HWPE: A CNN accelerator for RISC-V

author: chenhaoc chenqiang

HWPE is a CNN hardware acceleration coprocessor based on RISC-V. It is currently implemented on the EAI interface of hummingbird E200 MCU.

Features:

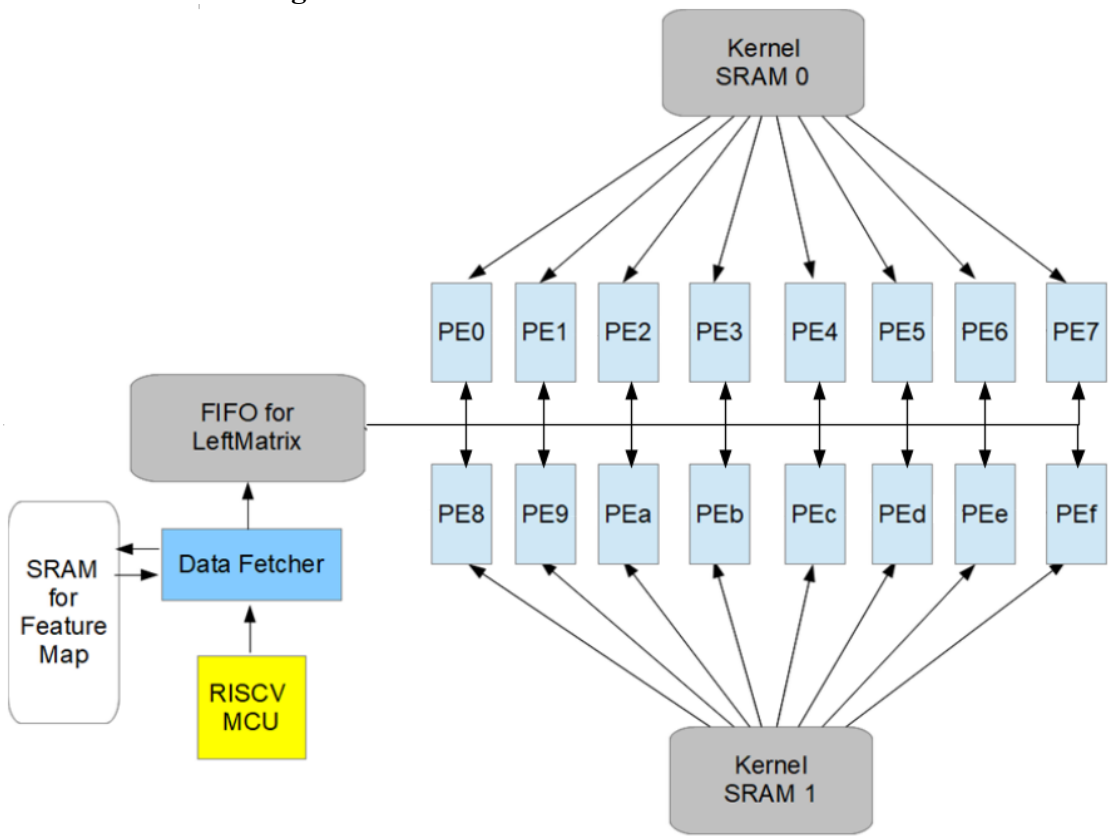
HWPE supports the mainstream CNN model acceleration, with kernel size from 3×3 to 11×11, data type INT8, EXP4 (4 bits of exponential scale) and ternary.

16 dot-product operations are completed in each cycle, and the operands of dot-product are 64 bits (8x8, 16x4 or 32x2).

The convolution operation is implemented by transforming convolution into matrix multiplication.

The PE utilization is 100% in the internal layer calculation. For the input layer, the utilization is 84.375%(11×11×3),82.5%(7×7×3),87.5%(5×5×3),93.75%(3×3×3).

Architecture diagram of HWPE:



This accelerator includes:

1. A FIFO is used to buffer the left matrix with 64-bit entries, and data fetcher constantly pushes new data into it. The FIFO pops the left matrix data out in turn per cycle.

2. Data fetcher accepts the command from MCU and fetch data from Fmap (Feature Map) SRAM and push them into FIFO. Fmap SRAM has at least two banks for data fetcher to fetch two 64-bits data at the same time.

3. 16 PE, each of which stores a column (64 bits) of the right matrix, and a column of the result matrix (which requires eight 32-bit result registers which called *AccRegs*).

4. Two Kernel SRAMs, each one broadcasts the data of the right matrix to 8 PE.

Convolution Acceleration

HWPE performs hardware acceleration of the convolution layer in CNNs. HWPE must be reset before performing normal operations. The MCU shall send a *HWPEReset* instruction to perform a synchronous reset for the HWPE registers. A convolution task is determined by the configuration registers. The *HWPEWriteFmapAddrReg* instruction configures eight base address registers which point to the eight initial points of the H-W plane. *HWPEWriteCfgReg* configures control registers for convolution. *HWPEMatrixMac* starts the first round of convolution operations. Each round of convolution operations can be performed by multiplying 8 points in the H-W plane of input feature map (calculated by 8 base address registers and the current state) and the 16 different kernels to obtain 128 output points. This operation is called for a basic calculation. After this instruction is completed, MCU sends the *HWPEReadAccReg* instruction to read the output back to the general register for further operation, or sends the *HWPEReLUMemWriteAccReg* to directly execute ReLU operation and transfers the 32 bits *AccRegs* to 8 bits, then store to the specified memory address as the final result of part of the current convolution layer, which is also the input for the next layer. When *HWPEReadAccReg* / *HWPEReLUMemWriteAccReg* is executed last time, the HWPE continues the next round of basic operations after outputting. According to the configuration information, the entire convolution task is completed after calculating *H_count*W_count*K_count* times.

The *HWPEWriteAccReg* instruction adds the bias in convolution layers. This instruction should also be used to reset or set *AccRegs* before performing a convolution operation. (To reduce area and power consumption, the *AccRegs* does not have reset logic.) *AccRegs* are automatically reset to 0 after the *HWPEReadAccReg* instruction and the *HWPEReLUMemWriteAccReg* instruction.

HWPE contains 8 first address registers of convolution window. In a round of computation, the data is fetched to perform dot multiplication in the way of im2col from the first address of convolution window. Then the first address of the convolution window increase by the size of stride, first in the H direction and then in the W direction, and the next round computation will start. A H-W plane need H_count*W_count rounds computation.

If the initially selected 8 points on the H-W plane can cover the entire H-W plane in accordance with the configuration registers to move, the calculation of the whole convolution layer will be completed in once configuration. If not, then 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 at once.

The ReLU layer can be done directly by *HWPEReLUMemWriteAccReg*. Other layers such as fully connected layers are still calculated by MCU.

In conclusion, HWPE is a flexible architecture for implementing convolution operations in CNNs. HWPE will automatically update the address and access the storage to perform matrix multiplication according to the configuration registers once configured. Under normal circumstances, one configuration can complete the convolution operation for a layer in a neural network. This architecture is suitable for implementing CNNs acceleration on the resource limited devices.

H: Height of input feature map W: Width of input feature map
H_count: Iterations needed in the H dimension of input feature map.
W_count: Iterations needed in the W dimension of input feature map.

K_count: Iterations needed in the K dimension of weights.

Appendix: List of HWPE Instructions

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

Github: https://github.com/chenhaoc/cnnhwpe