66
图 3-55 sim.cfg 以 Simulator Configuration Editor 模式打开 .	67
图 3-56 sim.cfg 以 Text Editor 模式打开 .	68
图 3-57 Simulator 按钮	70
图 3-58 Prototxt 编辑.	71
图 3-59 Prototxt Format 选项	71
图 3-60 Prototxt 格式化前后对比.	72
图 3-61 Terminate 按钮	72
图 3-62 Dot 按钮	72
图 3-63 Dot 图效果	73
图 3-64 保存 Dot 图	73
图 3-65 保存 Dot 图路径及文件名设置.	74
图 3-66 性能分析按钮	74
图 3-67 性能分析结果	75
图 3-68 目标检测示图	76
图 3-69 坐标文件	76
图 3-70 目标检测结果	76
图 3-71 代码格式	77
图 3-72 代码提示快捷键设置	78
图 3-73 设置自动提示首字符	78
图 3-74 代码提示效果	79
图 3-75 颜色设置	79



1 概述

1.1 SVP 简介

SVP(Smart Vision Platform)是海思媒体处理芯片智能视觉异构加速平台。该平台包含了 CPU、DSP、NNIE(Neural Network Inference Engine)等多个硬件处理单元和运行在这些硬件上的基于 OpenVX1.1 框架的软件开发环境，以及配套的工具链开发环境。

本文档主要介绍 SVP 的硬件特性、配套工具链及开发流程，旨在帮助用户快速入门以及开发出充分利用 SVP 硬件特性的智能应用。基于 OpenVX 的软件开发介绍请参考《HiSVP API 参考》文档。

1.2 开发框架

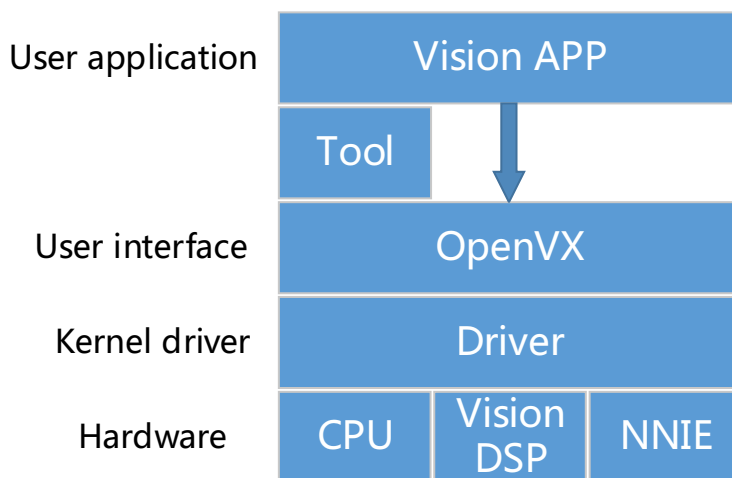
SVP 开发框架如[图 1-1](#) 所示。目前 SVP 中包含的硬件处理单元有 CPU、vision DSP、NNIE，其中某些硬件可能有多核或者多个。

整个 SVP 的用户软件接口大部分符合 OpenVX 1.1 软件框架标准，如[图 1-2](#)，仅有少部分不支持，同时为适应海思媒体平台做了一些扩展接口，《HiSVP API 参考》文档。

不同的硬件有不同的配套工具链，用户的应用程序需要结合这些工具的使用来开发。



图1-1 SVP 开发框架



1.3 硬件资源

不同的芯片 SVP 会使用不同硬件资源，如表 1-1：

表1-1 不同芯片下的 SVP 硬件资源

芯片	SVP 硬件资源		
	CPU	DSP	NNIE
Hi3559AV100ES	双核 A73+双核 A53	2 个	1 个

针对 CPU 的具体规格，请参考 ARM 官方文档。

DSP 和 NNIE 的硬件规格，可以参考《Hi3559AV100ES ultra-HD Mobile Camera SoC 用户指南》。



注意

不同的芯片 SVP 可能会使用不同的硬件资源，即便是使用相同的硬件型号，硬件的配置也不一定相同。

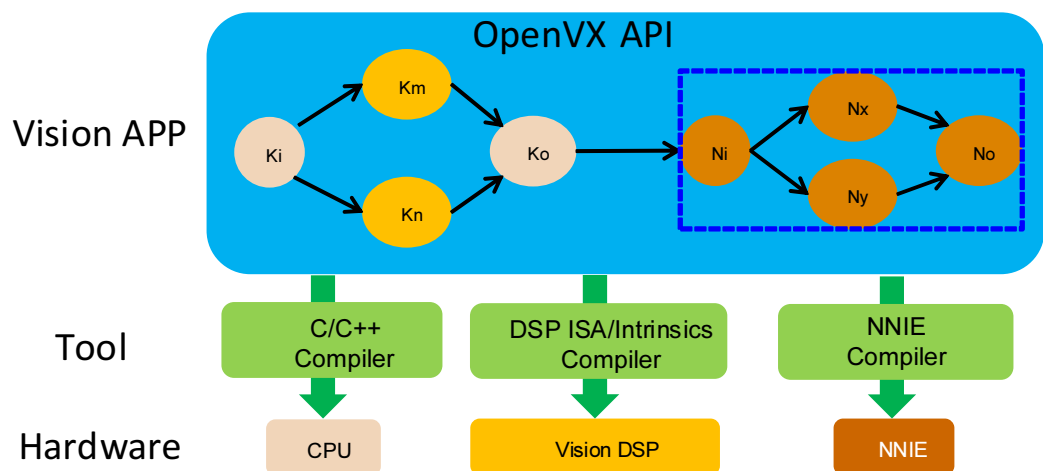


1.4 软件开发

SVP 采用基于 OpenVX1.1 的软件开发框架，如图 1-2 所示。用户程序可以形象的以 Node 组成的 Graph 来表示，不同的 Node 可以执行到不同的硬件中（Graph、Node 的定义参考《HiSVP API 参考》）。用户可以根据 SVP 的软硬件特性开发出能最大化利用 SVP 硬件资源的视觉处理应用。

SVP 是海思媒体处理芯片的智能加速平台，因此需要结合海思 MPP 平台一起来进行软件开发，可参考相关文档《HiMPP V4.0 媒体处理软件开发参考》。

图1-2 SVP 基于 OpenVX 的开发框架



1.5 开发环境

不同的芯片 SVP 会在不同的环境上运行，如表 1-2 所示。

表1-2 不同芯片下的 SVP 运行环境

芯片	操作系统
Hi3559AV100ES	Linux(big.LITTLE)

对于 Hi3559AV100ES: SVP 布局在双核 A73 和双核 A53 组成的 big.LITTLE 上，芯片媒体业务布局在单核 A53 上，智能业务需要的图像数据是通过核间通信从单核 A53 上获取。具体多核业务开发可以参考《HiMPP 多核 使用指南》和 SDK 包提供的 sample。



1.6 相关文档

《Hi3559AV100ES ultra-HD Mobile Camera SoC 用户指南》

《HiSVP API 参考》

《HiMPP V4.0 媒体处理软件开发参考》

《HiMPP 多核 使用指南》



2 DSP 开发指南

2.1 开发工具介绍

Xtensa Xplorer 是 Cadence 为客户提供的针对其 DSP 进行软件开发的一个集成开发环境，其包含了软件开发(software development)、编译(compile)、调试(debugging)、仿真(simulation)、性能分析(profiling)、硬件跟踪(hardware trace)等功能。

Xtensa Xplorer 的安装使用可参考 Cadence 提供的官方文档，以下均简称为 Xplorer。

由于 SVP 在不同芯片上可能使用不同的 DSP；即便是相同型号的 DSP，硬件配置也可能不一样。在有 DSP 的芯片配套开发包中，海思为用户提供了与硬件相同配置的配置核供用户在 Xplorer 下仿真开发。

配置核在 HiSVP_PC_Vx.x.x.x 的 tool 目录下与芯片中 DSP 的型号一致的目录中，如 tool/vp6，其中 windows、linux 目录分别在 windows、linux 上 Xplorer 的配置核。



注意

HiSVP_PC_Vx.x.x.x 的 tool 中可能提供了多个 dsp 配置核，分别对应到实际芯片中多个 dsp 的配置。

2.2 Windows 环境下安装 DSP 工具链和配置核

本文假定用户在 windows 环境中已经安装好了 Xplorer7.0.4，并且获取到了海思提供的 Hi3559AV100ES VP6 配置核 VP61_1124，以下均以此为例进行配置核安装，不同芯片不同配置核的在不同 Xplorer 版本的安装类似：

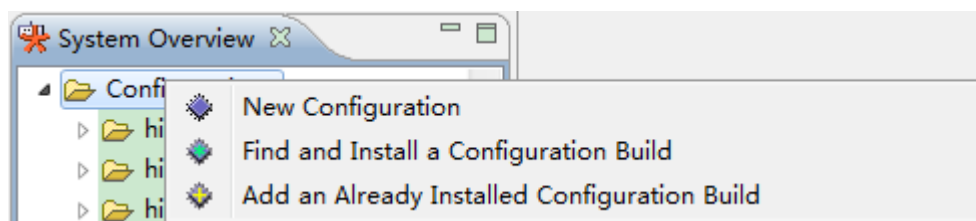


注意

Hi3559AV100ES VP6 配置核 VP61_1124 需与 RG2016.4 版本的工具包配套使用，Xplorer7.0.4 默认集成了 RG2016.4 工具包，如用户使用其他版本的 Xplorer，需要另外安装 RG2016.4 工具包。

- 步骤 1. 打开 Xplorer 7.0.4，右键 System Overview 窗口下的 Configurations，如图 2-1 所示，点击 Find and Install a Configuration Build；

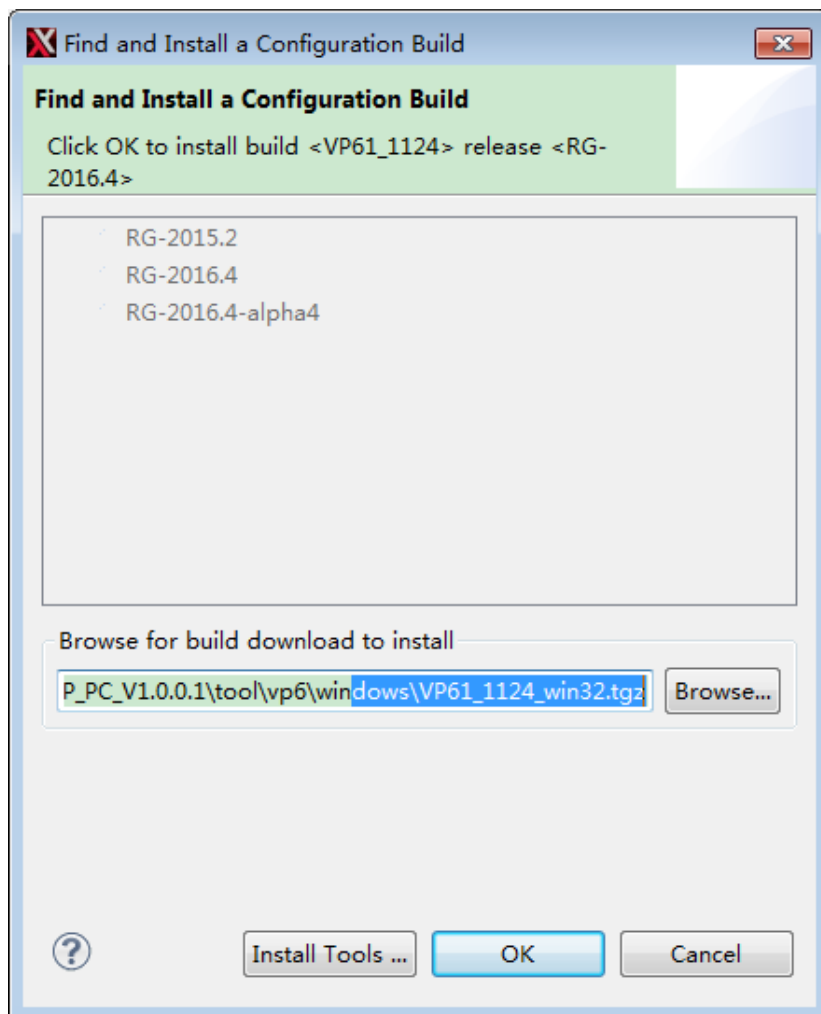
图2-1 System Overview



- 步骤 2. 进入 Find and Install a Configuration Build 界面，点击“Browse”，选择待安装配置核（此处以 VP61_1124_win32 为例），点击“OK”即开始安装；

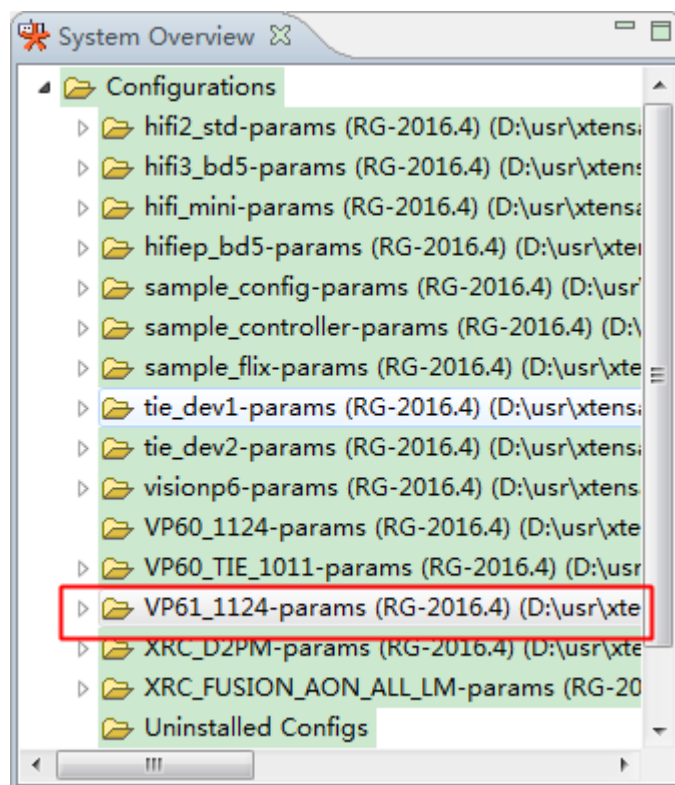


图2-2 Find and Install a Configuration Build



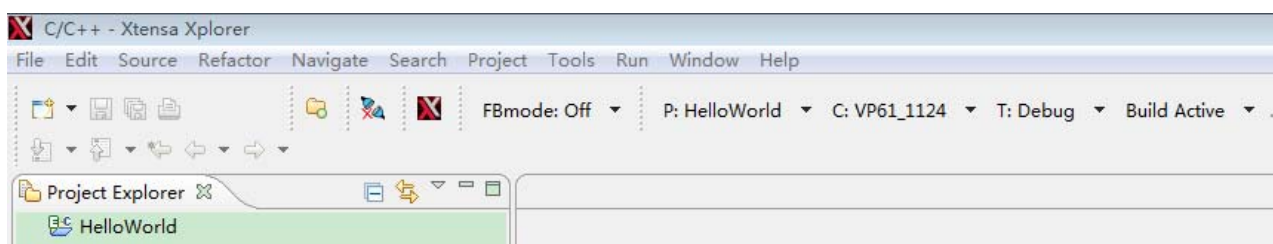
安装好后，在 SystemOverview 界面的 Configurations 下会出现刚刚安装的配置核，如图 2-3 所示，至此，VP61_1124 配置核安装成功。

图2-3 查看安装好的配置核



步骤 3. 运行工程（图 2-4 以 HelloWorld 为例）时，按照图 2-4 选择 Configuration 为图 2-4 中的 C:VP61_1124，即表示使用的是海思提供的配置核。

图2-4 创建的工程界面



----结束

2.3 修改 memmap 配置

用户可以参考 Xtensa® Linker Support Packages (LSPs) Reference Manual 的第二章。下面以海思开发包提供的 dsp0 memmap.xmm 文件做说明，如下所示：

- SRAM 部分是 DSP 在 DDR 执行指令和在 DDR 上的数据段空间，不可修改；
- dram0 和 dram1 是 DSP 内部数据存储，各有 192KB 分配给算法开发使用；



- fixvector 作为 DSP 复位使用空间，系统固定分配；
- iram0 作为 DSP 向量空间，系统固定分配；
- iobypass 为系统固定分配的串口调试空间。

```
//discription: memmap.xmm file
```

```
VECBASE = 0x40000400          //dynamic vector base address
```

```
ENTRY = "_PostResetVector"
```

```
PLACE SECTIONS(.PostResetVector.text) WITH_SECTION(.ResetVector.text)
```

```
//PLACE SECTIONS(.MemoryExceptionVector.literal)
```

```
WITH_SECTION(.MemoryExceptionVector.text) //move .literal to .text segment
```

```
//os_init_data
```

```
//3M
```

```
BEGIN SRAM_00
```

```
0x40008000: sysram: SRAM_00: 0x300000: writable ;
```

```
SRAM_00:C: 0x40008000-
```

```
0x40307fff: .os.hshell.data .os.jmptable.data .os.init.data .os.data .data .os.shared.data .os.shar  
ed.bss .os.init.bss .os.bss .bss;
```

```
END SRAM_00
```

```
//1M
```

```
BEGIN SRAM_01
```

```
0x40308000: sysram: SRAM_01: 0x100000: executable, writable ;
```

```
SRAM_01:C: 0x40308000-
```

```
0x40407fff: .os.data.icunit .os.shared.test.data .literal .text .os.kernel.literal .os.kernel.text .os.  
monitor.literal .os.monitor.text;
```

```
END SRAM_01
```

```
//1M
```

```
BEGIN SRAM_02
```

```
0x40408000: sysram: SRAM_02: 0x100000: executable, writable ;
```

```
SRAM_02:C: 0x40408000-
```

```
0x40507fff: .os.literal .os.text .os.minor.literal .os.minor.text .os.init.literal .os.init.text ;
```

```
END SRAM_02
```



```
//2.99M

BEGIN SRAM_03

0x40508000: sysram : SRAM_03: 0x2fe000 : executable, writable ;

sram0 : C : 0x40508000 - 0x40805fff : HEAP : .sram.rodatta .rodatta ;

END SRAM_03


//256KB

BEGIN dram0

0x15100000: dataRam : dram0: 0x40000 : writable ;

dram0_0 : C : 0x15100000 - 0x1512ffff : .dram0.rodatta .dram0.literal .dram0.data .dram0.bss;

dram0_1 : C : 0x15130000 - 0x15130fff : STACK ;;

dram0_2 : C : 0x15134000 -
0x1513ffff : .dram0_2.rodatta .dram0_2.literal .dram0_2.data .dram0_2.bss;

dram0_3 : C : 0x15131000 -
0x15133fff : .dram0_3.rodatta .dram0_3.literal .dram0_3.data .dram0_3.bss;

END dram0


//192KB

BEGIN dram1

0x15200000: dataRam : dram1: 0x30000 : writable ;

dram1_0 : C : 0x15200000 - 0x1522ffff : .dram1.rodatta .dram1.literal .dram1.data .dram1.bss;

END dram1


//fixed vector

BEGIN fixvector

0x40000000: instRam : fixvector: 0x300 : executable, writable ;

RESET : F: 0x40000000-0x400002ff : .ResetVector.text;

END fixvector


BEGIN iram0

0x40000400: instRam : iram0: 0x7c00 : executable, writable ;

WINDOW : F : 0x40000400 - 0x40000577 : .WindowVectors.text;

LEVEL2_LIT : C : 0x40000578 - 0x4000057f : .Level2InterruptVector.literal;

LEVEL2 : F : 0x40000580 - 0x400005b7 : .Level2InterruptVector.text;
```



```
DEBUG_LIT : C : 0x400005b8 - 0x400005bf : .DebugExceptionVector.literal;
DEBUG      : F : 0x400005c0 - 0x400005f7 : .DebugExceptionVector.text;
NMI_LIT    : C : 0x400005f8 - 0x400005ff : .NMIExceptionVector.literal;
NMI        : F : 0x40000600 - 0x40000637 : .NMIExceptionVector.text;
KERNEL_LIT : C : 0x40000638 - 0x4000063f : .KernelExceptionVector.literal;
KERNEL     : F : 0x40000640 - 0x40000677 : .KernelExceptionVector.text;
USER_LIT   : C : 0x40000678 - 0x4000067f : .UserExceptionVector.literal;
USER       : F : 0x40000680 - 0x400006b7 : .UserExceptionVector.text;
DOUBLE_LIT : C : 0x400006b8 - 0x400006ff : .DoubleExceptionVector.literal;
DOUBLE     : F : 0x40000700 - 0x4000073f : .DoubleExceptionVector.text;
IRAM0_LIT  : C : 0x40000740 - 0x400007ff : .iram0.literal;
IRAM0      : F : 0x40000800 - 0x400007ff : .iram0.text;
END iram0

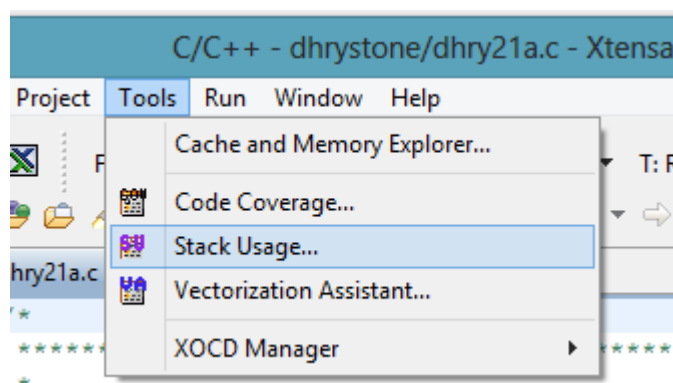
BEGIN iobypass
0x12100000: io : iobypass: 0x4000 : device, executable, writable ;
END iobypass
```

2.4 查看 stack 使用

具体操作步骤如下：

步骤 1. 点击 Xplorer 的 Tools->Statck Usage 选项。

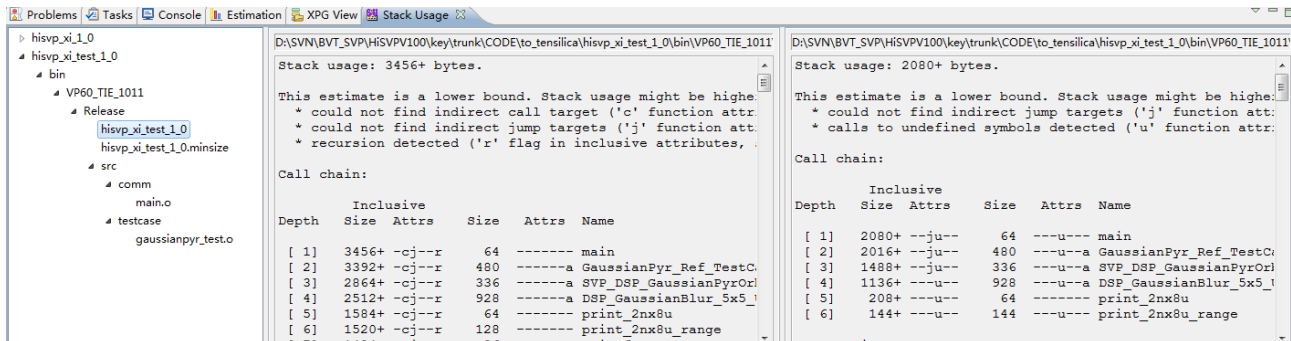
图2-5 查看栈信息操作界面



步骤 2. 此时会出现 Stack Usage 的窗口。选择一个工程，右键点击 Expand All，选择可执行文件右键点击“Show in Left Panel”或“Show in Right Panel”查看该文件的 Stack 使用情况。



图2-6 显示栈信息窗口



----结束

2.5 Linux 环境下安装 DSP 工具链和配置核

在 Linux 环境下安装 Cadence DSP 的工具链和配置核可参考 Cadence 提供的官方文档 [dev_tools_install_guide.pdf](#)。用户在利用海思提供的 SDK 包进行板端 DSP 程序开发时，推荐按照以下方式安装 DSP 的工具链和配置核。

2.5.1 安装 DSP 工具链

下面以 Xplorer-7.0.4-linux-installer.bin 版本介绍工具链的安装。步骤如下：

- 步骤 1. 首先需要以 root 权限在服务器上创建目录/opt/xtensa。
- 步骤 2. 把 Xplorer-7.0.4-linux-installer.bin 安装包拷贝到/opt/xtensa 目录下，然后执行安装./Xplorer-7.0.4-linux-installer.bin，如下面所示：

```
root@SZX1000040224:/opt/xtensa# ./Xplorer-7.0.4-linux-installer.bin
```

```
-----  
Welcome to the Xtensa Xplorer Setup Wizard.
```

```
-----  
Please read the following License Agreement. You must accept the terms of  
this  
agreement before continuing with the installation.
```

```
Press [Enter] to continue :
```

```
Cadence Tools Software Use Agreement
```

```
Your use of the software you are about to install is subject to one or  
more  
licensing agreements.
```

```
Portions of this software are subject to the terms of either (1) a
```



technology
license agreement between Cadence and the user ("you") - a direct Cadence licensee, (2) an end user license agreement between an existing Cadence licensee and you, or (3) a limited use evaluation agreement between Cadence and you. In addition, each directory contains files that identify any open source licenses or copyrights that apply to each component.

For a copy of the license agreement that applies to your use of this software, please make an inquiry to the organization that provided this copy of the software to you.

Press [Enter] to continue :

Do you accept this license? [y/n]:y

Xtensa Xplorer Installation Directory

Please enter the path to the Xtensa Xplorer root directory. The Xtensa Xplorer 7.0.4 and XtDevTools directories will be installed in this directory.

It is recommended that you use the same Xplorer root directory as your previous installations of Xplorer so that the XtDevTools directory can be shared, which will allow this version of Xplorer to use all previous installations of Xtensa configurations and tools.

Xtensa Xplorer Root Directory [/opt/xtensa]:

Warning: The installer did not detect an XtDevTools directory in the location that you specified. If you have previously installed any version of Xtensa Xplorer, please install this version in the same root directory so this version can share the same XtDevTools tree.

Do you have an existing installation of Xtensa Xplorer?

Click "Yes" to specify the previous Xplorer root directory location(usually /usr/xtensa), otherwise click "No" if this is your



first installation of Xplorer.

[Y/n]: n

Select the components you want to install; clear the components you do
not want
to install. Click Next when you are ready to continue.

Xplorer and Xtensa Development Tools : Y (Cannot be edited)

General Sample Configs [Y/n] :y

Xtensa Fusion F1 DSP [Y/n] :y

Xtensa HiFi Family Samples [Y/n] :y

Xtensa TIE base examples builds [Y/n] :y

Xtensa ConnX D2 DSP Engine Reference Cores [Y/n] :y

Is the selection above correct? [Y/n]: y

Xplorer components selected

Selected the following components:

- + Xtensa Xplorer and Development Tools
- + Xtensa Fusion F1 DSP core
- + Xtensa HiFi Family Samples
- + Xtensa ConnX D2 DSP Engine Reference Cores
- + Xtensa TIE base example
- + Xtensa General Samples

Press [Enter] to continue :

Disk Space Report



Installation space report

Required disk space is : 3640 MB

Current disk has free space : 7570 MB

Press [Enter] to continue :

Installation Summary

Xtensa products will be installed as follows

Xtensa Xplorer will be installed to:

/opt/xtensa/Xplorer-7.0.4

Xtensa Tools will be installed to:

/opt/xtensa/XtDevTools/install/tools/RG-2016.4

Xtensa Tools and samples bundles will be stored at:

/opt/xtensa/XtDevTools/downloads/RG-2016.4

Xtensa Xplorer workspace default location:

/opt/xtensa/Xplorer-7.0.4-workspaces

Press [Enter] to continue :

Pre-installation Message

The Xplorer installer runs in 2 phases. The last phase (post-installation) is installing tools and any configurations selected, and may run for several minutes without appearing to make progress. Please be patient.

Press [Enter] to continue :

Setup is now ready to begin installing Xtensa Xplorer on your computer.

Do you want to continue? [Y/n]: y

Please wait while Setup installs Xtensa Xplorer on your computer.



```
Installing
0% _____ 50% _____ 100%
#####

Post Installation Script Result
Congratulations !! You have finished installing Xplorer-7.0.4
Please review the following message log to make sure of the success of
the
installation.

=====
INSTALLING XtensaTools
===== LOCATE utils plugin =====
WHERE_UTILS_RESULT=/opt/xtensa/Xplorer-
7.0.4/eclipse/plugins/other.xide.external.
utils_7.0.4.2000
INSTALL XTTOOLS COMMAND :: /opt/xtensa/Xplorer-7.0.4/eclipse/jre/bin/java
-cp
/opt/xtensa/Xplorer-
7.0.4/eclipse/plugins/other.xide.external.utils_7.0.4.2000/ut
ils.jar other.xide.external.utils.io.Unpack
/opt/xtensa/XtDevTools/downloads/RG-
2016.4/tools/XtensaTools_RG_2016_4_linux.tgz
/opt/xtensa/XtDevTools/install/tools/
/opt/xtensa/Xplorer-
7.0.4/eclipse/plugins/other.xide.external.utils_7.0.4.2000
INSTALL XtensaTools RESULT :: 0 , SUCCESS
===== LOCATE dynamic plugin files =====
INSTALL XOS Document Plugin
WHERE_XOS_RESULT=/opt/xtensa/XtDevTools/install/tools/RG-2016.4-
linux/XtensaTools
/doc/xos-1.12.zip
INSTALL XOS COMMAND :: /opt/xtensa/Xplorer-7.0.4/eclipse/jre/bin/java -cp
/opt/xtensa/Xplorer-
7.0.4/eclipse/plugins/other.xide.external.utils_7.0.4.2000/ut
Press [Enter] to continue :
ils.jar other.xide.external.utils.io.Unpack
/opt/xtensa/XtDevTools/install/tools/RG-2016.4-linux/XtensaTools/doc/xos-
1.12.zip
/opt/xtensa/Xplorer-7.0.4/eclipse/plugins/
/opt/xtensa/Xplorer-
7.0.4/eclipse/plugins/other.xide.external.utils_7.0.4.2000
INSTALL XOS HELP PLUGIN RESULT :: 0 , SUCCESS
INSTALL XIPC Document Plugin
```



```
WHERE_XIPC_RESULT=/opt/xtensa/XtDevTools/install/tools/RG-2016.4-  
linux/XtensaTool  
s/doc/xipc.zip  
INSTALL XIPC COMMAND :: /opt/xtensa/Xplorer-7.0.4/eclipse/jre/bin/java -  
cp  
/opt/xtensa/Xplorer-  
7.0.4/eclipse/plugins/other.xide.external.utils_7.0.4.2000/ut  
ils.jar other.xide.external.utils.io.Unpack  
/opt/xtensa/XtDevTools/install/tools/RG-2016.4-  
linux/XtensaTools/doc/xipc.zip  
/opt/xtensa/Xplorer-7.0.4/eclipse/plugins/  
/opt/xtensa/Xplorer-  
7.0.4/eclipse/plugins/other.xide.external.utils_7.0.4.2000  
INSTALL XIPC HELP PLUGIN RESULT :: 0 , SUCCESS  
  
Checking XtensaRegistry dir...  
Check XtensaRegistry RG-2016.4 DIR  
DONE:: /opt/xtensa/XtDevTools/XtensaRegistry/RG-2016.4-linux  
  
INSTALLING User selected Xtensa config builds from  
XtDevTools/downloads/RG-2016.4/builds IF EXISTS  
=====  
Press [Enter] to continue :  
UNPACK CONFIG RESULT :: 0, SUCCESS  
INSTALL CONFIG XRC_D2PM RESULT :: 0, SUCCESS  
  
=====  
UNPACK CONFIG RESULT :: 0, SUCCESS  
INSTALL CONFIG XRC_FUSION_AON_ALL_LM RESULT :: 0, SUCCESS  
  
=====  
UNPACK CONFIG RESULT :: 0, SUCCESS  
INSTALL CONFIG hifi2_std RESULT :: 0, SUCCESS  
  
=====  
UNPACK CONFIG RESULT :: 0, SUCCESS  
INSTALL CONFIG hifi3_bd5 RESULT :: 0, SUCCESS  
  
=====  
UNPACK CONFIG RESULT :: 0, SUCCESS  
INSTALL CONFIG hifiep_bd5 RESULT :: 0, SUCCESS  
  
=====  
UNPACK CONFIG RESULT :: 0, SUCCESS
```



```

INSTALL CONFIG hifi_mini RESULT :: 0, SUCCESS

Press [Enter] to continue :
=====
UNPACK CONFIG RESULT :: 0, SUCCESS
INSTALL CONFIG sample_config RESULT :: 0, SUCCESS

=====
UNPACK CONFIG RESULT :: 0, SUCCESS
INSTALL CONFIG sample_controller RESULT :: 0, SUCCESS

=====
UNPACK CONFIG RESULT :: 0, SUCCESS
INSTALL CONFIG sample_flix RESULT :: 0, SUCCESS

=====
UNPACK CONFIG RESULT :: 0, SUCCESS
INSTALL CONFIG tie_dev1 RESULT :: 0, SUCCESS

=====
UNPACK CONFIG RESULT :: 0, SUCCESS
INSTALL CONFIG tie_dev2 RESULT :: 0, SUCCESS

Setting Xplorer configuration and then initializing configuration cache
INITIALIZE Xtensa Xplorer RESULT :: 0, SUCCESS
Press [Enter] to continue :
-----
Setup has finished installing Xtensa Xplorer on your computer.

Run Xtensa Xplorer now (Recommended)
(This initializes workspace location defaults) [Y/n]:n

```

----结束

2.5.2 安装 DSP 配置核

下面以 VP60_1124 核为参考说明配置核的安装，其他配置核安装过程类似。

- 步骤 1. 在 root 权限下，把 VP60_1124_linux.tgz 压缩包拷贝到之前创建的/opt/xtensa 目录，然后解压压缩包，命令 tar -xzf VP60_1124_linux.tgz，解压出/opt/xtensa/RG-2016.4-linux/VP60_1124/目录。
- 步骤 2. 进入/opt/xtensa/RG-2016.4-linux/VP60_1124/目录安装配置核，执行./install。

如下所示：

```
root@SZX1000040224:/opt/xtensa/RG-2016.4-linux/VP60_1124# ./install
```



```
Xtensa Processor Configuration Installation Tool
Copyright (c) 2001-2016 Tensilica Inc.
For Xtensa Tools Version 12.0.4
```

Before you can use a new Xtensa processor configuration, you must run this tool to complete the installation. Two separate packages are required:

- 1) The Xtensa Tools cross-development software toolkit package. These tools are configuration-independent and are shared by all your Xtensa processor configurations. You do not need a separate copy for each configuration.
- 2) The configured Xtensa processor files, of which this script is a part.

You must have already downloaded both packages and extracted the files on your system before you can continue.

```
Are you ready to proceed? [y] y
Continuing...
```

```
Enter the path to the Xtensa Tools directory:
/opt/xtensa/XtDevTools/install/tools/RG-2016.4-linux/XtensaTools
```

The files for this Xtensa processor configuration will now be set up to work with the installation directories that you have chosen. This process will take a few minutes, and once it has begun the installation directories cannot be changed. If you abort this script after this point, or if you need to change the installation directories for some reason, you will need to start over with the original files that you downloaded for this configuration. (The Xtensa Tools files are not modified in this process so you do not need to reinstall the Xtensa Tools package.) The directories to be used are:

```
Xtensa Tools:      /opt/xtensa/XtDevTools/install/tools/RG-2016.4-
linux/XtensaTools
Configured Processor: /opt/xtensa/RG-2016.4-linux/VP60_1124/.
```

```
Do you want to continue? [y] y
```

The files for this processor configuration have now been set to use the directory names you have chosen.



The next installation step is to add this processor configuration to the list of available configurations in a registry of Xtensa cores. The configuration will be registered with the default name, which is the Core ID from the Xtensa Processor Generator. You must ensure that each core in the registry has a unique name. Please refer to the "Xtensa Software Development Toolkit User's Guide" to learn how to register this configuration with a different name.

This script will update only one registry of Xtensa cores, and in most cases, you should use the default Xtensa registry. If you are sharing the Xtensa Tools installation with others, and you do not want this processor configuration to be shared, you can specify an alternate registry. Please refer to the "Xtensa Software Development Toolkit User's Guide" for instructions on adding this configuration to additional Xtensa core registries.

The default registry is:

```
/opt/xtensa/XtDevTools/install/tools/RG-2016.4-linux/XtensaTools/config
```

What registry would you like to use? [default]

```
/opt/xtensa/XtDevTools/XtensaRegistry/RG-2016.4-linux
```

Do you want to make "VP60_1124" the default Xtensa core? [y] y

The installation process is now complete.



注意

使用海思 SDK 开发时配置核路径最好要安照上面的安装路径指定, 否则在开发 DSP 程序时, 需要针对实际安装路径去修改 makefile, 指定工具链和配置核路径才能编译 (可参考 Cadence 提供的《xtensa_xcc_compiler_ug.pdf》文档)

----结束

2.5.3 配置环境变量

在/etc 目录下打开 profile 文件, 添加以下环境变量配置:

- 设置 Cadence license:

```
export XTENSAD_LICENSE_FILE=port@serverip
```

例如: port 是 28001, serverip 是 192.168.1.100

- 设置配置核参数路径:

```
export XTENSA_SYSTEM=/opt/xtensa/XtDevTools/XtensaRegistry/RG-2016.4-  
linux
```

- 设置默认的配置核：
export XTENSA_CORE=VP60_1124
- 设置交叉编译工具链路径：
export PATH="/opt/xtensa/XtDevTools/install/tools/RG-2016.4-
linux/XtensaTools/bin:\$PATH"

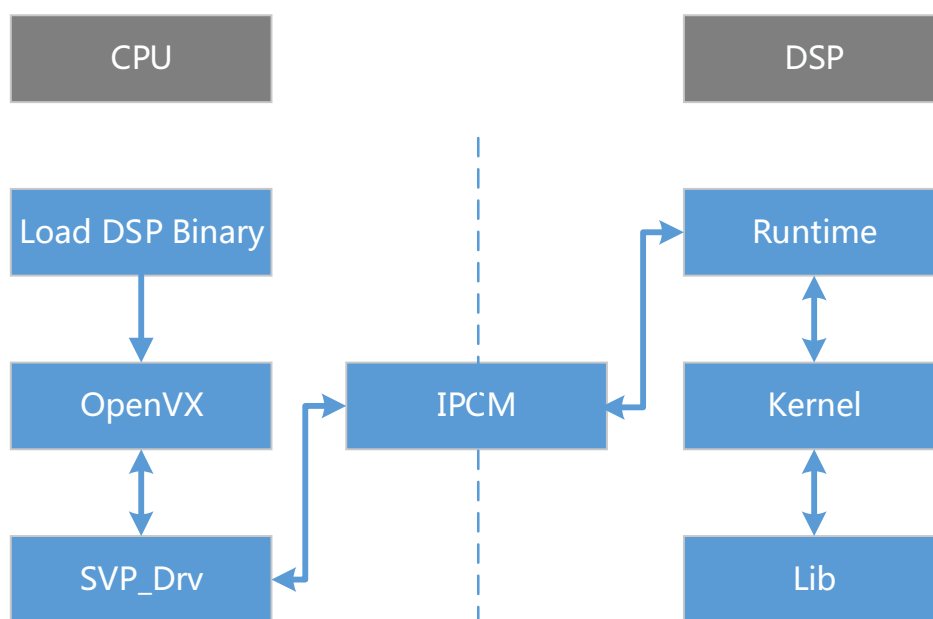
2.6 开发流程

DSP 开发分为 PC 端和板端环境：

- PC 端环境：利用 Xplorer 集成工具进行 DSP 算法开发，利用 Xplorer 集成工具提供的仿真环境可以快速实现对算法的开发和验证，开发完算法代码可以直接移植到板端编译调试。Xplorer 环境参考前面的介绍。
- 板端环境：基于海思提供的 SDK 开发包进行板端程序开发。

DSP 应用程序开发框图如图 2-7 所示。

图2-7 DSP 应用程序开发框图



如图 2-7 所示，DSP 程序的开发和使用分为几部分：

- 在 CPU 端
 - Load DSP Binary：加载 DSP 可执行程序，由相应的应用层接口完成。
 - OpenVX：基于 OpenVX 框架进行算法开发。



- SVP_Drv: 驱动模块。
- IPCM: 核间通信模块。
- 在 DSP 端
 - Runtime: DSP 上算子的调度管理。
 - Kernel: 解析参数, 调用 DSP 算子。
 - Lib:DSP: 算子库。
 - IPCM: 核间通信模块。

2.6.1 开发模式一

只使用海思提供的 DSP 算子

- 在 CPU 端

步骤 1. 在程序初始化时调用 vxSvpLoadBin 把 4 个 bin (hi_iram.bin、hi_sram.bin、hi_dram0.bin、hi_dram1.bin) 分别加载至对应位置。

步骤 2. 调用 vxSvpCoreEnable 使能 DSP。

步骤 3. 调用海思提供的 OpenVX 接口结合自己算法开发即可。



说明

OpenVX HiSVP API .pdf

----结束

- 在 DSP 端

无需任何操作, 只使用海思提供的 4 个 bin 即可。



说明

4 bin hi_iram.bin hi_sram.bin hi_dram0.bin hi_dram1.bin runtime/bin.

2.6.2 开发模式二

用户开发私有 DSP 算子:

- 在 CPU 端

步骤 1. 在程序初始化时调用 vxSvpLoadBin 把 4 个 bin (hi_iram.bin、hi_sram.bin、hi_dram0.bin、hi_dram1.bin) 分别加载至对应位置。

步骤 2. 调用 vxSvpCoreEnable 使能 DSP。

步骤 3. 开发基于 OpenVX 框架算子。



说明

HiSVP API .pdf vxLoadKernels vxAddUserKernel

----结束

- 在 DSP 端



步骤 1. 在发布包 kernel/include/int 目录下开发基于帧级或者 Tile 级的算子。



说明

kernel/include/int/svp_dsp_frm.h kernel/src/svp_dsp_frm.h

步骤 2. 在发布包 kernel/src/svp_dsp_kernel.c 源文件 SVP_DSP_ProcessKernel 函数添加对私有算子的调用。

步骤 3. 执行 kernel/Makefile 重新对 kernel 目录下的代码编译成库。

步骤 4. 执行 runtime/obj/Makefile 把.o、.a 等编译成 elf 文件并转换为 4 个 binary。



说明

4 bin hi_iram.bin hi_sram.bin hi_dram0.bin hi_dram1.bin runtime/obj/bin.

----结束



说明

DSP

IRAM/SRAM/DRAM



注意

在 DSP 端开发时，对地址有以下约束：

- 从 ARM 端传递给 DSP 端处理的地址必须是通过 SDK 的 MMZ 申请内存地址。
- 由于 ARM 系统的地址是 64 位，DSP 系统的地址是 32 位，因此 ARM 端传递的地址给 DSP 端时，DSP 需要先做 64 位到 32 位的转换，可以参考发布包 kernel/src/svp_dsp_kernel_comm.c 的函数 SVP_DSP_ConvertPhyAddr。
- DSP 要访问 ARM 端传递过来的地址的内容时，需要通过 IDMA 拷贝到 DSP 的地址空间才能访问。IDMA 拷贝函数可以参考发布包 kernel/src/svp_dsp_frm.c 函数 SVP_DSP_CopyData。



3 NNIE 开发指南

3.1 NNIE 介绍

NNIE 是 Neural Network Inference Engine 的简称，是海思媒体 SoC 中专门针对神经网络特别是深度学习卷积神经网络进行加速处理的硬件单元，支持现有大部分的公开网络，如 Alexnet、VGG16、Googlenet、Resnet18、Resnet50 等等，以及 Faster R-CNN、SSD、RFCN 等类型的网络。

目前 NNIE 配套软件及工具链仅支持以 Caffe 框架的网络描述文件 prototxt 和训练得到的模型来进行开发。

3.1.1 NNIE 支持规格

图 3-1 列出了 NNIE 以 Caffe 层描述的硬件支持的层间链接约束。

图3-1 NNIE 层间约束

Next Layer Layer Type	conv	pooling	Inner Product	LRN	BN	scale	eltwise	tanh	sigmoid	relu	concat	as input	as output
conv	●	●	●	●	●	●	●	●	●	●	●	●	●
Avg Pooling	●	●	●	●	×	×	●	×	×	×	●	×	●
Max Pooling	●	●	●	●	×	×	●	×	×	×	●	×	×
Inner Product	×	×	●	×	●	●	×	●	●	●	×	●	●
LRN	●	●	●	●	×	×	●	×	×	×	●	×	×
BN	●	●	●	●	×	●	●	●	●	●	●	×	○
scale	●	●	●	●	●	×	●	●	●	●	●	×	○
eltwise	●	●	●	●	×	×	●	×	×	●	●	×	×
tanh	●	●	●	●	×	×	●	×	×	×	●	×	○
sigmoid	●	●	●	●	×	×	●	×	×	×	●	×	○
relu	●	●	●	●	×	×	●	×	×	×	●	×	○
concat	●	●	×	×	×	×	×	×	×	×	×	×	×

●:Support; ×:Not Support; ○: Only behind conv、Inner Product layer can be supported;

具体规格如下：

- 支持卷积核大小(方形 N*N): 1*1, 2*2, 3*3, 4*4, 5*5, 6*6, 7*7, 8*8, 9*9, 10*10, 11*11;



- 支持 MaxPooling 和 AveragePooling, 支持矩形 N*M Pooling, 其中 $1 \leq N \leq 11$, $1 \leq M \leq 11$;
- 卷积和 Pooling 支持 Stride 规格:
Kernel_size=1: Stride=1、2;
Kernel_size=2: Stride=1、2;
Kernel_size=3~11: Stride<Kernel_size
- 卷积和 Pooling 支持 Pad, Pad_size < Kernel_size, 支持上下 Pad_h、左右 Pad_w 单独配置, Pad 填充数据为 0;
- 支持激活函数类型: Relu、Sigmoid 和 Tanh;
- 支持 LRN 运算, local_size(通道间归一化时表示求和的通道数)为 5, 要求输出通道数大于等于 5;
- 支持 Inner Product 层的向量维数: 64~100000;
- 支持 Concat 输入节点个数最大为 4;
- 支持 Eltwise 加法运算;
- 支持数据和参数为 8bit 模式;
- 支持参数压缩和参数稀疏;
- 支持输入图像 Size 为: 16x16~1920x1080;
- 支持图像预处理: 除以 256、减通道均值、减图像均值, 减通道均值再除以 256, 减图像均值后在除以 256;
- 支持结果上报的层为 Conv、AvePooling 和 Inner Product;
- 参数规模限制: 卷积层一个 kernel 的规格为 input_channel * kernel_height * kernel_width 需小于等于 4608Byte (其中, input_channel 为输入 feature map 维度, kernel_width、kernel_height 为卷积核宽、高维度);
- 数据规模限制: 由于数据需通过内部 buffer 进行传输, 而数据 buffer 总大小为 260KB, 且被分为 15 个内部 ram 被使用, 因此, 载入数据需符合规则: feature map 将会被分多次载入, 假设一次能载入 feature map 的行数记为 row_cnt, 当 row_cnt < kernel_height 时不支持。

row_cnt 的计算方式如下所示:

由于计算阵列结构, 可同时计算的 feature map 数为:

parall_map_size = **Floor**(RAM_CNT / kernel_height)

每个 feature_map 可用的 buffer 空间为:

useable_ram_cnt = **Floor**(RAM_CNT / parall_map_size) * parall_map_size

useable_buffer_size = (totle_buffer_size / RAM_CNT) * useable_ram_cnt

可以存的行数为:

row_cnt = useable_buffer_size / (input_channel * **Align32**(input_width))

其中 RAM_CNT = 15, totle_buffer_size = 260KB, kernel_height 卷积核高, input_width 为 feature map 宽, input_channel 为输入 feature map 维度, **Floor** 为向下取整操作, **Align32** 为按 32 对齐操作。

3.1.2 NNIE 硬件资源利用率

在现有硬件规格下，用户可以参考以下准则提升硬件利用率：

- 总体原则：
 - 一般情况下，相同计算量场景，尽量增加单层的计算量，减少网络层数，以减少层之间切换造成的利用率损失，提高端到端的利用率。
 - 尽量使用卷积、FC 等乘加计算量大的层，减少 LRN 层的使用。
- 特别地针对卷积、Pooling 运算：
 - 基于硬件计算单元特点，kernel 在 1*1、3*3、5*5、7*7 时，计算资源的利用率更高。
 - 输出特征数据尺寸在 14*14 时，读参数的 DDR 带宽与卷积计算基本匹配；特征数据尺寸更小时，计算单元受限于 DDR 带宽会无法全部利用。
 - 硬件计算时，数据按 14 组并行计算，输出特征数据尺寸尽量等于或接近 14 的倍数，利用率更高。

3.2 工具链介绍

SVP NNIE 在 HiSVP_PC_Vx.x.x.x.rar 组件包中，提供如下的工具链：

- 网络过滤器：用户设计好网络后，在训练之前使用该工具对网络描述文件进行有效性检查。目前仅支持过滤 Caffe 的 prototxt 文件。
- 编译器：将用户开通过开源深度学习框架训练得到的模型编译成在 Hi35xx 芯片上或者在仿真库、仿真器中可以加载的数据指令文件。目前仅支持 Caffe 框架。
- 仿真库：模拟 NNIE 的硬件执行和软件接口调用，在相同的“输入”下仿真库与硬件得到相同的结果。仿真库可以使用户脱离硬件在 PC 环境下仿真，且便于调试，有助于用户提前快速开发算法原型。
- 仿真器：基于仿真库实现的模拟 NNIE 硬件的仿真执行器。用户无需编程仅通过简单地输入编译后的模型、图像数据、label 文件即可获得该模型的准确率数据，简单性能评估以及带宽分析。

网络过滤器、编译器、仿真器集成在一个 UI 环境 HiSVP_CNNIE8bit 中，该 UI 还包含了其他的一些功能，如根据用户输入的网络描述文件生成可视化的网络拓扑结构图、目标检测结果可视化等功能。



相同的“输入”是指相同的原始模型，相同的图像数据输入。由编译器转化得到的仿真库上的模型和在芯片上的稍有不同。

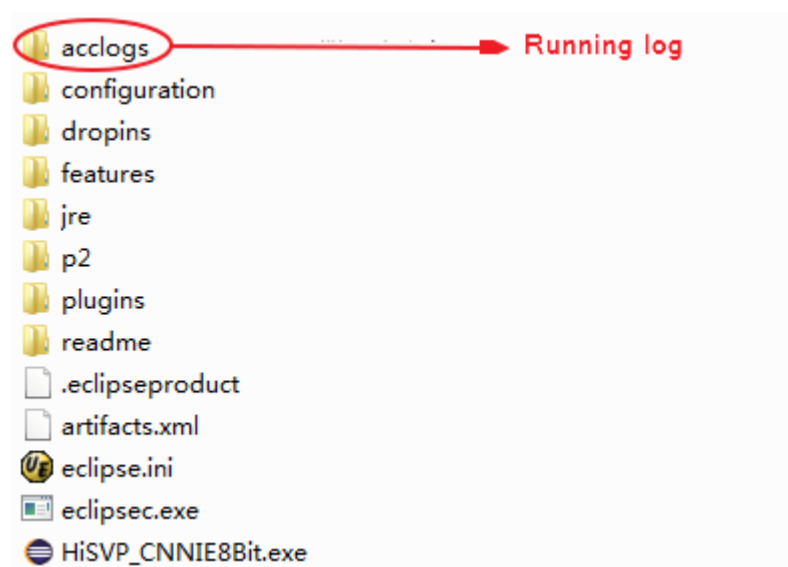
3.3 工具链安装

NNIE 工具链运行依赖于 Protobuf2.5.0 和 Opencv3.0.0 第三方开源软件的 lib 文件。其中，Protobuf2.5.0 由于是以源码形式获取，需在 Windows7 64bit 系统中采用 Visual Studio 2010 完成编译（本文默认用户已经完成 Windows 7 和 Visual Studio2010 的安装），具体编译过程参考 3.3.2 “Protobuf 安装” 部分。而 Opencv3.0.0 可以参考本文提供的下载链接直接获取 lib 文件。

3.3.1 NNIE 工具链本体安装

解压 HiSVP_Tool_CNNIE8Bit_Win64_Vx.x.x.rar 后，得到如图 3-2 所示目录。

图3-2 HiSVP_CNNIE8Bit 集成 UI 目录



3.3.2 Protobuf 安装

- 步骤 1. 请至 <https://github.com/google/protobuf/releases/tag/v2.5.0>，点击 protobuf2.5.0.zip 下载对应的 windows 版本 protobuf，并解压得到 protobuf 文件夹。
- 步骤 2. 进入解压后目录下的 vsprojects 目录，使用 Windows 命令行依次执行以下命令：

```
devenv protobuf.sln /Upgrade
devenv protobuf.sln /Project libprotobuf /Rebuild "Release|x64"
devenv protobuf.sln /Project libprotobuf-lite /Build "Release|x64"
devenv protobuf.sln /Project libprotoc /Build "Release|x64"
devenv protobuf.sln /Project lite-test /Build "Release|x64"
devenv protobuf.sln /Project protoc /Build "Release|x64"
```
- 步骤 3. 进入到 vsprojects\Release 目录下，打开 Windows 的命令行，执行 tests.exe 或 lite-test.exe，如果 “passed”，则表明编译成功；

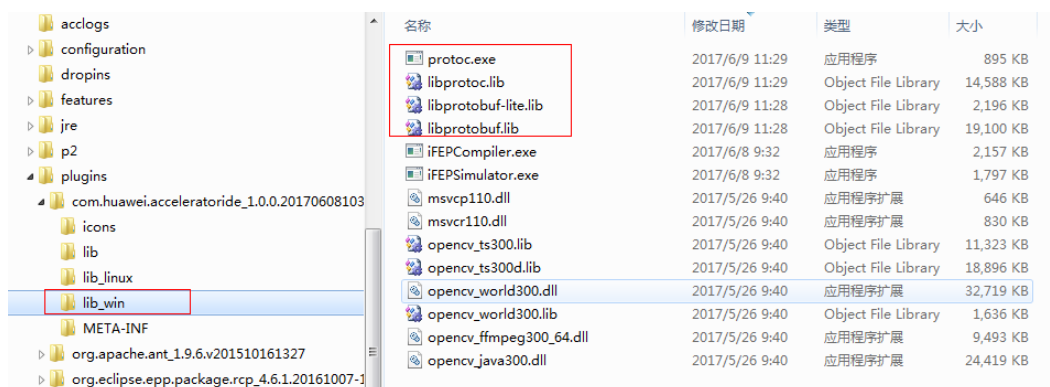


图3-3 执行 tests.exe 结果

```
[-----] Global test environment tear-down  
[=====] 868 tests from 112 test cases ran. <5682 ms total>  
[ PASSED ] 868 tests.
```

步骤 4. 将 vsprojects\Release 目录下的 protoc.exe, libprotobuf.lib, libprotobuf-lite.lib, and libprotoc.lib 拷贝到 HiSVP_CNNIE8Bit 集成 UI 目录的 lib_win 文件夹中, 如图 3-4 所示。

图3-4 第三方开源库放置目录



----结束

3.3.3 OpenCV 安装

步骤 1. 请至 <https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.0.0/opencv-3.0.0.exe/download> 下载 opencv, 并解压到 opencv 文件夹。

步骤 2. 将 build\x64\vc11\lib 目录下的 opencv_ts300.lib, opencv_world300.lib 和 build\x64\vc11\bin 目录下 opencv_world300.dll, opencv_ffmpeg300_64.dll, opencv_java300.dll 拷贝到 HiSVP_CNNIE8Bit 集成 UI 目录的 lib_win 文件夹中, 如图 3-4 所示。

----结束



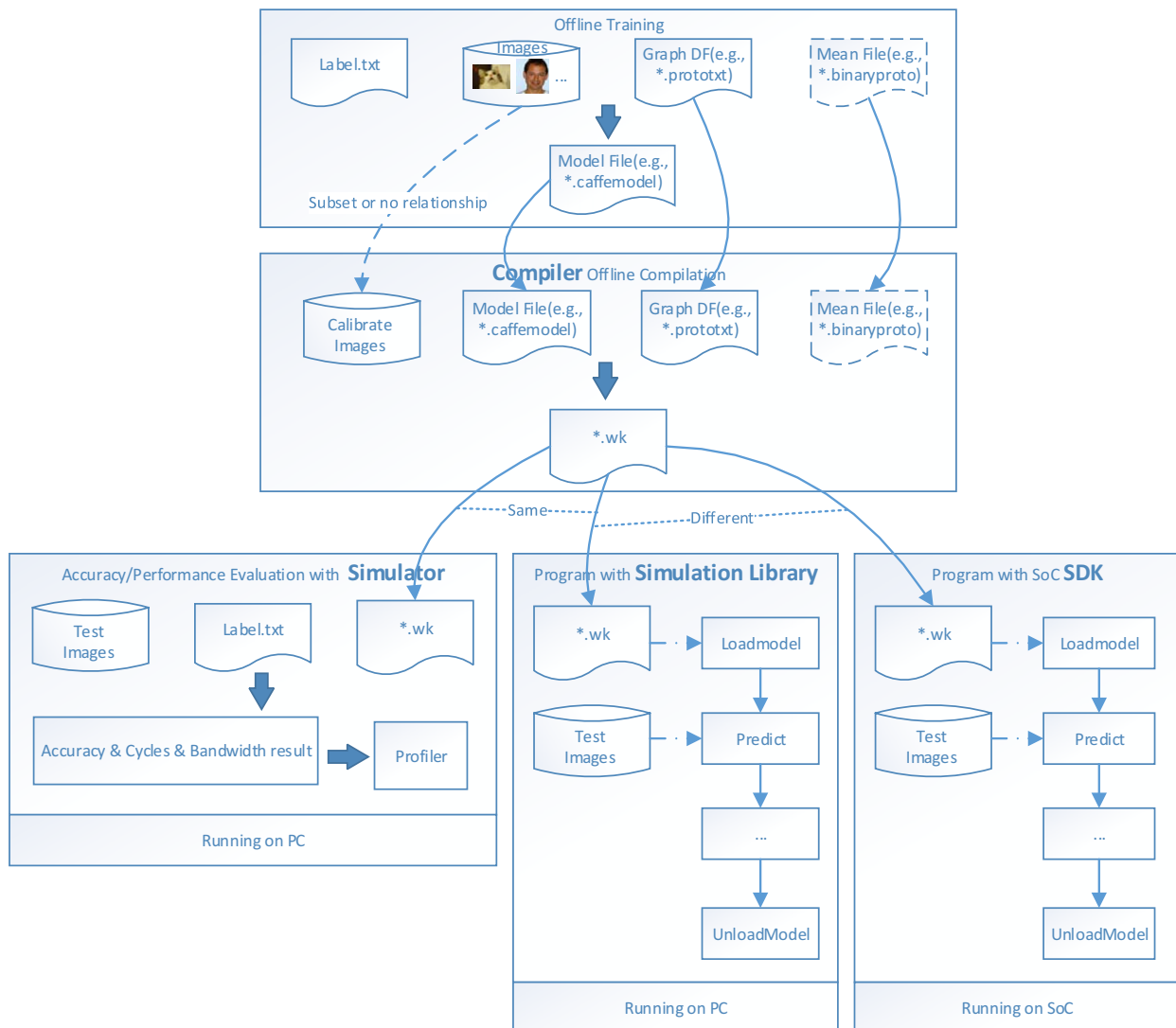
注意

一般情况下, 应该需要使用 OpenCV 在 Visual Studio 2010 win32 环境下编译的 lib、dll, 但是经过验证, 上述的 lib、dll 可以使用。

3.4 开发流程

以 Caffe 框架上训练的模型为例，NNIE 的开发流程如图 3-5 所示。在 Caffe 上训练、使用 NNIE 的编译工具编译都是离线的。通过设置不同的模式，编译器将*.caffemodel 编译成在仿真器、仿真库或板端上可加载执行的数据指令文件。一般在开发前期，用户可使用仿真器对训练出来的模型进行精度、性能、带宽进行初步评估，符合用户预期后再使用仿真库进行完整功能的仿真，最后将程序移植到板端。

图3-5 NNIE 软件开发流程





3.5 Prototxt 要求及模型下载

3.5.1 Prototxt 要求

3.5.1.1 deploy.prototxt 开头格式

deploy.prototxt 开头支持如下两种格式:

格式一:

```
input:"data"
input_shape{
  dim:1
  dim:3
  dim:224
  dim:224
}
```

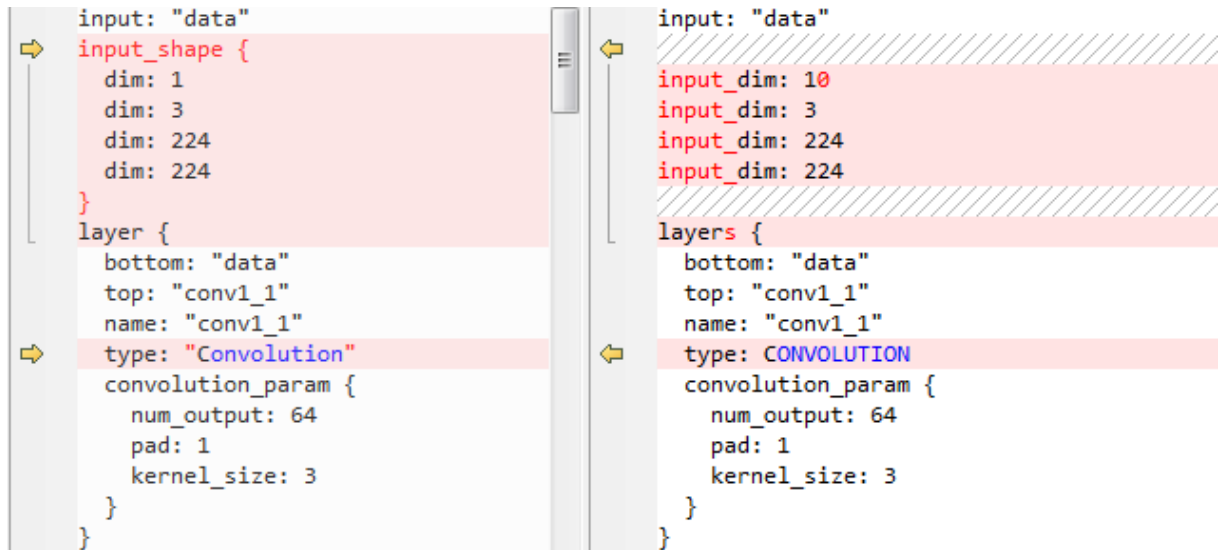
格式二:

```
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param {
    shape: {
      dim: 10
      dim: 3
      dim: 227
      dim: 227
    }
  }
}
```

3.5.1.2 prototxt layer 描述格式

prototxt 格式上只支持图 3-6 左边的格式;层以 layer 开头, type 以带双引号字符串表示。

图3-6 HiSVP_CNNIE8Bit.exe 支持 prototxt 格式说明示意图（左边支持）



3.5.1.3 prototxt 特殊 layer 描述

- 1) batchnorm, scale, relu, sigmoid, tanh 要求 bottom 与 top 名称相同:

```

layer {
  bottom: "res5c_branch2a"
  top: "res5c_branch2b"
  name: "res5c_branch2b"
  type: "Convolution"
  convolution_param {
    num_output: 512
    kernel_size: 3
    pad: 1
    stride: 1
    bias_term: false
  }
}

layer {
  bottom: "res5c_branch2b"
  top: "res5c_branch2b"
  name: "bn5c_branch2b"
  type: "BatchNorm"
  batch_norm_param {
    use_global_stats: true
  }
}
  
```




```
layer {  
    bottom: "res5c_branch2b"  
    top: "res5c_branch2b"  
    name: "scale5c_branch2b"  
    type: "Scale"  
    scale_param {  
        bias_term: true  
    }  
}
```

- 2) batchnorm 层必须有 batch_norm_param 参数，并且 use_global_stats 必须为 true，格式如下所示：

```
layer {  
    bottom: "res5c_branch2b"  
    top: "res5c_branch2b"  
    name: "bn5c_branch2b"  
    type: "BatchNorm"  
    batch_norm_param {  
        use_global_stats: true  
    }  
}
```

3.5.1.4 分段执行网络 prototxt 特殊格式

针对 FasterRCNN、iFDT、RFCN、SSD 类型的网络，NNIE 采用软硬件协同的方法来完成这类网络的运算。

根据 NNIE 支持规格，Faster RCNN 类型网络分四阶段执行：

- 第 1 阶段，RPN 非卷积部分之前的网络，由 NNIE 硬件运算（包含 RPN 卷积层运算）；
- 第 2 阶段，RPN 非卷积部分和 ROI Pooling，由 CPU 软件运算；
- 第 3 阶段，FC 部分，由 NNIE 硬件运算；
- 第 4 阶段，GetResult 部分，由 CPU 软件运算，如图 3-7 所示。

iFDT 类型网络分两段式执行，跟 Faster RCNN 的前两段一致。

RFCN 网络同样分两段式执行，前一段在 NNIE 上执行，后一段在 CPU 上执行。其中，CPU 执行部分通过三个接口调用完成，1) RPN，2) PS_ROIpooling，3) GetResult。



图3-7 Faster RCNN 类型网络四段式执行

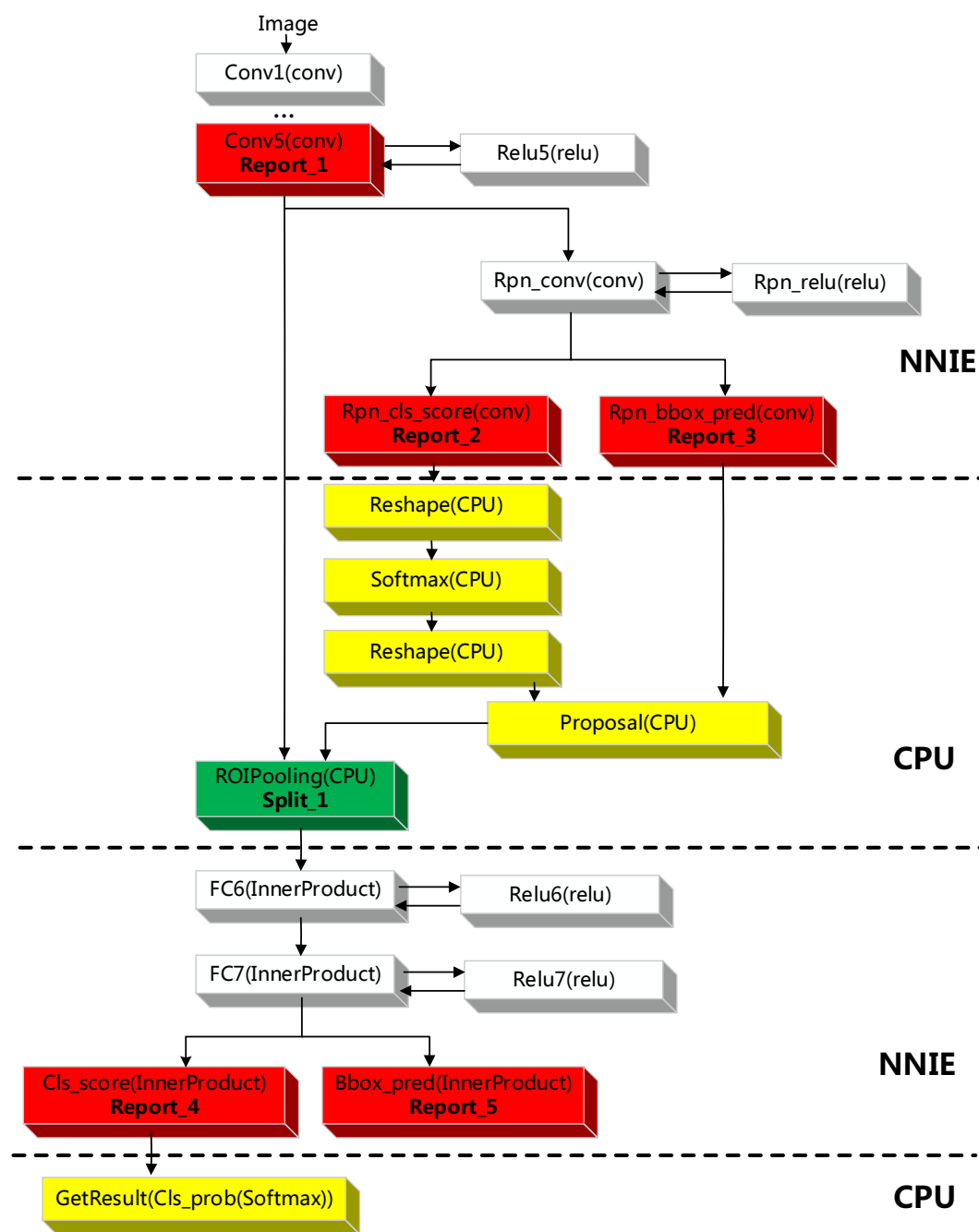
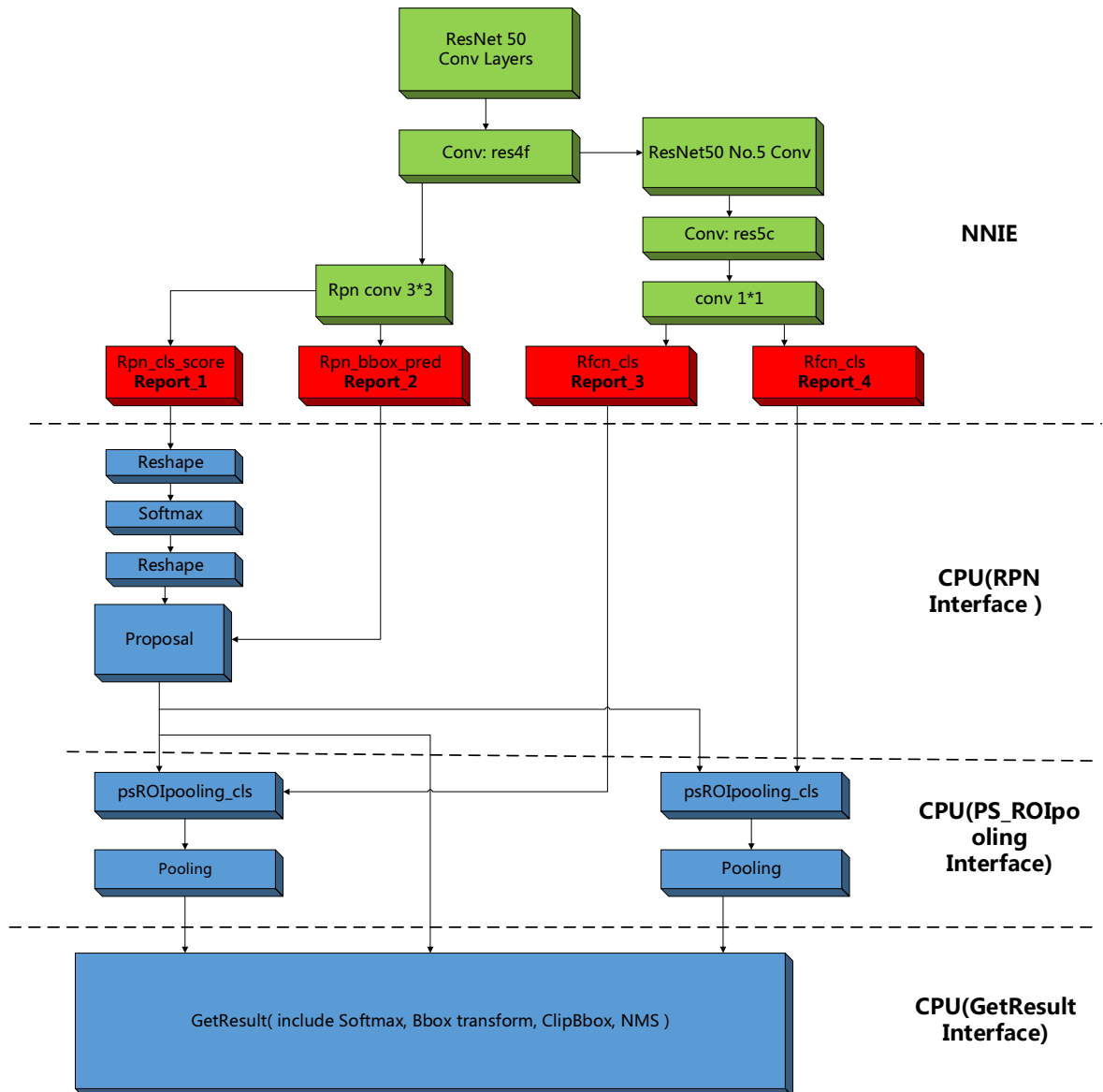


图3-8 RFCN 类型网络两段式执行



为配合 NNIE 和 CPU 软硬件协同执行 FasterRCNN、iFDT、RFCN、SSD 类型网络，NNIE 的编译器制定了一套规则来完成网络层任务的划分，用户需要在网络描述文件 prototxt 中按照约定标识完成标记以便 NNIE 编译器按照预设完成编译。

在 Faster RCNN、iFDT、RFCN、SSD 等类型网络中，NNIE 和 CPU 的配合，主要有 2 类：

- NNIE 硬件部分完成，向 CPU 上报，请 CPU 完成后续软件部分；
- CPU 软件部分完成，向 NNIE 下发后续任务，请 NNIE 执行后续硬件任务；

由此，NNIE 编译器规定，用“Report”标识表示第 1 类的 NNIE 向 CPU 上报，用“Split”标识完成 CPU 向 NNIE 的任务切换。



根据 NNIE 支持的规格，以 Faster RCNN 为例，参考图 3-7 所示，在红色网络节点处采用“Report”标识进行标注，完成 NNIE 向 CPU 的上报。同时，由于 NNIE 支持 ROI Pooling 的后续计算，因此，可以参考绿色网络节点，在该处添加“Split”标识完成 CPU 向 NNIE 的切换。类似的，RFCN 网络的 NNIE 和 CPU 协同处理可以参考图 3-8。

现基于 Alexnet(no_group_no_lrn)的 FasterRCNN 的 prototxt 为例，结合图 3-7，说明“Report”节点和“Split”节点的修改格式，接下来图 3-9~图 3-14 中左边均为原始 prototxt (caffe_alexnet_faster_rcnn_no_group_no_lrn_deploy.prototxt)，右边为加了标识的 prototxt 文件 (cnnie_alexnet_faster_rcnn_no_group_no_lrn_deploy.prototxt，这两个 prototxt 可以在 PC 端 sample 的 SvpSample\data\detection\fasterrcnn\alexnet\model 目录中)。

● Report 节点标识

通过在对应 layer 的 name 域中原有字符末尾添加“_report_x”来标识为 report 节点。

Faster RCNN 第一段 report 节点为：report_1; report_2; report_3;

Faster RCNN 第三段 report 节点为：report_4; report_5

注：Report1 支持层类型 convolution, maxpooling, avepooling, nnerproduct, lrn, eltwise; Report2-5 支持层类型 convolution, avepooling, innerproduct。

1) Report1

由于 RPN 前层卷积（附带 Relu 操作）即需要上报，因此在该卷积层添加 report 标识，修改前后对比如图 3-9 所示。



图3-9 Report1 修改前后对比

<pre>layer { name: "conv5" type: "Convolution" bottom: "conv4" top: "conv5" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 256 pad: 1 kernel_size: 3 #group: 2 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0.1 } } }</pre>	<pre>layer { name: "conv5_report_1" type: "Convolution" bottom: "conv4" top: "conv5" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 256 pad: 1 kernel_size: 3 #group: 2 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0.1 } } }</pre>
<pre>layer { name: "relu5" type: "ReLU" bottom: "conv5" top: "conv5" }</pre>	<pre>layer { name: "relu5" type: "ReLU" bottom: "conv5" top: "conv5" }</pre>

2) Report2-3

由于 RPN 卷积层后续由 CPU 执行，NNIE 需要在执行完后将结果上报给 CPU，以便 CPU 完成 FasterRCNN 第 2 阶段的处理，因此，需在此处添加 report 标识。

图3-10 Report2 修改前后对比

<pre>layer { name: "rpn_cls_score" type: "Convolution" bottom: "rpn/output" top: "rpn_cls_score" convolution_param { num_output: 18 # 2(bg/fg) * 9(anchors) kernel_size: 1 pad: 0 stride: 1 weight_filler { type: "gaussian" bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "rpn_cls_score_report_2" type: "Convolution" bottom: "rpn/output" top: "rpn_cls_score" convolution_param { num_output: 18 # 2(bg/fg) * 9(anchors) kernel_size: 1 pad: 0 stride: 1 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0 } } }</pre>
--	---

图3-11 Report3 修改前后对比

<pre> layer { name: "rpn_bbox_pred" type: "Convolution" bottom: "rpn/output" top: "rpn_bbox_pred" convolution_param { num_output: 36 # 4 * 9(anchors) kernel_size: 1 pad: 0 stride: 1 weight_filler { type: "gaussian" bias_filler { type: "constant" value: 0 } } } </pre>	<pre> layer { name: "rpn_bbox_pred_report_3" type: "Convolution" bottom: "rpn/output" top: "rpn_bbox_pred" convolution_param { num_output: 36 # 4 * 9(anchors) kernel_size: 1 pad: 0 stride: 1 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0 } } } </pre>
---	--

3) Report4-5

由于需要将 NNIE 处理的最终结果上报给 CPU，以便 CPU 提取最终选框信息和对应的概率信息，因此末尾需添加 report 标识。

图3-12 Report4 修改前后对比

<pre> layer { name: "cls_score" type: "InnerProduct" bottom: "fc7" top: "cls_score" inner_product_param { num_output: 2 } } </pre>	<pre> layer { name: "cls_score_report_4" type: "InnerProduct" bottom: "fc7" top: "cls_score" inner_product_param { num_output: 2 } } </pre>
--	---

图3-13 Report5 修改前后对比

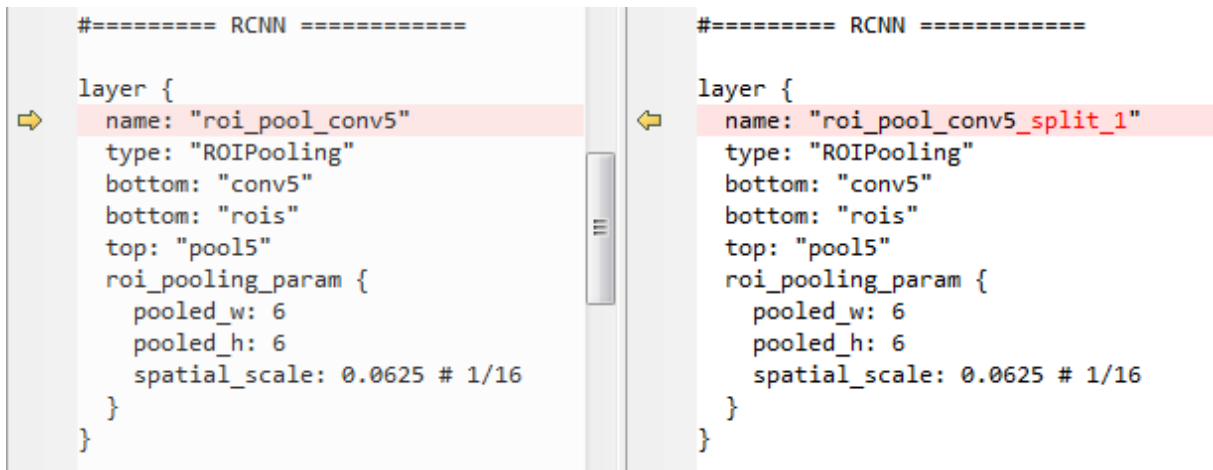
<pre> layer { name: "bbox_pred" type: "InnerProduct" bottom: "fc7" top: "bbox_pred" inner_product_param { num_output: 8 } } </pre>	<pre> layer { name: "bbox_pred_report_5" type: "InnerProduct" bottom: "fc7" top: "bbox_pred" inner_product_param { num_output: 8 } } </pre>
--	---

4) Split 节点标识

Split 节点通过在对应该 layer 域中原有字符串末尾加入“_split_1”来区分 FasterRCNN 第二段和第三段，从第一段上报节点到分割节点间的所有节点为 CPU 处理节点，即 FasterRCNN 第二段。

注：split 节点标识切分需考虑 NNIE 支持规格，且仅支持一个 split 节点。

图3-14 Split 修改前后对比



RFCN 网络采用两段式执行，只需通过 report 标记完成 NNIE 向 CPU 上报即可。

Prototxt 文件的修改可以参考图 3-15~图 3-18，对比的两个 prototxt 在 PC 端 SvpSample\data\detection\rfcn\model 目录中。

图3-15 Report1 修改前后对比

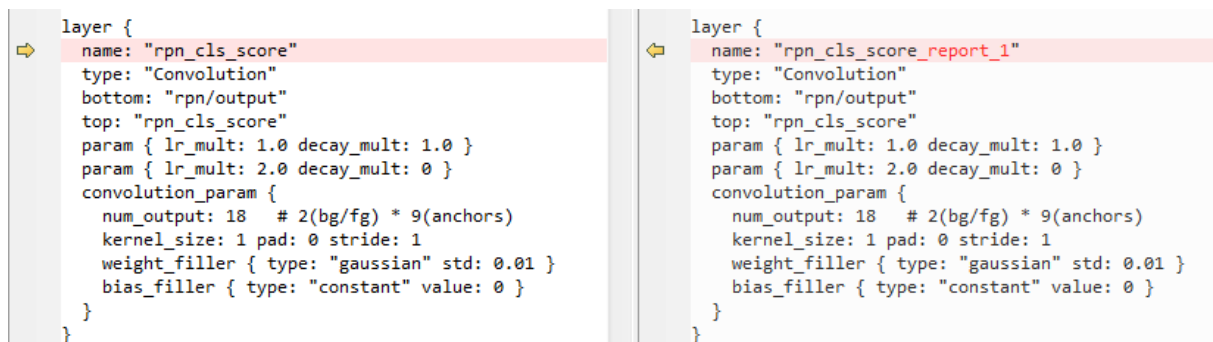


图3-16 Report2 修改前后对比

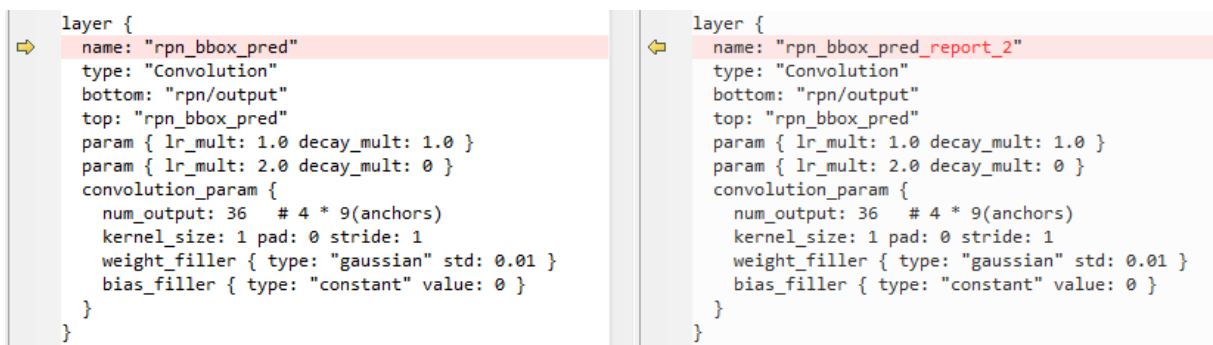


图3-17 Report3 修改前后对比

<pre> layer { bottom: "conv_new_1" top: "rfcn_cls" name: "rfcn_cls" type: "Convolution" convolution_param { num_output: 1029 #21*(7^2) cls_num*(score_map kernel_size: 1 pad: 0 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0 } } param { lr_mult: 1.0 } param { lr_mult: 2.0 } } </pre>	<pre> layer { bottom: "conv_new_1" top: "rfcn_cls" name: "rfcn_cls_report_3" type: "Convolution" convolution_param { num_output: 1029 #21*(7^2) cls_num*(score_maps_size^2) kernel_size: 1 pad: 0 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0 } } param { lr_mult: 1.0 } param { lr_mult: 2.0 } } </pre>
---	---

图3-18 Report4 修改前后对比

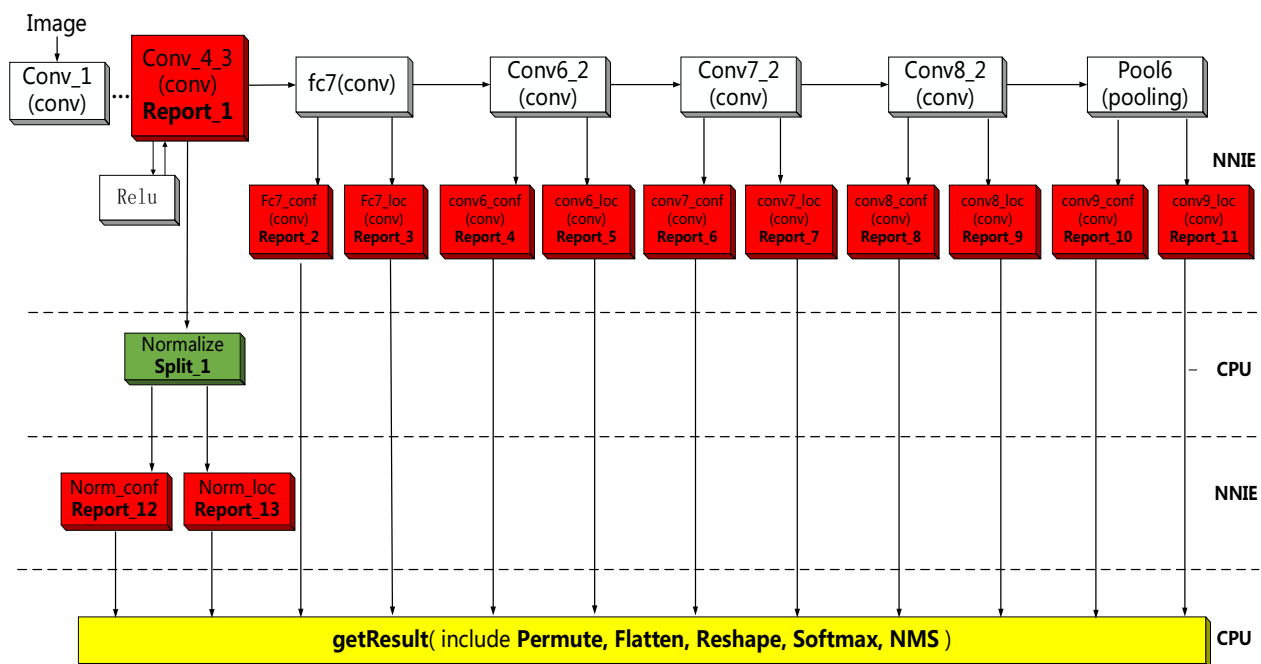
<pre> layer { bottom: "conv_new_1" top: "rfcn_bbox" name: "rfcn_bbox" type: "Convolution" convolution_param { num_output: 392 #8*(7^2) cls_num*(score_maps_ kernel_size: 1 pad: 0 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0 } } param { lr_mult: 1.0 } param { lr_mult: 2.0 } } </pre>	<pre> layer { bottom: "conv_new_1" top: "rfcn_bbox" name: "rfcn_bbox_report_4" type: "Convolution" convolution_param { num_output: 392 #8*(7^2) cls_num*(score_maps_size^2) kernel_size: 1 pad: 0 weight_filler { type: "gaussian" std: 0.01 } bias_filler { type: "constant" value: 0 } } param { lr_mult: 1.0 } param { lr_mult: 2.0 } } </pre>
---	---

类似地，SSD 网络也可以采用 **report** 和 **split** 标识进行标记以便编译器完成相应指令编译，实现软硬件协同处理。

由于 SSD 网络中存在 NNIE 不支持层类型(Normalize 层类型)，该层需切换成 CPU 实现，即类似 FasterRCNN 网络类型中的第二段 RPN 和 ROI Pooling 由 CPU 实现。待 NNIE 将 SSD 中的所有卷积计算过程完成后（其中 Normalize 层执行完重新切换回 NNIE 执行卷积），最终通过 CPU 实现 permute、flatten、priorbox 等操作（NNIE 当前不支持前述操作，统一在 GetResult 中完成，如图 3-19）完成最终检测。



图3-19 SSD 类型网络四段式执行



Prototxt 文件的修改可以参考图 3-20~图 3-33。其中，左图为原始 SSD prototxt 文件，右图为 NNIE 对应的 prototxt 文件。对比的两个 prototxt 在 PC 端 SvpSample\data\detection\ssd\model 目录中。

图3-20 Report1 修改前后对比

<pre>layer { name: "conv4_3" type: "Convolution" bottom: "conv4_2" top: "conv4_3" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 512 pad: 1 kernel_size: 3 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv4_3_report_1" type: "Convolution" bottom: "conv4_2" top: "conv4_3" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 512 pad: 1 kernel_size: 3 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
--	---

图3-21 Split1 修改前后对比

<pre>layer { name: "conv4_3_norm" type: "Normalize" bottom: "conv4_3" top: "conv4_3_norm" norm_param { across_spatial: false scale_filler { type: "constant" value: 20 } } channel_shared: false }</pre>	<pre>layer { name: "conv4_3_norm_split_1" type: "Normalize" bottom: "conv4_3" top: "conv4_3_norm" norm_param { across_spatial: false scale_filler { type: "constant" value: 20 } } channel_shared: false }</pre>
--	--



图3-22 Report13 修改前后对比

修改前	修改后
<pre>layer { name: "conv4_3_norm_mbox_loc" type: "Convolution" bottom: "conv4_3_norm" top: "conv4_3_norm_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 16 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv4_3_norm_mbox_loc_report_13" type: "Convolution" bottom: "conv4_3_norm" top: "conv4_3_norm_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 16 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>

图3-23 Report12 修改前后对比

修改前	修改后
<pre>layer { name: "conv4_3_norm_mbox_conf" type: "Convolution" bottom: "conv4_3_norm" top: "conv4_3_norm_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 84 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv4_3_norm_mbox_conf_report_12" type: "Convolution" bottom: "conv4_3_norm" top: "conv4_3_norm_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 84 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>



图3-24 Report3 修改前后对比

<pre>layer { name: "fc7_mbox_loc" type: "Convolution" bottom: "fc7" top: "fc7_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 24 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "fc7_mbox_loc_report_3" type: "Convolution" bottom: "fc7" top: "fc7_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 24 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
---	--



图3-25 Report2 修改前后对比

<pre>layer { name: "fc7_mbox_conf" type: "Convolution" bottom: "fc7" top: "fc7_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 126 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "fc7_mbox_conf_report_2" type: "Convolution" bottom: "fc7" top: "fc7_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 126 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
--	---



图3-26 Report5 修改前后对比

<pre>layer { name: "conv6_2_mbox_loc" type: "Convolution" bottom: "conv6_2" top: "conv6_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 24 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv6_2_mbox_loc_report_5" type: "Convolution" bottom: "conv6_2" top: "conv6_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 24 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
---	--

图3-27 Report4 修改前后对比

<pre>layer { name: "conv6_2_mbox_conf" type: "Convolution" bottom: "conv6_2" top: "conv6_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 126 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv6_2_mbox_conf_report_4" type: "Convolution" bottom: "conv6_2" top: "conv6_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 126 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
--	---



图3-28 Report7 修改前后对比

<pre>layer { name: "conv7_2_mbox_loc" type: "Convolution" bottom: "conv7_2" top: "conv7_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 24 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv7_2_mbox_loc_report_7" type: "Convolution" bottom: "conv7_2" top: "conv7_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 24 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
---	--



图3-29 Report6 修改前后对比

修改前	修改后
<pre>layer { name: "conv7_2_mbox_conf" type: "Convolution" bottom: "conv7_2" top: "conv7_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 126 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv7_2_mbox_conf_report_6" type: "Convolution" bottom: "conv7_2" top: "conv7_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 126 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>



图3-30 Report9 修改前后对比

修改前	修改后
<pre>layer { name: "conv8_2_mbox_loc" type: "Convolution" bottom: "conv8_2" top: "conv8_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 16 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv8_2_mbox_loc_report_9" type: "Convolution" bottom: "conv8_2" top: "conv8_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 16 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>



图3-31 Report8 修改前后对比

<pre>layer { name: "conv8_2_mbox_conf" type: "Convolution" bottom: "conv8_2" top: "conv8_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 84 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv8_2_mbox_conf_report_8" type: "Convolution" bottom: "conv8_2" top: "conv8_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 84 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>
---	--



图3-32 Report11 修改前后对比

修改前	修改后
<pre>layer { name: "conv9_2_mbox_loc" type: "Convolution" bottom: "conv9_2" top: "conv9_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 16 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>	<pre>layer { name: "conv9_2_mbox_loc_report_11" type: "Convolution" bottom: "conv9_2" top: "conv9_2_mbox_loc" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 16 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } }</pre>

图3-33 Report10 修改前后对比

<pre> layer { name: "conv9_2_mbox_conf" type: "Convolution" bottom: "conv9_2" top: "conv9_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 84 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } } </pre>	<pre> layer { name: "conv9_2_mbox_conf_report_10" type: "Convolution" bottom: "conv9_2" top: "conv9_2_mbox_conf" param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0 } convolution_param { num_output: 84 pad: 1 kernel_size: 3 stride: 1 weight_filler { type: "xavier" } bias_filler { type: "constant" value: 0 } } } </pre>
---	---



注意

Report 节点的顺序和硬件输出的顺序一致，建议用按照图示流程来设置，《HiSVP API 参考》中的输出内存排布跟本节的图示 report 节点顺序是一致的。

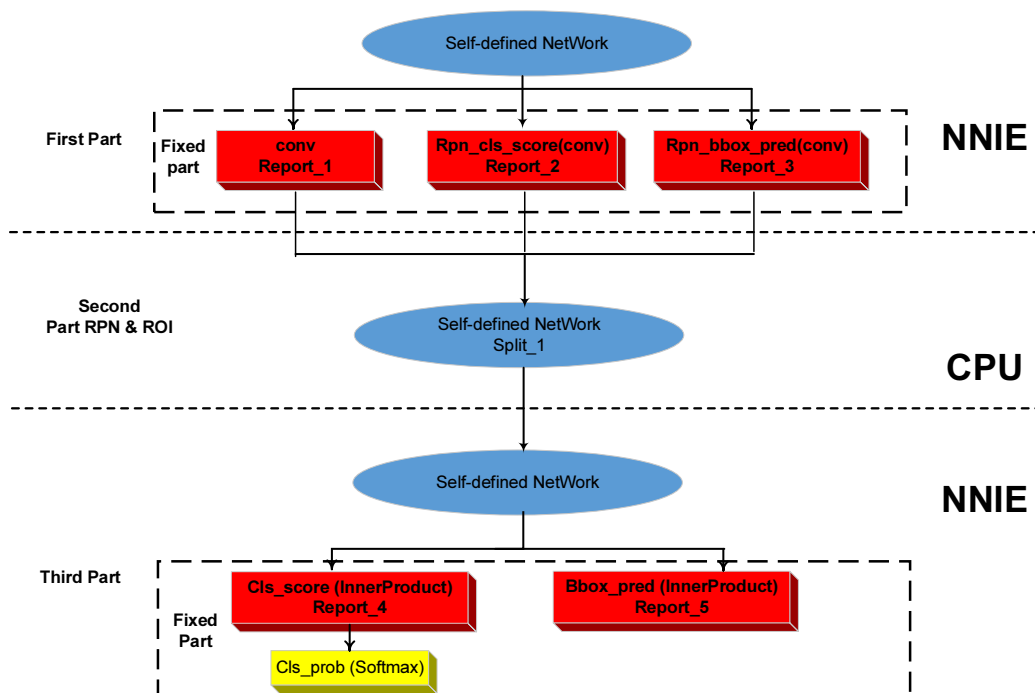
3.5.2 网络扩展说明

一般地，网络的最后一部分是软件接口，典型的的就是获取最终结果的 GetResult 接口。若用户希望自行编写最后一段的软件接口来获取最终结果，需要参考以下说明：

- 在编译前应将网络的最后一段在 CPU 上执行的所有层对应的 prototxt 描述删除，然后再进行编译；
- 由于编译器并没有把最后一段在 CPU 上执行的各层网络参数编译进去，因此，最后这段 CPU 上执行各层的参数需由用户读取出来，结合前段硬件输出，实现这部分 CPU 上执行的运算，获取最终结果；
- 此外，仿真器采用海思提供的 GetResult 等标准接口实现，因此，仿真器不支持用户采用第 1 步操作编译得到的知识库文件。

FasterRCNN、iFDT 类型的网络，基于前一节标识约定，在保持软硬件切换处结构不变的情况下，即 Report 节点和 Split 节点数目、顺序、结构不变，用户可以对其余部分在 NNIE 支持的规格内进行自定义，具体如图 3-34 所示，可自定义部分包含 3 个椭圆部分和最后的 GetResult 部分

图3-34 Faster RCNN、iFDT 类型网络扩展



SSD 网络目前支持第一段网络结构自定义（需符合 NNIE 支持规格）和 GetResult 自实现，如图 3-35 红色虚线框出部分，其余部分需要满足以下规格：

- Report 和 Split 的相对位置、组合以及个数均需满足图 3-35 对应关系；
- 图 3-35 中 CPU 实现的 Normalize 仅支持一个；
- GetResult 部分自定义需遵循前述自定义说明，若采用海思提供的标准 GetResult 接口需要符合 sample SSD 网络；
- 海思提供的标准 Normalize 和 GetResult 接口遵循以下规格：
 - 最多两级 concat 结点串联, concat 个数上限为 32；
 - 仅支持 1) permute, 2) flatten, 3) priorbox, 4) softmax, 5) detectionoutput, 6) normalize, 其中 permute、flatten、priorbox、softmax 和 normalize 仅支持单输入（含 concat 输入），Detectionoutput 多输入（含 concat）个数上限为 32；
- Prior box 层参数需要满足表 3-1 所示的规格。

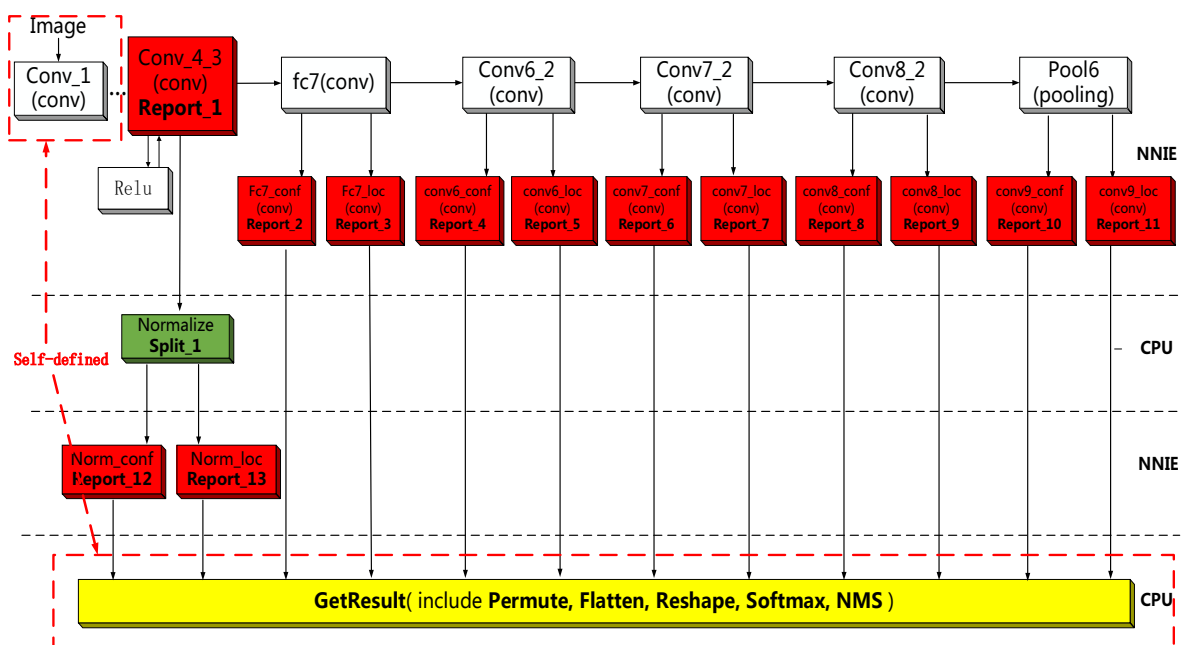
表3-1 规格说明

名称	规格说明
min_size	个数大于等于 1 个，小于等于 32 个； 值必须大于 0；
max_size	个数大于等于 0 个，小于等于 32 个； 如果个数大于 0 个，则个数必须等于 min_size 的个数；



名称	规格说明
	值必须大于 0，且大于 min_size 的值；
img_size	img_size 和 img_h/img_w 二者必选一，不可缺省；
img_h/img_w	img_size 和 img_h/img_w 二者必选一，不可缺省；
step	step 和 step_h/step_w 二者必选一，不可缺省；
step_h/step_w	step 和 step_h/step_w 二者必选一，不可缺省；
aspect_ratio	flip 为 true 时，个数大于等于 0 个，小于等于 16 个； flip 为 false 时，个数大于等于 0 个，小于等于 32 个。
variance	个数为 0 或者 1 或者 4
offset	不可缺省

图3-35 SSD 类型网络扩展

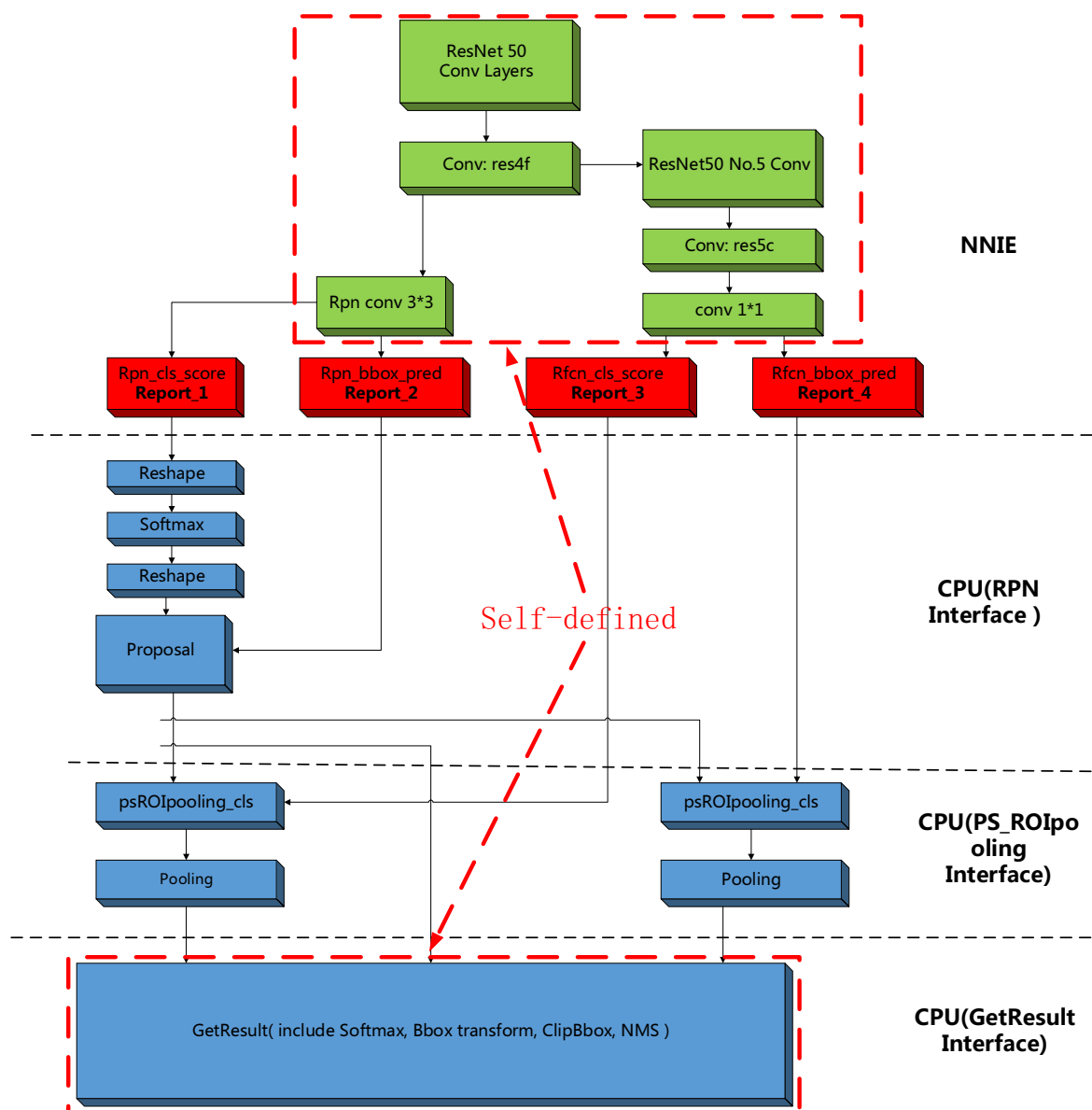


和 SSD 类似，RFCN 网络支持第一段网络结构自定义（需符合 NNIE 支持规格）和 GetResult 自实现，如图 3-36，其余部分需要满足以下规格：

- Report 相对位置以及个数符满足图 3-36；
- CPU 实现 RPN 部分可参考 Faster-RCNN Sample 代码；
- GetResult 部分自定义需遵循前述自定义说明，若采用海思提供的标准 GetResult 接口需要符合 sample RFCN 网络；

- CPU 实现 PSROIpooling 部分规格：仅支持 1) psroipooling, 2) average pooling, 其中 PSROIpoolng 仅支持两路输入（含 roi 输入），average pooling 仅支持单输入。

图3-36 RFCN 类型网络扩展



3.5.3 模型下载

- Googlenet
下载链接: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet



deploy.prototxt	[examples] switch examples + models to Input layers	11 months ago
quick_solver.prototxt	Added bvlc_googlenet prototxt and weights	2 years ago
readme.md	Remove Gist from BVLC GoogleNet	2 years ago
solver.prototxt	Added bvlc_googlenet prototxt and weights	2 years ago
train_val.prototxt	minor typo	10 days ago

name	caffemodel	caffemodel_url	license
BVLC GoogleNet Model	bvlc_googlenet.caffemodel	http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel	unrestricted

- VGG16

prototxt 下载链接(注: prototxt 格式需按 3.5.1 “Prototxt 要求” 修改):
http://www.robots.ox.ac.uk/~vgg/research/very_deep/

Models

We release our two best-performing models, with 16 and 19 weight layers (denoted as configurations *D* and *E* use the models.

The models are compatible with the [Caffe](#) toolbox. They are available in the Caffe format from the [Caffe Zoo](#)

- 16-layer model: [information page](#) in the Caffe Zoo, [weights \(528 MB\)](#), [layer configuration](#)
- 19-layer model: [information page](#) in the Caffe Zoo, [weights \(548 MB\)](#), [layer configuration](#)

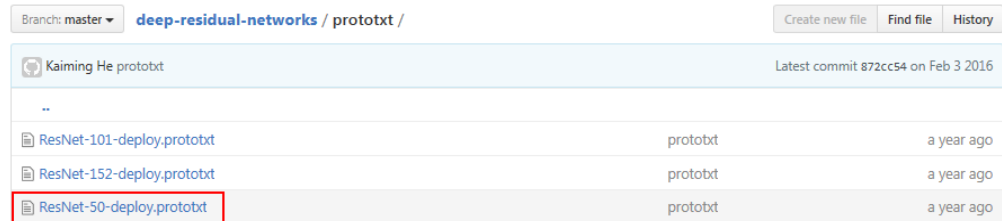
The models can also be used with the [MatConvNet](#) toolbox. They are available in the MatConvNet format from

caffemodel 下载链接: https://deeptdetect.com/applications/list_models/

	Caffe	Tensorflow	Source
AlexNet	Y	N	BVLC
SqueezeNet	Y	N	DeepScale
Inception v1 / GoogLeNet	Y	Y	BVLC / Google
Inception v2	N	Y	Google
Inception v3	N	Y	Google
ResNet 50	Y	Y	MSR
ResNet 101	Y	Y	MSR
ResNet 152	Y	Y	MSR
Inception-ResNet-v2	N	Y	Google
VGG-16	Y	Y	Oxford
VGG-19	Y	Y	Oxford

- Resnet-50、resnet-101、resnet-152

prototxt 下载链接: <https://github.com/KaimingHe/deep-residual-networks/tree/master/prototxt>



Caffemodel 下载链接: https://deeptdetect.com/applications/list_models/

Models are provided for image and text classification

Generic image models

These models are pre-trained on the 1000 classes of the ImageNet ILSVRC Challenge. They can be used as is or finetuned to more targeted applications.

	Caffe	Tensorflow	Source	Top-1 Accuracy (ImageNet)
AlexNet	Y	N	BVLC	57.1%
SqueezeNet	Y	N	DeepScale	59.5%
Inception v1 / GoogleNet	Y	Y	BVLC / Google	67.9%
Inception v2	N	Y	Google	72.2%
Inception v3	N	Y	Google	76.9%
ResNet 50	Y	Y	MSR	75.3%
ResNet 101	Y	Y	MSR	76.4%
ResNet 152	Y	Y	MSR	77%
Inception-ResNet-v2	N	Y	Google	79.79%
VGG-16	Y	Y	Oxford	70.5%
VGG-19	Y	Y	Oxford	71.3%

3.6 HiSVP_CNNIE8Bit 使用

安装好 NNIE 工具链后，双击 HiSVP_CNNIE8Bit.exe，启动后选择工作空间，待加载完毕后计入软件主页面。

3.6.1 创建工程

3.6.1.1 创建过程

步骤 1. 可以采用多种方式进行工程创建，如图 3-37 到图 3-40:

- 1) 依次点击 File→New→Project→NNIE→CNNIE8Bit Project→Next;
- 2) 从快捷工具栏 New 下拉选择 Project→NNIE→CNNIE8Bit Project→Next;
- 3) 在 Project Explorer 视图中鼠标右键 New→Project→NNIE→CNNIE8Bit Project→Next;



图3-37 创建工程方式一

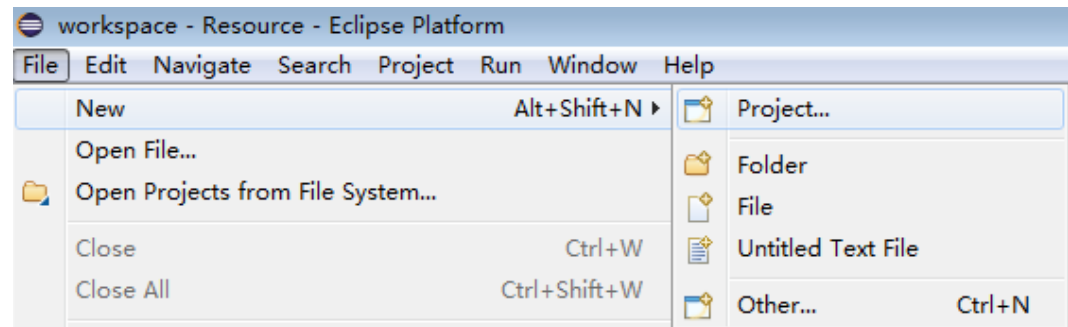


图3-38 创建工程方式二

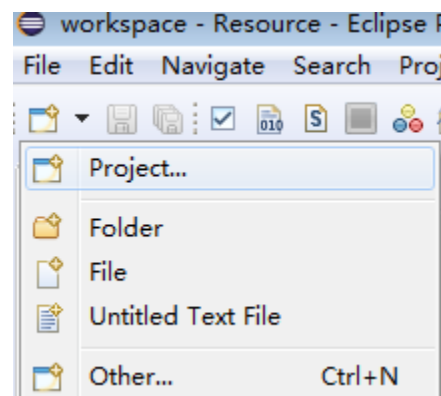


图3-39 创建工程方式三

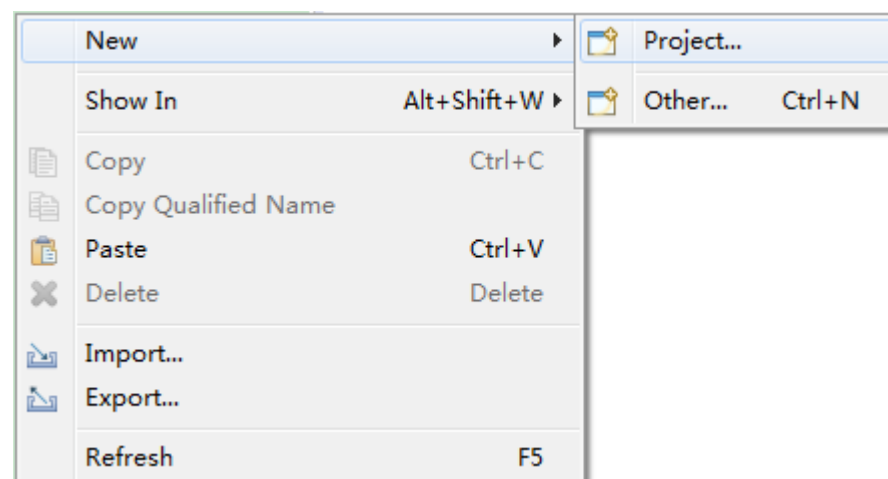
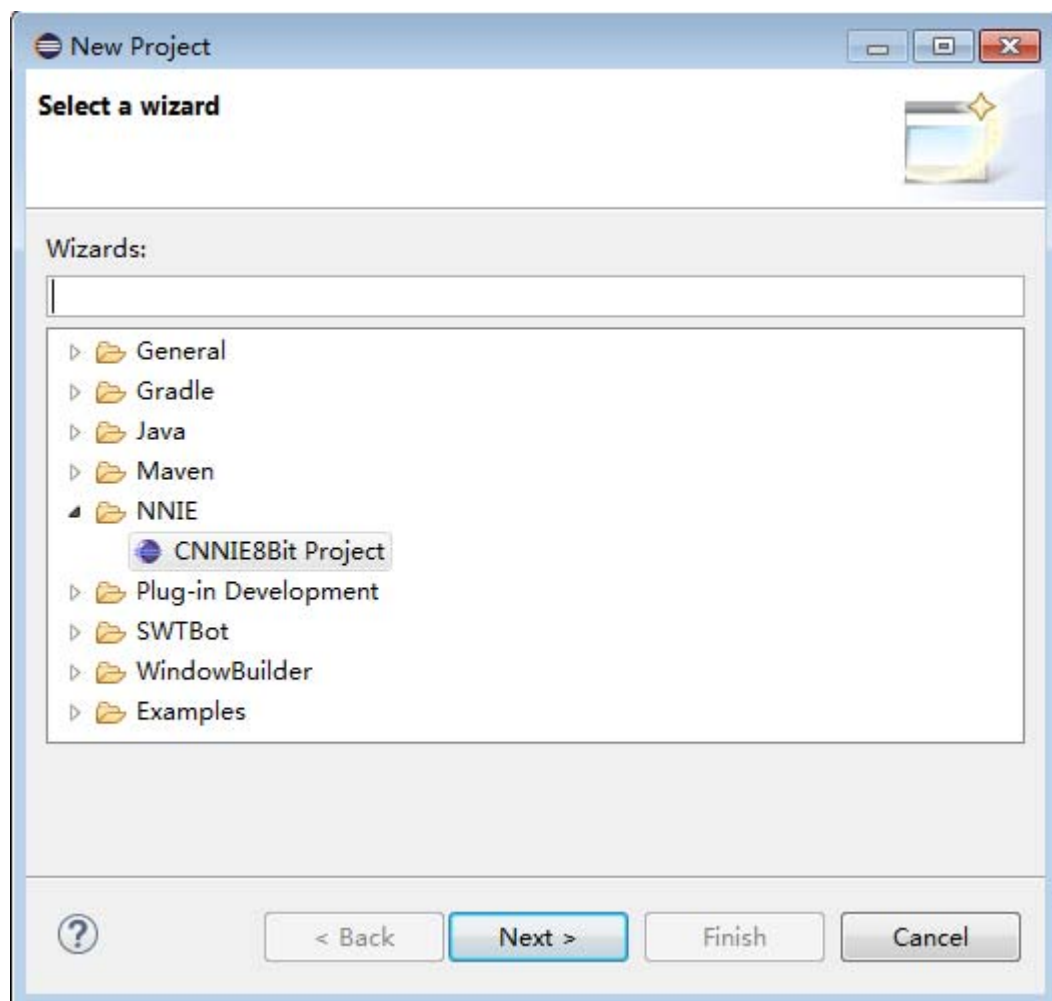




图3-40 选择 CNNIE8Bit Project



步骤 2. 设置工程名称及工程路径，如图 3-41，点击 Finish 按钮后成功创建工程，得到如图 3-42 所示工程。



图3-41 设置工程名及工程路径

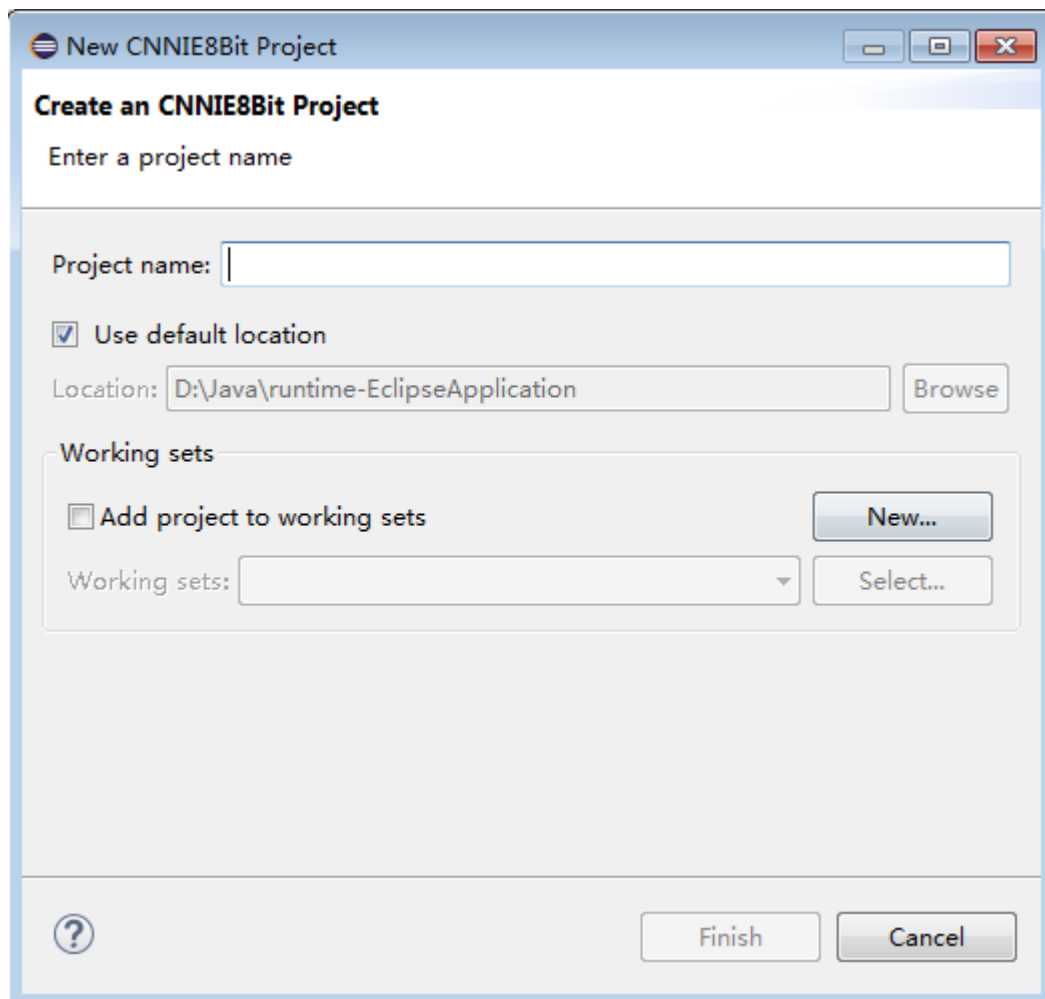
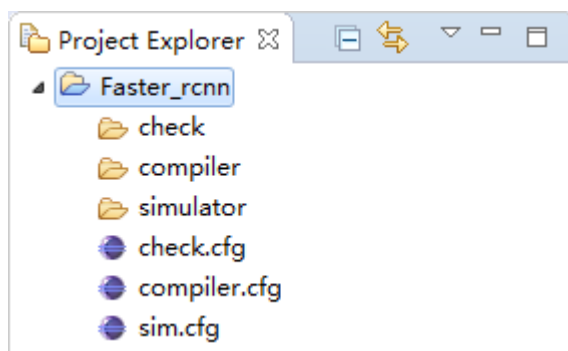


图3-42 创建工程后的工程视图



----结束

3.6.1.2 工程文件说明

如图 3-42，创建后的工程包含三个空文件夹和三个配置文件：

check、compiler 和 simulator 文件夹分别用于存放网络过滤、编译和仿真过程的输出，文件夹中的内容在编译或者仿真后都会自动刷新，故而不能用于存放其他内容。

- check.cfg：用于配置网络过滤参数；
- compile.cfg：用于配置编译器运行参数；
- sim.cfg：用于配置仿真器运行参数。

3.6.2 网络过滤器

3.6.2.1 check.cfg 文件编辑模式

check.cfg 文件有两种编辑模式：

- 如图 3-43，选择用 Check Net Type Configuration Editor 打开，打开后如图 3-44，使用界面的方式来预览、编辑、保存该 cfg 文件；
- 如图 3-43，可以选择使用 Text Editor 来打开，打开后如图 3-46，使用文本方式来浏览、编辑、保存该 cfg 文件；

用户在编辑完后可通过工具栏上的保存图标或 Ctrl+S 来保存修改。



说明

cfg 文件默认打开方式为上一次的打开方式，即若用户右键以 Text Editor 的方式打开配置文件修改并关闭后，下一次双击打开文件亦是以 Text Editor 的方式。

图3-43 check.cfg 文件两种打开方式

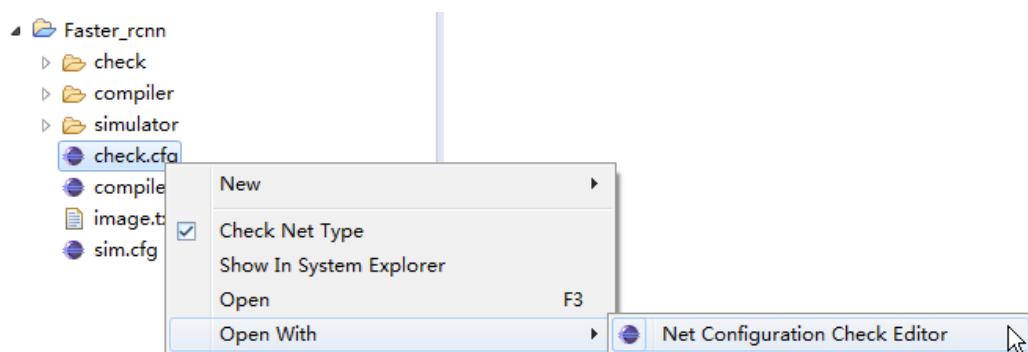




图3-44 check.cfg 以 check Net Type Configuration Editor 模式打开

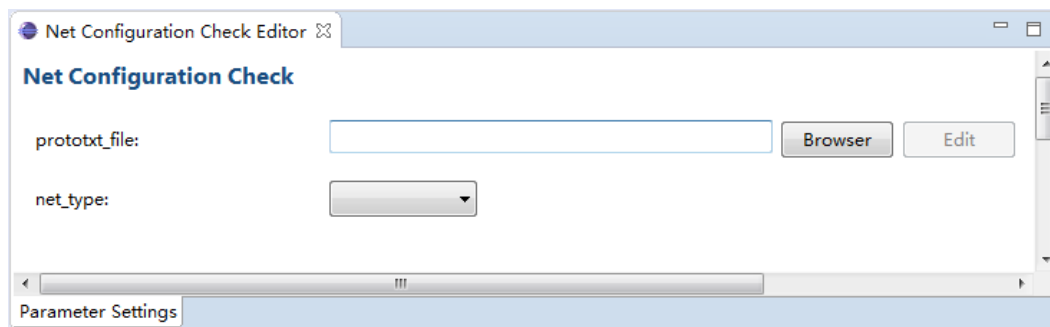


图3-45 check.cfg 选择 faster RCNN 网络

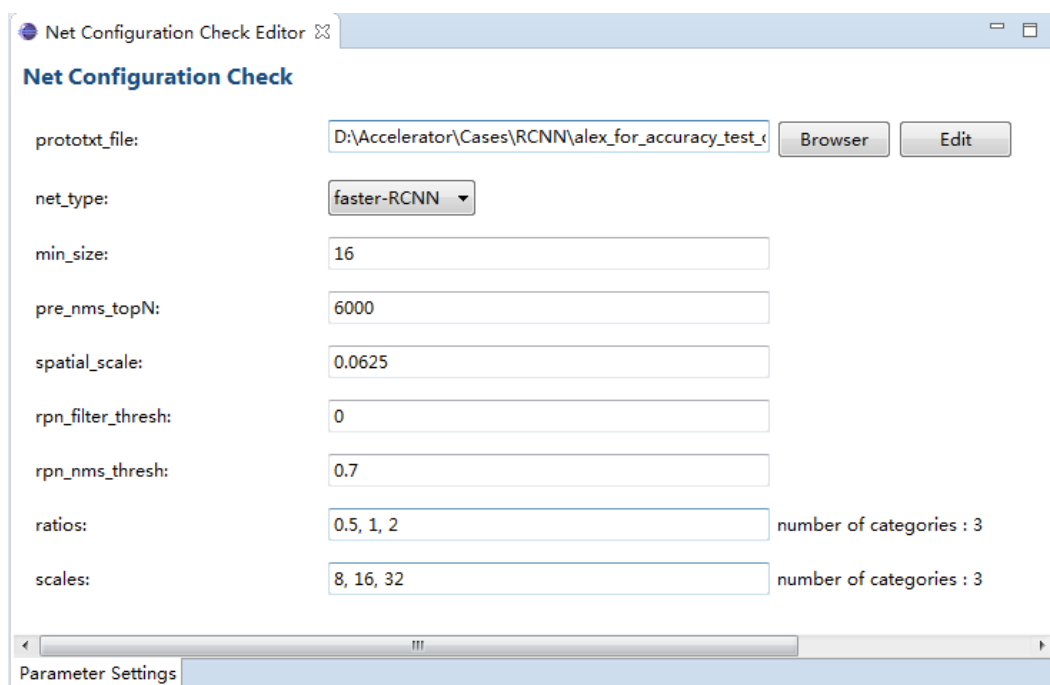
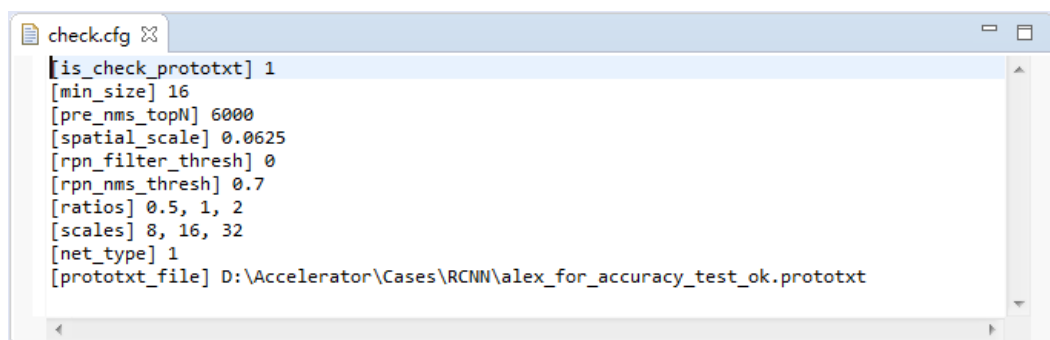


图3-46 check.cfg 选择 Text Editor 模式打开





3.6.2.2 配置文件说明

网络过滤器配置选项说明如表 3-1 所示。

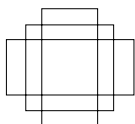
表3-2 网络过滤器配置选项说明

配置选项	取值范围	描述
prototxt_file	-	网络描述文件 deploy.prototxt 文件。
net_type	{0, 1, 2, 3, 4}	网络的类型。 0: CNN; 1: faster RCNN; 2: iFDT, 海思自研快速版 faster RCNN; 3: SSD 4: RFCN
min_size	[1, 100]	Anchor 最小值。 默认值: 16。
pre_nms_topN	[100, 20000]	RPN 层保留候选 anchor 进入非极大抑制操作的数目。 默认值: 6000
spatial_scale	(0, 2]	RoiPooling 卷积层输入分辨率跟原始输入图像分辨率的比例。 默认值: 0.0625 (=1/16)。
rpn_filter_thresh	[0, 1)	在 RPN 阶段, 会根据该阈值进行候选框的筛选, 该参数主要用来提高 RPN 的运行速度。 默认值: 0.01。
rpn_nms_thresh	(0, 1)	在 RPN 阶段, 做非极大值抑制的阈值参数。 默认值: 0.7。
ratios	(0, 100)	对 anchor 进行形变的比例参数数组, 参数间用逗号分隔, 最多 32 个。 默认值: 0.5, 1, 2。
scales	(0, 500)	对 anchor 进行形变的缩放参数的数组, 参数间用逗号分隔, 最多 32 个。 默认值: 8, 16, 32。
is_check_prototxt	{0, 1}	检查网络描述文件标志。 0: 编译模式, 对 prototxt、caffemodel 等进行编译。 1: 网络过滤器模式, 对 prototxt 文件是否符合编译支持规格进行检查。

【注意】

- UI 下使用字符串表示的选项，在 Text Editor 编辑模式下使用对应数值表示，参考表 3-1 中的说明。
- 网络过滤器只检查 deploy.prototxt，且符合 3.5.1 “Prototxt 要求”的约束条件；
- ratios、scales 分别表示 Faster RCNN、iFDT 在 RPN 阶段用于生成参考 anchors 的长宽比例参数、缩放比例参数。

假设 ratios 配置为[0.5, 1, 2]，则表示会根据 ratios 参数生成 3 种不同长宽比例的 anchors，这 3 种 anchor 的长宽比例分别为 1:2, 1:1, 2:1，如下图所示：



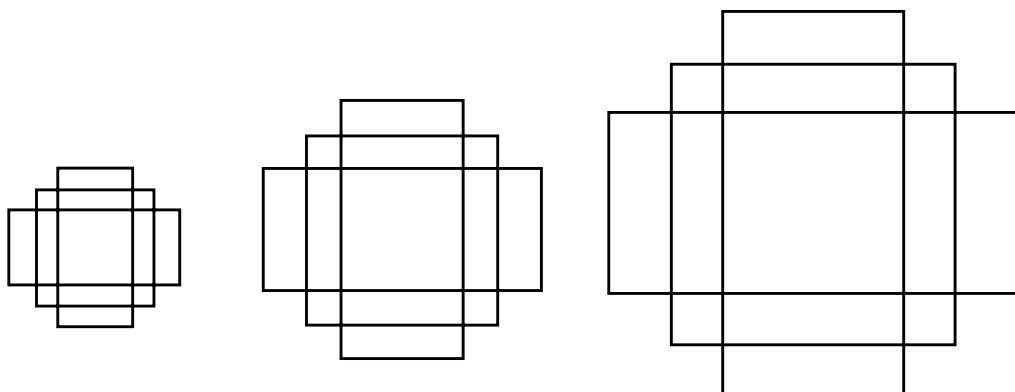
基础 anchor 的高(h)、宽(w)都为 min_size，生成不同 ratio 的 anchor 的公式为

$$w_i = \sqrt{(w * h) / ratios(i)}$$

$$h_i = \sqrt{(w * h) * ratios(i)}$$

$ratios(i)$ 表示第*i*个ratio值

假设 scales 配置为[8, 16, 32]，则上面根据 ratios 配置得到的 anchor 再扩大相应的倍数，如下图所示：



新生成的 anchor 的高、宽计算公式为

$$w_j = w_i * scales(j)$$

$$h_j = h_i * scales(j)$$

$scales(j)$ 表示第*j*个scale值， (w_i, h_i) 表示第*i*个ratio生成的anchor

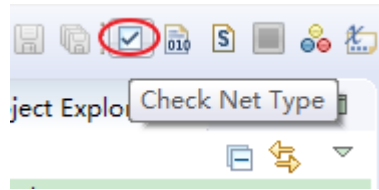
由此得到 9 种 anchor 配置。

3.6.2.3 网络过滤器运行

在 Project Explorer 视图选择指定项目需要检查网络的工程，在 UI 界面上方工具栏上点击网络检查(check)按钮，执行检查。



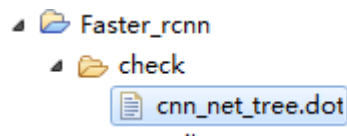
图3-47 Check 按钮



3.6.2.4 网络过滤器输出

检查结果存放在工程目录下的“check”文件夹中。

图3-48 网络过滤器输出



网络过滤器输出如下文件：

编译生成的网络结构 dot 描述 Cnn_net_tree.dot，可显示为 dot 视图，具体参考 [3.6.7 “Dot 图”](#) 章节。

图3-49 网络 check 成功输出示意图

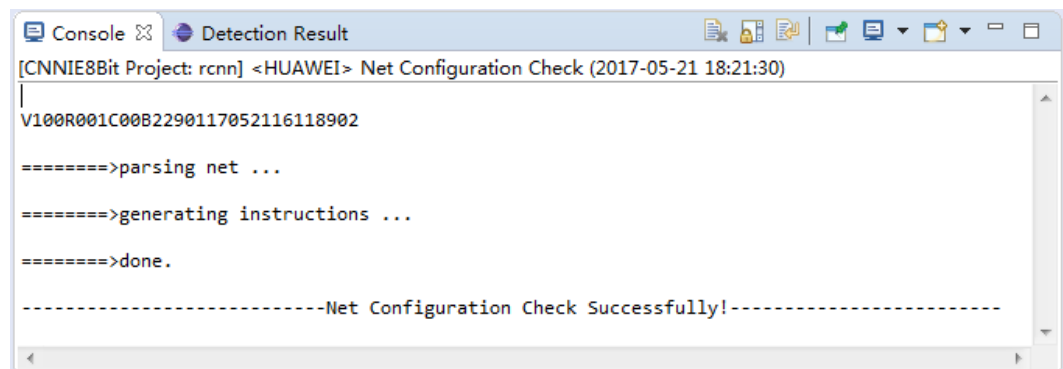
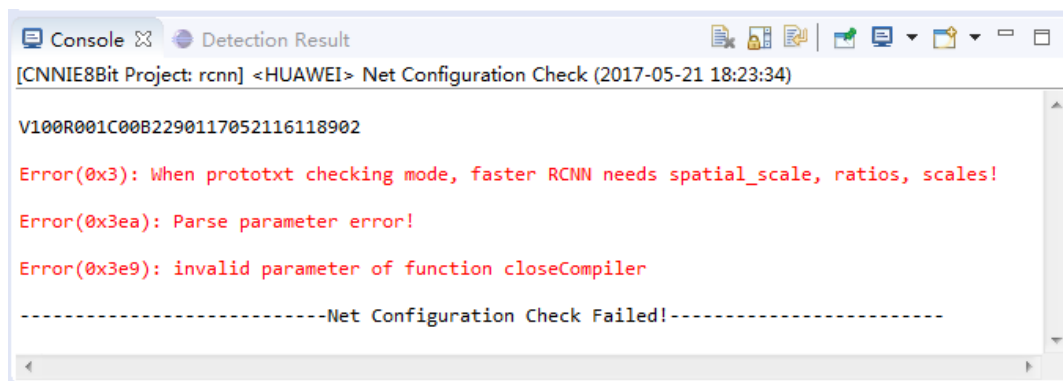


图3-50 网络 check 失败示意图



网络检测失败时，生成文件和打印信息随错误的不同打印的信息也不相同，同时会在工程目录下生成错误日志 `compiler_error.log`。

3.6.3 编译器

3.6.3.1 compiler.cfg 文件编辑模式

同 `check.cfg`，`comiler.cfg` 也有两种编辑模式。如 图 3-51 到 图 3-53。

图3-51 compiler.cfg 以 Compiler Configuration Editor 模式打开

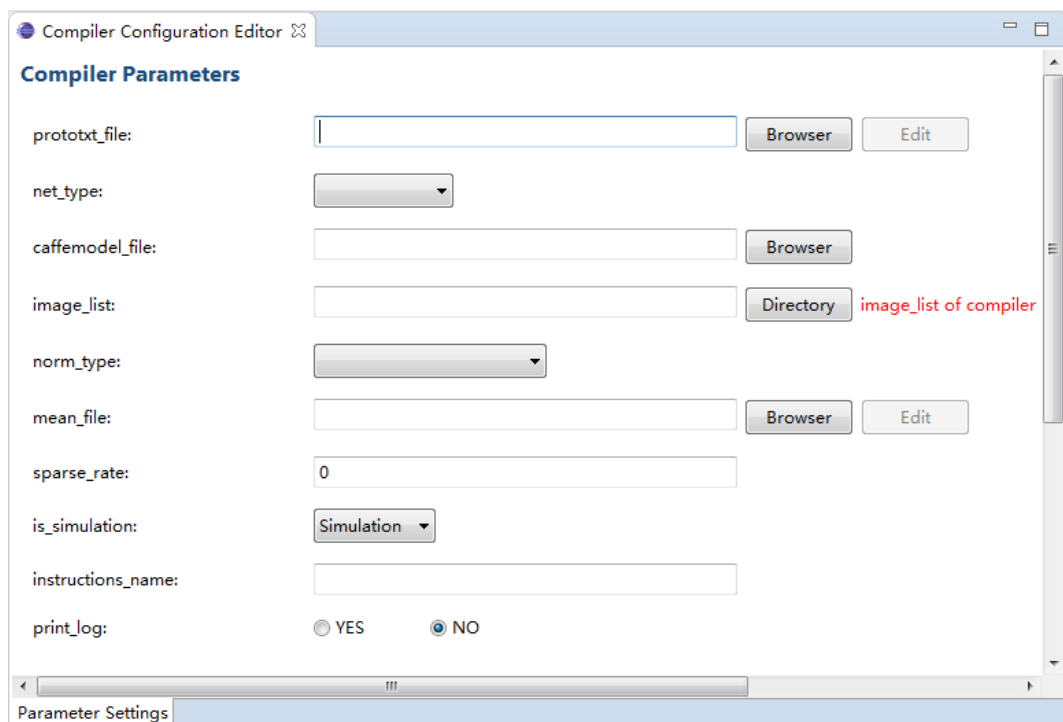




图3-52 compiler.cfg 选择 faster RCNN

Compiler Configuration Editor

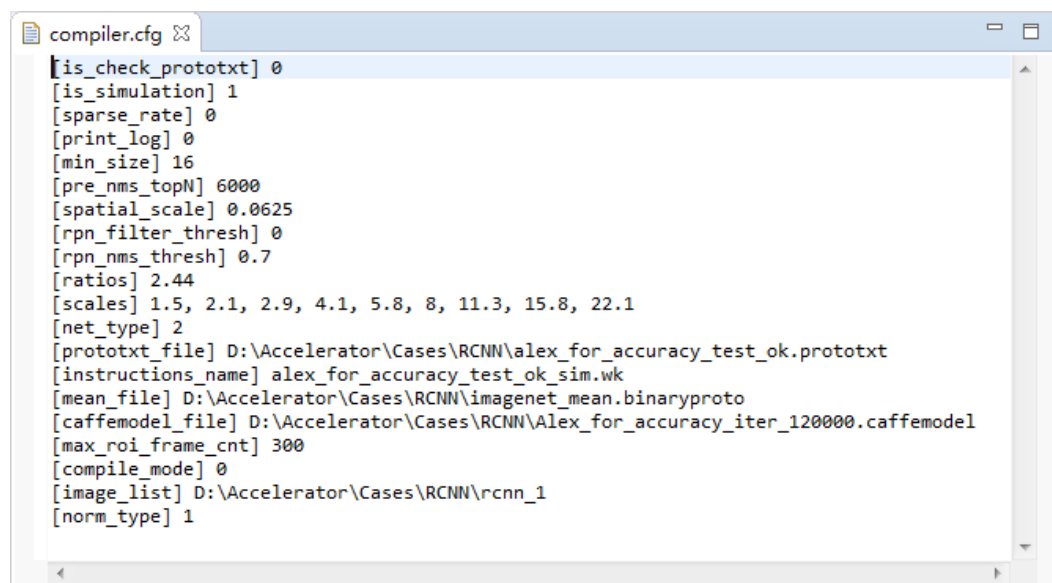
Compiler Parameters

prototxt_file:	D:\Accelerator\Cases\RCNN\alex_for_accuracy_test_1	Browser	Edit
net_type:	iFDT		
caffemodel_file:	D:\Accelerator\Cases\RCNN\Alex_for_accuracy_iter_1	Browser	
image_list:	D:\Accelerator\Cases\RCNN\rcnn_1	Directory	
norm_type:	mean file		
mean_file:	D:\Accelerator\Cases\RCNN\imagenet_mean.binary	Browser	Edit
sparse_rate:	0		
is_simulation:	Simulation		
instructions_name:	alex_for_accuracy_test_ok_sim.wk		
print_log:	<input type="radio"/> YES <input checked="" type="radio"/> NO		
max_roi_frame_cnt:	300	value range : [1, 5000]	
min_size:	16	value range : [1, 100]	
pre_nms_topN:	6000	value range : [100, 20000]	
spatial_scale:	0.0625	value range : (0, 2]	
rpn_filter_thresh:	0	value range : [0, 1)	
rpn_nms_thresh:	0.7	value range : (0, 1)	
ratios:	2.44	number of categories : 1	
scales:	1.5, 2.1, 2.9, 4.1, 5.8, 8, 11.3, 15.8, 22.1	number of categories : 9	

Parameter Settings



图3-53 compiler.cfg 选择 Text Editor 模式打开



3.6.3.2 配置文件说明

编译器配置选项说明如表 3-2 所示。

表3-3 编译器配置选项说明

配置选项	取值范围	描述
prototxt_file	-	网络描述文件。
net_type	{0, 1, 2, 3, 4}	网络的类型。 0: CNN 1: faster RCNN 2: iFDT, 海思自研快速版 faster RCNN 3: SSD 4: RFCN
caffemodel_file	-	网络模型数据文件。
image_directory	-	用于数据量化的参考图像目录。 建议选取典型场景图像 20~50 张。
norm_type	{1, 2, 3, 4, 5}	图像预处理的减均值和归一化方法。 1: mean file, 减图像均值; 2: channel mean_value, 减通道均值; 3: 1/256, 对图像像素值固定缩放 1/256; 4: mean file/256, 减图像均值后再乘以 1/256; 5: channel mean_value/256, 减通道均值后再乘以 1/256。



配置选项	取值范围	描述
mean_file	-	<p>norm_type 为 1: mean file、4: mean file/256 时，表示均值文件 xxx.binaryproto;</p> <p>norm_type 为 2: channel mean_value、5: channel mean_value/256，表示通道均值文件。</p> <p>通道均值文件 mean.txt 中每一行表示对应的通道均值，如单通道只有一个值。</p>
sparse_rate	[0, 1)	<p>NNIE 引擎采用了参数压缩技术以减少带宽占用，为了提高压缩率，可对 FC 参数进稀疏处理。</p> <p>用户通过 sparse_rate 数值指定多少比例的 FC 参数稀疏为 0，例如配 0.5，则 FC 参数有 50%将被稀疏为 0，</p> <p>由于数据变的稀疏，压缩模块会获得更好的压缩率。稀疏值越高，计算 FC 时所需参数带宽越低，但精度会有所下降。</p>
max_roi_frame_cnt	[1, 5000]	<p>Faster RCNN 或 iFDT 网络的 RPN 阶段输出的候选框最大数目。</p> <p>默认值：300。</p>
is_simulation	{0, 1}	<p>编译类型。</p> <p>0: Chip，芯片模式，编译成在芯片上加载的指令文件；</p> <p>1: Simulation，仿真模式，编译成在 PC 端仿真上加载的指令文件；</p>
instructions_name	-	<p>编译器生成的知识库文件名称。</p> <p>在配置好 is_simulation 后，会默认生成如下格式的知识库名：“prototxt 文件名”_“sim/chip”.wk；用户也可以自行修改生成的知识库名字。</p>
print_log	{0, 1}	<p>设置是否开启日志文件。</p> <p>0: NO，不打印日志；</p> <p>1: YES，打印日志。</p>
min_size	[1, 100]	<p>Anchor 最小值。</p> <p>默认值：16。</p>
pre_nms_topN	[100, 20000]	<p>RPN 层保留候选 anchor 进入非极大抑制操作的数目。</p> <p>默认值：6000。</p>
spatial_scale	(0, 2]	<p>RoiPooling 卷积层输入分辨率跟原始输入图像分辨率的比例。</p> <p>默认值：0.0625 (=1/16)。</p>
rpn_filter_thresh	[0, 1)	<p>在 RPN 阶段，会根据该阈值进行候选框的筛选，该参数主要用来提高 RPN 的运行速度。</p> <p>默认值：0.01。</p>



配置选项	取值范围	描述
rpn_nms_thresh	(0, 1)	在 RPN 阶段，做非极大值抑制的阈值参数。 默认值：0.7。
ratios	(0, 100)	对 anchor 进行形变的比例参数的数组，参数间用逗号分隔，最多 32 个。 默认值：0.5，1，2。
scales	(0, 500)	对 anchor 进行形变的缩放参数的数组，参数间用逗号分隔，最多 32 个。 默认值：8，16，32。
is_check_prototxt	{0, 1}	检查网络描述文件标志。 0：编译模式，对 prototxt、caffemodel 等进行编译。 1：网络过滤器模式，对 prototxt 文件是否符合编译支持规格进行检查。

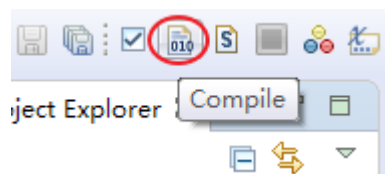
【注意】

UI 下使用字符串表示的选项，在 Text Editor 编辑模式下使用对应数值表示，参考表 3-2 中的说明。

3.6.3.3 编译器运行

在 Project Explorer 视图中选择指定项目需要编译的工程，在 UI 界面上方工具栏上点击编译(Compile)按钮，执行编译。

图3-54 Compile 按钮



3.6.3.4 编译器输出

编译器默认输出如下文件：

- *.wk：编译成功时生成的知识库文件；用户可通过 instructions_name 配置选项更改保存路径和文件名。
- Cnn_net_tree.dot：编译成功时生成的网络结构 dot 描述；保存在工程所在目录中，可显示为 dot 视图，具体参考 3.6.7 “Dot 图” 章节。
- Compiler_error.log：编译器出错时生成的错误日志。
- 当 print_log 配置为 YES（Text Editor 模式下为 1）时，编译器还将在工程目录下输出如下日志，建议用户只在出现错误定位问题时才输出该日志：



- Compiler_debug.log: 编译器运行日志。

3.6.4 仿真器

3.6.4.1 sim.cfg 文件编辑模式

同 check.cfg, sim.cfg 也有两种编辑模式。如图 3-55 到图 3-56。

图3-55 sim.cfg 以 Simulator Configuration Editor 模式打开

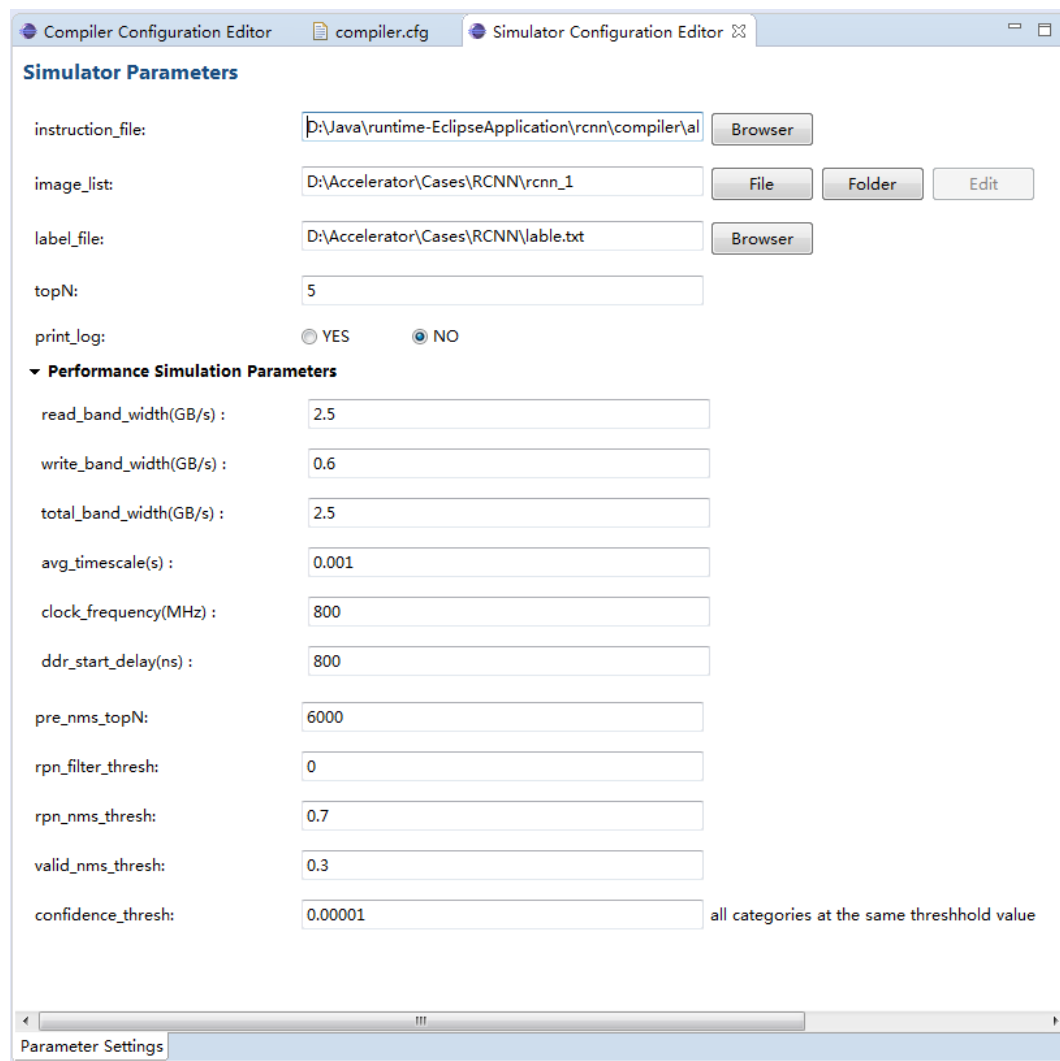




图3-56 sim.cfg 以 Text Editor 模式打开

```
[print_log] 0
[topN] 5
[pre_nms_topN] 6000
[rpn_filter_thresh] 0
[rpn_nms_thresh] 0.7
[valid_nms_thresh] 0.3
[read_band_width] 2.5
[write_band_width] 0.6
[total_band_width] 2.5
[avg_timescale] 0.001
[clock_frequency] 800
[ddr_start_delay] 800
[instruction_file] alex_for_accuracy_test_ok_sim.wk
[label_file] D:\Accelerator\Cases\RCNN\lable.txt
[confidence_thresh] 0.00001
[image_list] D:\Accelerator\Cases\RCNN\rcnn_1
```

3.6.4.2 配置文件说明

表3-4 仿真器配置选项说明

配置选项	取值范围	描述
instruction_file	-	编译器在仿真模式下生成的指令文件*.wk。
images	-	仿真器测试图片。 File 选项要求配置测试图像列表的文本文件； Folder 选项要求配置测试图像的目录；
label_file	-	CNN 网络下，跟 images 对应的分类标记文件。
topN	[1, class_num]	分类结果按置信度排序且取前 N 的结果。 class_num 是模型对应的分类类别数目。
print_log	{0, 1}	设置是否开启日志文件。 0: NO，不打印日志； 1: YES，打印日志。
pre_nms_topN	[100, 20000]	RPN 层保留候选 anchor 进入非极大抑制操作的数目。 默认值：6000。
rpn_filter_thresh	[0, 1)	在 RPN 阶段，会根据该阈值进行候选框的筛选，该参数 主要用来提高 RPN 的运行速度。 默认值：0.01。
rpn_nms_thresh	(0, 1)	在 RPN 阶段，做非极大值抑制的阈值参数。 默认值：0.7。



配置选项	取值范围	描述
valid_nms_thresh	(0, 1)	最终输出候选框前进行非极大值抑制的阈值参数。 默认值：0.3。
confidence_thresh	(0, 1)	对检测的每个类别，过滤掉小于该阈值的候选框。 阈值个数与网络分类类别数一致，参数间用逗号分隔。
read_band_width(GB/S)	(0, DDR 最大读带宽)	DDR 能够提供给 CNNIE8Bit 的平均读带宽。用于仿真数据从 DDR 加载到 CNNIE8Bit 片上 RAM 的时间。 建议值：2.5。 DDR 最大带宽 = DDR bit 位宽 * DDR 频率。
write_band_width(GB/s)	(0, DDR 最大写带宽)	DDR 能够提供给 CNNIE8Bit 的平均写带宽。用于仿真数据从 CNNIE8Bit 片上 RAM 写到 DDR 的时间。 建议值：2.5。
total_band_width(GB/s)	(0, DDR 最大读写带宽)	DDR 能够提供给 CNNIE8Bit 的读写平均总带宽。用于仿真 DDR 与 RAM 之间同时读写的时间。 建议值：2.5。
avg_timescales(s)	(0, 1)	平均带宽限制窗口。 假设 DDR 能提供给 CNNIE8Bit 2.5G/s 的读带宽，平均带宽限制窗口设置为 0.001，表示在 1ms 内，CNNIE8Bit 最多读取 2.5Mbyte 数据量。
clock_frequency(MHz)	(0, 3000)	时钟频率。 建议值：800。
ddr_start_delay(ns)	[0, 10000]	DDR 延时。用于仿真从 DDR 加载数据到 CNNIE8Bit 片上 RAM 的时间。 建议值：800。
top_k	[1, 5000]	SSD 每一个分类的候选框进入 NMS 模块之前保留前 top_k 个 建议值：400
keep_top_k	[1, 2000]	SSD 所有类别的最终选定的框的个数总和不超过 keep_top_k 个 建议值：200
detectionout_nms_thresh	(0, 1)	SSD 在 DetectionOut 阶段，做非极大值抑制的阈值参数。 默认值：0.7。



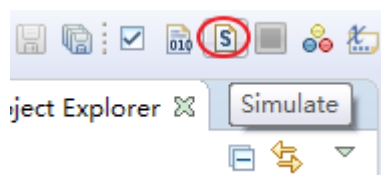
注意

仿真器上的性能、带宽评估值跟真实的硬件会有差异，仅作为参考。

3.6.4.3 仿真器运行

在 Project Explorer 视图中选择指定项目需要仿真的工程，在 UI 界面上方工具栏上点击仿真(Simulate)按钮，执行仿真。如图 3-57 所示。

图3-57 Simulator 按钮



3.6.4.4 仿真器输出

不同的网络类型，仿真器其输出稍有不同。

- CNN 类型网络仿真输出

默认情况下在工程目录下输出如下文件：

- topN.txt: 分类 top1 和 topN 准确率统计及每张图的 topN 概率输出；
- ieva.txt: 网络各层性能、带宽估计；
- simulator_error.log: 仿真器错误日志。

当 print_log 配置为 YES (Text Editor 模式下为 1) 时，仿真器还将在工程目录下输出如下日志，建议用户只在出现错误定位问题时才输出该日志：

- simulator_debug.log: 仿真器运行日志。

- Faster RCNN、iFDT 类型网络仿真输出

- ieva.txt: 网络各层性能、带宽估计；
- Faster_rcnn_result_*/iFDT_result_*: *代表的数字，对应模型分类类别编号，保存该类别下各图像检测得到的目标框的概率和位置。
- simulator_error.log: 仿真器错误日志

当 print_log 配置为 YES (Text Editor 模式下为 1) 时，仿真器还将在工程目录下输出如下日志，建议用户只在出现错误定位问题时才输出该日志：

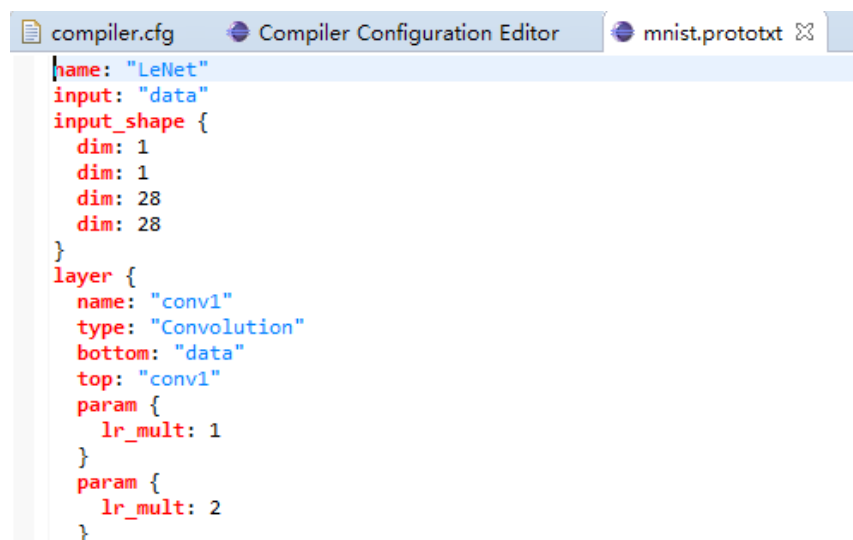
- simulator_debug.log: 仿真器运行日志

3.6.5 prototxt 编辑

在网络过滤器和编译器的 prototxt 参数配置选项中有 Edit 按钮，点击后可以对 prototxt 进行编辑、保存。



图3-58 Prototxt 编辑



编辑 prototxt 文件时，若发现内容格式不规范，可通过右键→Format 来对 prototxt 文件进行内容格式规范化。

图3-59 Prototxt Format 选项

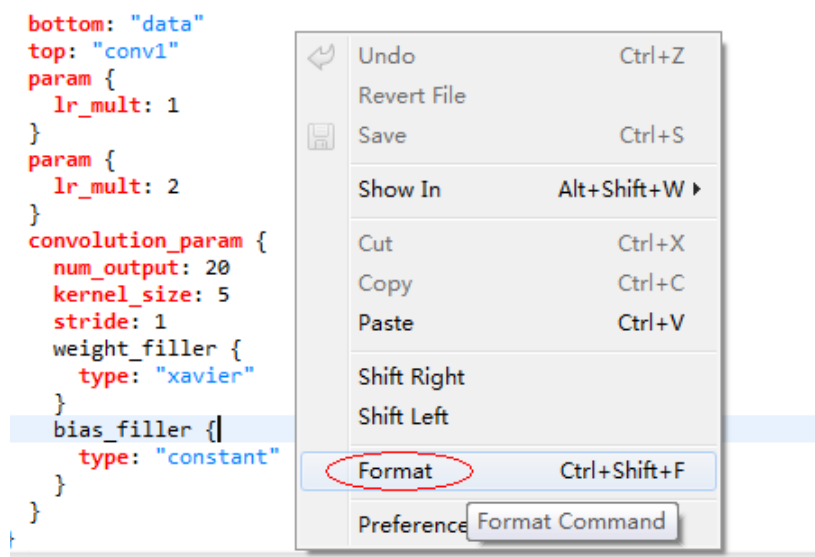


图3-60 Prototxt 格式化前后对比

```
name: "LeNet"
input: "data"
input_shape {
  dim: 1
  dim: 1
  dim: 28
  dim: 28
}

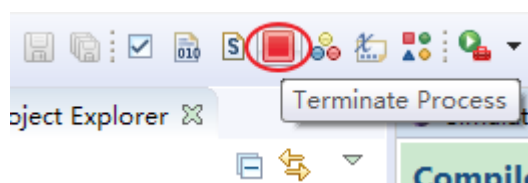
name: "LeNet"
input: "data"
input_shape {
  dim: 1
  dim: 1
  dim: 28
  dim: 28
}
```

用户可以通过偏好设置页面来配置 prototxt 文件编辑器，包括格式化规范、关键字颜色、代码提示等，具体参考 3.6.10 “偏好设置” 章节。

3.6.6 终止

当前版本中，同一时间只能运行一个编译进程或仿真进程。用户在编译、仿真过程中可通过点击工具栏中的终止(Terminate)按钮来结束编译或仿真工作。当编译或仿真工作正在进行时，Terminate 按钮可点击，显示为红色；终止任务或任务运行结束后，按钮不可点击，显示为灰色。

图3-61 Terminate 按钮

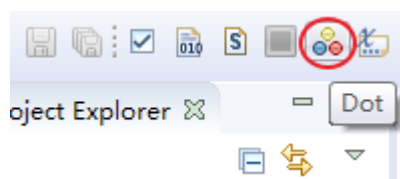


3.6.7 Dot 图

网络过滤器和编译器生成的 cnn_net_tree.dot 文件，可以生成可视化 Dot 图。

步骤 1. 点击 Dot 按钮将视图打开。

图3-62 Dot 按钮





步骤 2. 双击打开“cnn_net_tree.dot”文件，Dot View 会自动获取文件内容并绘制 Dot 图。如果 dot 节点较多，由于界面有限，节点可能出现重叠的情况，可以使用鼠标拖动，以达到想要的效果。

图3-63 Dot 图效果

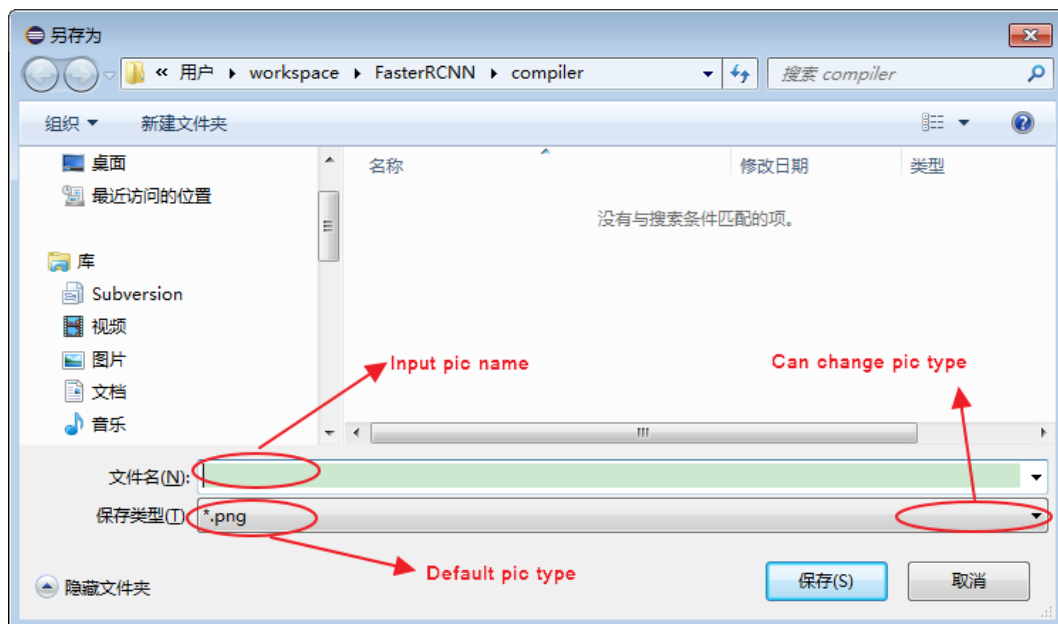


步骤 3. 点击视图右上角的 Export as image file 按钮可以将 Dot 图以*.png 或其他图片格式保存到指定路径；

图3-64 保存 Dot 图



图3-65 保存 Dot 图路径及文件名设置

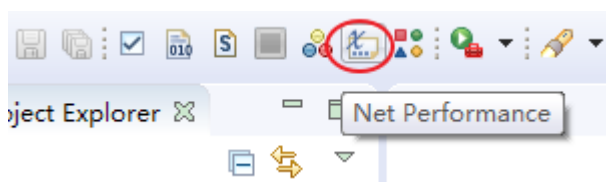


3.6.8 性能分析

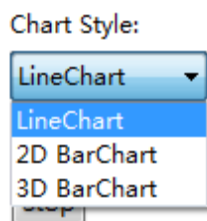
仿真器生成的文件 ieva.txt 文件，通过 Net Performance 视图生成可视化性能分析图。

- 步骤 1. 点击 Net Performance 按钮打开性能分析视图。性能分析会使用到仿真结果中的“ieva.txt”文件。

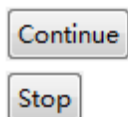
图3-66 性能分析按钮



- 步骤 2. 点击 ChartStyle 下面伸缩菜单栏，选择性能展示方式



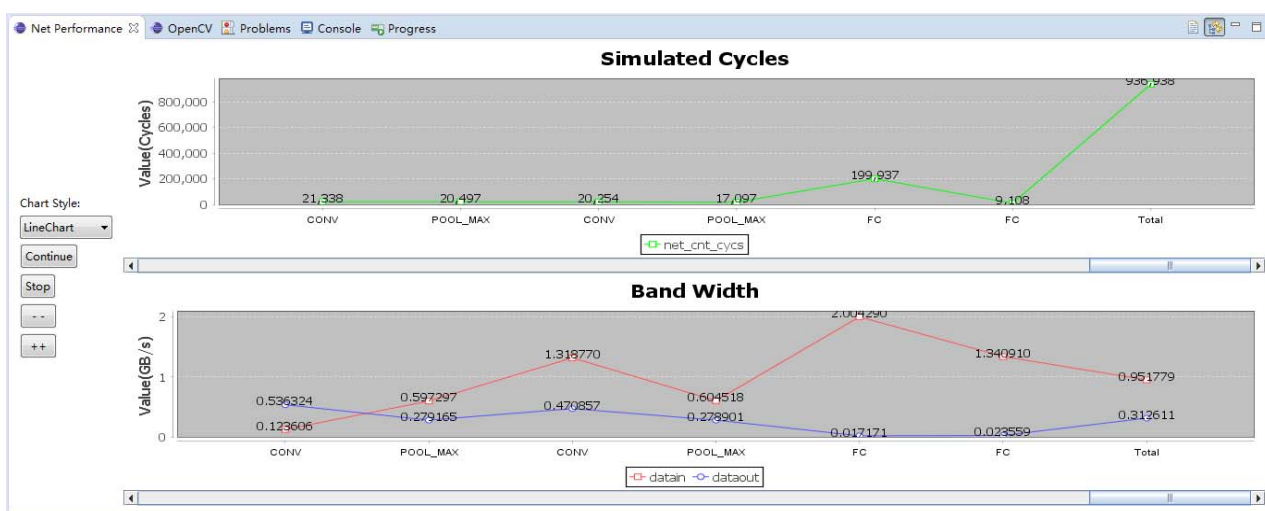
- 步骤 3. 可以通过点击 continue 或 stop 按钮实现暂停或继续运行控制



步骤 4. 点击 - 或 + 按钮可以实现对性能分析视图的缩放



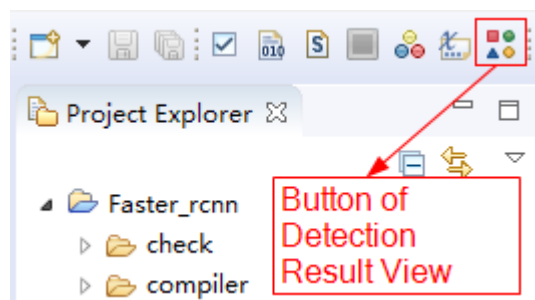
图3-67 性能分析结果



3.6.9 目标检测

目标检测视图用来查看框选结果。用户需将光标移动至任意工程，系统会检测该工程仿真时使用的图片并在视图上显示；同时，系统会自动检测该工程下的 **simulator** 目录，查看是否有能够解析的坐标文件，若有，则将坐标文件列表显示在视图上。

步骤 1. 点击 “Detection Result” 按钮，打开 Detection Result 视图。





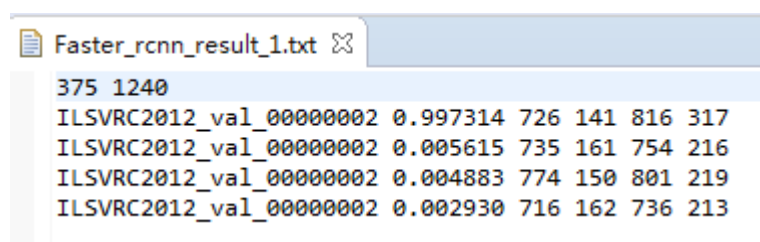
步骤 2. Detection Result 视图提供了三个按钮，分别用于选择图片、上下切换图片，以及图片切换列表。

图3-68 目标检测示图



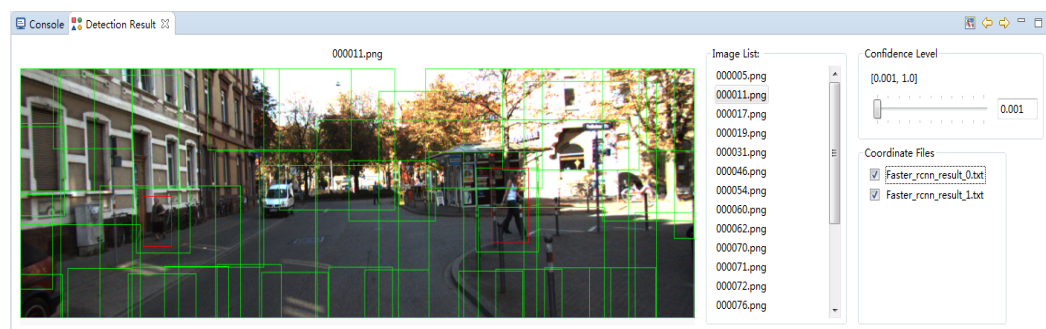
步骤 3. 系统会自动检索工程下 simulator 目录中的可解读坐标文件。坐标文件首行表示图片缩放到模型输入要求的像素大小，从第二行开始，每行由 6 列组成，每列分别对应图片名、置信度、选框左上角 x 坐标、选框左上角 y 坐标、选框右下角 x 坐标、选框右下角 y 坐标）

图3-69 坐标文件



步骤 4. 选择图片后勾选文件列表，左边的图像会显示出置信度大于等于 confidence level 文本框中阈值的矩形框。滑块可以调整矩形框置信度显示阈值，文本框可以手动调整阈值。

图3-70 目标检测结果



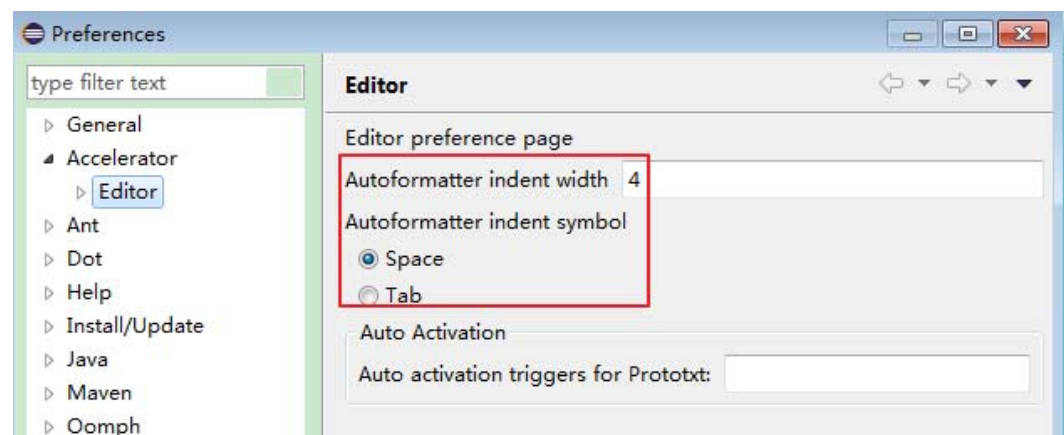
----结束

3.6.10 偏好设置

3.6.10.1 代码格式

依次选择菜单栏 Windows→Preferences→General→Keys，设置代码格式化缩进字符数和缩进标志。

图3-71 代码格式

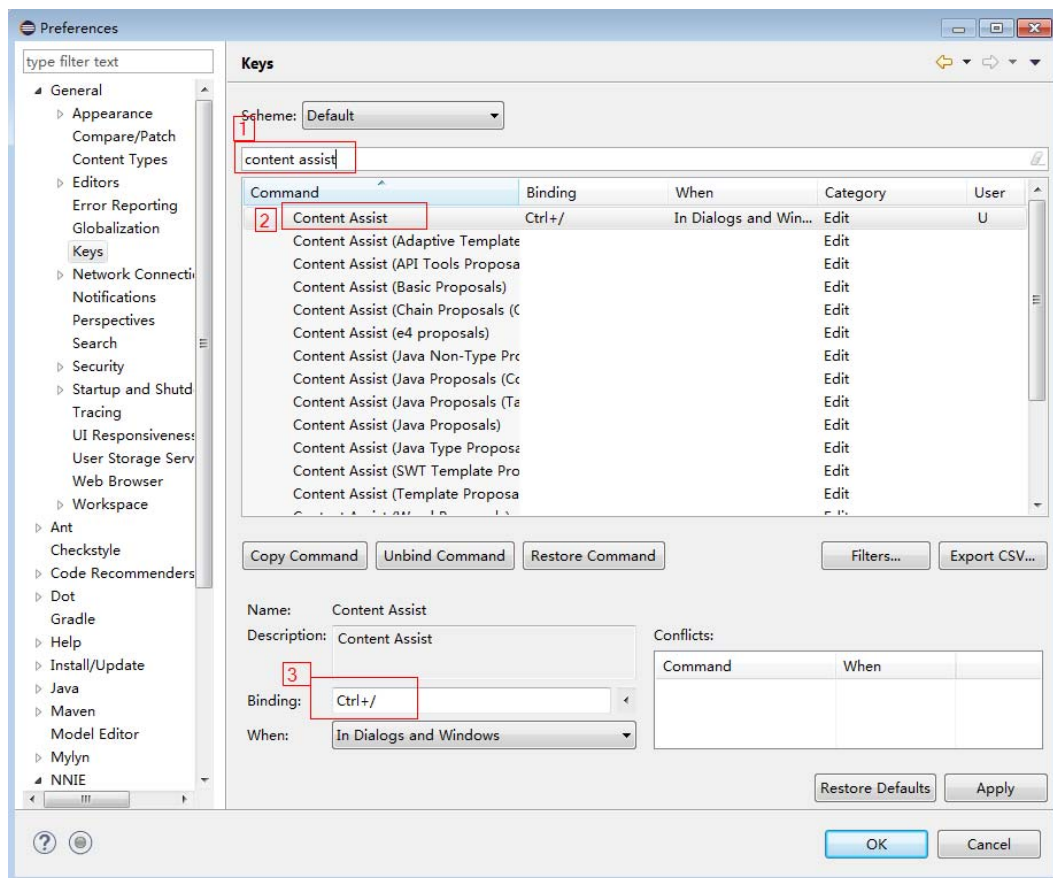


3.6.10.2 代码提示

步骤 1. 依次选择菜单栏 Windows→Preferences→General→Keys

步骤 2. 使用关键字 content assist 进行查找或替换快捷键，就会弹出内容助手窗口，在 Binding 上按下用户自己习惯的快捷键，如图 3-72，内容助手会自动根据光标前面的字符配出相应的关键字。

图3-72 代码提示快捷键设置



步骤 3. 选择依次菜单栏 Windows→Preferences→Accelerator→Editor，设置自动提示首字符，可同时设置多个字符，如图 3-73。

图3-73 设置自动提示首字符

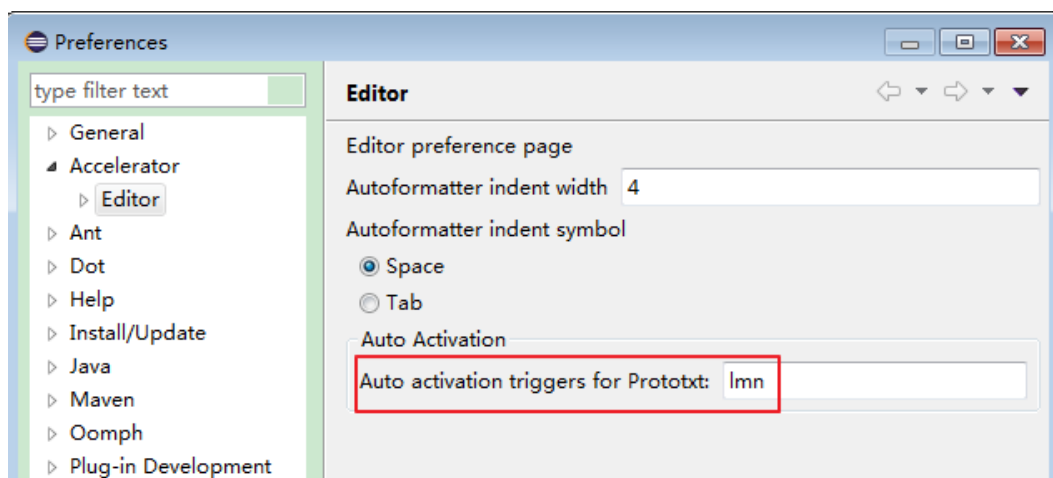
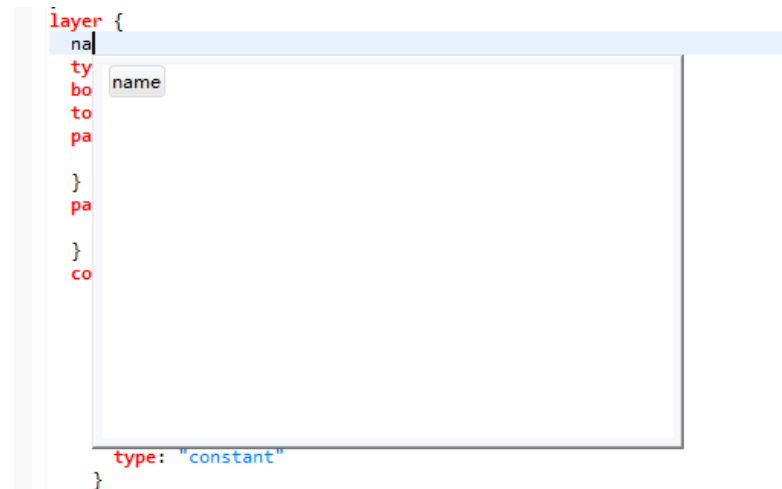


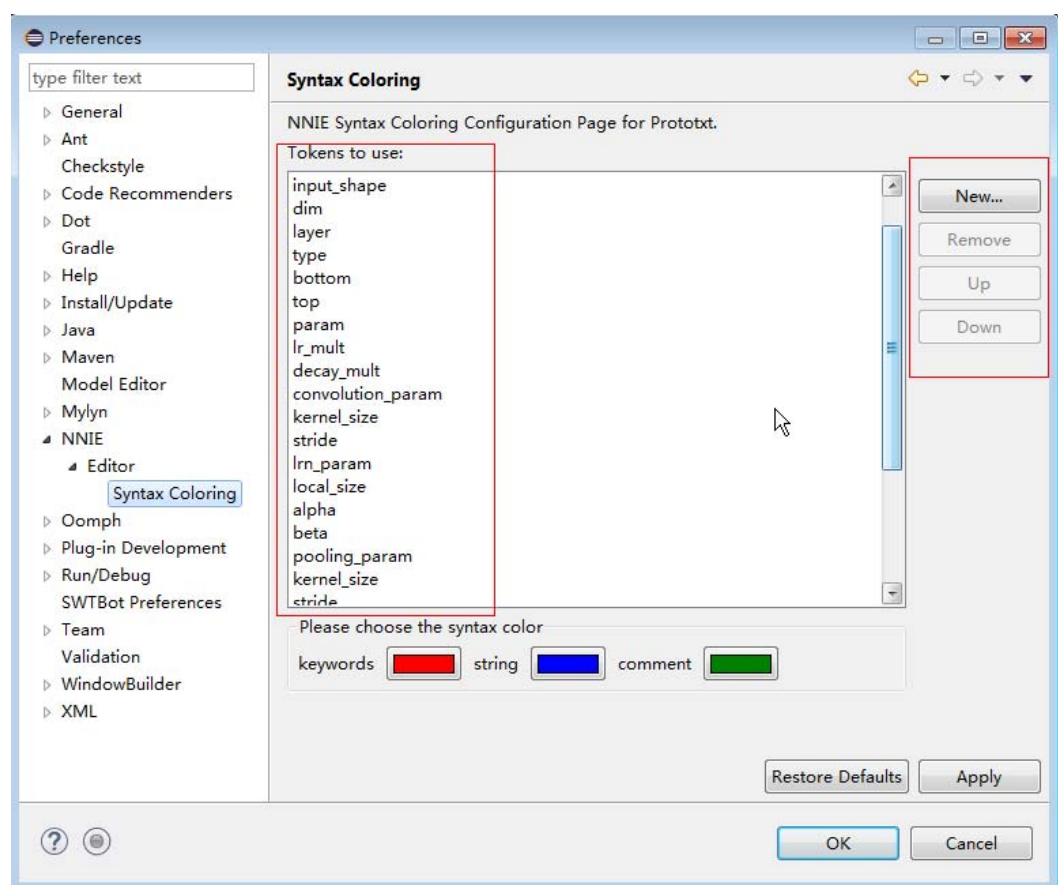
图3-74 代码提示效果



3.6.10.3 Prototxt 内容颜色设置

- 步骤 1. 依次选择菜单栏 Windows→Accelerator→Editor→Syntax Coloring，设置关键字列表、关键字、字符串、注释的字体颜色，如图 3-75。

图3-75 颜色设置





3.7 仿真库使用

HiSVP_PC_Vx.x.x.x.rar 中提供了仿真库，其使用说明参考《HiSVP API 参考》。



注意

HiSVP_PC_Vx.x.x.x.rar 中的仿真库不仅包含了 NNIE 的部分，也包含了 OpenVX 的框架和 DSP 的算子。

3.8 FAQ

3.8.1 软件运行期间报错 “periodic workspace save failed”

此报错对 NNIE 业务的编译和仿真不会造成任何影响，大多数是由于用户非法使用工作空间造成的。建议更换工作空间，且每个工作空间不要放置过多工程。在软件的使用过程中，不要在工程中添加额外的文件，每个 accelerator 工程就是如工程创建时所得到的目录结构。

3.8.2 软件运行崩溃，无法启动

使用不当导致 eclipse 软件无法启动，建议删除现有解压包，从发布版本中重新解压运行软件。

3.8.3 工具栏没有显示

关闭欢迎页。若依旧没有显示工具栏的话，菜单栏点击 Window->Appearance->Show Toolbar 来显示工具栏。

