HWPE: A CNN Accelerator for RISC-V

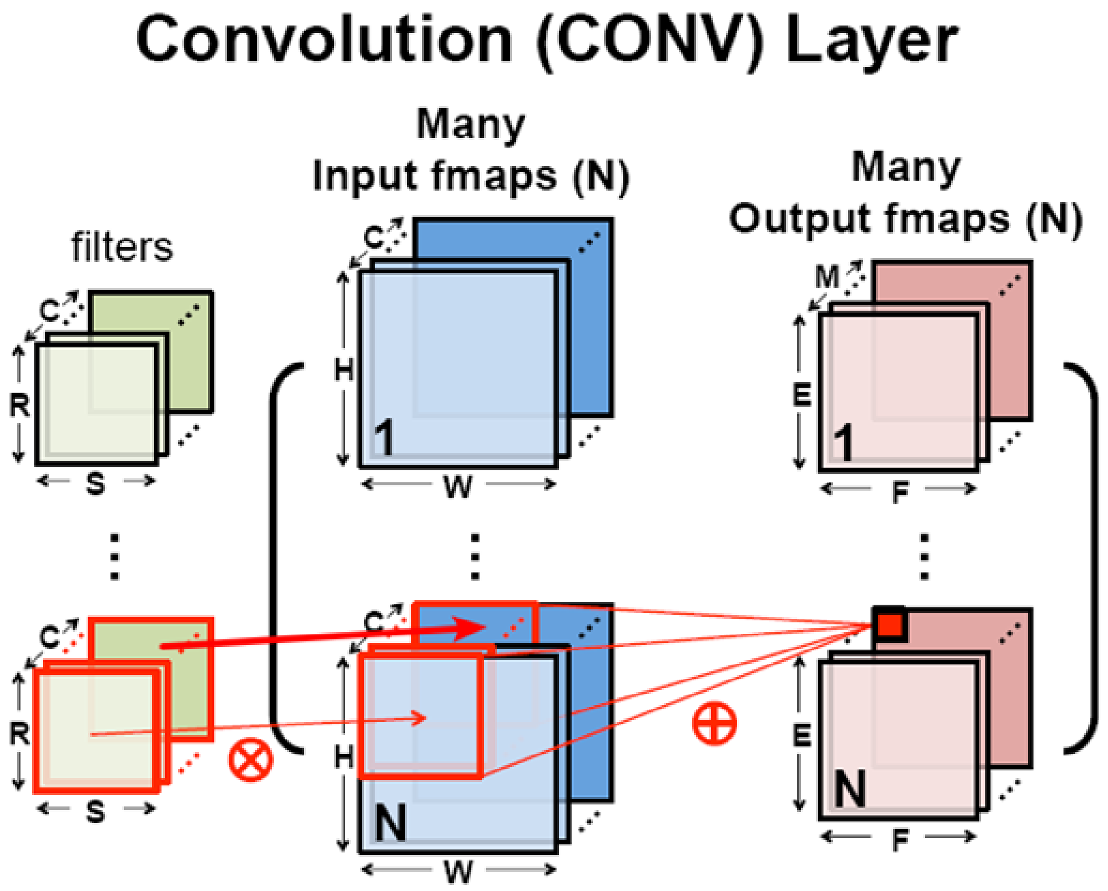
Author: Hao Chen , Qiang Chen

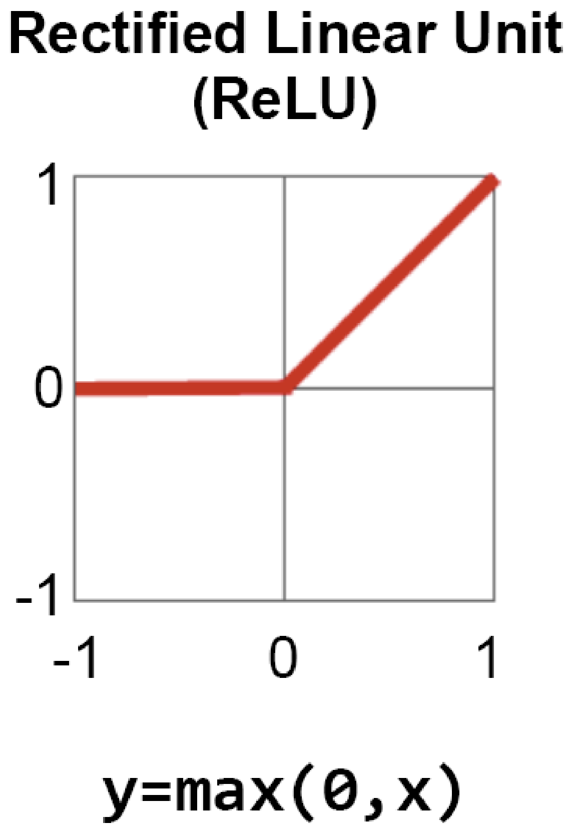
What can HWPE do?

HWPE（CNN Hardware Processing Engines）

- A CNN hardware acceleration coprocessor for RISC-V
- Support Convolution layer and ReLU layer
- Transform convolution into matrix multiplication (im2col on the fly)
- Kernel size from 3×3 to 11×11
- Support data type int8(uint8), exp4 (4 bits of exponential scale) and ternary
- 16 dot-product operations of 64 bits operands(8 int8, 16 exp4 or 32 ternary) per cycle
- Complete a convolution layer operation with one configuration

Convolution & ReLU

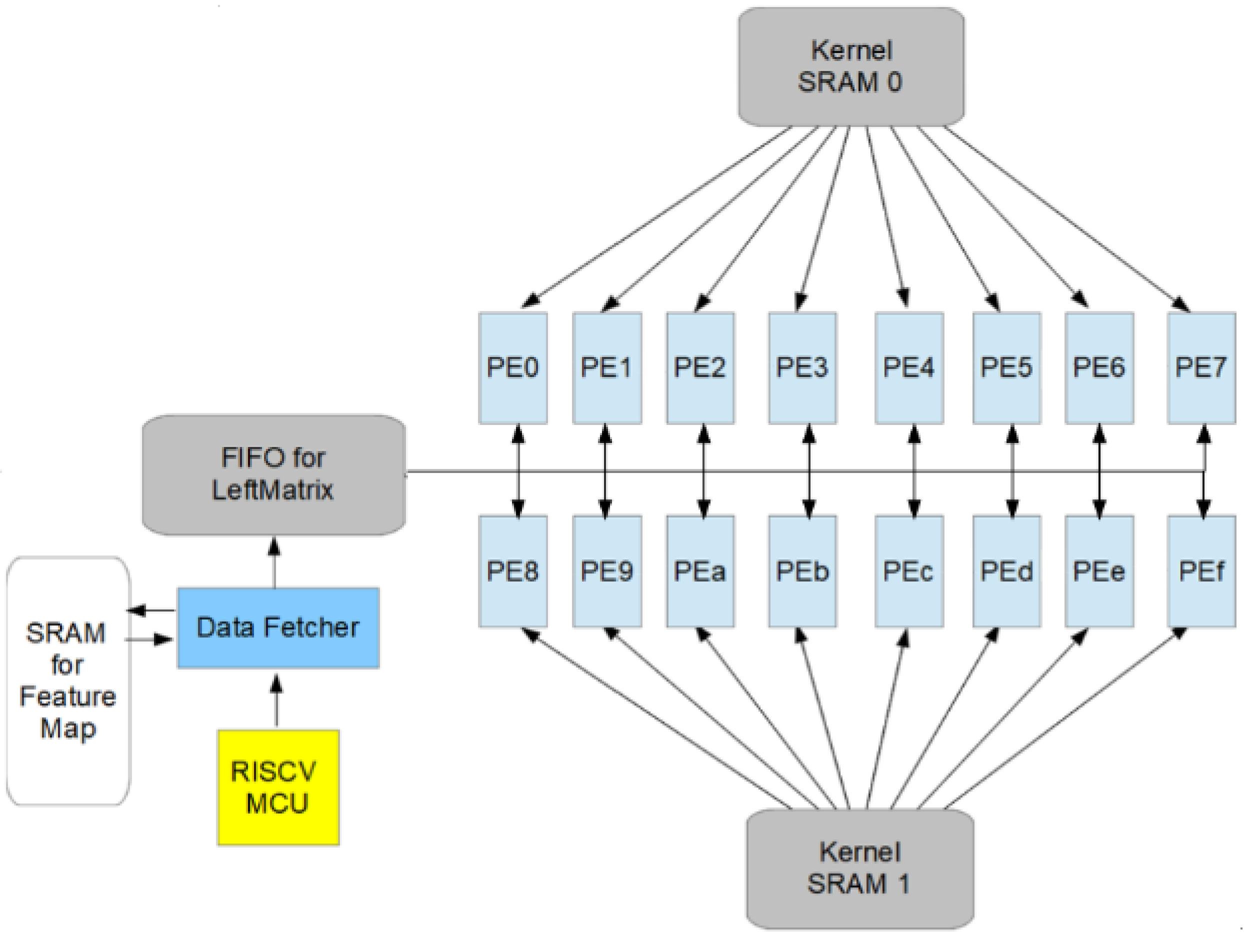




- Fully-Connected layer and pooling layer are still calculated by MCU

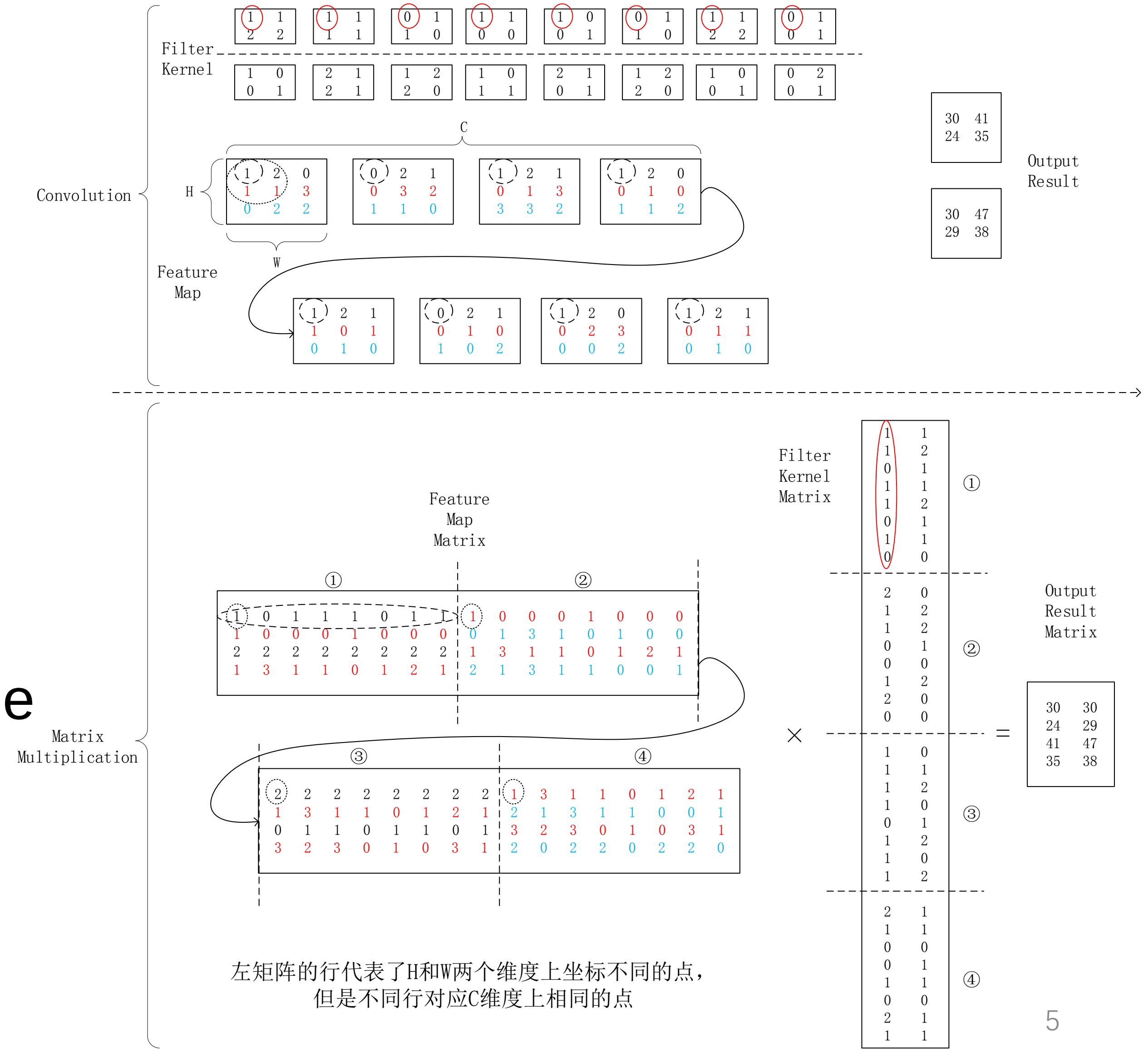
Architecture Diagram

- Data fetcher accepts the command from MCU and fetches data from Feature Map SRAM and pushes them into FIFO in the way of im2col
- FIFO is used as a buffer for the left matrix
- 16 PE, each one stores a 64 bits right matrix, performs multiply-accumulate and stores a column result matrix
- Two kernel SRAMs, each one broadcasts the data of the right matrix to 8 PE



Convolution / Matrix Multiplication

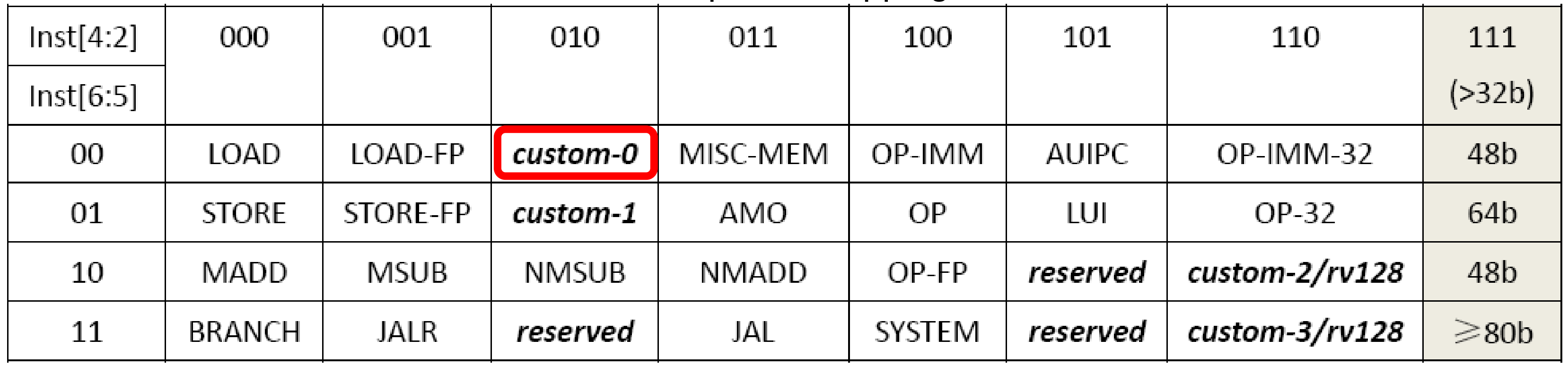
- Transform convolution into matrix multiplication
- Left Matrix : data-fetcher calculate memory address to access data in the way of im2col
- Right Matrix : store the kernel in the way of channel priority in advance



HWPE Instructions

- HWPE has only 7 instructions based on custom-0 opcode of RISC-V extended instruction set

RISC-V basic opcode mapping table



Inst[4:2]	000	001	010	011	100	101	110	111
Inst[6:5]								(>32b)
00	LOAD	LOAD-FP	***custom-0***	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	***custom-1***	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	***reserved***	***custom-2/rv128***	48b
11	BRANCH	JALR	***reserved***	JAL	SYSTEM	***reserved***	***custom-3/rv128***	⩾80b

HWPE Instructions table

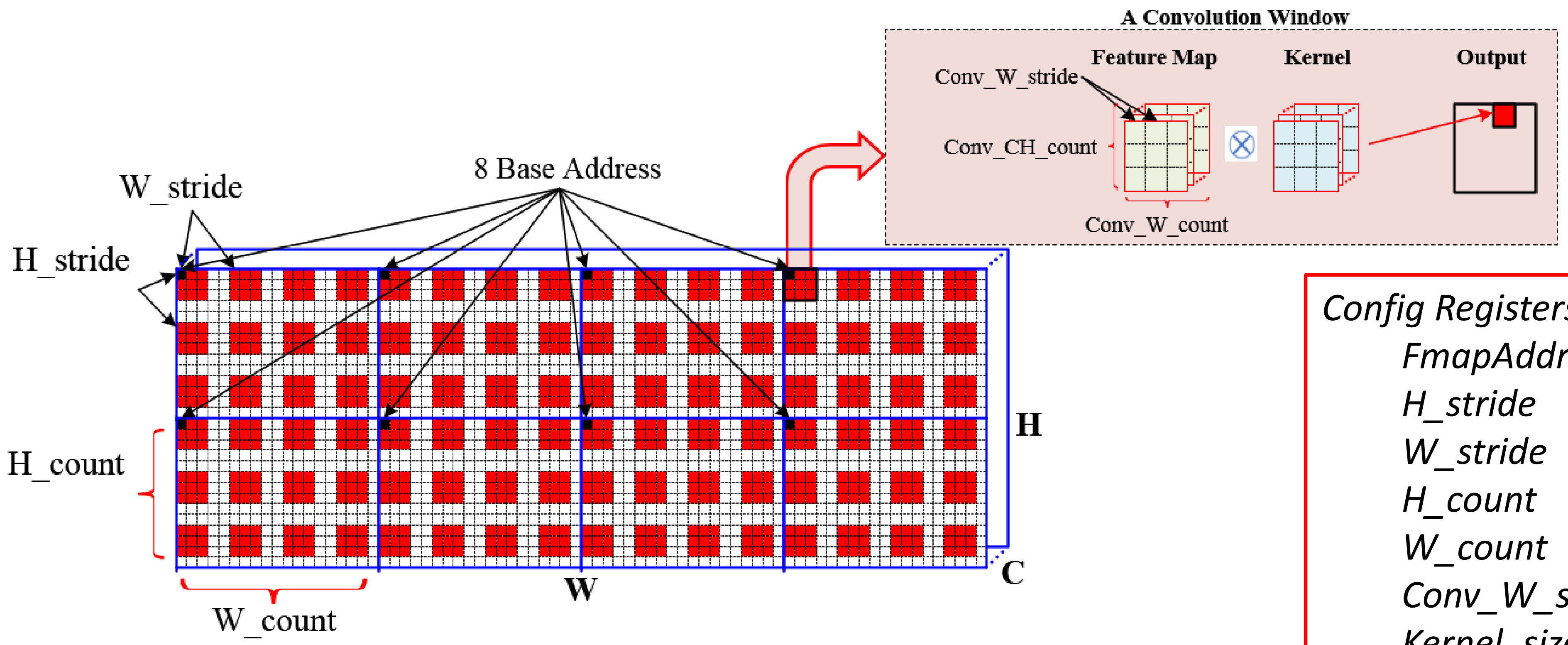
Instruction	funct	rd	xd	rs1	xs1	rs2	xs2
HWPEWriteFmapAddrReg	1	addr_idx	0	addr1	1	addr2	1
HWPEWriteCfgReg	2	——	0	cfg	1	cfg	1
HWPEMatrixMac	4	——	0	W/H_count	1	H/W_stride	1
HWPEWriteAccReg	8	AccReg_ID	0	M(idx)	1	PE_ID	0
HWPEReadAccReg	16	M(idx)	1	Acc_Reg_ID	0	PE_ID	0
HWPEReLUMemWriteAccReg	32	——	0	M(addr)	1	AccReg_ID	0
HWPEReset	64	——	0	——	0	——	0

What do HWPE Instructions do?

1	*HWPEReset*	perform a synchronous reset for the HWPE registers
2	*HWPEWriteFmapAddrReg*	configure 8 base address registers which point to the 8 initial points of the H-W plane
3	*HWPEWriteCfgReg*	configure control registers for convolution
4	*HWPEWriteAccReg*	reset or set AccRegs / adds the bias in convolution layers
5	*HWPEMatrixMac*	start first round of convolution operation
6	*HWPEReadAccReg*	read the AccRegs back to the MCU general registers
7	*HWPEReLUMemWriteAccReg*	execute ReLU operation and transfer the 32 bits AccRegs to 8 bits, then store the results into the specified memory address

How to run CNN: Config Registers

- *A convolution task is determined by the configuration registers.*



Config Registers:
FmapAddrBase[8]
H_stride
W_stride
H_count
W_count
Conv_W_stride
Kernel_size
Conv_CH_count

Head address of convolution windows:

■ FmapConvAddr[i] = FmapAddrBase[i] + x·W_stride + y·H_stride   0≤x≤H_count; 0≤y≤W_count

SRAM memory access address:

🟥 FmapSramAddr[i] = FmapConvAddr[i] + m·Conv_W_stride + n·Conv_CH_stride

0≤m≤Conv_CH_count; 0≤n≤Kernel_size

How to run CNN

1. Configure the configuration registers with *HWPEWriteFmapAddrReg* & *HWPEWriteCfgReg*
   - select 8 initial points on the H-W plane as base address, each one is the head address in the first convolution window.
2. Start the first round of convolution operations with *HWPEMatrixMac.* 128(8x16) output points are calculated.
   - In a round of computation, the data is fetched to perform dot multiplication. The address of the data is calculated according to registers *Conv_W_stride and Kernel_size Conv_CH_count*
   - Multiply 8 points in the H-W plane of feature map and16 different kernels to obtain 128 output points.
3. Read the output back to the MCU general register for further operation with *HWPEReadAccReg*
   or execute ReLU operation and transfer the 32 bits output to 8 bits, then store into the specified memory address with *HWPEReLUMemWriteAccReg*
4. HWPE continues the next round calculation until the last *HWPEReadAccReg/ HWPEReLUMemWriteAccReg*
   - Convolution windows move by the size of **stride**, first in the H direction and then in the W direction.
   - Head address of convolution window will be updated according to registers *W_stride H_sride H_count W_count* every round*.* Then the data is fetched to perform dot multiplication in the way of im2col.
5. When calculate *H_count*W_count*K_count* rounds, the entire convolution task is completed.
   - A H-W plane needs *H_count*W_count* rounds computation.
   - For K kernels, the entire convolution task needs H_count*W_count*K_count rounds.(K_count=K/16)

Convolution with One Configuration

- *In general, one configuration can complete a convolution layer operation.*

- Select 8 initial points and move according to the configuration registers to cover the entire H-W plane as far as possible.
- If not, reconfigure the configuration registers and select new 8 initial points to cover the entire H-W plane.
- MCU can also pad the input feature map before storage, making it easy to finish the convolution operation with one configuration.

PE Utilization

Type	Kernel size	PE utilization
internal layer	any size	100%
input layer	3x3x3	84.38%
	5x5x3	93.75%
	7x7x3	87.50%
	9x9x3	96.43%
	11x11x3	82.50%

Conclusion

- Currently implemented on the EAI interface of hummingbird E200 MCU(not verified yet)
- Can be easily transplanted to other RISC-V coprocessor interfaces
- *https://github.com/chenhaoc/cnnhwpe/*
- Include Matlab Model / C Model / RTL
- **Total equivalent gate count : 190K**
- **Peak performance(16PE): 256xFreq OPS(INT8)/512xFreq OPS(EXP4)**

  /1024xFreq OPS(Ternary)
- HWPE architecture is flexible and efficient. It is suitable for CNNs acceleration on the resource-limited devices.

In the Coming Work

- Simulate with EAI interface of hummingbird E200
- software framework

- ***Looking for volunteers to finish it!***

Thanks For Your Attention!