MobileNet\_VC709 Design specification

**v1.0 2021/8/19**  **spec writer 赵忠宇（澳门大学）**

设计目标

本项目目标为搭建一个部署在FPGA上的图像识别与分类系统，该系统能够读取摄像头或片外DRAM的图像信息，经过预处理后输入神经网络模型，完成inference操作后，通过HDMI协议将原图像与分类结果打印至显示器。

硬件平台

VC709 Connectivity Kit

· Virtex-7 VX690T

其资源总数见图[1].

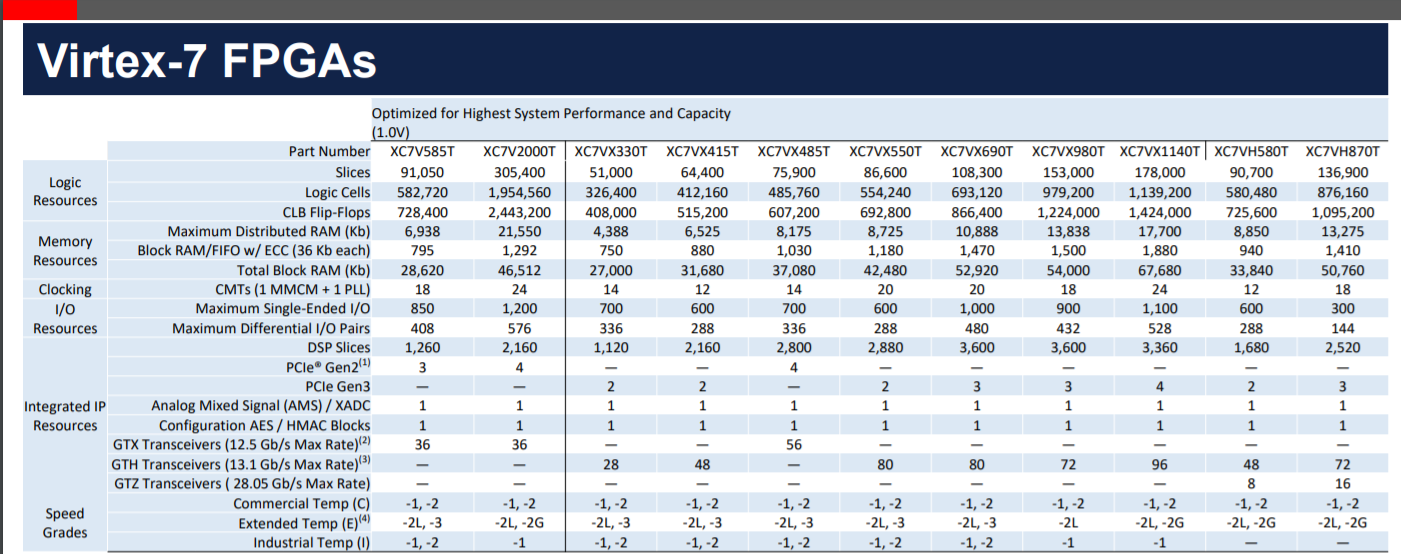


图 1 Virtex-7 系列芯片资源数目统计

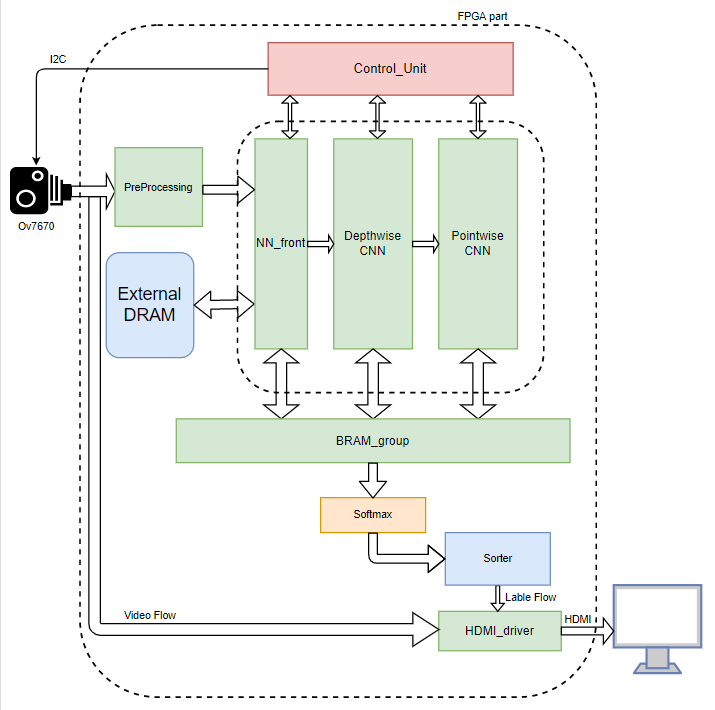
1960\*1024液晶HDMI显示屏

Ov7670摄像头模块

功能分解

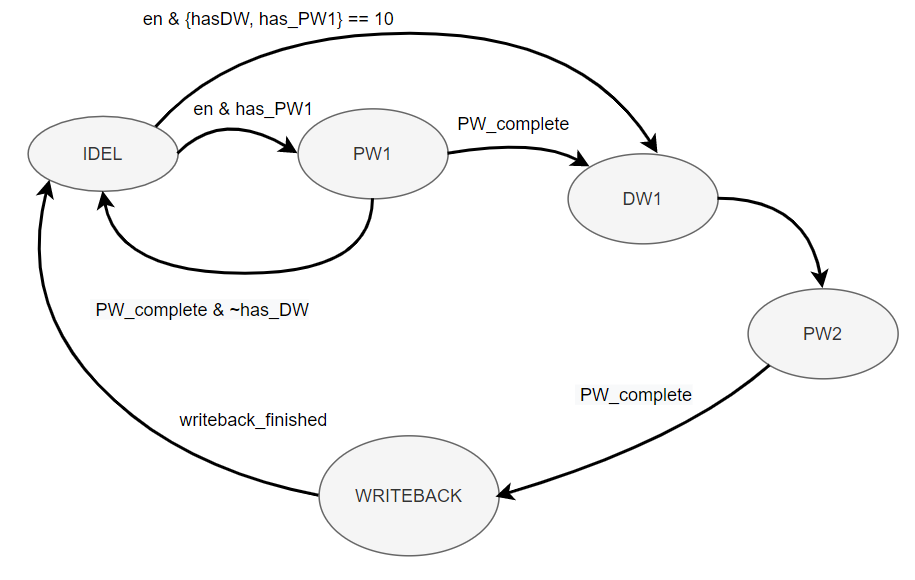
1. 系统在上电后，固化好的bit流能自动从Flash中下载至FPGA，对FPGA进行编程。
2. 系统分为DRAM工作模式与摄像头工作模式。DRAM工作模式下，系统从DRAM的指定位置，读取待inference的图片，以流的形式输入神经网络的输入层，之后启动神经网络，完成inference，将图片与结果共同显示在屏幕上。摄像头工作模式下，上电后系统对摄像头进行初始化，读取摄像头输入的视频流，之后分成两个线程，一个线程对输入帧进行预处理并以流的形式存入神经网络的输入层，神经网络处于连续inference模式，得出lable。另一个线程对视频流进行输出，lable的字符串像素与输出视频流进行merge后显示在屏幕上。
3. 神经网络的输入层为流输入形式。由于MobileNet第一层为普通3\*3卷积，故第一层NN\_front独立于其他层进行单独实现。该层的主要目的为将以流传输的数据流转化为以package存入BRAM中的数据流
4. MobileNet V2的主要部分为Inverse Residual Block（IRB），所有IRB层都由一个可配置的模块进行inference计算，该模块对输入值的内存读取模式与对输出值的内存写入模式相同，故可以在一个模块内连续进行多次的IRB层运算。该模块通过配置控制寄存器来实现对不同层的操作。
5. MobileNet V2的输出值经过softmax模块存入输出BRAM组中。该BRAM组将被HDMI显示模块调用，排序后将分类的前五名打印在屏幕上。

系统架构图

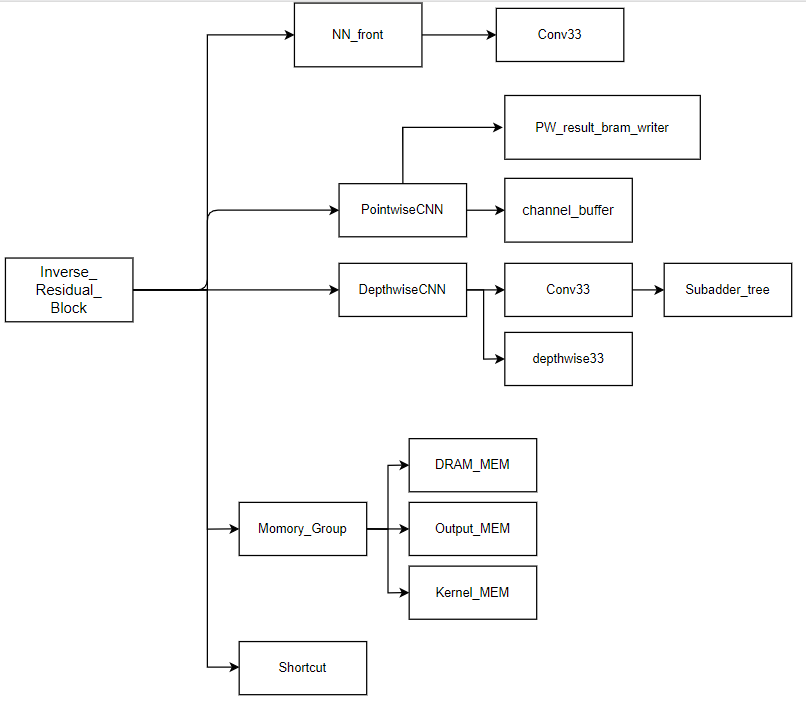


Inverse\_Residual\_Block Design Spec.

在接收到Control\_Unit的命令后，Inverse\_Residual\_Block模块状态机转化示意图为：



依照其功能设定，对其模块层次做如下安排：



1. DepthwiseCNN spec.

该模块用于对BRAM中存储的ifmap做depthwise卷积操作。其可配置的参数定义如下：

1. Ifmap\_size: 用于指示输入特征矩阵的尺寸
2. Ifmap\_channel：用于指示输入特征矩阵的通道数
3. stride：指示步长 0：stride = 1 1: stride =2
4. ena：状态机使能信号，当完成当下ifmap的所有运算后，应将ena置0。
5. Ifmap\_base\_addr：存储当前Ifmap的基地址
6. ifmap\_group\_sel：Ifmap输入输出组别选择。 **0：1组输入，2组输出1：1组输出，2组输入**
7. Weight\_base\_addr：存储当前weight的基地址
8. Bias\_base\_addr：存储当前bias的基地址
9. Ofmap\_base\_addr：存储Ofmap输出的目标地址

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **寄存器名** |  | | | | | |
| **DW\_config** | **[7:0]: ifmap\_size**  **输入特征图像尺寸** | **[17:8]: ifmap\_channel，**  **输入特征图像通道** | **[28:18]: ofmap\_channel,**  **输出特征图像通道** | **[29]: stride,**  **步长** | **[30]: ena**  **使能信号** | **[31]: conv**  **是否为第一层的普通卷积** |
| **Input\_base\_addr** | **[18:0]: ifmap\_base\_addr**  **输入特征图像基地址** | **[30:19]: Weight\_base\_addr**  **输入权重基地址** | | **[31]: ifmap\_group\_sel**  **Data\_MEM输入输出组配置，**  **0：1组输入，2组输出**  **1：1组输出，2组输入** | | |
| **Output\_base\_addr** | **[18:0]: Ofmap\_base\_addr**  **卷积结果输出基地址** | **[30:19]: Bias\_base\_addr**  **输入偏置基地址** | | | | |

Feature Map在BRAM中的存储方式见图2，存储一个Feature Map需要使用3个BRAM模块，Feature Map的行在三个BRAM中交叉存储，每三行称为一个raw group，每一行中的每一个点包含其所有channel的数据。

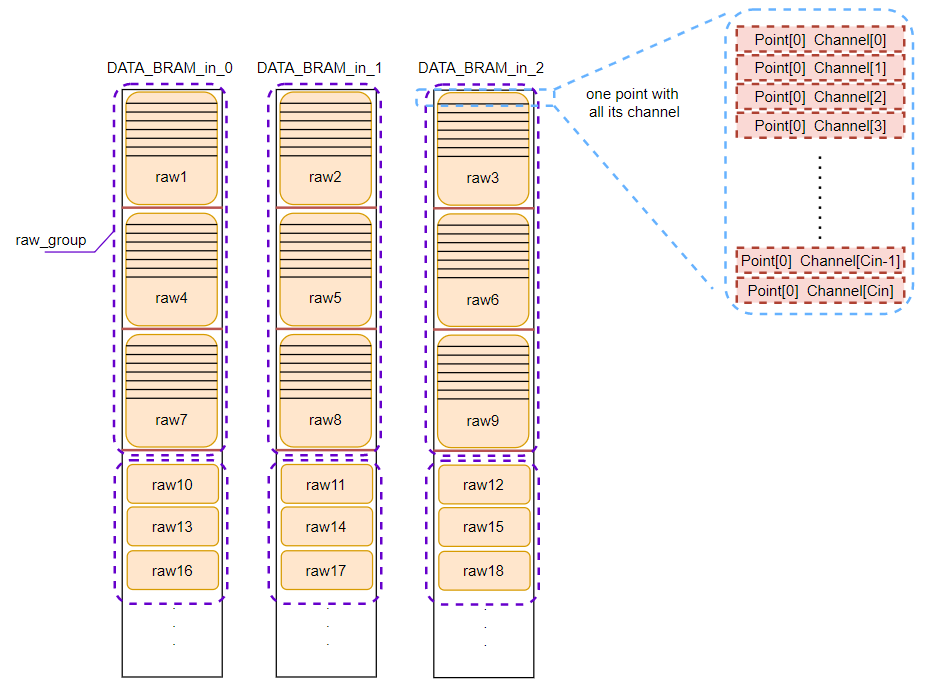


图 2 Feature Map在BRAM中的存储方式

Depthwise模块在计算ofmap中的一个点时，需保证成功交付后，才能再进行下一个点的计算。所谓交付，是指当前depthwise计算后的点将直接参与后续pointwise的计算，只有当后续操作进行完毕后，depthwiseCNN模块接受到conv\_channel\_calculation\_finishing （CCCF）信号后，才进行下一个通道的计算。

1. PointwiseCNN spec.

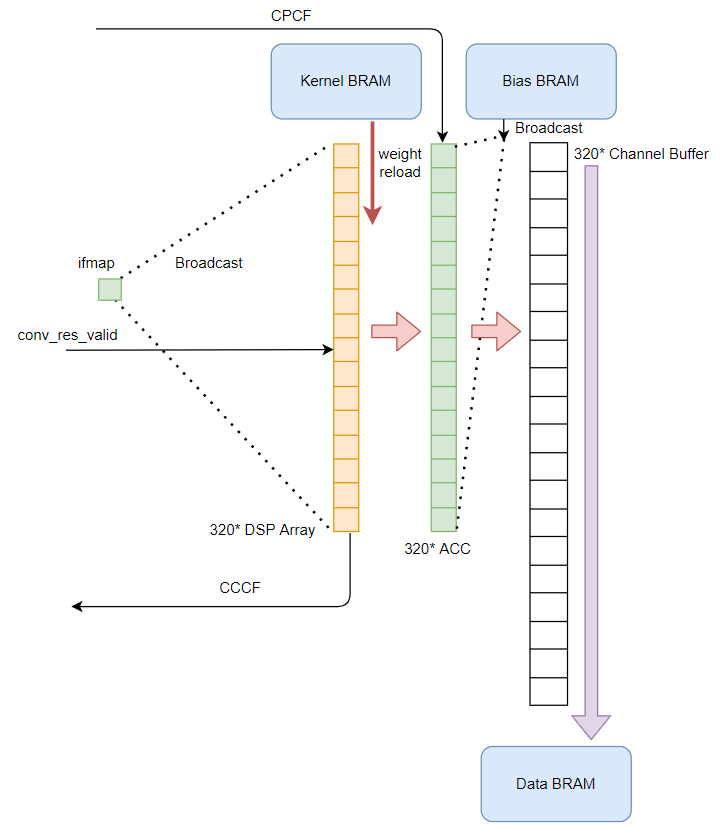
该模块用于完成pointwise与FC层的操作，可配置参数定义如下：

1. Mode: 工作模式： 0：IDLE模式 1：PW模式， 2：DLM模式， 3：FC模式
2. （PW模式下调用）ifmap\_size
3. （PW模式下调用）ifmap\_channel
4. Ofmap\_channel，输出特征图像的通道数目
5. （PW模式下调用）ifmap\_base\_addr，输入特征图像基地址
6. （PW模式下调用）Weight\_base\_addr, 输入权重基地址
7. Ofmap\_base\_addr，卷积结果输出基地址
8. （PW模式下调用）Bias\_base\_addr，输入偏置基地址

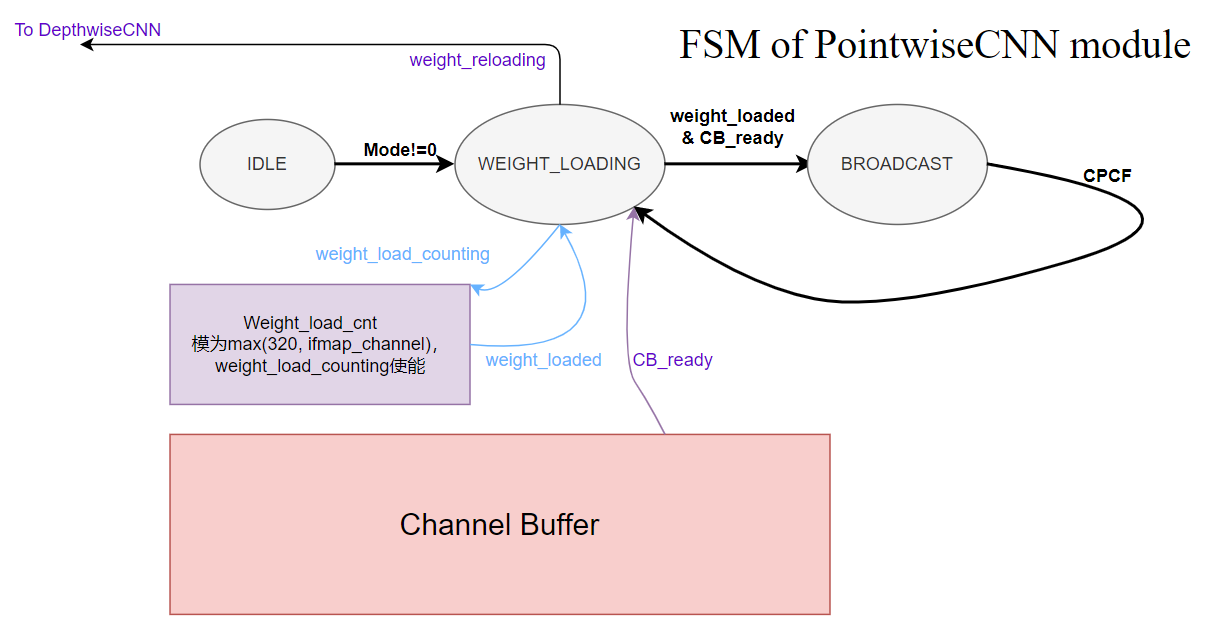
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **寄存器名** |  | | | | |
| **PW\_config** | **[7:0]: ifmap\_size**  **输入特征图像尺寸** | **[18:8]: ifmap\_channel，**  **输入特征图像通道** | **[29:19]: ofmap\_channel,**  **输出特征图像通道** | | **[31:30]: Mode,**  **工作模式** |
| **Input\_base\_addr** | **[18:0]: ifmap\_base\_addr(data)**  **输入特征图像基地址** | **[30:19]: Weight\_base\_addr (group\_select)**  **输入权重基地址** | | | **[31]: ifmap\_group\_sel**  **Data\_MEM输入输出组配置，**  **0：1组输入，2组输出**  **1：1组输出，2组输入** |
| **Output\_base\_addr** | **[18:0]: Ofmap\_base\_addr**  **卷积结果输出基地址** | **[30:19]: Bias\_base\_addr**  **输入偏置基地址** | | **[31]: has\_SC**  **存在shortcut层** | |

PointwiseCNN模块架构如下，其由一个320\*DSP阵列、320\*ACC阵列与320\*Channel Buffer组成，功能分解如下：

1. PointwiseCNN模块可配置为三种工作模式，第一种为PW模式，负责完成PW1卷积，在该模式下，ACC阵列清0的同时，Bias BRAM输出的bias将被broadcast到所有的ACC中；PointwiseCNN直接向BRAM组输出地址与使能信号获得数据流，对于ofmap channel大于320的情况，DSP Array需重新装载weight。规定，只有当当前frame计算完毕之后进行weight的装载，装载接下来320个ofmap channel对应的weight值，所有计数器重新扫当前ifmap的各个point。weight装载过程中输出weight\_reloading用于暂停地址产生计数器；第二种为DLM模式，负责完成double layer MAC中的PW2卷积，同样在weight装载期间，需输出weigth\_reloading用于打断前级DepthwiseCNN的流水线与状态机；第三种为FC模式，负责完成FC层运算，在该模式下，weight与bias同时进行装载。
2. DLM模式下，前级输入准备好后，发送conv\_res\_valid信号，此时若DSP Array已完成weight的装载，则输入将被broadcast到DSP Array中所有的DSP模块里，与其中的weight相乘，在一个时钟周期后，结果存入后级ACC(Accumulator) 模块，当接受到CPCF信号时，表明前一级一个point的所有channel全部输入完毕，此时ACC中的所有值向后缓存至320\*Channel Buffer模块中，ACC阵列清零。下一个周期中，ACC模块在计算下一个point的同时，Channel Buffer发出PWOV(Pointwise Output Valid)信号，向输出BRAM中或SC模块输出上一个point的计算结果，如此实现流水线化处理。
3. 对于PW2而言，ifmap的channel往往比ofmap的channel大很多，故一般在上一个point的所有ofmap channel输出完毕后，下一个point的ACC阶段还未结束，故当下一个point ACC阶段结束后可以放心地向Channel Buffer写入值。但我们希望该模块能被PW1以及FC模块复用，故存在ofmap channel大于ifmap channel的层，将这种情况称为**输出缓冲延迟**。解决输出缓冲延迟的方法，在于将ifmap输入数据流与ofmap输出数据流分开。由于load的weight数等于Channel Buffer一次性可以输出的channel数，故可以当weight\_load\_buffer装载weight的同时，Channel\_buffer同时进行ofmap channel的卸载（disboarding）。待ofmap把存在Channel Buffer中的point的所有channel均完成写入，并将当前ACC中的point的所有channel缓存完毕，ACC清零，DSP Array装载完新的weight值后，输入下一个point的数据流。



1. 综上，可以总结PointwiseCNN状态机如下：



**输入信号定义：**

* 从DepthwiseCNN输入：

Channel\_cnt：在DLM模式中用于指示当前正在计算的ifmap channel索引

conv\_res\_valid：在DLM模式中表示当前channel的DW计算结果值已经可以取用

conv\_res\_input：DW计算结果输入

CPCF：用于表示当前ifmap的所有channel全部处理完毕。CPCF经过一定时间的延时生成CPCF\_delay信号，当接收到该信号时，系统将发生两种变化：一：DSP Array中的weight值将被重载；二：ACC中的所有值被存入Channel Buffer。

* 从Control\_Unit输入：

PW\_config\_input：PointwiseCNN模块用户定义寄存器配置数据

PW\_config\_valid：PointwiseCNN模块用户自定义寄存器配置使能

* 从BRAM输入：

Bram\_data\_input(PW与FC模式)

Bram\_weight\_input

Bram\_bias\_input

**输出信号定义**

weight\_reloading：用于暂停计数器及状态机数据流，为应对输出缓冲延迟问题，当CB\_ready信号为0时，即当前Channel Buffer写入未完成时，该信号同样为1。

Bram\_data\_en：用于DATA\_BRAM使能，在PW与FC模式下，所有BRAM使能

Bram\_weight\_en：用于Kernel\_BRAM使能

Bram\_bias\_en：用于Bias\_BRAM使能

CCCF：用于表示当前channel全部处理完毕，DW可开始处理下一个channel

PWOV(Pointwise Output Valid)：用于指示当前Channel Buffer可以输出有效数据。

**内部信号定义：**

CB\_ready：Channel Buffer写入完毕信号，由内部计数器Channel\_buffer\_cnt产生。

channel\_load：

weight\_loaded：指示weight以装载完毕。

ofmap\_round\_channel：初始化为ofmap\_channel，OCR计数器自增后，ofmap\_round\_channel = ofmap\_channel-320

frame\_end：计数完一个frame后，用于清零CVLOC

**计数器组定义：**

由于在PW模式下，逐点完成ifmap的pointwise卷积，故需要与DW相同的计数器时序，故需要添加计数器Ifmap\_hori\_base\_cnt(IHB)，Ifmap\_vert\_base\_cnt(IVB)与Channel\_cnt (CC)，另外为了产生针对Data BRAM存储模式的地址流，需要Conv33\_local\_vert\_cnt(CVLOC)作为row group的索引。此外，需要Weight\_load\_cnt（WL）来指示weight已装载完毕， Buffer\_cnt（BC）用于表示buffer中的ofmap已全部输出。对于ofmap channel大于320的情况，需要各个计数器反复扫过整个frame，这需要引入一个新的计数器：Ofmap\_channel\_round\_cnt(OCR)，其模等于ofmap channel/320。另外，当处于DLM模式下时，只使能WL、BL与OCR计数器；FC模式下时，使能CC、WL、BL与OCR计数器。

故个计数器行为描述总结如下：

1. IHB

使能：BROADCAST阶段，Mode==1

自增条件：CC计满

模：ifmap\_size

归零条件：IHB计满且CC计满

1. IVB

使能：BROADCAST阶段，Mode==1

自增条件：IHB计满且CC计满

模：ifmap\_size

归零条件：IVB计满且IHB计满且CC计满

1. CC

使能：(Mode==1||MODE==3) && BROADCAST

自增条件：

1clk自增一次

模：ifmap\_channel

归零条件：跳出BROADCAST状态归0

1. CVLOC

使能：Mode==1&& BROADCAST

自增条件：IHB计满且CC计满

模：3

归零条件：CVLOC计满且IHB计满且CC计满

1. WL

使能：WEIGHT\_LOADING

自增条件：1clk自增一位

模：ofmap\_round\_channel

归零条件： WEIGHT\_LOADING状态结束

1. OCR

使能：Mode==1||Mode==3

自增条件：IVB计满且IHB计满且CC计满

模：ofmap\_ channel /320

归零条件：OCR计满且IVB计满且IHB计满且CC计满

**BRAM访存模式定义：**

定义subaddress如下：

bram\_row\_subaddress：用于指示当前bram输出的行idex

bram\_point\_subaddress：用于指示当前输出的point，

bram\_channel\_start\_address: 用于指示当前输出的channel组号

bram\_channel\_address：用于指示当前输出的channel index

**BRAM结果写回策略：**

实现PW\_result\_bram\_writer模块，该模块用于接受向BRAM的特定地点写入ofmap数据，检测到bram\_write\_back信号时，即开始进行写回，模块内部拥有计数器，用于产生写回地址。

1. Short\_cut spec.

可将Short\_cut模块设计为一个可配置深度的FIFO用于缓存ifmap，问题在于，我们一次缓存多少的数据呢，是缓存ifmap全部的内容还是仅缓存一个point连同它所有的channel？由于SC操作仅存在于PW2层之后，PW2是在一个特定点依次按channel进行读取，Channel Buffer也是一次性输出一个point的所有channel，故可在PointwiseCNN模块后跟一个判断是否经过shortcut层的逻辑，若经过，则对PW\_result\_bram\_writer进行配置，其用于结果写入的各个计数器同样用于产生SC ifmap的通道数据流。如此，Short\_cut模块共享PW\_result\_bram\_writer的计数器时序，也不用设计为一个FIFO了。

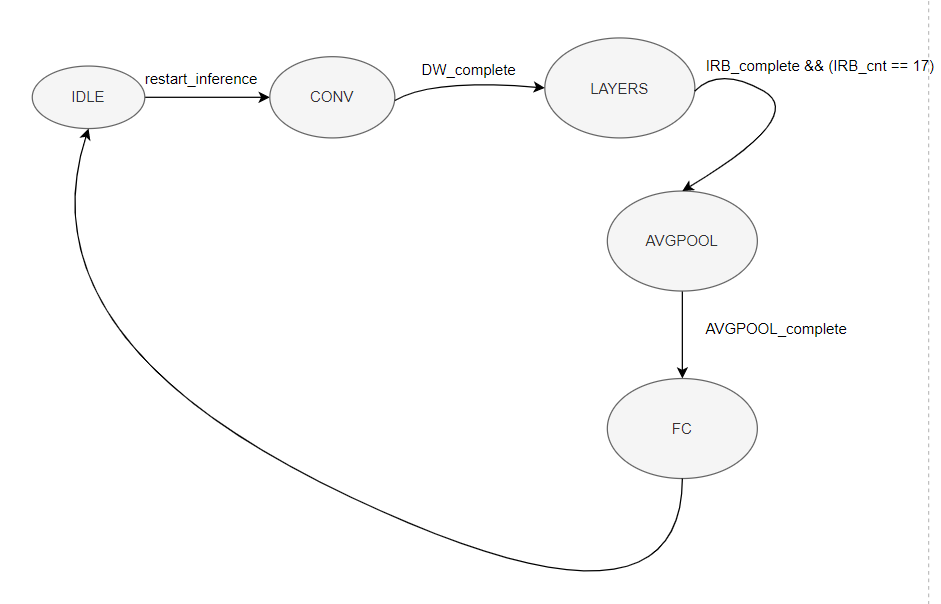
寄存器配置

|  |  |  |
| --- | --- | --- |
| **寄存器名** |  | |
| **SC\_config** | **[18:0]: ifmap\_base\_addr(data)**  **输入特征图像基地址** | **[19 ]: ena**  **SC使能信号** | |

1. Control\_Unit spec.

Control\_unit模块作用为，记录各个模块当前的状态，并依照MobileNet V2各层的次序配置各个硬件模块的寄存器，使硬件模块依次完成MobileNet V2网络各个层的forward propagate过程。

拟将Control\_unit设计为一个状态机，其中LAYERS状态对应一个计数器IRB\_cnt，用于计数当前正在进行inference的layer。状态机示意图如下：



我们希望Inverse\_Residual\_Block能通过合理地配置参数完成MobileNet V2所有的操作，故综合起来Inverse\_Residual\_Block需要从Control\_Unit获得的信息如下：

1. hasDW，判断当前操作是否为单个PW1层
2. hasPW1，判断当前操作是否存在PW1层
3. hasSC，判断当前操作是否含有SC操作，若有，DW层Stride为1，没有则判断hasPW1是否为0，为0则说明当前运行的是layer1，stride设为1，否则stride设为2。
4. PW1.fmin，PW1层的ifmap size，若hasPW1为0，则该参数自动流用为DW1层的ifmap size
5. PW1.Cin，PW1层的ifmap channel，若hasPW1为0，则该参数自动流用为DW1层的ifmap channel
6. PW1.Cout，PW1层的ofmap channel，若hasPW1为0，则该参数disable
7. PW2.Cout，PW2层的ofmap channel

**DRAM工作模式下Control Unit对各层的配置：**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 层数 | hasDW | hasPW1 | hasSC | PW1.fmin | PW1.Cin | PW1.Cout | PW2.Cout |
| 0(CONV) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 112 | 32 | 0 | 16 |
| 2 | 1 | 1 | 0 | 112 | 16 | 96 | 24 |
| 3 | 1 | 1 | 1 | 56 | 24 | 144 | 24 |
| 4 | 1 | 1 | 0 | 56 | 24 | 144 | 32 |
| 5 | 1 | 1 | 1 | 28 | 32 | 192 | 32 |
| 6 | 1 | 1 | 1 | 28 | 32 | 192 | 32 |
| 7 | 1 | 1 | 0 | 28 | 32 | 192 | 64 |
| 8 | 1 | 1 | 1 | 14 | 64 | 384 | 64 |
| 9 | 1 | 1 | 1 | 14 | 64 | 384 | 64 |
| 10 | 1 | 1 | 1 | 14 | 64 | 384 | 64 |
| 11 | 1 | 1 | 0 | 14 | 64 | 384 | 96 |
| 12 | 1 | 1 | 1 | 14 | 96 | 576 | 96 |
| 13 | 1 | 1 | 1 | 14 | 96 | 576 | 96 |
| 14 | 1 | 1 | 0 | 14 | 96 | 576 | 160 |
| 15 | 1 | 1 | 1 | 7 | 160 | 960 | 160 |
| 16 | 1 | 1 | 1 | 7 | 160 | 960 | 160 |
| 17 | 1 | 1 | 0 | 7 | 160 | 960 | 320 |
| 18（调用AVGPOOL） | 0 | 1 | 0 | 7 | 320 | 1280 | 0 |
| 19（FC） | 0 | 0 | 0 | 1 | 1280 | 1000 | 0 |

此外还有：

8．Ifmap.base.addr，

9．Ofmap.base.addr，

10．DW1.kernel.base.addr,

11．DW1.bias.base.addr，

12．PW1.kernel.base.addr, 若处理到FC层（即hasDW、hasPW1、hasSC皆为0）时，该参数自动流用为FC层的权值基地址。

13．PW1.bias.base.addr，若处理到FC层（即hasDW、hasPW1、hasSC皆为0）时，该参数自动流用为FC层的偏置基地址。

14．PW2.kernel.base.addr,

15．PW2.bias.base.addr，

**Control Unit配置方法：**

Control\_unit的配置信号线共有三条，1：32bits的module\_config； 2：module\_config\_valid； 3：module\_config\_addr；

当module\_config\_valid，Control\_unit将由module\_config\_addr指示的目标寄存器写入module\_config的值。

各目标寄存器地址列表：

|  |  |  |
| --- | --- | --- |
| 寄存器名 | 模块基地址 2bits | 寄存器地址 3bits |
| **DW\_config** | 0 | 0 |
| **Input\_base\_addr(DW)** | 0 | 1 |
| **Output\_base\_addr(DW)** | 0 | 2 |
|  |  |  |
| **PW\_config** | 1 | 0 |
| **Input\_base\_addr(PW)** | 1 | 1 |
| **Output\_base\_addr(DW)** | 1 | 2 |
|  |  |  |
| **SC\_config** | 2 | 0 |
|  |  |  |

其他设计时关注的问题

1. 如何完成ReLU操作

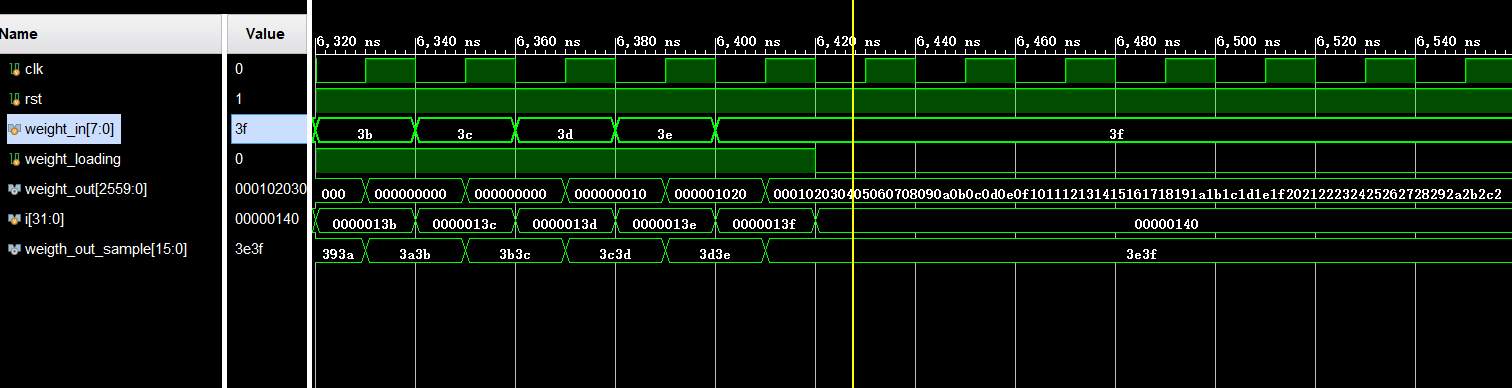
由于在MobileNet V2中，所有的PW2层都没有ReLU层。

仿真与验证

**对Weight\_loaded\_buffer.v模块的验证：**

Coverage goal 1

当weight load为1时，buffer表现为shift register，为0时所有值不动。

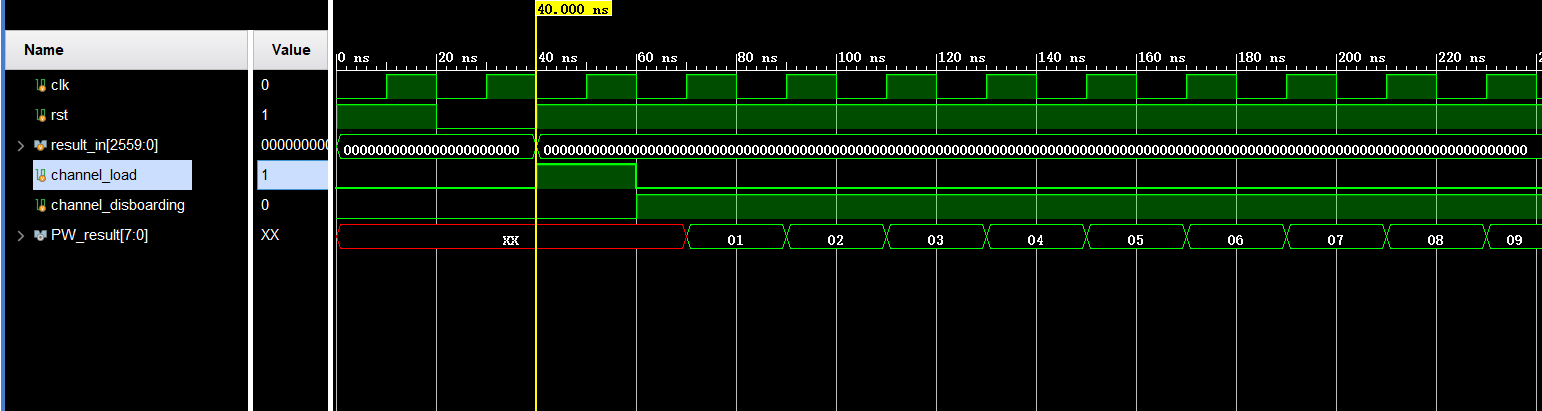


验证成功。

**对Channel\_buffer.v模块的验证：**

Coverage goal 1

当channel\_load为1时，buffer装载所有的ACC输出结果，channel\_load 1clk后置0，之后channel\_disboarding置1，表现为shift register。

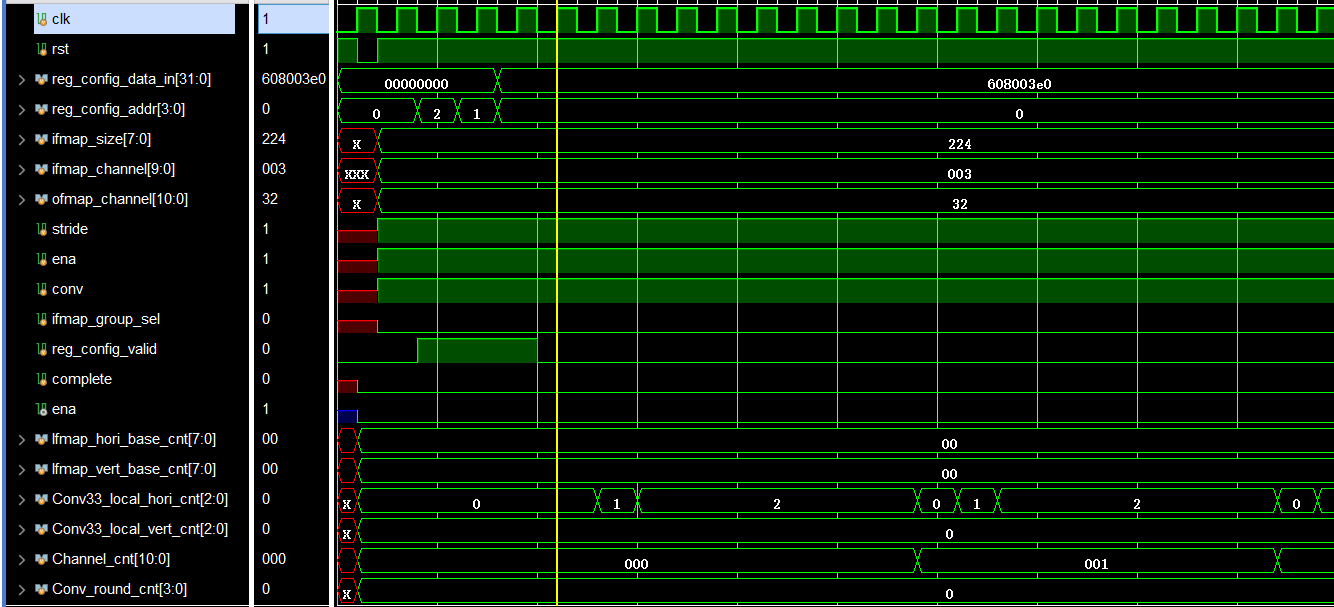


验证成功

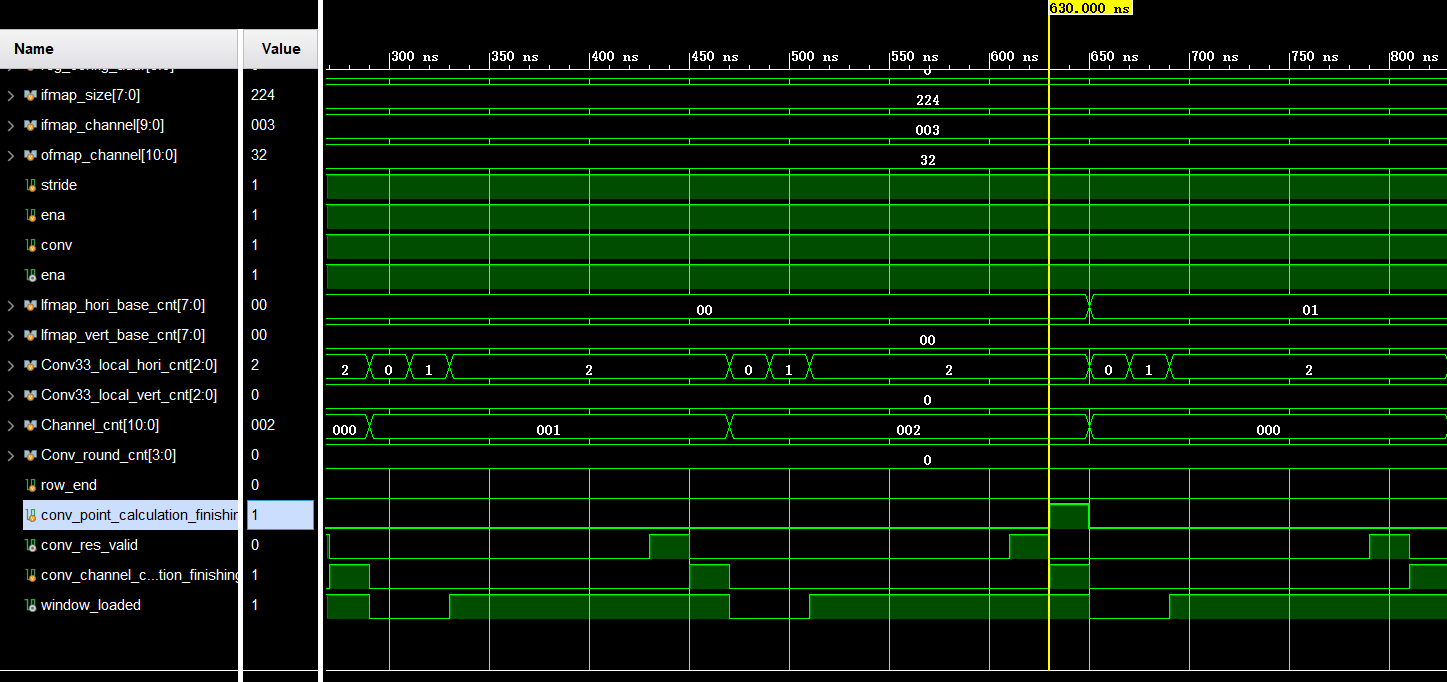
**对depthwise33模块的验证：**

Coverage goal 1

确认各个计数器时序输出正确，以及CPCF，CCCF，row\_end信号时序输出正确。



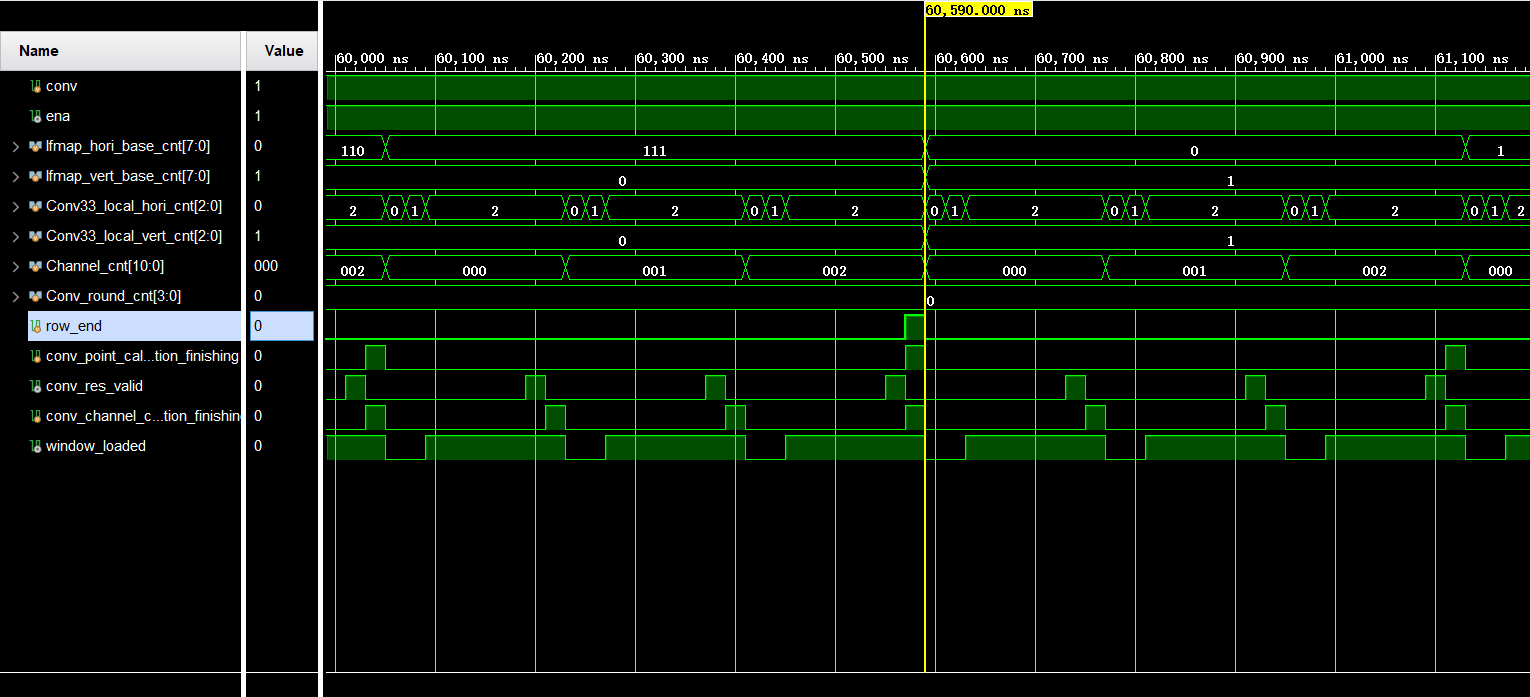
√ 当ena配置成功后，计数器开始计数。



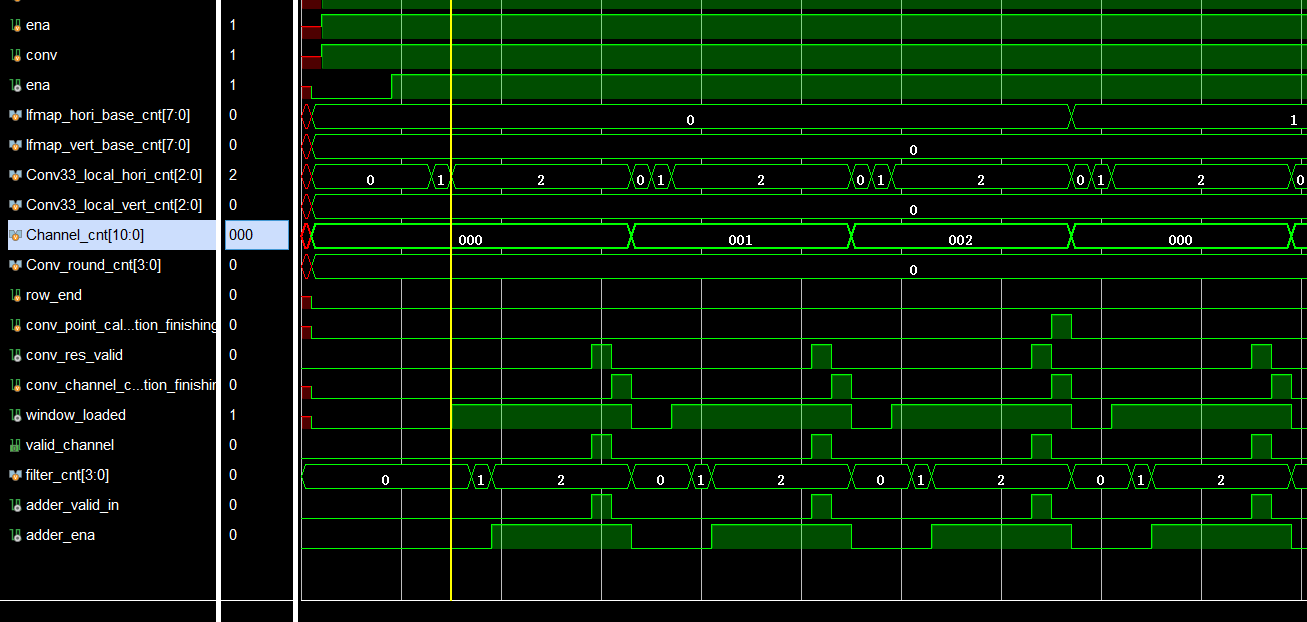
√ Channel\_cnt计满后，CPCF信号置1，且IHB自增

√ CHLOC计满后，CCCF置1，且Channel\_cnt自增

√ window\_loaded信号输出正常，表示2clk后收到BRAM输出的ifmap与weight数据

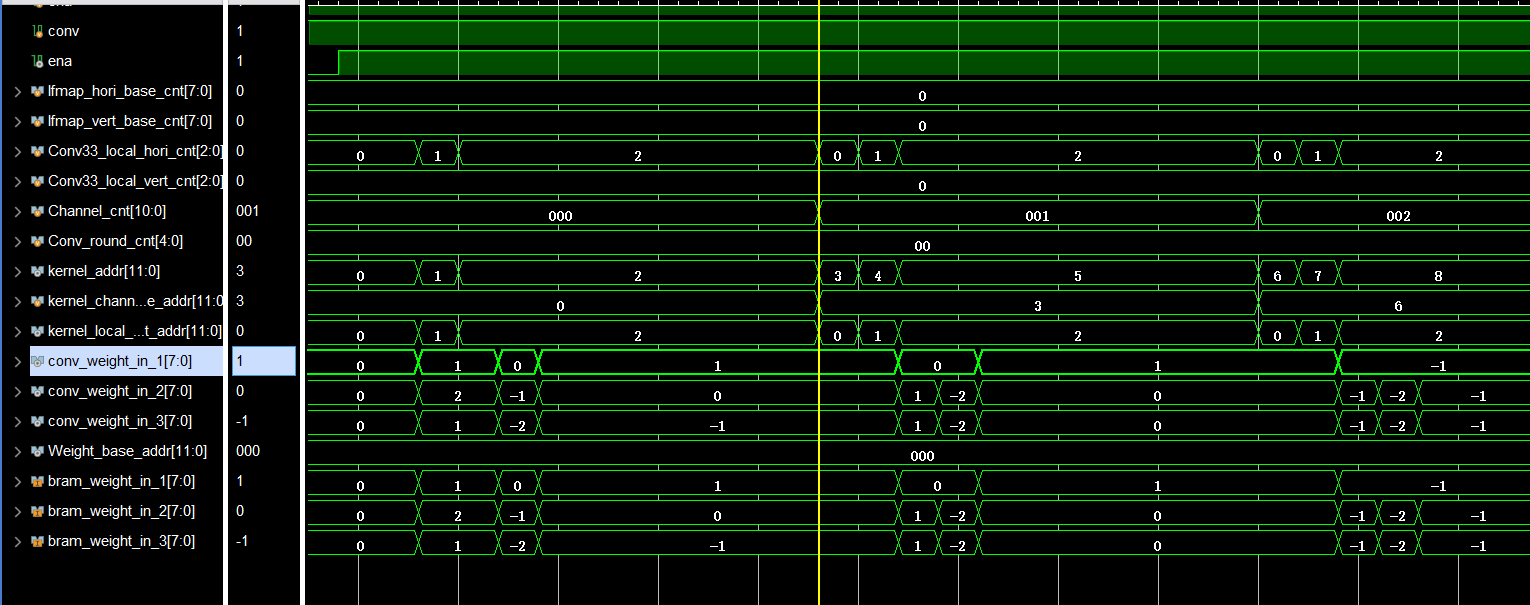


√ row\_end信号输出正常，检测到row\_end信号后，IHB归0，IVB信号自增。

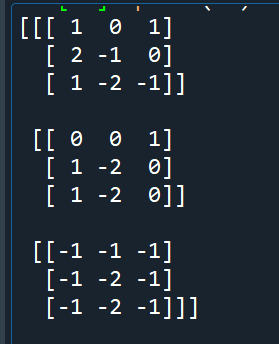


√ 因BRAM输出延迟，当CHLOC计数到1时，window\_loaded信号置1，此时filter\_cnt开始计数。

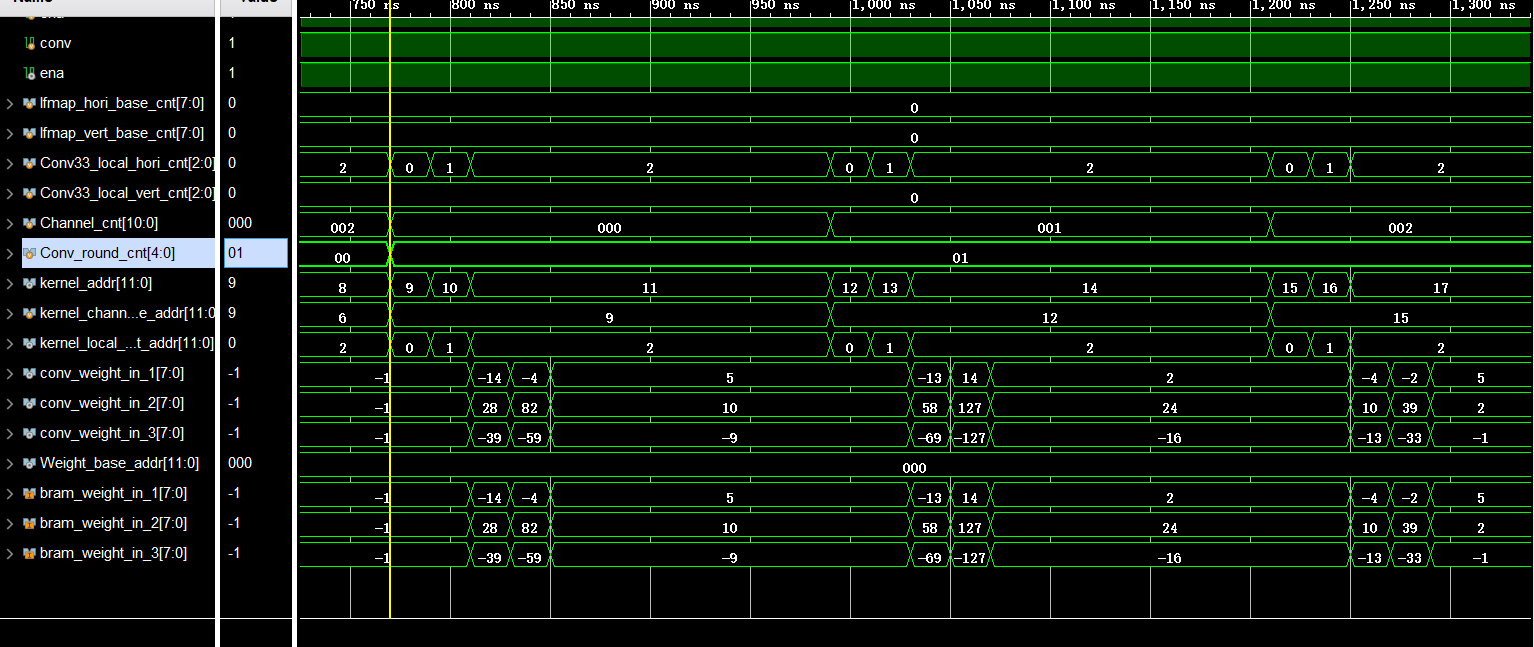
Coverage goal 2



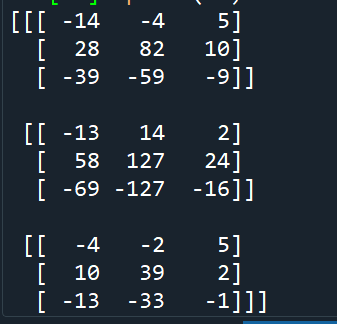
Kernel Bram layer0.kernel0：



√ Kernel Bram layer0.kernel0数据输出正常

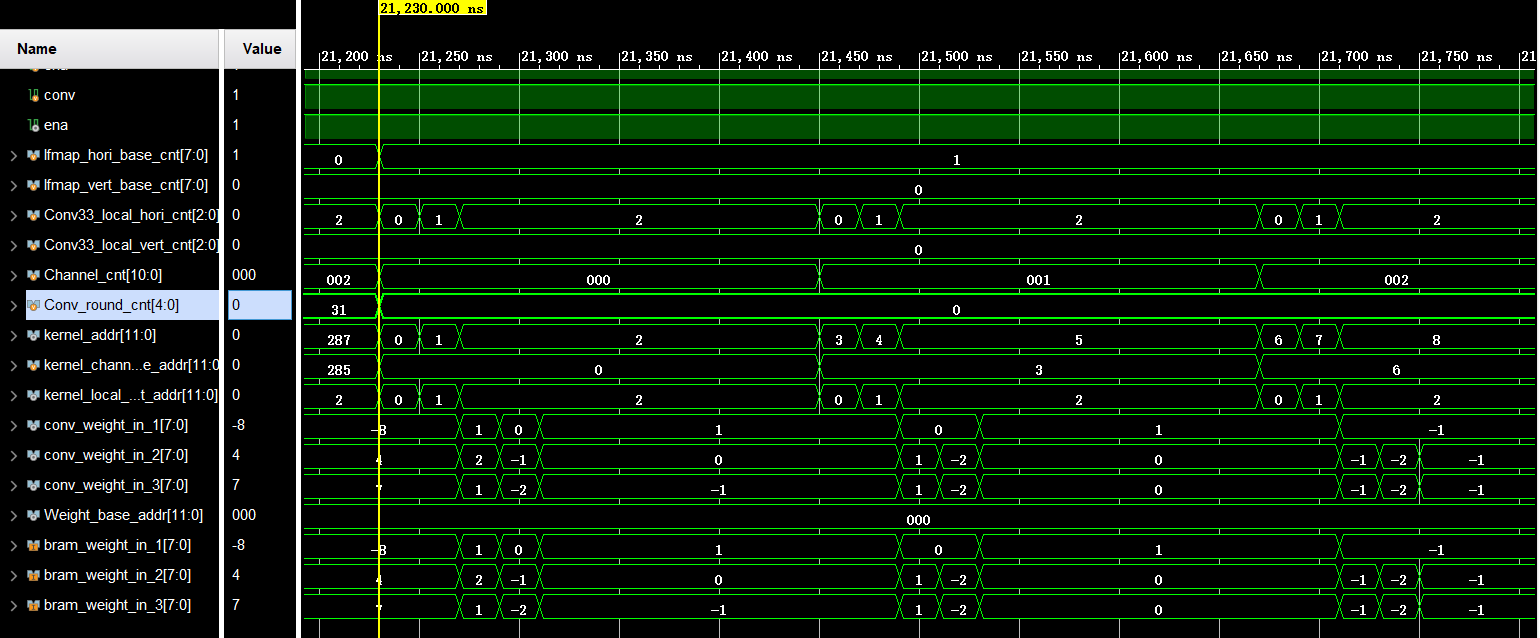


Kernel Bram layer0.kernel1：

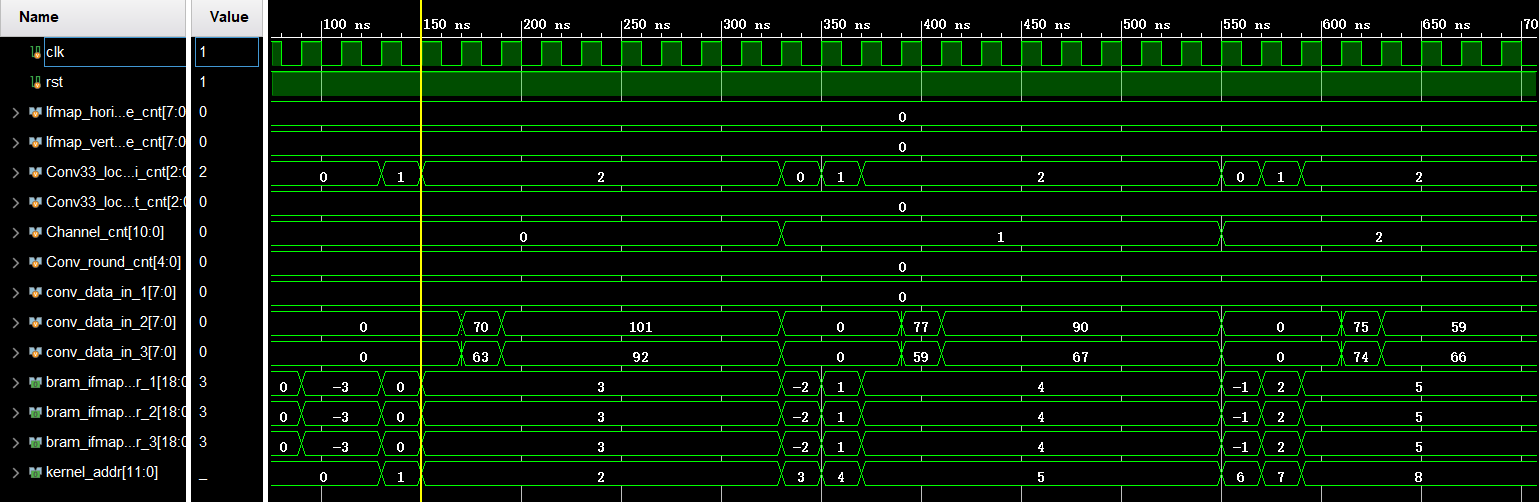


√ Kernel Bram layer0.kernel0数据输出正常

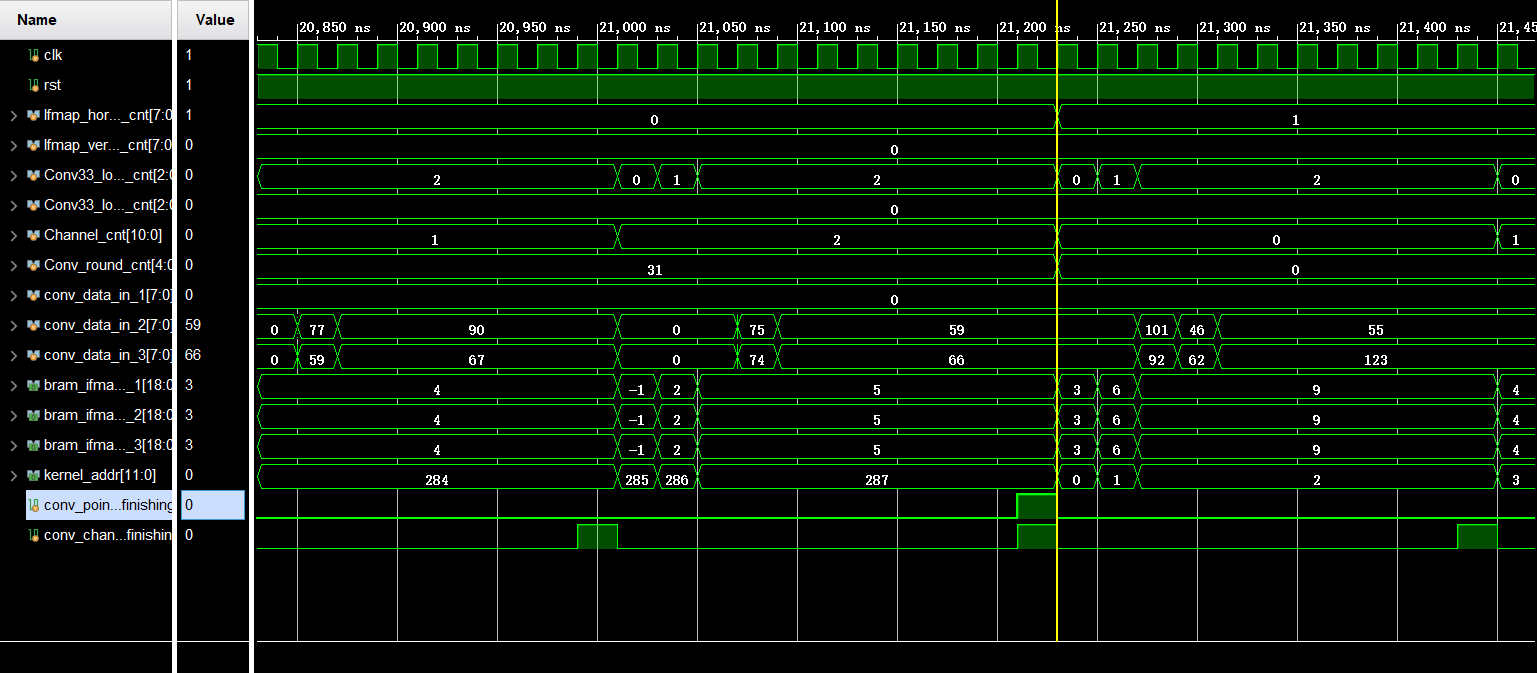
√ Conv\_rount\_cnt计数器时序正常



√ Conv\_rount\_cnt计满后，kernel\_addr归零，IHB自增。



√ Bram\_ifmap\_addr与kernel\_addr地址输出正确

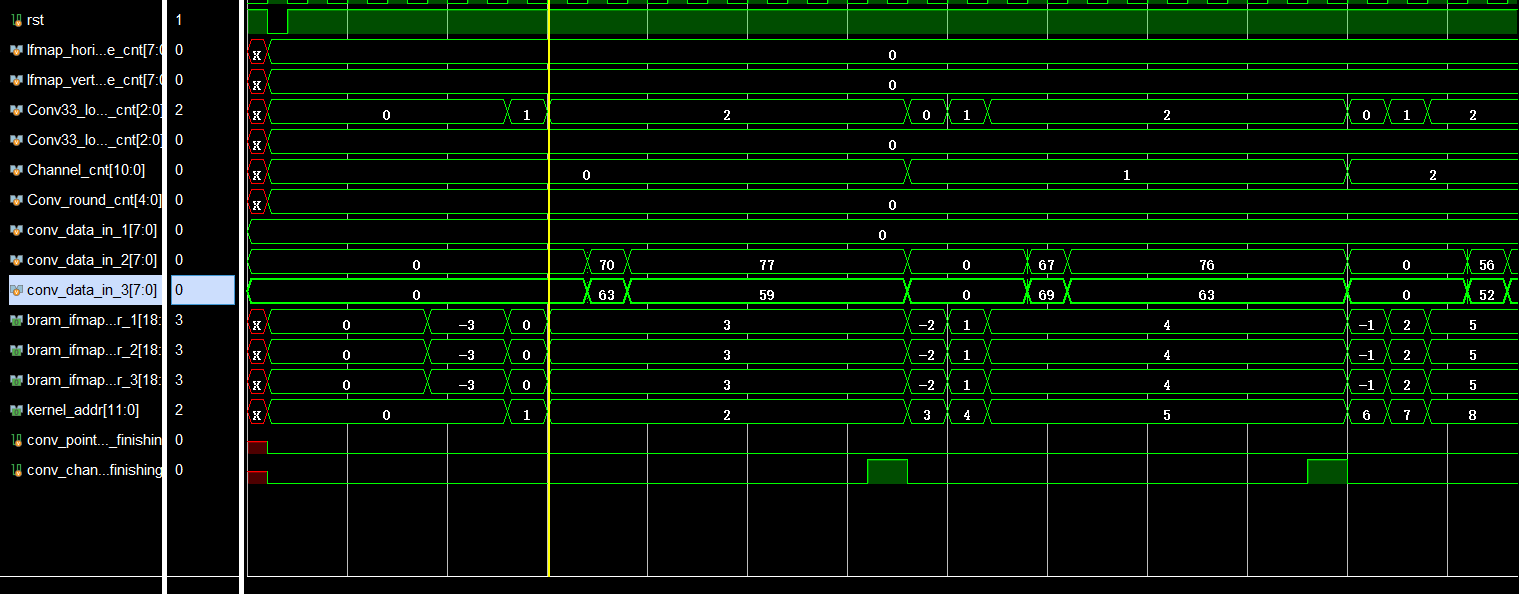


√ 收到CPCF信号后kernel\_addr重新自增，而bram\_ifmap\_addr开始计数下一个点。

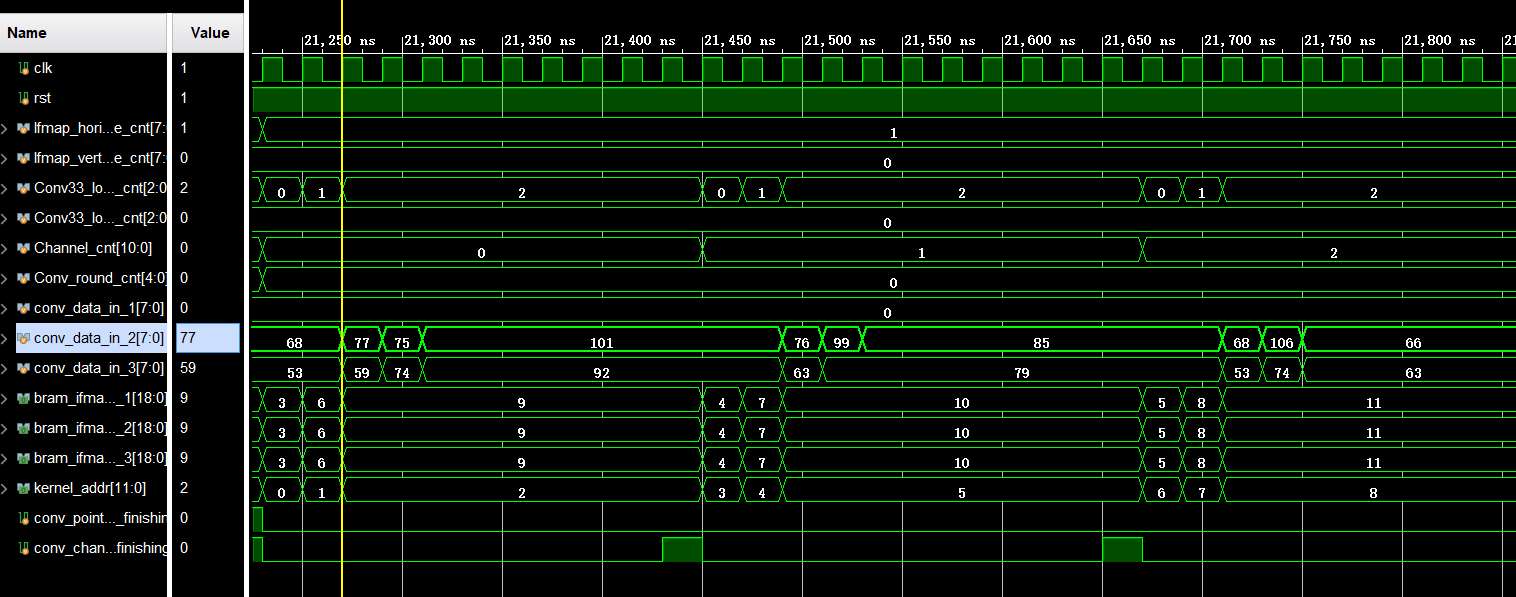
Coverage goal 3

查看ifmap输入是否正常

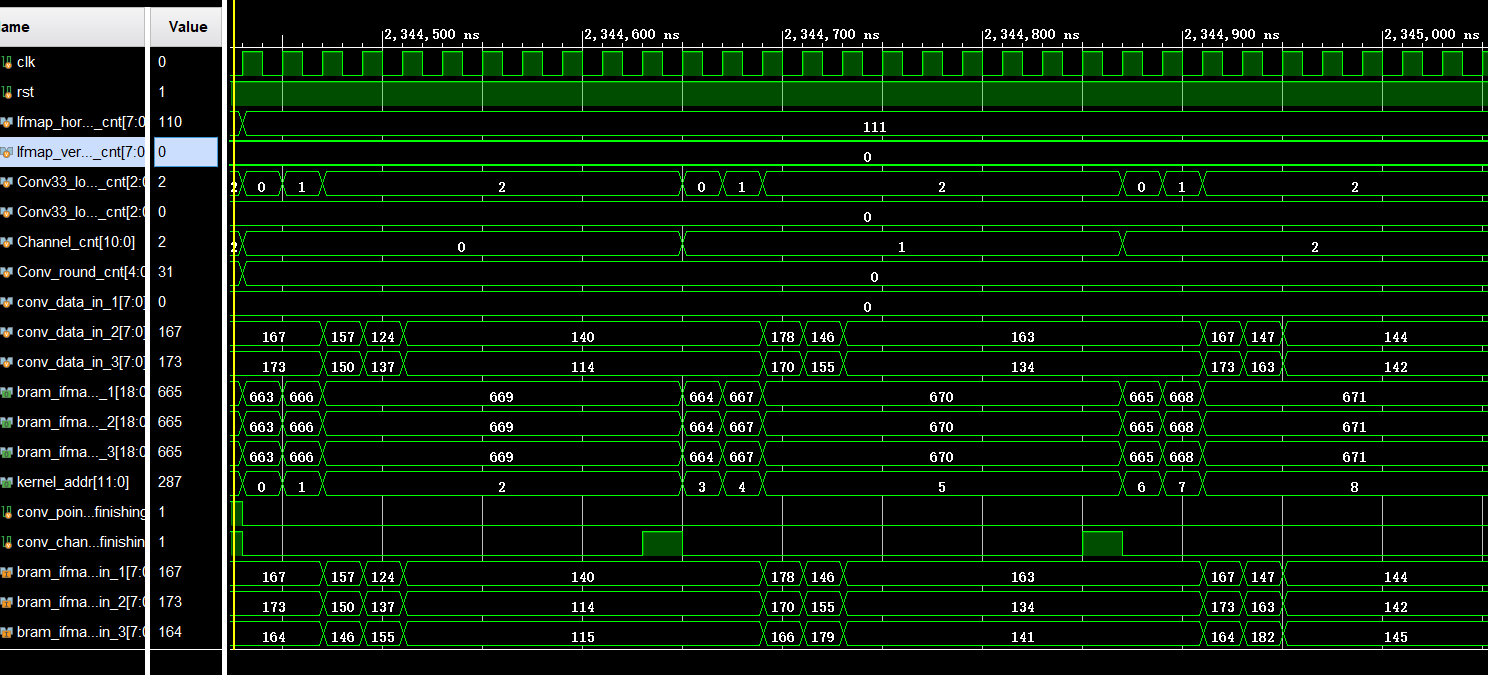
第一行第一个点输出：



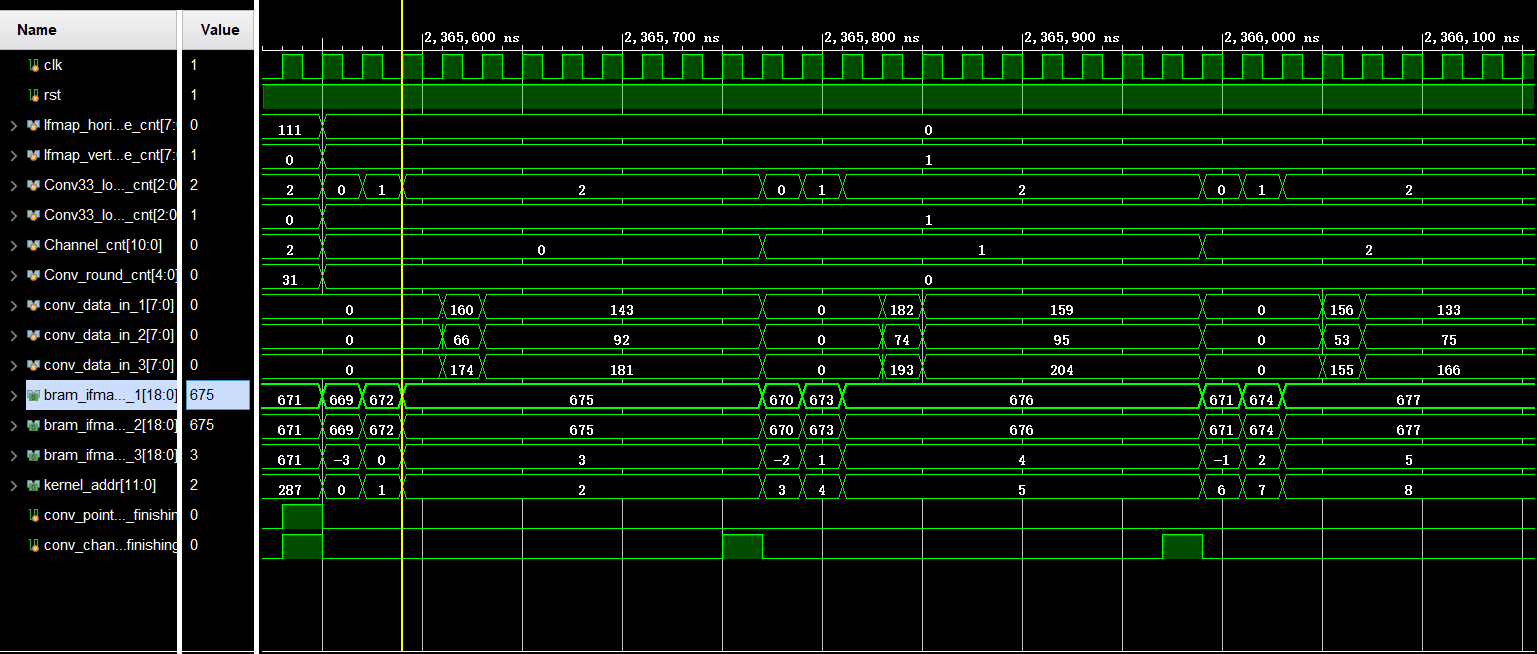
第一行第二个点输出



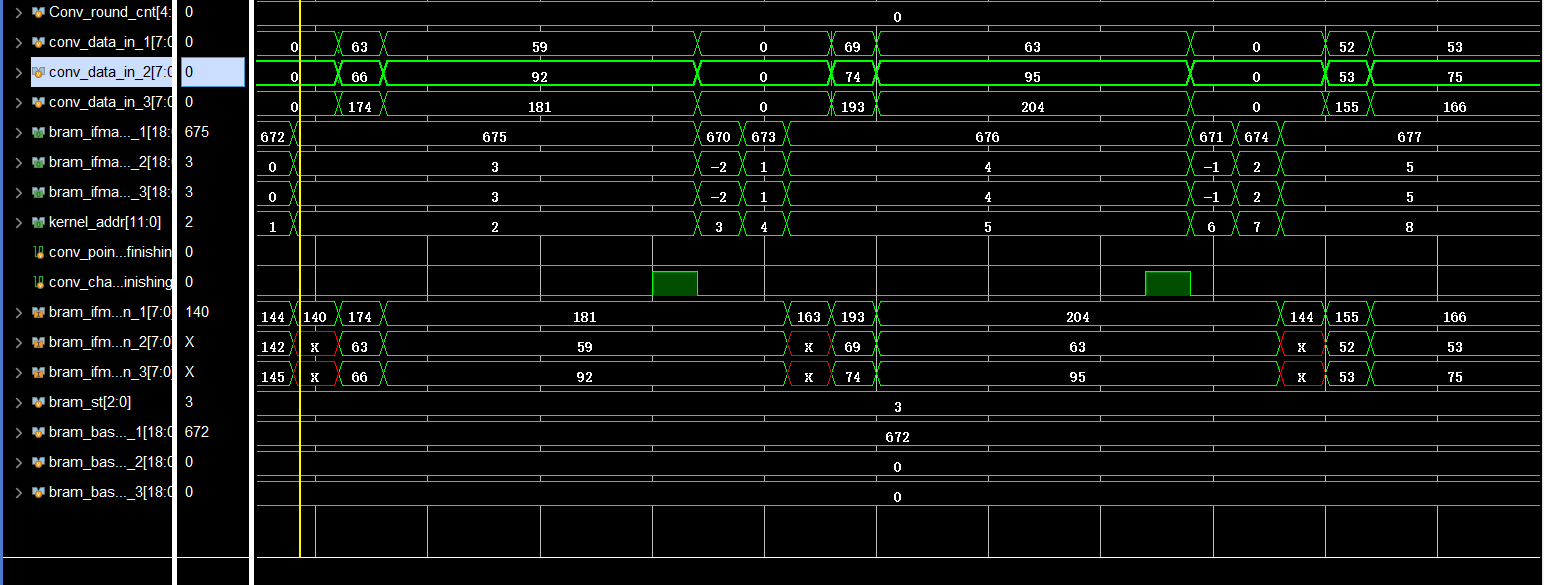
第一行最后一个点输出



√ 第一行输出正确



第二行第一个点输出

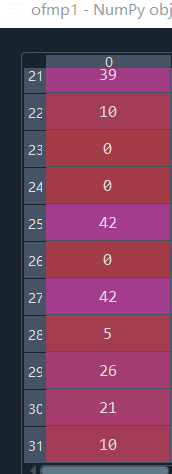
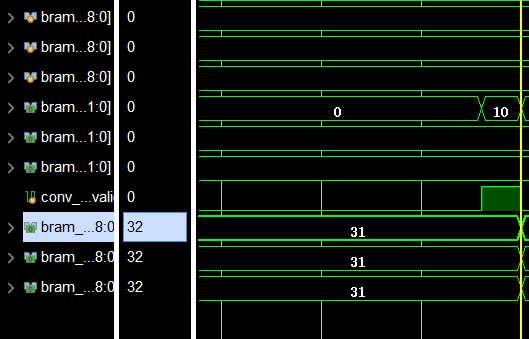


√ 第二行输出正确

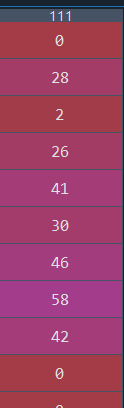
Coverage goal 4

查看ofmap写入是否正确

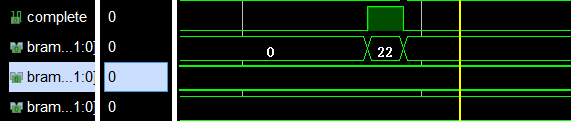
第一行第一个点最后一个channel输出



√ 第一个点输出正确



√ 第一个行输出正确



√ depthwise33模块 CONV层功能验证完成！

**对PointwiseCNN模块的验证：**

Coverage goal 1

验证PW模式下，状态机，计数器及控制信号时序正确