## 加速器整体框架图

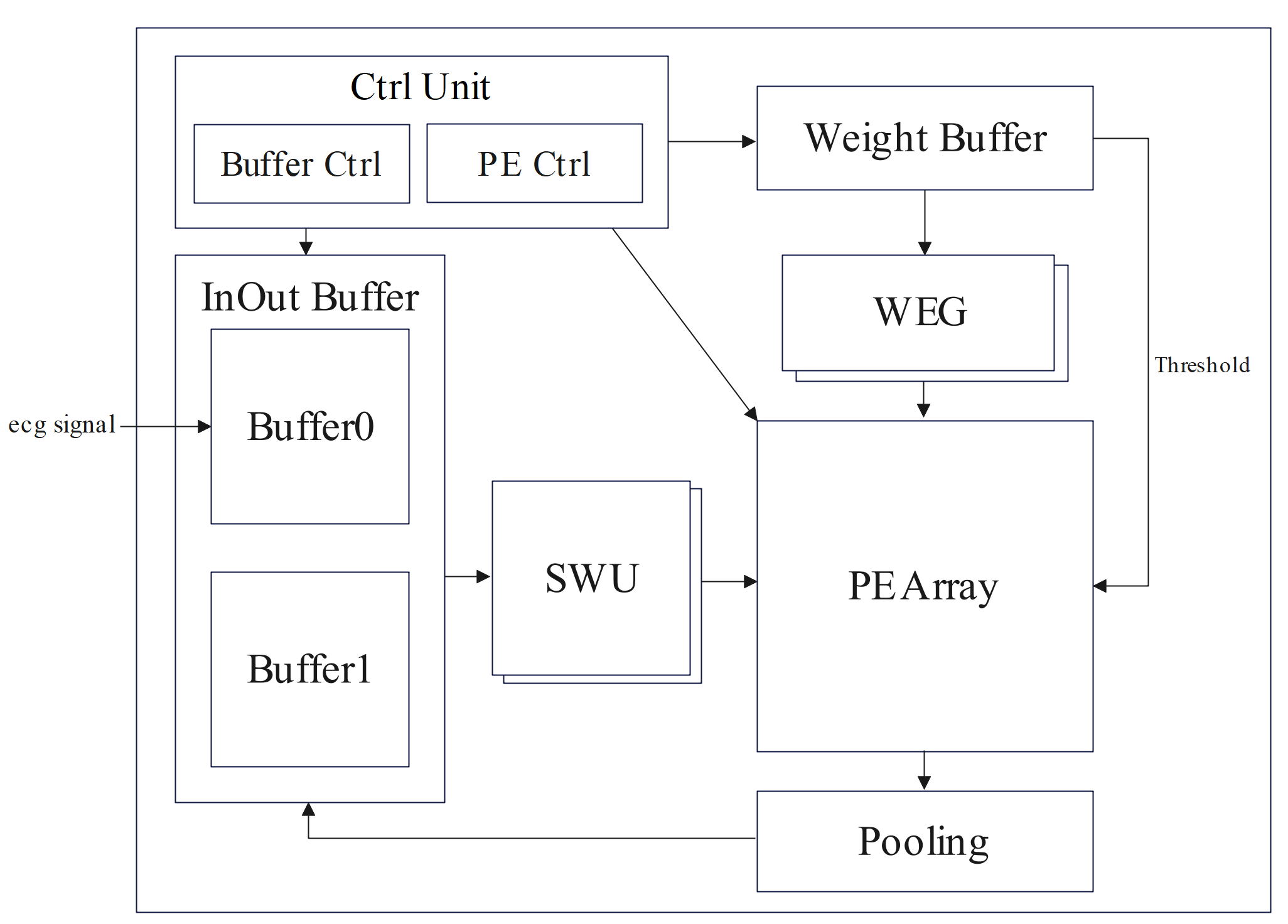


图1 加速器整体框架图

加速器各个模块的功能如表1所示。Contrl Unit是加速器的控制单元，它负责整个加速器工作状态的控制，以及缓存数据的调度。存储单元包含InOut Buffer、Weight Buffer。InOut Buffer模块是负责激活值存储的模块，其负责向SWU输出待计算的激活值，以及写回经过计算单元PE计算完毕和Pooling单元池化之后的结果。Weight Buffer则负责向WEG输入待计算的权重，以及为PE提供当前输出通道对应的阈值。SWU和WEG是缓存待计算参数的单元，通过对缓存单元中的参数复用实现更少的数据读写次数。SWU单元负责向PE阵列广播激活值输入，其将从InOut Buffer读取的输入按照卷积滑动的方式进行滑动，然后将选中的激活输入到PE单元之中。而WEG则向每个PE单元分发权重。计算单元主要包括PE阵列和Pooling单元，分别执行核心的卷积运算和激活结果的池化运算，最终将计算结果返回给InOut Buffer。其中PE阵列由多个PE单元组成，它们可以并行的进行计算。

表1 硬件加速器主要模块及模块功能说明表

|  |  |
| --- | --- |
| **模块** | **模块说明** |
| **Contrl Unit** | **控制模块，负责控制整个加速器的工作状态、InOut Buffer和Weight Buffer读写地址的计算以及网络参数的传输。** |
| InOut Buffer | **特征数据存储模块，存放输入的ECG信号和各层计算的中间值。** |
| Weight Buffer | 权重数据存储模块，存储权重数据以及阈值数据。 |
| **SWU** | **激活数据寄存器堆，缓存单批次待计算的特征数据，通过滑动操作对数据进行输出，提高激活输入的数据复用。** |
| **WEG** | **权重数据寄存器堆，缓存计算所需的权重。** |
| **PE Array** | **高并行度卷积计算阵列，执行CNN网络模型中的卷积计算、批处理归一化运算以及激活量化操作。** |
| **Pooling** | 池化模块，执行MaxPooling操作 |

## 主控模块

主控模块中分为两部分，分别是存储控制和计算控制。存储控制包括ctrl,ctrl\_a和ctrl\_w，计算控制则是ctrl\_pe。在存储控制中，ctrl负责生成缓存模块swu，weg从存储模块中读取待计算参数的地址生成。ctrl\_a负责swu的写入，ctrl\_w负责weg的写入。在计算控制中，ctrl\_pe负责控制pe阵列从swu,weg中读参数，以及pe阵列的计算和输出控制。

### 存储控制

CTRL负责生成SWU,WEG向MEM的读地址，CTRL\_A通过状态机控制SWU的轮流写入，CTRL\_W通过状态机控制WEG的轮流写入。

#### CTRL



通过localparam的方式定义网络层参数，包括每层的输入通道数IC，输出通道数OC，输出长度OL（池化之前），输入长度IL。

1. 激活控制

**wswuolcnt:**

对写swu进行计数，每当一个swu被写入完毕，即wswufin拉高，wswuolcnt就进行累加。对于第一层INT CMP而言，每个SWU只能计算一个输出点，而对于BIN CMP每个SWU能计算出TMPWID个输出点。TMPWID是PE中的缓存个数，长度为12。

**wswuiccnt:**

wswuolcnt计数到当前层的最大值之后，wswuiccnt递增32。这是因为在BIN CMP中，每次SWU中同时被写入32个IC的数据。

**wswuoccnt:**

wswuiccnt计数到当前层的最大值之后，wswuoccnt递增32。这是因为每次计算并行计算32个OC的数据。

对于DEN CMP，一个SWU即可存放所有数据，所以wswufin之后直接wswulyrcnt加一。

根据以上的计数，可以生成SWU读ACT MEM的基地址。由于采用乒乓操作，每层激活的起始地址都是0。当wswuolcnt增长时，对INT CMP地址增长3，这是由卷积步长决定的。对BIN CMP地址则增长TMPWID，因为卷积步长为1，而一次计算了TMPWID个数。

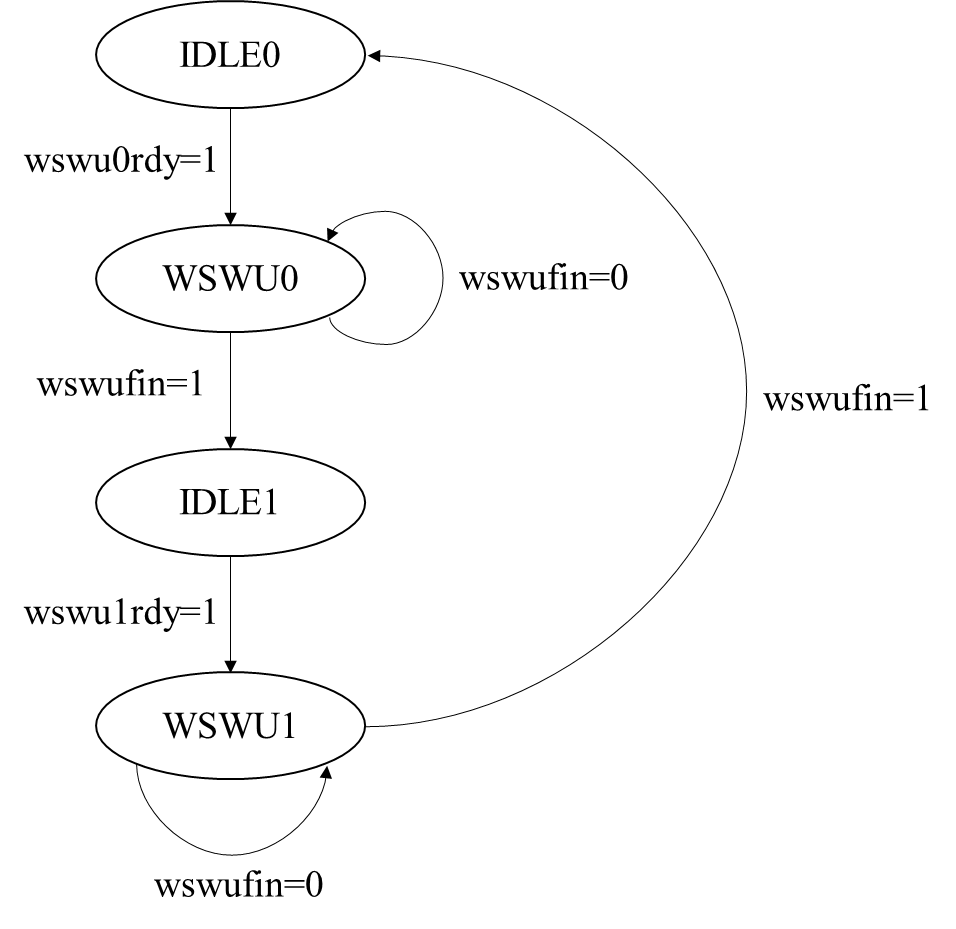
1. 权重控制

和激活控制类似，通过wwegfin来进行计数。然后通过计算值来输出基地址。

#### CTRL\_A

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 2 | wswulyrcnt | Ctrl\_a写swu的lyr层数，ctrl输入 |
| input | 32 | swumemdata | Ctrl\_a从act\_mem读数据 |
| input |  | rswumembias | Ctrl\_a从act\_mem读数据的基地址 |
| input | 1 | wswu0rdy | SWU0从ctrl\_a读数据有效 |
| input | 1 | wswu1rdy | SWU1从ctrl\_a读数据有效 |
| input | 3 | rlyrcnt | ctrl\_pe正在读取的层，避免读写顺序错误 |
| input | 1 | wmemnum | Pooling写回的MEM块，避免读写顺序错误 |
| output | 1 | actmem0ena | act\_mem0 使能 |
| output | 1 | actmem1ena | act\_mem1 使能 |
| output | 1 | actmem0wea | Actmem0 读写选择 |
| output | 1 | actmem1wea | Actmem1 读写选择 |
| output |  | actmem0addr | Ctrl\_a从act\_mem0读数据的地址 |
| output |  | actmem1addr | Ctrl\_a从act\_mem1读数据的地址 |
| output |  | wswuaddr | Ctrl\_a向swu写地址 |
| output | 1 | wact0vld | Ctrl\_a向swu0写数据有效 |
| output | 1 | wact1vld | Ctrl\_a向swu1写数据有效 |
| output | 32 | wswu0data | Ctrl\_a向swu0写数据 |
| output | 32 | wswu1data | Ctrl\_a向swu1写数据 |
| output | 1 | wswufin | Ctrl\_a写swu完毕 |

状态机：



轮流写两块SWU，两块SWU形成双缓冲，减少PE的等待。

进入WSWU状态后，根据CTRL提供的ACT MEM基地址向ACT MEM读数据，每读取一个数据wswucnt计数一次，达到最大值之后wswufin拉高。对于INT CMP而言，需要写入20行，即80个输入点。对于BIN CMP则是TMPWID+KL-1。对于DEN CMP则根据具体输入长度计算。

BRAM读延迟两周期，需要进行打拍。

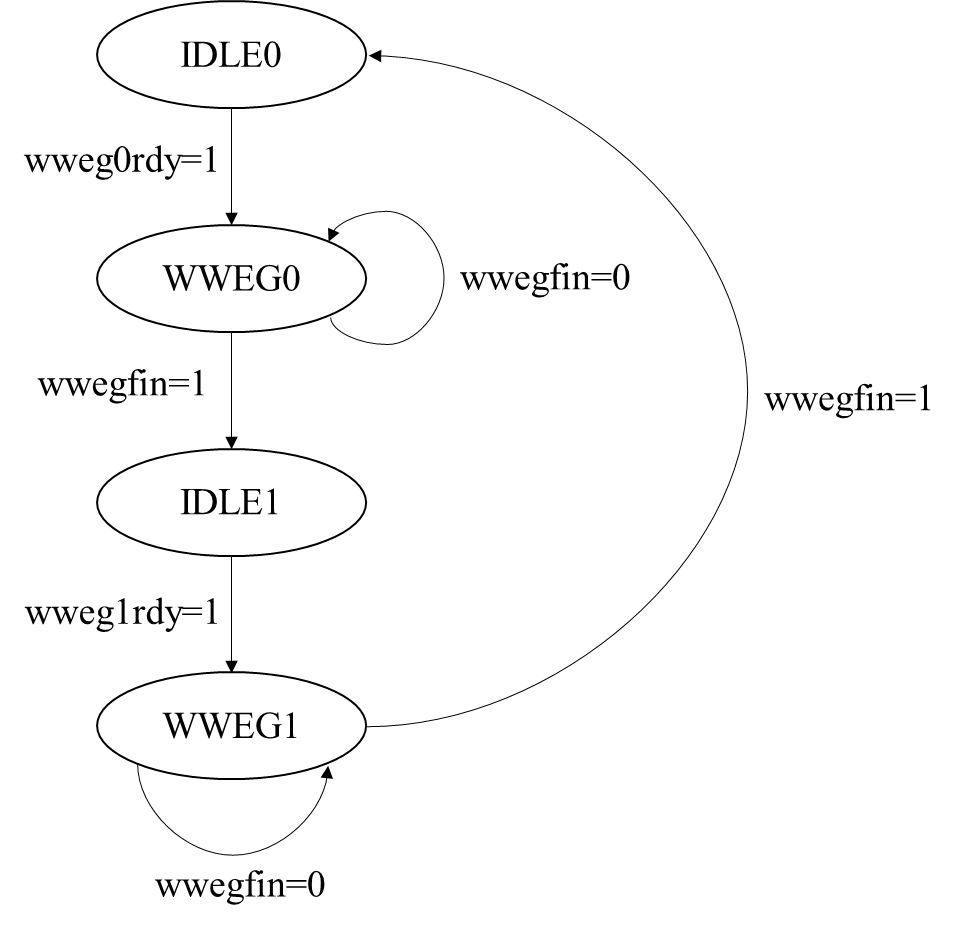
和SWU的握手：

CTRL\_A准备好写入SWU时，拉高vld信号。SWU接收到vld信号之后，且其中数据已经消耗完毕，就拉高rdy信号。握手成功之后进入wswu状态开始从mem中读数据写入swu。

#### CTRL\_W

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 2 | wweglyrcnt | Ctrl\_w写weg的lyr层数，ctrl输入 |
| input | 32 | wegmemdata | Ctrl\_w从weg\_mem读数据 |
| input |  | rwegmembias | Ctrl\_w从weg\_mem读数据的基地址 |
| input | 1 | wweg0rdy | Weg0从ctrl\_w读数据有效 |
| input | 1 | wweg1rdy | Weg1从ctrl\_w读数据有效 |
| output | 1 | wegmemena | Wegmem 使能 |
| output | 1 | wegmemwea | Wegmem 读写选择 |
| output |  | wegmemaddr | Ctrl\_w从weg\_mem读数据的地址 |
| output |  | wwegaddr | Ctrl\_w向weg写地址 |
| output | 1 | wweg0vld | Ctrl\_w向weg0写数据有效 |
| output | 1 | wweg1vld | Ctrl\_w向weg1写数据有效 |
| output | 32 | wweg0data | Ctrl\_w向weg0写数据 |
| output | 32 | wweg1data | Ctrl\_w向weg1写数据 |
| output | 1 | wwegfin | Ctrl\_w写weg完毕 |

状态机：



轮流写入两个WEG

和CTRL\_A类似，握手成功后开始写WEG，并计数，计数完毕之后写入完毕，等待写入下一块WEG。

### 计算控制

#### CTRL\_PE

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 1 | rswu0vld | SWU0读有效 |
| input | 1 | rswu1vld | SWU1读有效 |
| input | 1 | rweg0vld | WEG0读有效 |
| input | 1 | rweg1vld | WEG1读有效 |
| input | 1 | rswu0fin | SWU0读完成，拉低rswu0rdy |
| input | 1 | rswu1fin | SWU1读完成，拉低rswu1rdy |
| input | 1 | ecg\_start | 计算开始信号 |
| output | 1 | rswu0rdy | Ctrl\_pe可读SWU0 |
| output | 1 | rswu1rdy | Ctrl\_pe可读SWU1 |
| output | 1 | rweg0rdy | Ctrl\_pe可读WEG0 |
| output | 1 | rweg1rdy | Ctrl\_pe可读WEG1 |
| output | 1 | rswuicfin | 指示BIN\_CMP时反复读同一块SWU |
| output | 1 | rwegolfin | 指示INT\_CMP时反复读同一块WEG |
| output | 3 | pe\_cs | PE状态，指示SWU和WEG在BIN/INT/DEN\_CMP |
| output | 1 | Thren | 向PE写阈值有效 |
| output | 256 | threshold | 32个8bit的阈值 |
| output | 3 | rlyrcnt | 当前PE计算的层，控制PE ARRAY |
| output | 1 | ricfin | 输入通道读取完毕，控制PE ARRAY计算完毕输出 |
| output | 3 | Cmp\_cs | 计算状态，控制PE ARRAY |
| output | 3 | Cmp\_ns | 计算状态，控制PE ARRAY |
| output | 1 | rdenilfin | 输入长度读取完毕，控制PE ARRAY计算完毕输出 |
| output | 1 | rolfin | 输出长度读取完毕，控制PE ARRAY计算完毕输出 |

使用状态机控制当前PE阵列计算从哪个SWU,WEG中读参数来计算。

INT CMP：

此情况下，SWU需要每次计算就更换，而WEG则不需要。卷积核长度为80，一个WEG可以存放32个输出通道长度为80的卷积核。一个WEG可以用到一组32个输出通道都被计算完毕。

BIN CMP：

此情况下，WEG每次计算就要更换。因为SWU中有32个IC的输入数据，而WEG中只有32个OC，2个IC的数据。

DEN CMP：

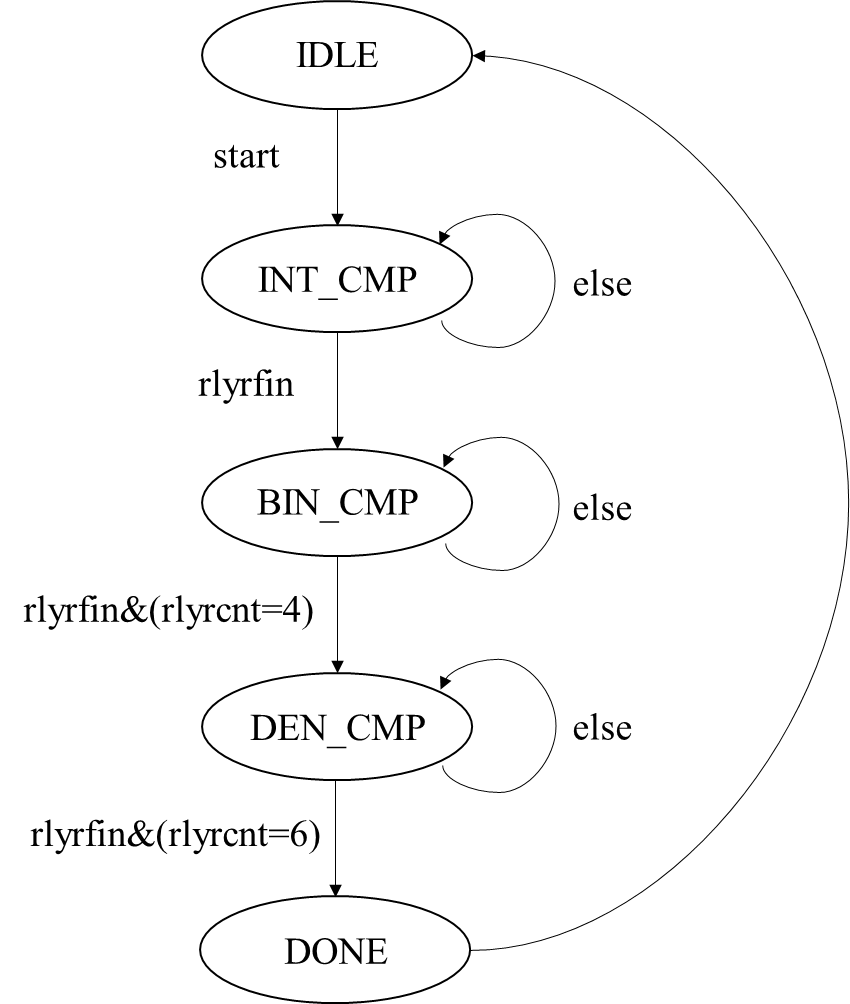
此情况下，一个SWU就可以存放所有数据，而WEG则需要更换。

下面是详细介绍：

主状态机的状态跳转逻辑，状态的具体含义如表所示。当外部ECG信号写入BRAM后，主状态机收到START信号，开始执行前向推导的计算流程。由于卷积神经网络包含整形输入卷积层、二值输入卷积层、全局平均池化层和全连接层，它们需要不同的计算策略，所以主状态机会根据当前的前向推导进度，切换到不同的状态。首先，主状态机进入INT\_CMP状态，执行整型输入的卷积计算。当第一层计算完成后，进入BIN\_CMP状态，执行二值输入的卷积计算。当3层二值输入卷积计算完成后，主状态机进入DEN\_CMP状态，执行全连接层的计算。当2层全连接层计算完成后，进入DONE状态，输出计算结果。输出结果后，主状态机返回IDLE状态，等待新的ECG信号写入。

加速器主状态机各状态功能描述表

|  |  |
| --- | --- |
| **状态** | **状态描述** |
| IDLE | **空闲状态,等待ECG写入** |
| INT\_CMP | **整型输入的卷积层计算** |
| **BIN\_CMP** | **二值输入的卷积层计算** |
| DEN\_CMP | **全连接层计算** |
| DONE | **输出分类结果** |



计算控制模块主状态机

在主状态机的每个计算状态，即INT\_CMP、BIN\_CMP和DEM\_CMP中，均需要PE阵列进行循环的乘累加计算才能够完成最终的计算结果。在本加速器的设计中，PE阵列进行计算所需要的激活值和权重值均通过双缓冲的方式存储在SWU和WEG中。因此从状态机需要根据主状态机表示的当前计算类型，使用不同的计算策略对SWU和WEG中的参数进行读取。从状态机的各个状态及其功能如表所示。

加速器从状态机各状态功能描述表

|  |  |
| --- | --- |
| **状态** | **状态描述** |
| **S0W0IDLE** | **等待SWU0和WEG0中参数写入完毕** |
| **S0W0CMP** | **读取SWU0和WEG0中参数输入PE计算** |
| **S0W1IDLE** | **等待SWU0和WEG1中参数写入完毕** |
| **S0W1CMP** | **读取SWU0和WEG1中参数输入PE计算** |
| **S1W0IDLE** | **等待SWU1和WEG0中参数写入完毕** |
| **S1W0CMP** | **读取SWU1和WEG0中参数输入PE计算** |
| **S1W1IDLE** | **等待SWU1和WEG1中参数写入完毕** |
| **S1W1CMP** | **读取SWU1和WEG1中参数输入PE计算** |

根据主状态机的三种计算状态INT\_CMP、BIN\_CMP和DEN\_CMP，从状态机有三种不同的跳转策略。接下来将分别介绍主状态机处于三种不同状态时从状态机的跳转策略。

（a）INT\_CMP

展示了从状态机在整型输入计算时的状态转移策略。从状态机有两类状态：空闲状态IDLE和计算状态CMP。在空闲状态IDLE时，状态机等待SWU和WEG的读有效信号。当SWU和WEG都准备好被读取时，状态机进入计算状态CMP。在计算状态CMP时，模块内部开始计数，并且输出读ready信号。同时，PE阵列也开始读取SWU和WEG的参数。INT\_CMP状态下PE阵列每轮计算需要从SWU读取80个8bit的有符号整型数，从WEG读取32组80个1bit的权重。每个周期，SWU输出4个8bit的整型数，WEG输出32个4bit的权重。因此，模块内部需要计数20个周期，然后输出rswufin信号。当rswufin信号为高时，状态机返回空闲状态IDLE，并且取消读ready信号。整体读参数的时序图如图4.7所示。



图4.7 计算控制模块从状态机读参数时序示意图

由于本网络的第一层只有一个输入通道，所以每个输出通道的输出特征图都使用同一个长度为80的卷积核。这意味着当一个输出通道的计算没有完成时，即rolfin信号为低时，下一次计算只需要更换激活值，不需要更换权重。因此，状态机会进入相同SWU、不同WEG的空闲状态。反之，当一个输出通道的计算完成时，即rolfin信号为高时，下一次计算需要同时更换SWU和WEG。因此，状态机会进入不同SWU、不同WEG的空闲状态。

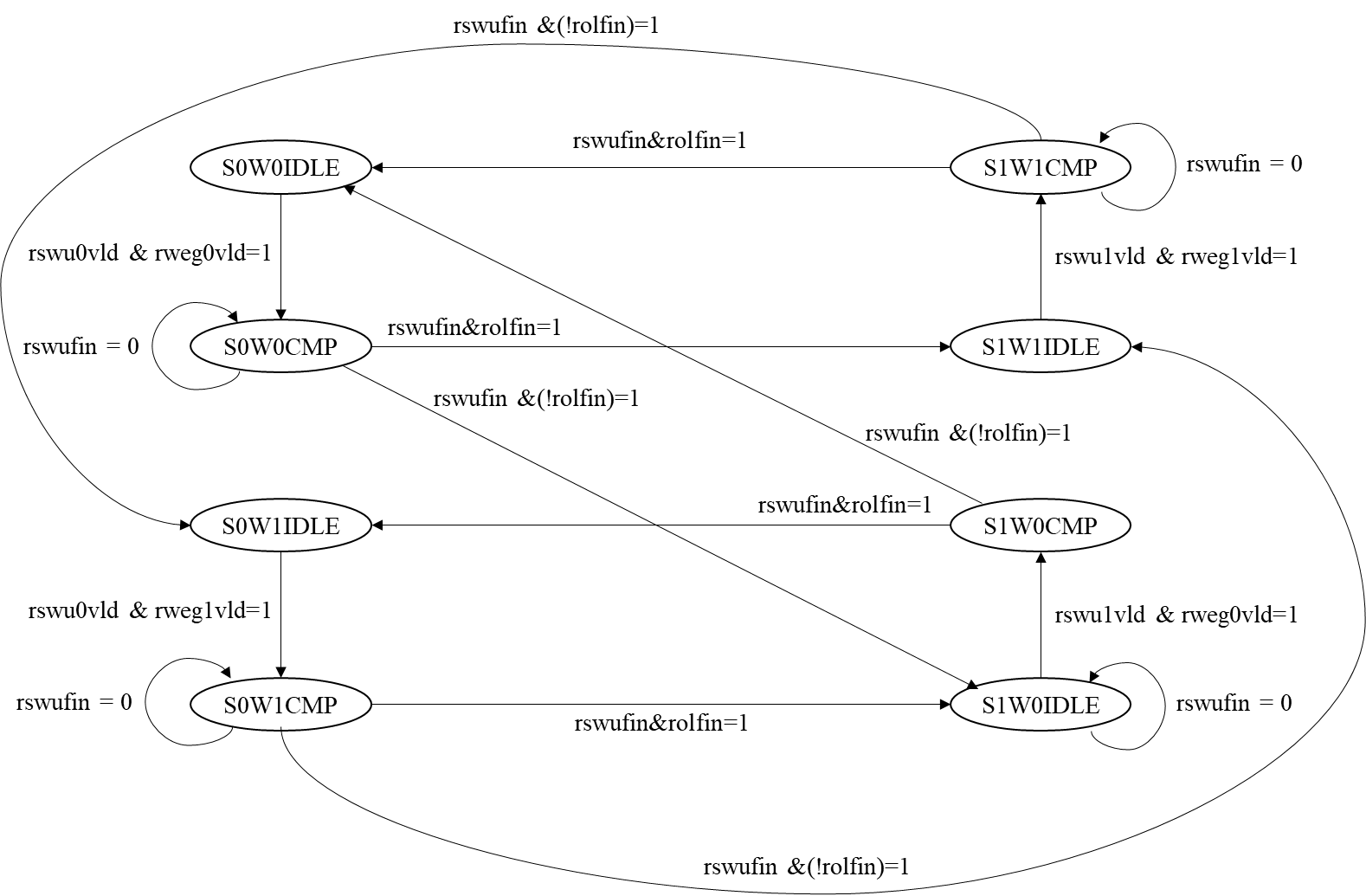


图4.8计算控制模块从状态机在整形输入卷积时的跳转策略

对主状态机INT\_CMP下的计算状态跳转进行仿真验证，如图4.9所示。从图中可以看到其主状态机pe\_cs处于INT\_CMP状态，而从状态机则根据和缓存单元的握手信号进行跳转。仿真结果符合预期的功能，在一条输出特征图计算完毕之前，反复读取同一WEG中的权重，即rweg0rdy反复被拉高而rweg1rdy一直置低。而SWU中的数据则在计算完毕后被切换，即从状态机跳转状态后和不同的SWU进行握手。

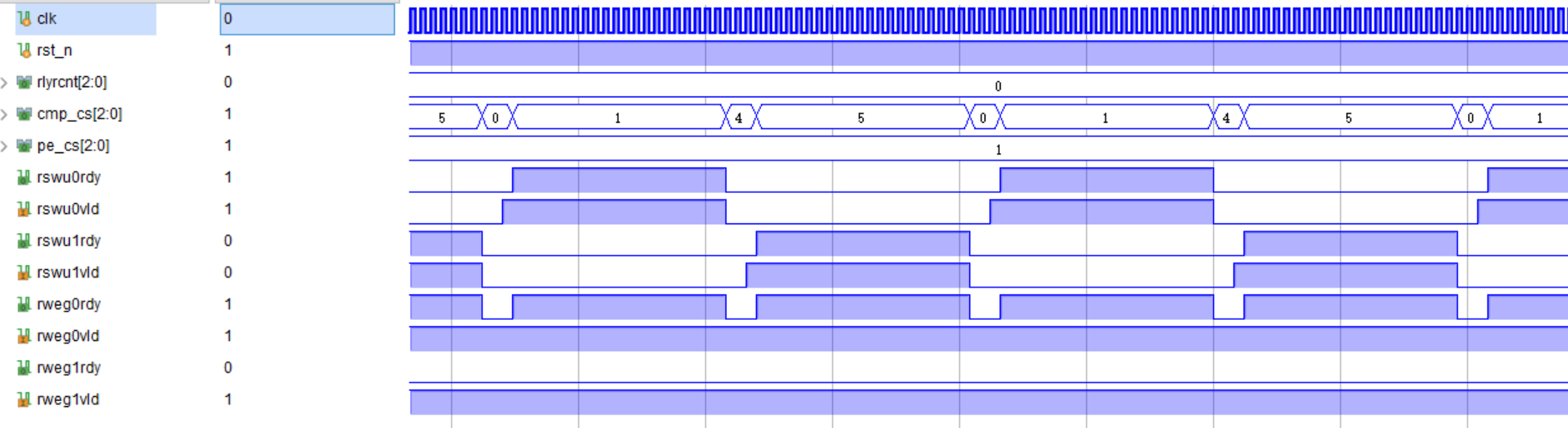


图4.9整型输入时计算控制模块仿真波形图

(b)BIN\_CMP

图4.10展示了在进行二值输入的卷积运算时，计算控制模块的从状态机的跳转策略。和主状态机在INT\_CMP时相同，当SWU和WEG均读有效时，从状态机从空闲状态IDLE跳转到计算状态CMP。在计算状态CMP时，读ready信号被拉高，PE阵列读取SWU和WEG输入的参数。在全二值运算中，SWU使用滑动的方式对WEG中的权重进行复用。在PE阵列的一轮计算中，每个周期SWU输出6bit的二值输入，而WEG则保持相同的6bit权重输出不变。当SWU滑动完毕之后，rswufin被拉高，从状态机跳转到空闲状态IDLE。由于每次写SWU时一次性写入了32个输入通道的激活值，而一轮PE计算的滑动只消耗了两个输入通道的激活值。因此，在SWU中的所有输入通道被消耗完之前，即rswuicfin低位时，从状态机跳转到SWU相同、WEG不同的空闲状态。相反，若rswuicfin是高位，则跳转到SWU和WEG均不同的空闲状态。

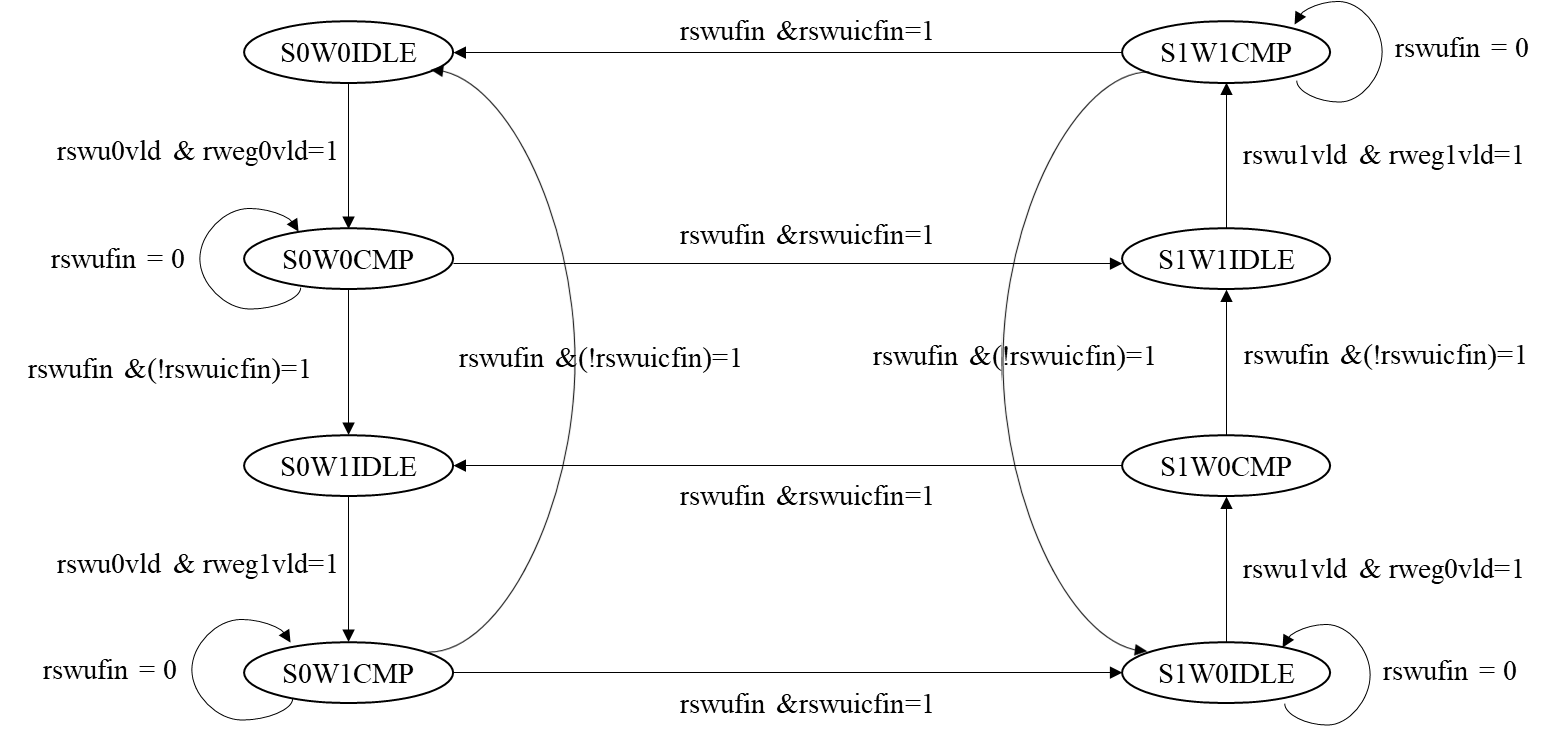


图4.10计算控制模块从状态机在计算二值输入卷积时的跳转策略

计算控制模块主状态机处于BIN\_CMP时候其状态跳转的行为及仿真如图4.11所示。从图中可以看到其主状态机pe\_cs处于BIN\_CMP状态，SWU中的参数在所有输入通道被消耗完之前被反复的读取。而WEG则在一次计算后就进行切换，同时也可以看出，计算消耗参数的周期大于向WEG中写数据所需要的周期，所以PE可以无缝的进行连续的计算。

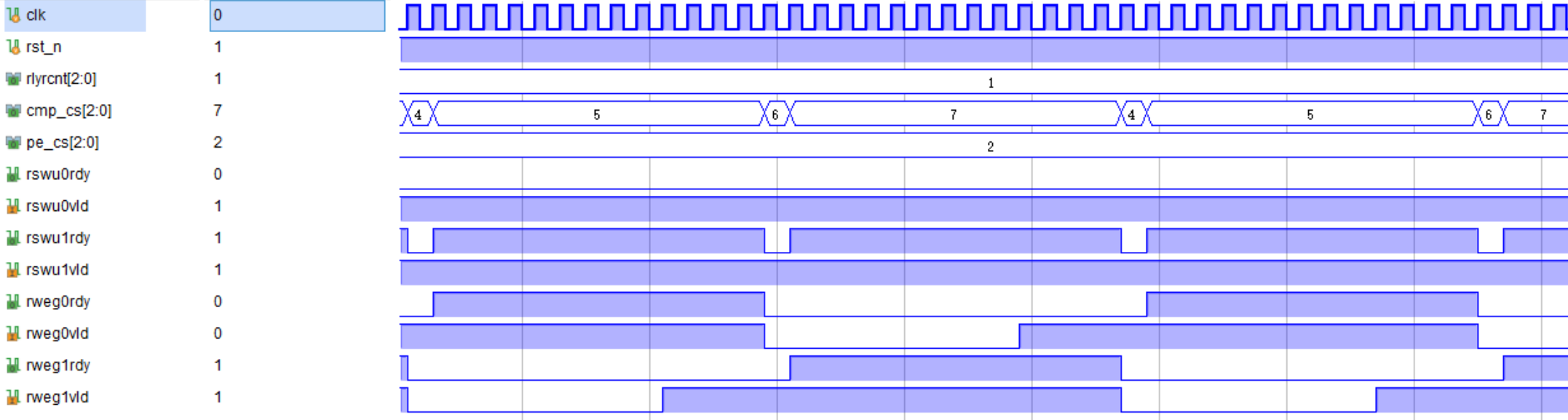


图4.11二值输入时计算控制模块仿真波形图

(c)DEN\_CMP

当主状态机位DEN\_CMP时，从状态机的跳转策略如图4.12所示。在全连接层的计算中，由于输入的数据量很小，两层的输入分别为256bit和64bit。因此仅使用一个SWU即可，而权重WEG则不断进行变换。同样的，当读参数的握手成功之后，从状态机从空闲状态IDLE跳转到计算状态CMP，PE阵列开始读取数据。每轮计算中向PE阵列输入6bit的激活值，以及32个6bit的权重值，一个周期即可完成此轮计算，然后从状态机跳转到SWU相同，WEG不同的空闲状态中。

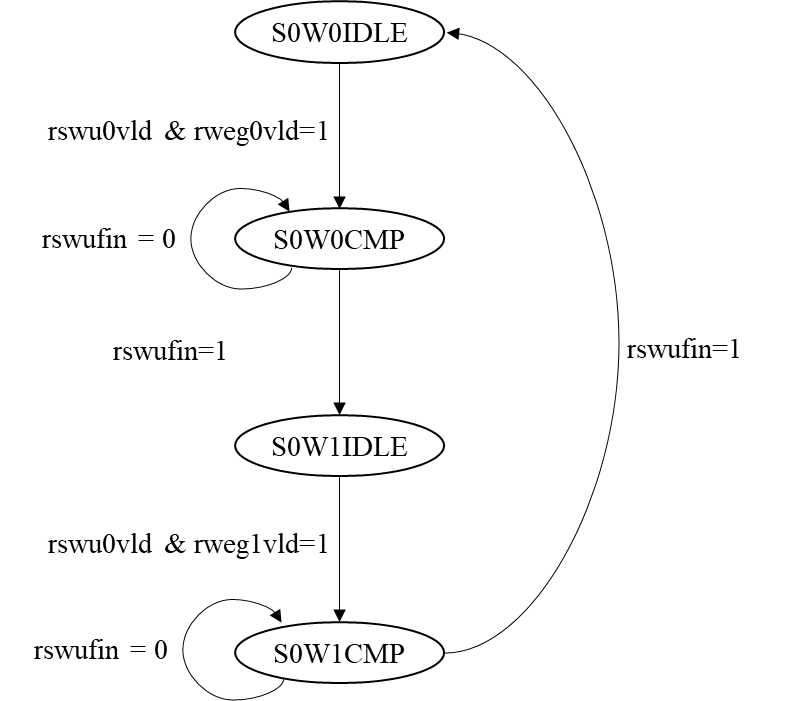


图4.12计算控制模块从状态机在二值输入卷积时的跳转策略

图4.13是全连接层计算时计算控制模块的状态跳转仿真波形图。从图中可以看到其主状态机pe\_cs处于DEN\_CMP状态。由于全连接层的输入数据量较少，可以被单个SWU存储，因此只需要反复从SWU0中读取参数即可。而WEG则在一次计算后需要切换，同时每次计算只需要一个周期即可完成，PE需要等待WEG参数的写入完成才能进行计算。

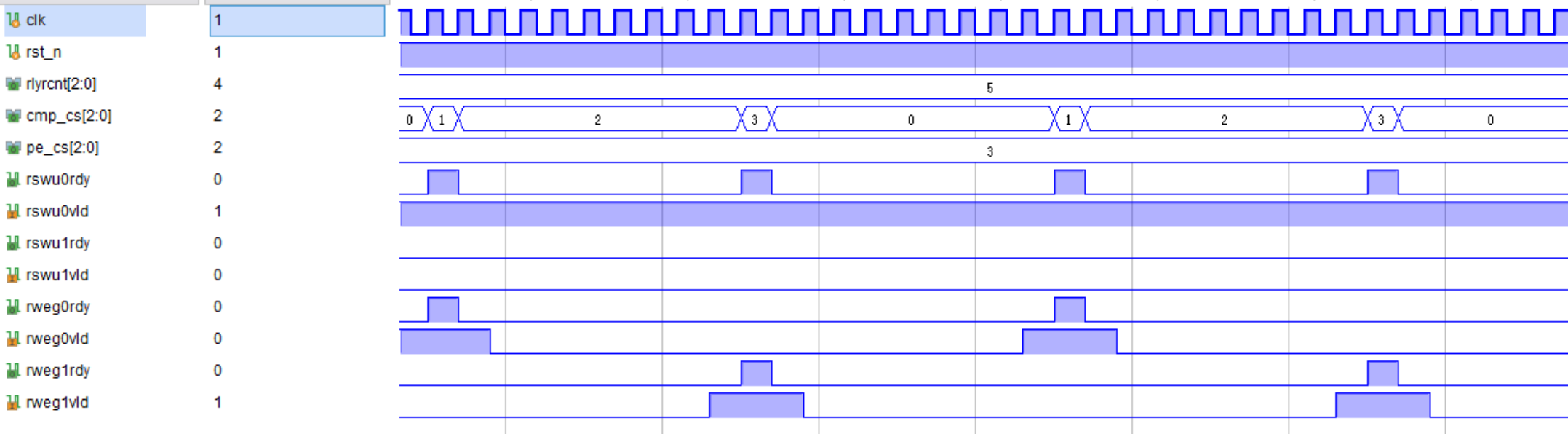


图4.13全连接层计算控制模块仿真波形图

## 计算模块

### PE\_ARRAY

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 3 | rlyrcnt | Ctrl pe当前计算层 |
| input | 3 | pe\_cs | Ctrl pe 主状态 |
| input | 3 | cmp\_cs | Ctrl pe 从状态 |
| input | 3 | cmp\_ns | Ctrl pe 从状态 |
| input | 6 | bin0input | Swu 0 的BIN输入 |
| input | 6 | bin1input | Swu 1 的BIN输入 |
| input | 32 | int0input | Swu 0 的INT输入 |
| input | 32 | int1input | Swu 1 的INT输入 |
| input | 192 | rweg0data | WEG 0 |
| input | 192 | rweg1data | WEG 1 |
| input | 1 | rswu0fin | SWU0 连续写入完毕 |
| input | 1 | rswu1fin | SWU1 连续写入完毕 |
| input | 1 | ricfin | 所有IC累加完毕，控制输出 |
| input | 1 | rolfin | 控制全局平均池化，所有输出累加完毕再输出 |
| input | 1 | rdenilfin | DEN CMP时所有IL计算完毕，控制输出 |
| input | 1 | thren | 输入阈值有效 |
| input | 256 | threshold | 32个8bit阈值 |
| output | 384 | Peout | 输出32个12bit的输出 |
| output | 1 | Peouten | 输出有效 |
| output |  | classout | 最后的5个输出 |

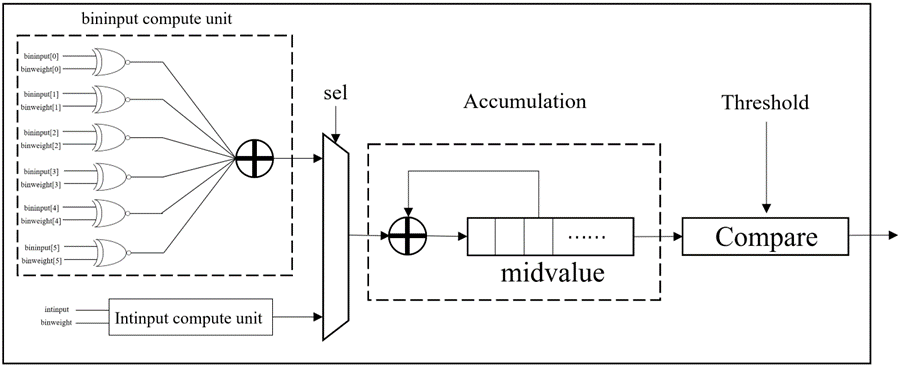
根据ctrl pe的主从状态机，控制pe en信号来使能PE阵列进行计算。

根据ctrl pe的主从状态机，来确定当前输入的权重和激活。

根据ricfin，rolfin等控制输出有效，使能PE阵列进行输出。

### PE

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 6 | bininput | BIN\_CMP |
| input | 32 | intinput | INT\_CMP |
| input | 6 | binweight | 权重 |
| input | 1 | outen | 输出使能 |
| input | 1 | sel | 选择int计算或bin计算结果 |
| input | 4 | acc\_sel | 选择中间值进行累加 |
| input | 11 | threshold | 阈值 |
| Input | 1 | avg\_pol\_signal | 全局平均池化时阈值放大18倍 |
| output | 12 | binoutput | 计算结果 |



Sel控制bin cmp还是int cmp，进行二选一。

Accsel控制中间值选取哪个值进行累加。特别的，对于AVG POL，accsel一致为0，实现所有数的累加。

## 缓存模块

### SWU

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 1 | wswuvld | Ctrl\_a写swu有效 |
| input | 1 | rswurdy | Ctrl\_pe读swu有效 |
| input | 32 | wswudata | Ctrl\_a写swu数据 |
| input |  | wswuaddr | Ctrl\_a写swu地址 |
| input | 1 | wswufin | Ctrl\_a写swu完毕 |
| input | 3 | pe\_cs | Ctrl 状态，用于指示ctrl\_pe读weg |
| intput | 1 | rswuicfin | Ctrl\_pe控制RSWU |
| input | 5 | rswuiccnt | Ctrl\_pe contrl read witch row in swu |
| output | 1 | wswurdy | SWU已被读取完毕，可以被写覆盖 |
| output | 1 | rswuvld | SWU中数据已经准备好被ctrl\_pe读取 |
| output | 32 | intinput | Ctrl\_pe从swu中读的数据,用于int计算 |
| output | 6 | bininput | Ctrl\_pe从swu中读的数据,用于bin计算 |
| output | 1 | rswufin | 拉低Ctrl\_pe的rswu\_rdy |

激活值缓存模块SWU

激活值缓存模块由一组32bit宽的存储单元组成。其深度由算法的卷积核长度以及PE阵列中输出缓存个数共同决定。在本设计中，由于第一层的卷积核长度较大，所以取其值为缓存模块的深度。SWU的示意图如图4.18所示，缓存控制模块从InOut Buffer中取一组32bit的数据写入SWU之中，然后SWU进行滑动，将选中的激活值广播到PE阵列。

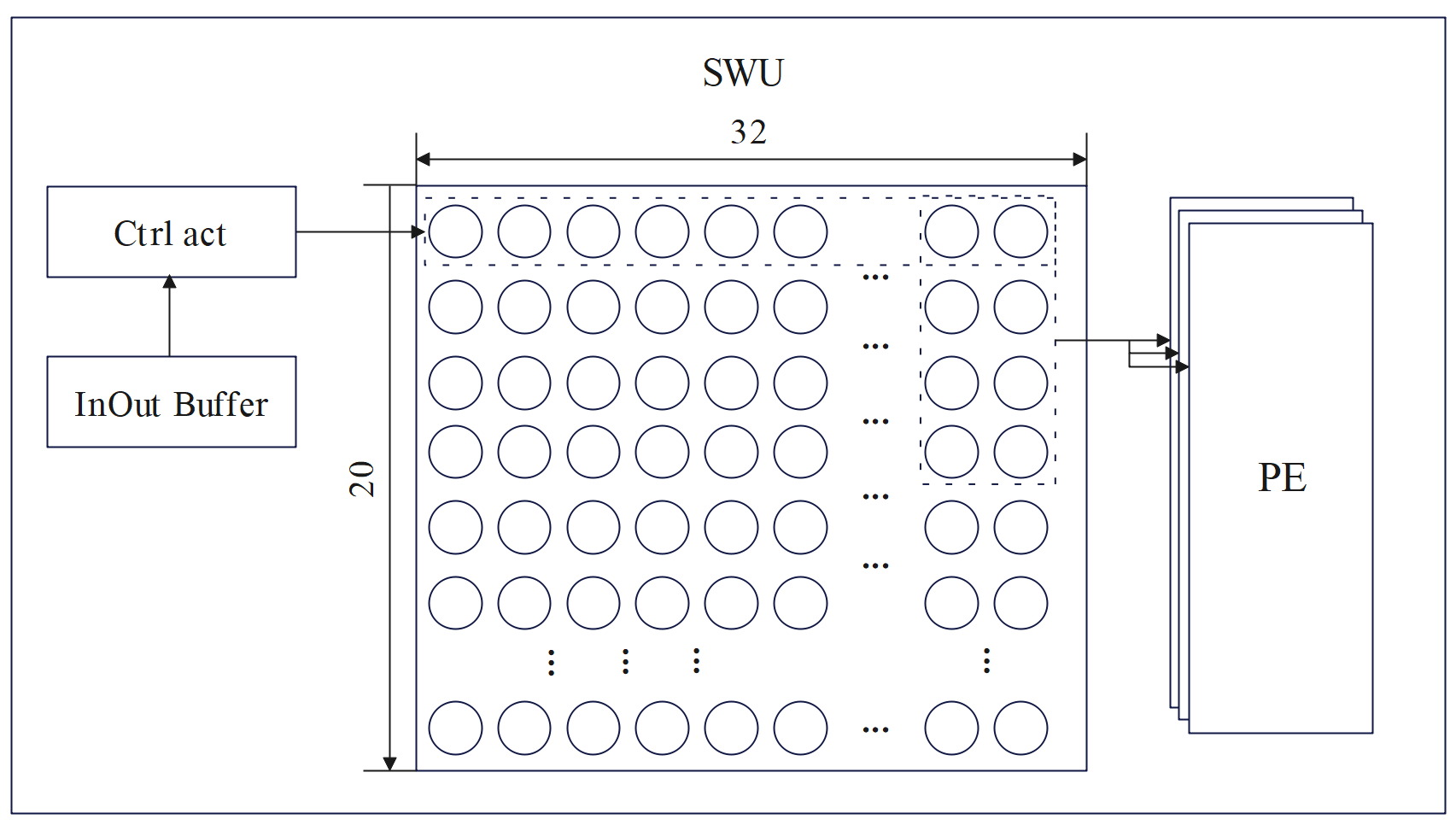


图4.18激活值缓存模块SWU

在存储控制单元从InOut Buffer中读取参数写入SWU时，写入的参数的位宽为32bit。SWU根据存储控制单元提供的写入地址，将写入的参数存储到对应的32bit存储单元之中。存放在SWU中的数据分为二值激活和整型激活，其存放方式与InOut Buffer中相同。对于整型数而言，每一行存储4个8bit的输入，一个SWU中存储80个8bit的输入，即一个完整的卷积核。PE中可以缓存的中间值数量为tmpwid个，即输出特征图的并行数TR。对于二值数而言，每行存放32个不同输入通道的1bit激活值，一共存放(tmpwid+kl-1)行，可供PE阵列滑动读取tmpwid次。

在计算控制单元从SWU中读取参数时，SWU根据计算控制单元的当前计算状态是整型输入计算还是二值输入计算，分别输出整型输入和二值输入。如图4.19(a)所示，当SWU输出二值输入时，每次选取TN列长度为KL的激活进行输出，在本文的设计中即是2列3行，共6bit的输出。如图4.19(b)所示，当SWU输出整型输入时，每次输出一行4个8bit的输出。激活缓存模块的输出是通过广播的方式发送到所有计算单元的，也就是说PE阵列中所有的PE单元共享相同的输入。

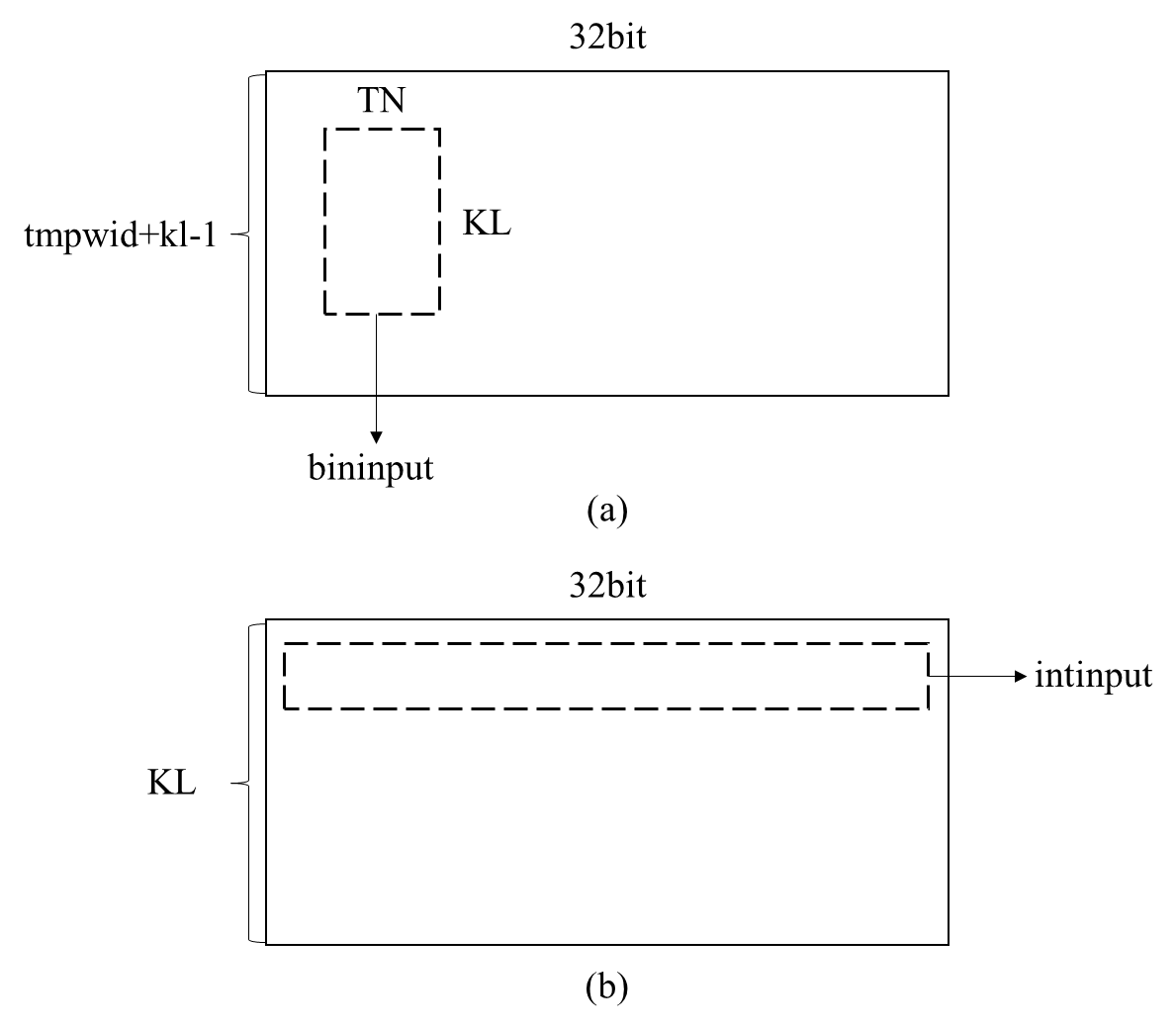


图4.19激活缓存模块SWU输出示意图

激活缓存模块的状态机如图4.20所示。当存储控制模块准备好写入时，且SWU处于写入空闲的状态IDLEW，则跳转到WSWU状态并进行参数的写入。当存储控制模块发出写入完毕信号wswufin之后，跳转到IDLER状态等待计算控制模块读取SWU内缓存的参数，同时拉高读有效的信号。当计算控制模块发送读准备信号rswurdy之后，跳转到RSWU状态进行参数的读取。SWU根据计算控制模块的状态进行计数，当参数读取完毕后拉高rswufin。对于二值的激活值输入而言，由于SWU中存放了多个输入通道的激活值，需要分多轮计算才能消耗完毕。因此在所有激活值消耗完毕即rswuicfin为低时，跳转到IDLER状态等待下一次的读取。否则在其他情况下，SWU中的参数已经被消耗完毕，可以被再一次写入覆盖，即跳转到IDLEW状态等待存储控制模块的写入。

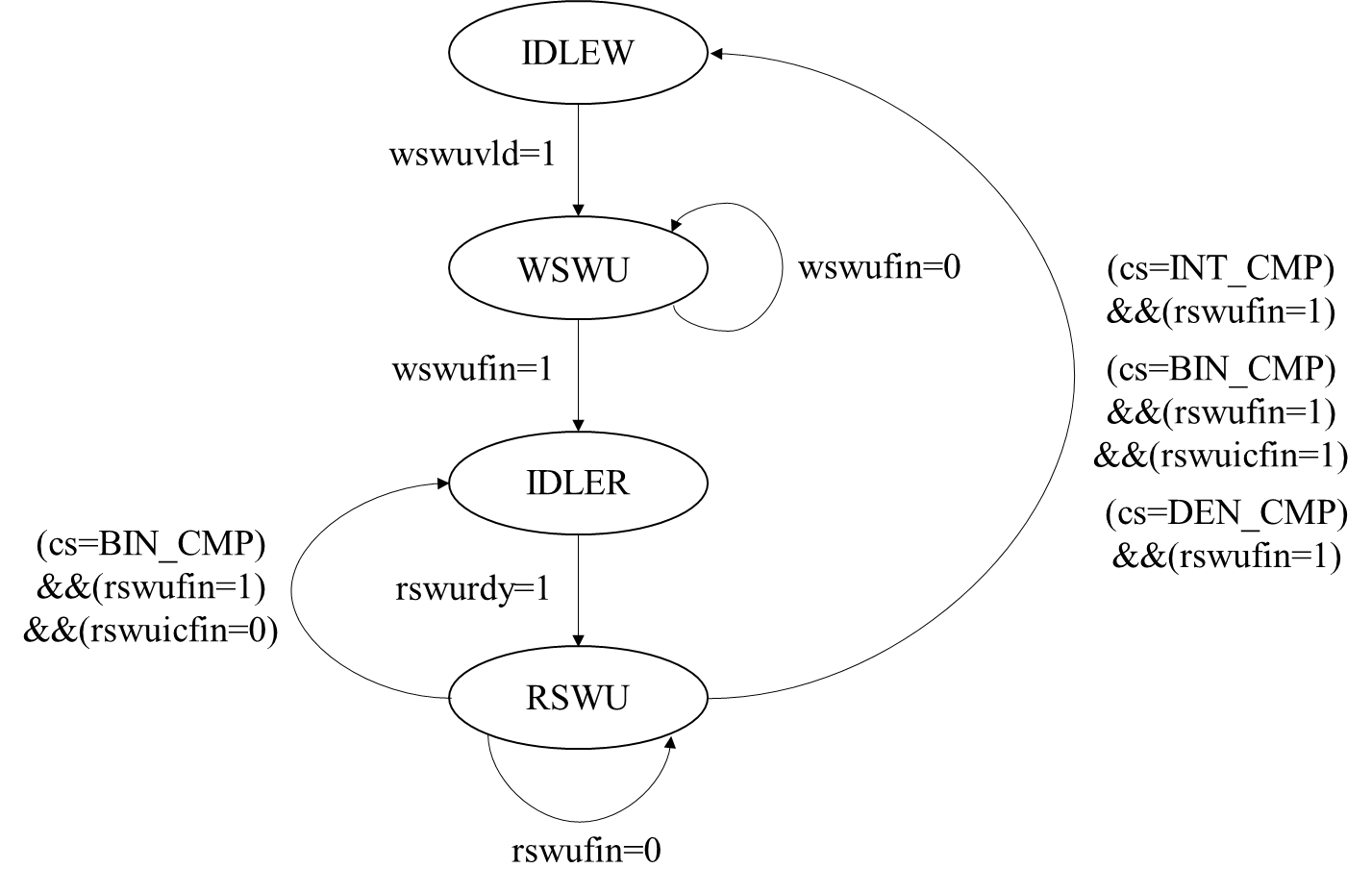


图4.20激活缓存模块SWU状态机

对激活缓存模块SWU进行功能仿真。图4.21是加速器进行第一层整型激活值卷积计算时SWU读写激活值的仿真图。在进行整型激活的卷积计算时，由于激活的位宽更大，有限的缓存空间难以进行复用，SWU需要不断的进行读写。输入的wswudata由两个SWU共用，通过握手协议控制SWU当前的读写状态。可以看到向SWU中写入20个数据后拉低wswurdy，同时拉高rswuvld等待被读取。

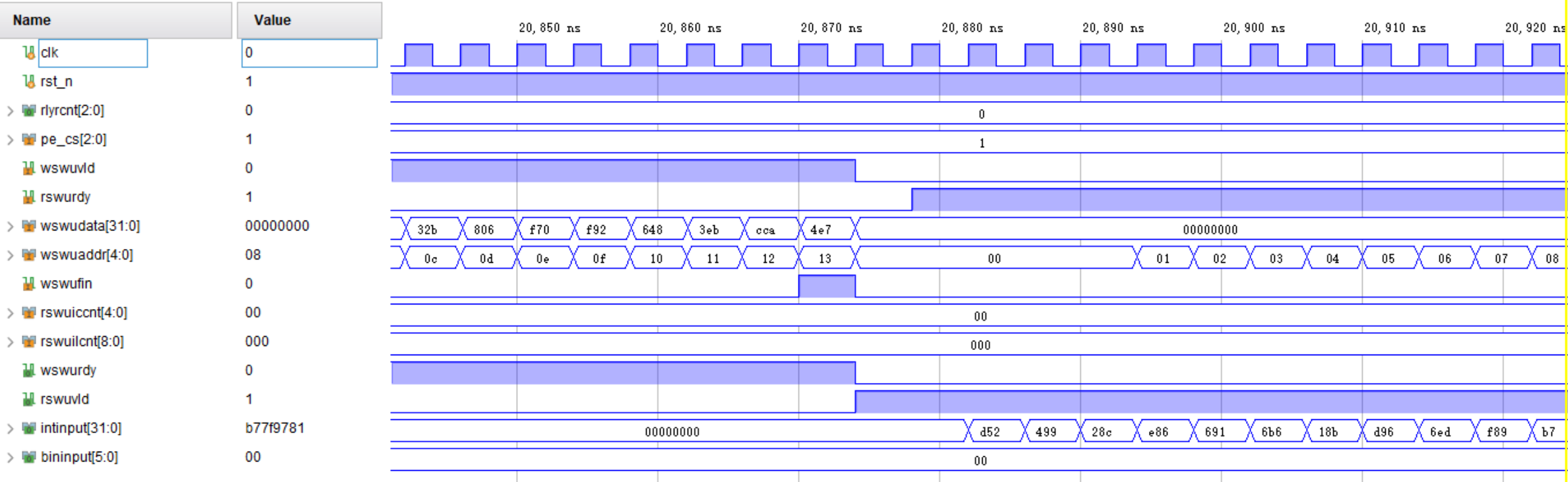


图4.21激活缓存模块整型激活值读写仿真图

当SWU缓存二值激活时，SWU可以滑动输出以对数据进行复用，对SWU读取二值激活的功能仿真如图4.22所示。当和计算控制模块握手成功之后，在下一个周期开始连续滑动12次，输出两个通道的12个输入激活值。输入完毕之后切换到下两个输入通道，等待计算阵列的读取。

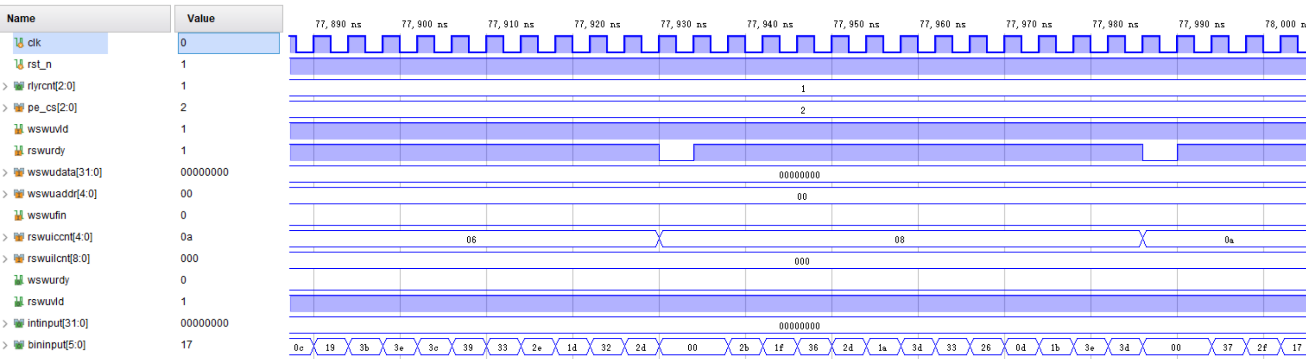


图4.22激活缓存模块二值激活值读取仿真图

### WEG

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 位宽 | 信号名 | 注释 |
| input | 1 | clk |  |
| input | 1 | rst\_n |  |
| input | 1 | wwegvld | Ctrl\_w写weg有效 |
| input | 1 | rwegrdy | Ctrl\_pe读weg有效 |
| input | 32 | wwegdata | Ctrl\_w写weg数据 |
| input |  | wwegaddr | Ctrl\_w写weg地址 |
| input | 1 | wwegfin | Ctrl\_w写weg完毕 |
| input | 3 | pe\_cs | Ctrl\_pe 状态，用于指示ctrl\_pe读weg |
| input | 1 | rwegolfin | Ctrl\_pe控制RWEG |
| output | 1 | wwegrdy | Weg已被读取完毕，可以被写覆盖 |
| output | 1 | rwegvld | Weg中数据已经准备好被ctrl\_pe读取 |
| output | 32 | rwegdata | Ctrl\_pe从weg中读的数据 |
| output | 1 | rwegfin | 拉低Ctrl\_pe的rweg\_rdy |

权重值缓存模块WEG

权重缓存模块WEG是由一组存储单元组成，其深度由卷积核的大小决定，而宽度则和输出通道的并行数TM相关。本加速器采用32个输出通道的并行计算，因此WEG的存储单元位宽是32bit。而本文算法中第一层的卷积核较大，因此深度采用第一层卷积核的大小为80。权重缓存模块的示意图如图4.23所示，缓存模块的每一列负责一个PE的权重的输入。

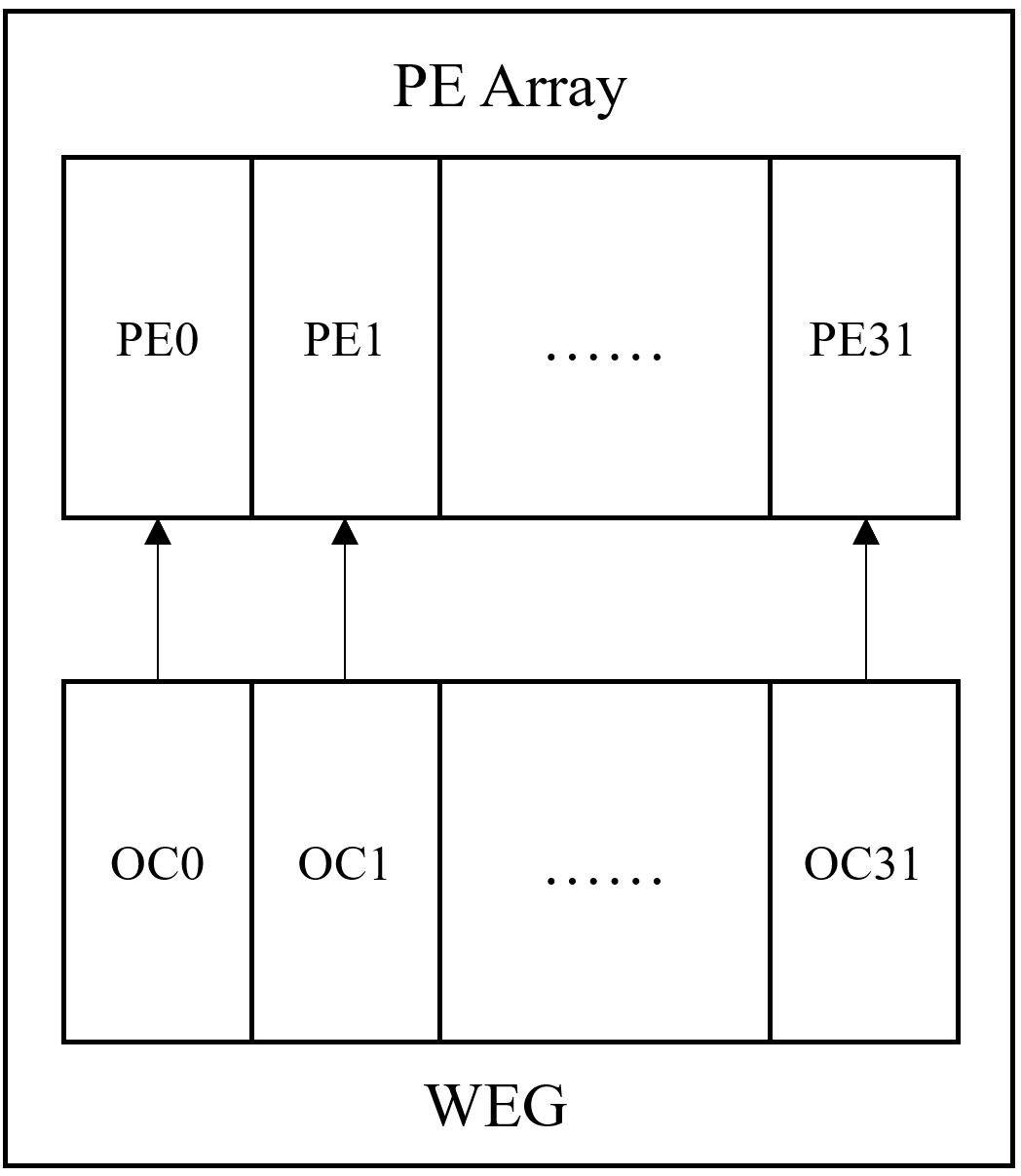


图4.23权重缓存模块示意图

在存储控制单元从Weight Buffer中读取参数写入WEG时，写入的参数的位宽为32bit。WEG根据存储控制单元提供的写入地址，将写入的参数存储到对应的32bit存储单元之中。存放在WEG中的权重参数的存储方式和在Weight Buffer中相同。整型输入的权重如图4.24(a)所示，每个32bit存储单元中的每一位权重属于不同的输出通道，即存储单元的每一列代表一个卷积核。二值输入的权重如图4.24(b)所示，和整型输入的权重相同，每列存储不同的卷积核。在此之外由于本加速器实现两个输入通道的并行，因此每列存储了一个输出通道的两个输入通道。

在计算控制模块从WEG中读取参数时，WEG根据计算控制模块的当前计算状态以不同的方式进行输出。若计算控制模块处于INT\_CMP状态，则从WEG中每次读取4行32bit的参数，且进行连续的参数输出，直至所有的参数被输出完毕。若计算控制模块处于BIN\_CMP或DEN\_CMP状态，则每次计算输出6行32bit的参数，并维持此参数输出不变直至计算控制模块跳转到空闲状态。WEG中的权重参数被分别送给32个不同的PE单元，因此在输出时每个PE单元会接收到输出参数的不同位。

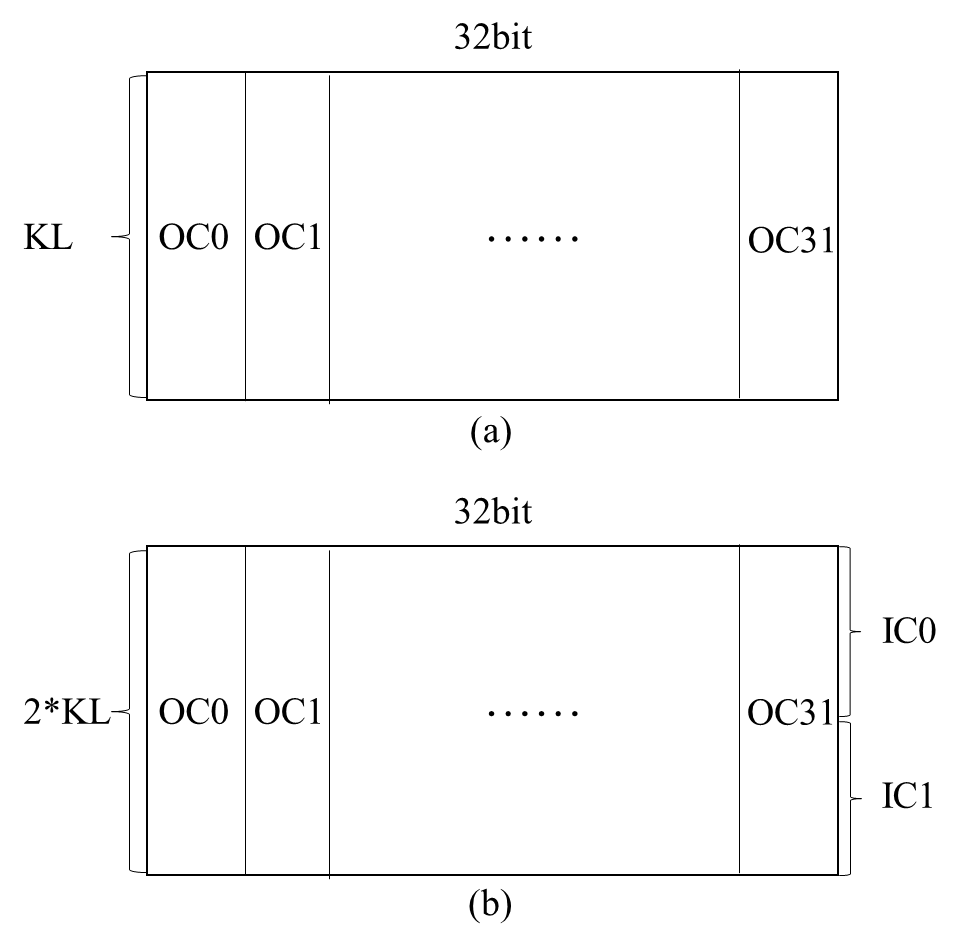


图4.24激活缓存模块WEG示意图

权重缓存模块的状态机如图4.25所示。当存储控制模块准备好写入时，且WEG处于写入空闲的状态IDLEW，则跳转到WWEG状态并进行参数的写入。当存储控制模块发出写入完毕信号wwegfin之后，跳转到IDLER状态等待计算控制模块读取WEG内缓存的参数，同时拉高读有效的信号。当计算控制模块发送读准备信号rwegrdy之后，跳转到RWEG状态进行参数的读取。WEG根据计算控制模块的状态进行计数，当参数读取完毕后拉高rwegfin。对于第一层整型输入的计算，即计算控制单元处于INT\_CMP时，由于第一层的输入通道只有一个，所以WEG中的权重无需进行更换即可完成整个输出通道的计算。因此WEG中的参数可以被多批次SWU中的参数重复使用，直至整个输出通道计算完毕即rwegolfin为高。因此在rwegolfin为低时跳转到IDLER等待计算控制单元下一次复用WEG中的参数。在其他情况下，WEG中的参数已经被消耗完毕，可以被再一次写入覆盖，即跳转到IDLEW状态等待存储控制模块的写入。

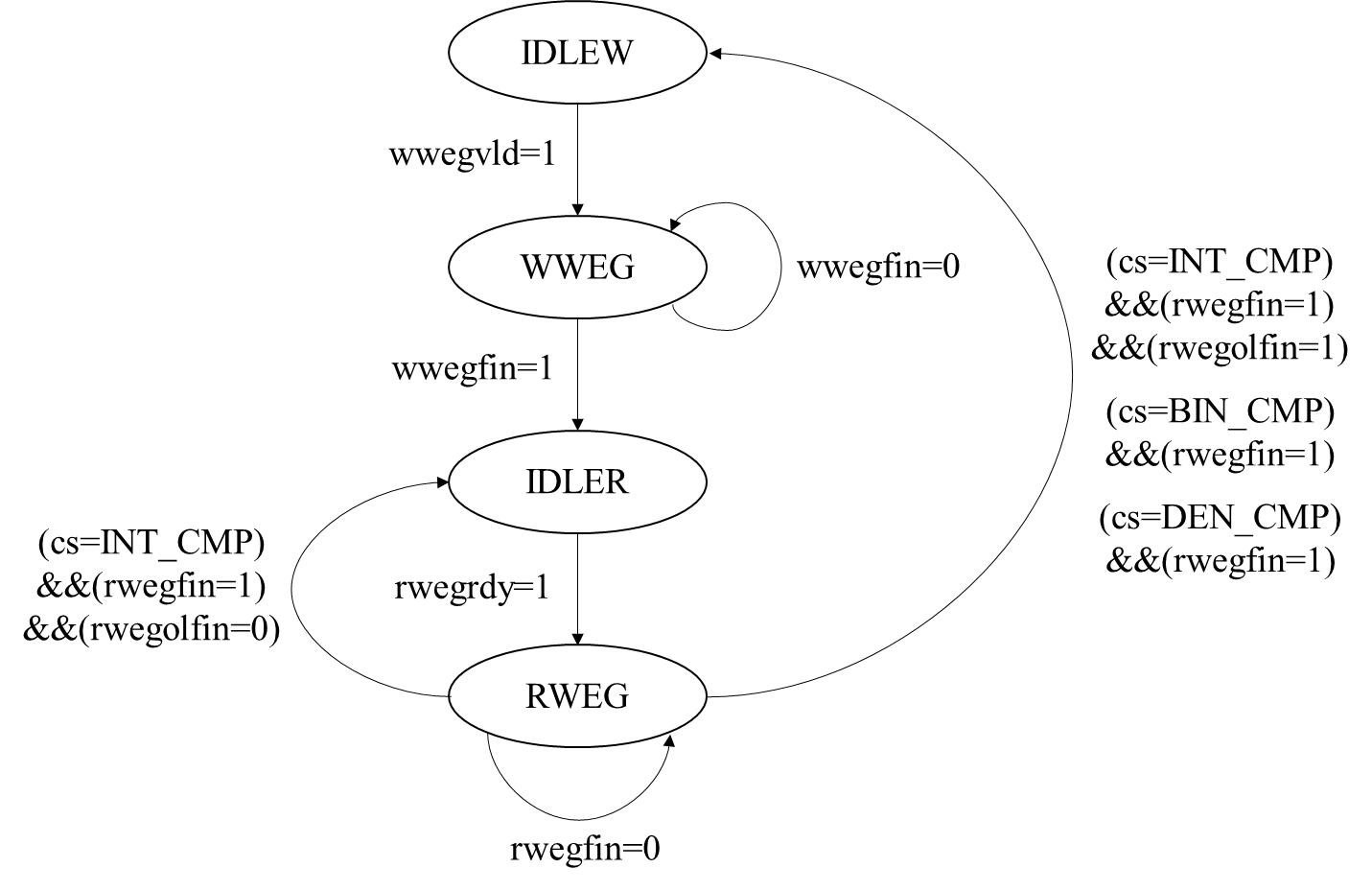


图4.25权重缓存模块WEG状态机

## 存储模块

### ACT MEM

**所有顺序都是从低到高**

激活值存储模块InOut Buffer

激活存储模块由两块Buffer组成，称之为Buffer0和Buffer1。这两块Buffer执行乒乓操作，轮流进行激活值参数的读写。即当计算卷积神经网络的一层时，存储控制单元从Buffer0中读取输入的激活值时，计算单元输出的激活值，也就是下一层的激活值输入会被存放到Buffer1之中。当进行下一层计算时，则从Buffer1中读取输入的激活值，而计算结果则直接输出到Buffer0中进行写覆盖。通过乒乓操作的方式提高了并行的效率，参数的写入和读出互不干扰。

激活值在InOut Buffer中的存储方式根据其类型不同也有所不同，如图4.26所示。在图4.26(a)中是整型激活的排布方式，每行存放4个8bit有符号整型输入。由于第一层的输出只有一个输入通道，是一个连续的一维数据，因此只需要连续的存放在InOut Buffer中即可。在图4.26(b)中是二值激活的排布方式，由于其他层拥有32的倍数个输入通道，所有InOut Buffer中每行的每一位都是不同的输入通道的激活值，即每一列是不同的输入通道。每列中一个输入通道的数被连续的存储，长度为输入的长度IL，其后则存储下一组32个输入通道的激活值。对于全连接层的激活值，由于其数量较少，仅为256bit或64bit，所以将其简单的连续存储到InOut Buffer中。

InOut Buffer的存储方式方便参数的存取，是由本加速器的并行方式所决定的。对于整型激活的读取，每次读取20行的输入，且下次读取的地址在上一次读取的地址基础上递增(KS/4)。对于二值激活的读取，每次读取(tmpwid+kl-1)行输入，一次取出32个输入通道。下一次则地址递增IL，读取下一组32个输入通道的激活值。同时，对于计算单元的输出而言，其存在32组输出，分别对应32个输入通道。因此可以将每组输入拼接至一个32bit数的不同位，然后存放到InOut Buffer中。

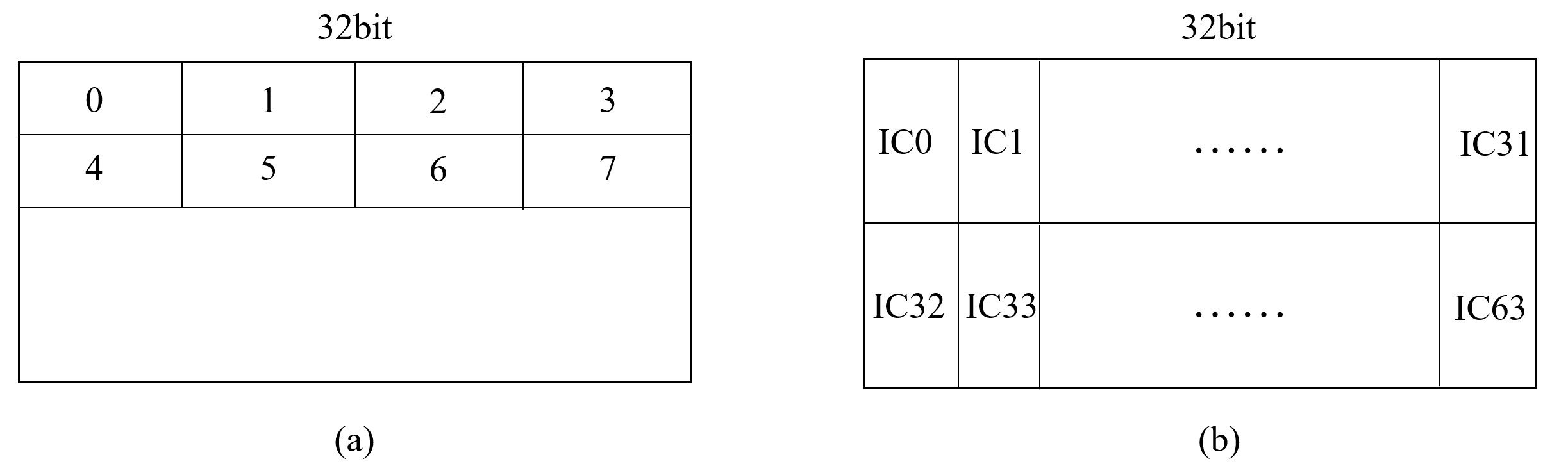


图4.26 InOut Buffer中数据排布方式

### WEG MEM

权重值存储模块Weight Buffer

权重存储模块是一块整体Buffer，且在计算过程中仅进行读取而不需要进行写入。如图4.27所示，权重存储模块的位宽是32bit，每bit存储不同输出通道的权重，即每一列存放不同输出通道的权重。由于卷积计算需要将所有输入通道计算完毕之后，中间值才可以计算完毕被输出。所以在同属于一个输入通道的32组输出通道权重之后，存放的是下一个输入通道的相同32个输出通道的权重。直到这组32个输出通道的权重的所有输入通道被存放完毕之后，再存放下一组32个输出通道的权重。对于全连接的权重而言，可以将全连接层的输出长度视为特征通道数，以相同的方式进行存储，而输入通道视为仅有一个。

权重存储模块的数据排布方式是由计算阵列的切片方式决定的。由于计算阵列的输出通道切片TM是32，输入通道切片TN是2，所以对于二值激活输入的计算，每次存储控制单元从Weight Buffer中读取一组32个输出通道的权重中的两个输入通道的权重。

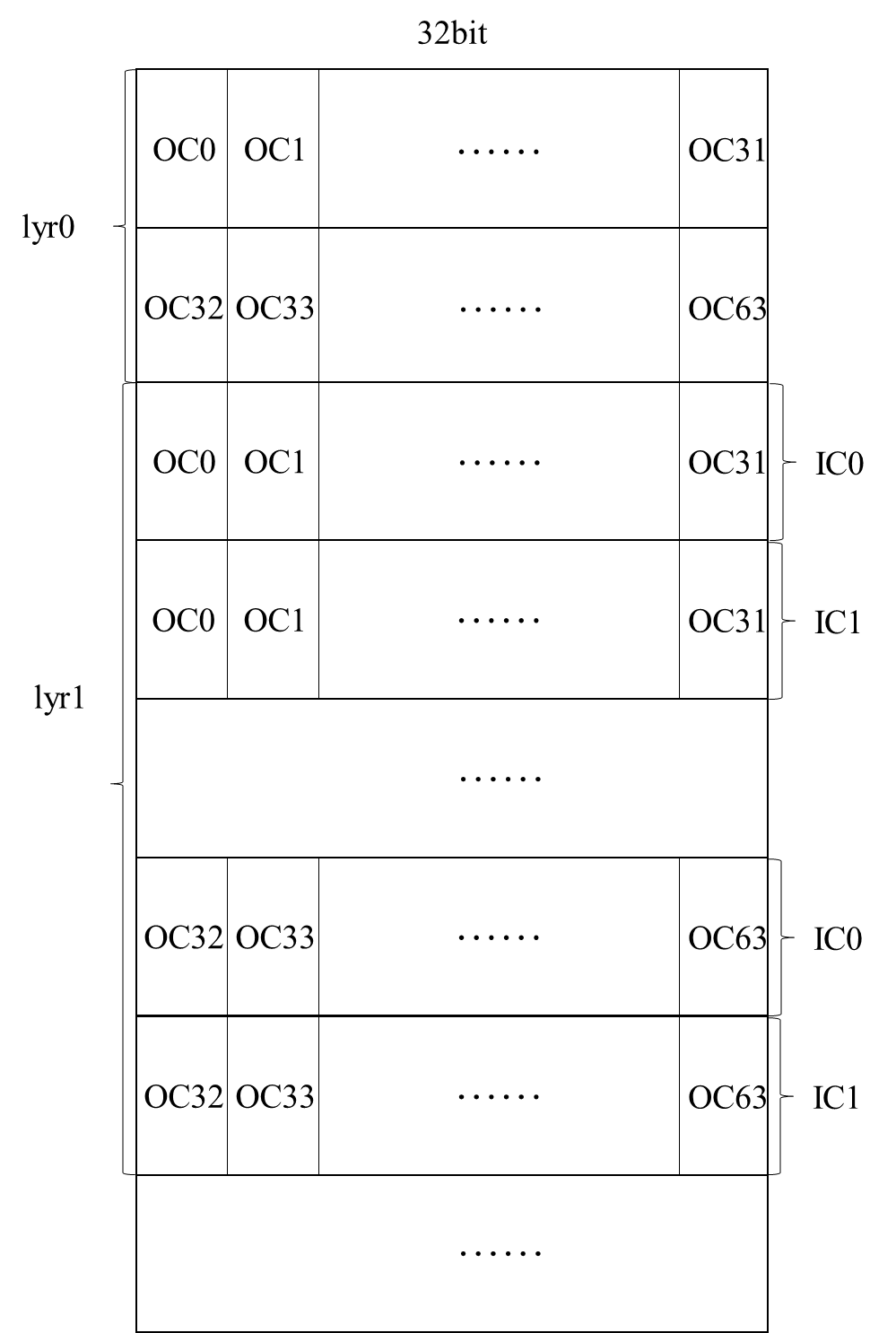
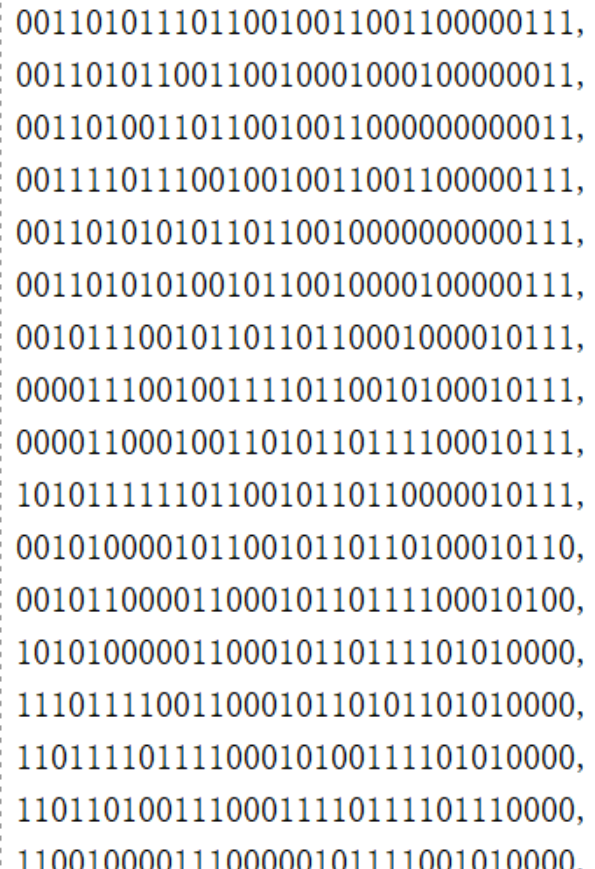


图4.27 Weight Buffer中数据排布方式

数据排布：

WEG：

WEG MEM中的数据排布



从高位到低位，代表的输出通道依次递增

地址从低到高，依次递增

WEG REG中的数据排布

一致

