QUICK-SPEC DO BLOCO DE CONVOLUÇÃO CQInt8

(para os íntimos, CONVOLUQUINHAS)

Esse bloco deve realizar uma convolução quantizada (Int8) que se adapta às dimensões da imagem e do kernel, por meio de controle ao acesso na memória. As entradas (imagens) e os kernels (filtros) são guardados de forma linearizada na memória. Para realizar tal processo, ele deve apresentar as seguintes funcionalidades:

- Ler valores guardados na memória:
    - **M_0 (32b),** que é a constante de quantização
    - **N (8b),** que é outra constante de quantização
    - **C (16b),** referente ao bias (viés) da rede quantizada
    - **Kernel (x_k*y_k com elementos de 8b),** o filtro aplicado
- Ler valores da entrada guardados na memória, de modo a indexar as janelas válidas a enviar para a UNIT.
- Caso haja *pooling*, realizar tal operação dentro do bloco e escrever na memória

Deve se comunicar com o bloco Unit de modo a utilizá-lo para finalizar o processamento da convolução em questão, sendo essa parte referente ao somatório da multiplicação ponto a ponto entre imagem e kernel. Para isso, o bloco deve se comunicar com a memória para ler ou escrever em endereços específicos. Essa interface deve conter os seguintes sinais de dados:

- uma entrada de leitura de dados (32b) vindo da memória
- uma entrada de envio de dados (32b) para a memória
- parâmetros das dimensões do imagem e do kernel (16b cada)
- uma saída de dados com o endereço para acesso à memória
- uma saída de dados com o dado para escrita na memória

Também deve conter os seguintes sinais de controle:

- um *valid* de entrada para indicar saída válida
- um *enable* de entrada, para indicar que o bloco está sendo usado
- uma de entrada de escrita bem sucedida na memória
- uma saída requisitando uma escrita na memória
- uma entrada indicando leitura válida na memória
- uma saída requisitando uma leitura da memória

Os sinais listados podem ser vistos na visão de topo do bloco, ilustrado na Figura 1. Cada sinal é associado à sua interface, na respectiva lateral do bloco.

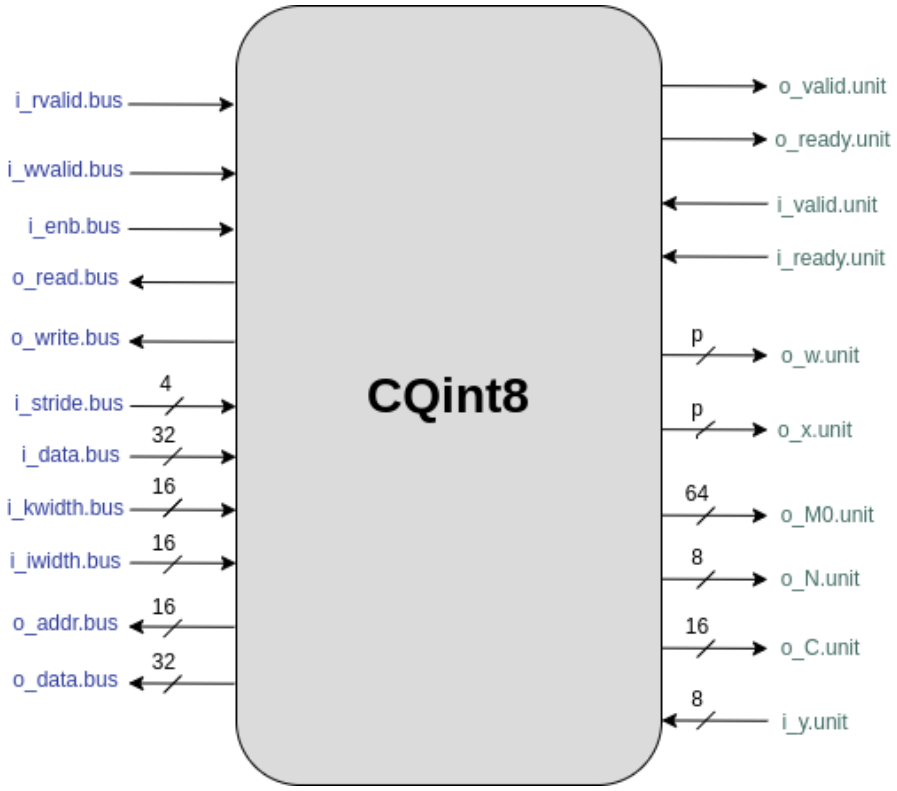

Figura 1: Topo do módulo CQint8.

De posse dessas informações, o bloco deve obedecer uma lógica de leitura da memória para obter as janelas válidas em ordem, de acordo com o passo (esse dado é enviado pelo controlador, em um registrador interno). Para cada janela válida, o módulo pode se comunicar com a Unit, indicando uma janela válida (sem as transições de coluna) e mandando essa, o kernel e as constantes de quantização. Os dois últimos são inicialmente lidos da memória antes da leitura das entradas, como disposto no mapa de memória da Figura 2. O resultado da inferência é escrito em uma região reservada para ela.

Os dados de quantização são lidos sequencialmente pois suas larguras definidas no projeto permitem regularidade no acesso. Como o kernel é um vetor unidimensional de bytes (8b), este pode ter uma quantidade de elementos que não preenche uma palavra inteira (32b) na sua extremidade. Um exemplo é o caso de um filtro 5x5, que possui 25 pixels e teria que ser guardado em 4 palavras (4*7 = 28, tendo (28-25)*8 = 24 bits ou 3 bytes preenchidos com zero). Os campos de cada byte depende de como o dado é lido.

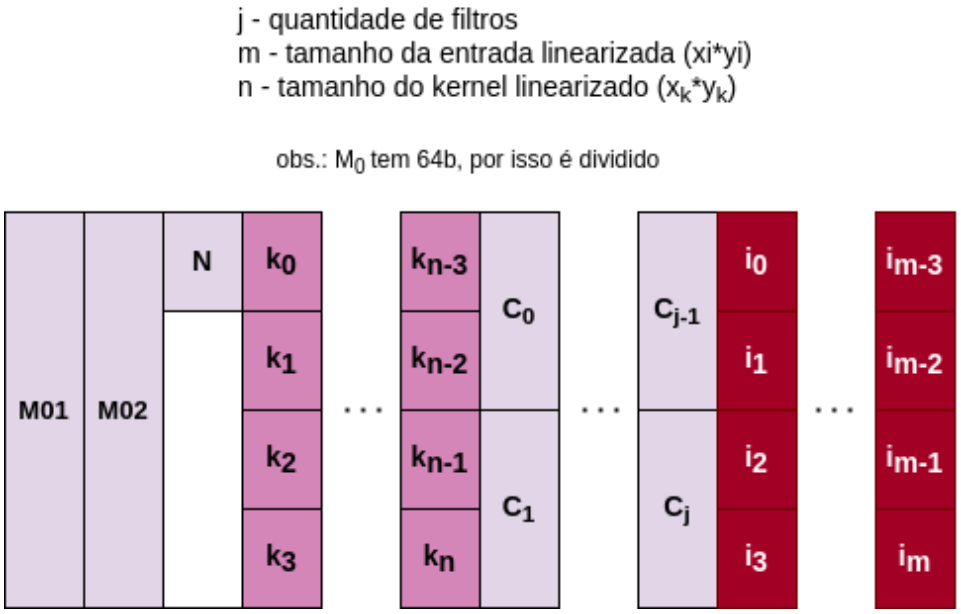

Figura 2: mapa de memória da convolução.

A leitura das entradas é a parte mais complexa do processo. A imagem é guardada assim como o kernel, mas os valores a serem carregados para uma única janela estão dispostos em colunas diferentes. Por isso, para um stride igual a 1, deve-se seguir a seguinte sequência de passos:

1. Ler da memória os 4 primeiros bytes da primeira linha, e verificar se a horizontal do filtro (x_k) é menor ou igual que a quantidade de bytes inserida. Se sim, pular para a próxima linha, e se não, repetir a etapa com os próximos 4 bytes da mesma linha.
2. Repetir o passo 1 nas próximas linhas até chegar no limite do eixo vertical do kernel (y_k).
3. Armazenar em uma FIFO essas primeiras janelas válidas. Nessa etapa, já tem-se dados suficientes para formar algumas janelas, então não ocorre leitura da memória. Essa quantidade é igual a VW = x_4w - x_k, sendo x_4w o mínimo múltiplo de 4 que comporta pelo menos uma janela, variando de 0 a 3.
4. Ler da memória 32 bits de cada uma das primeiras linhas (y_k).
5. Deslizar a janela 4 vezes, identificando quais são válidas.
6. Carregar 4 bytes da linha y_k.
7. Voltar para o passo 5, até o janelamento chegar na última janela válida.

Havendo uma camada de *pooling* na rede, o módulo deve receber e armazenar da Unit os valores da saída da camada para executar a função. Assim, o bloco deve calcular a saída reduzida e armazenar na memória, para então ser usada como entrada de outra camada de convolução ou da inferência.