A Reconfigurable Streaming Deep Convolutional Neural Network Accelerator for Internet of Things

Li Du, *Member, IEEE*, Yuan Du, Yilei Li and Mau-Chung Frank Chang, *Fellow, IEEE*

03/19/2022

The main contribution of this paper include:

1. A CNN accelerator design using streaming data flow to achieve optimal energy efficiency.

2. An interleaving architecture to enable parallel computing for multiple output features without SRAM input bandwidth increment.

3. A methodology to decompose large-sized filter computation to be many small-sized filter computation, achieving high reconfigurability without adding additional hardware penalty.

4. A supplementary pooling block that can support pooling function while the main engine serves for CNN computation.

5. A prototype design with FPGA verification, which can achieve a peak performance of 152 GOPS and energy efficiency of 434 GOPS/W.

Filter Decomposition

To minimize the hardware resource usage, a filter decomposition algorithm is proposed to compute any large kernel-sized(>3*3) convolution through using only 3*3-sized CU.

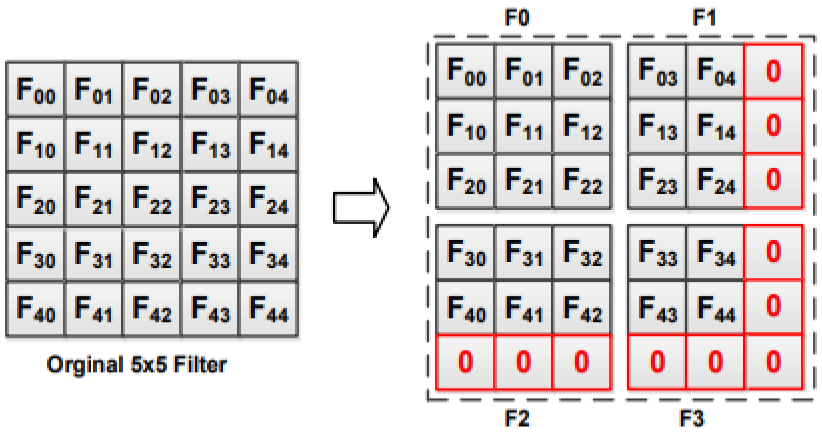
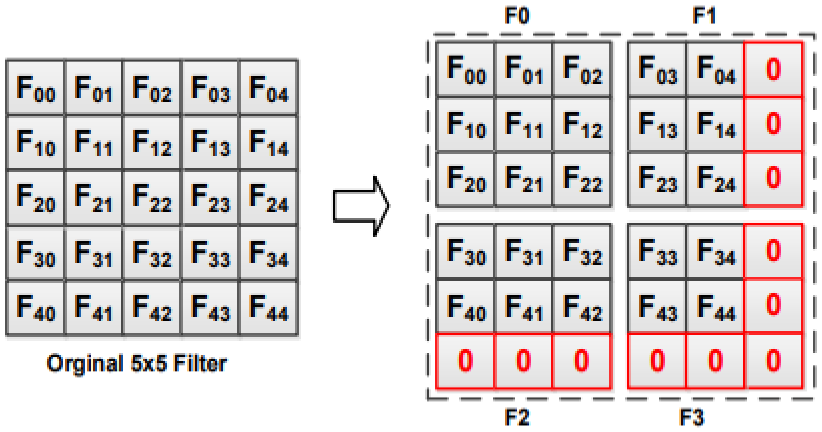


Fig.5 An 5x5 Filter decomposed into four 3x3 sub filter. F0, F1, F2, F3's shift address are (0,0), (0,3), (3,0), (3,3).

The Arithmetical Derivation of this Filter Decomposition

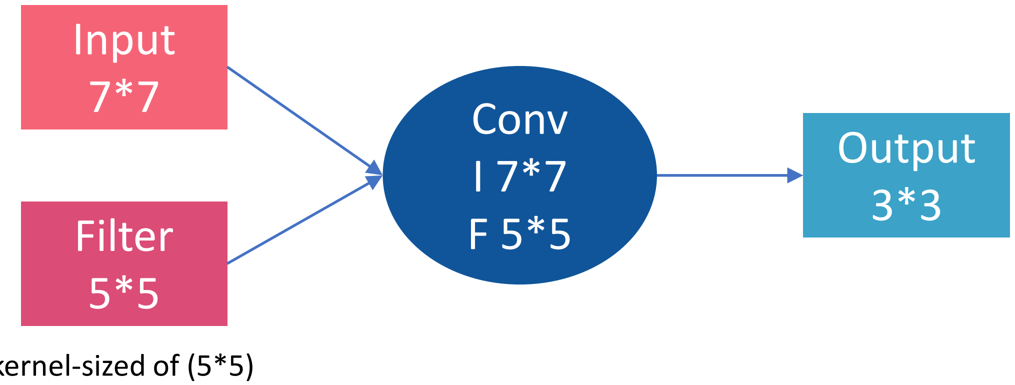


$$
\begin{aligned}
F_{3K}(a,b) &= \sum_{i=0}^{3K-1}\sum_{j=o}^{3K-1} f(i,j)\times I_i(a+i,b+j)\\
&= \sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{l=0}^{2}\sum_{m=0}^{2} f(3i+l,3j+m)\times I_i(a+3i+l,b+3j+m)\\
&= \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} F_{3\_i\_j}(a+3i,b+3j) \qquad (5)
\end{aligned}
$$

$$F_{3\_i\_j}(a,b) = \sum_{m=0}^{2}\sum_{l=0}^{2} f(3i+l,3j+m)\times I_i(a+3i+l,b+3j+m)$$

$$0\le i< K-1;\ 0\le k< K-1; \qquad (6)$$

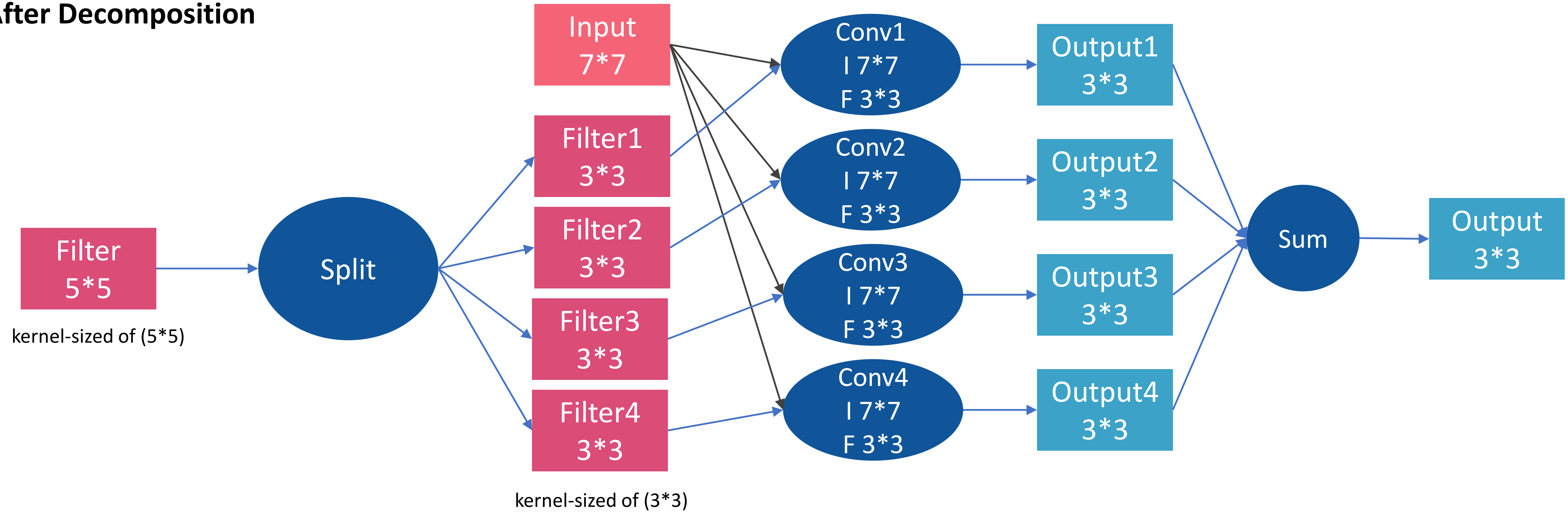
Before Decomposition



Example :

To decompose **kernel-sized of (5*5)** computation to **4 kernel-sized of (3*3)** computations.

After Decomposition



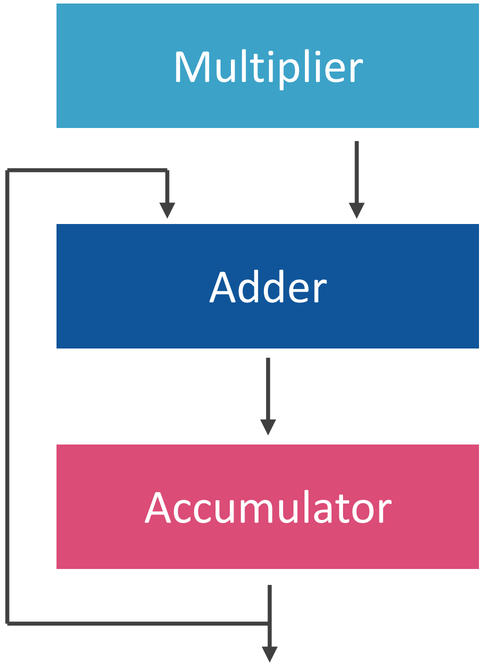
The arithmetical derivation of this filter decomposition can be described as (5)

$$
\begin{aligned}
F_{3K}(a,b) &= \sum_{i=0}^{3K-1}\sum_{j=o}^{3K-1} f(i,j)\times I_i(a+i,b+j) \\
&= \sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{l=0}^{2}\sum_{m=0}^{2} f(3i+l,3j+m)\times I_i(a+3i+l,b+3j+m) \\
&= \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} F_{3\_i\_j}(a+3i,b+3j)
\end{aligned}
\tag{5}
$$

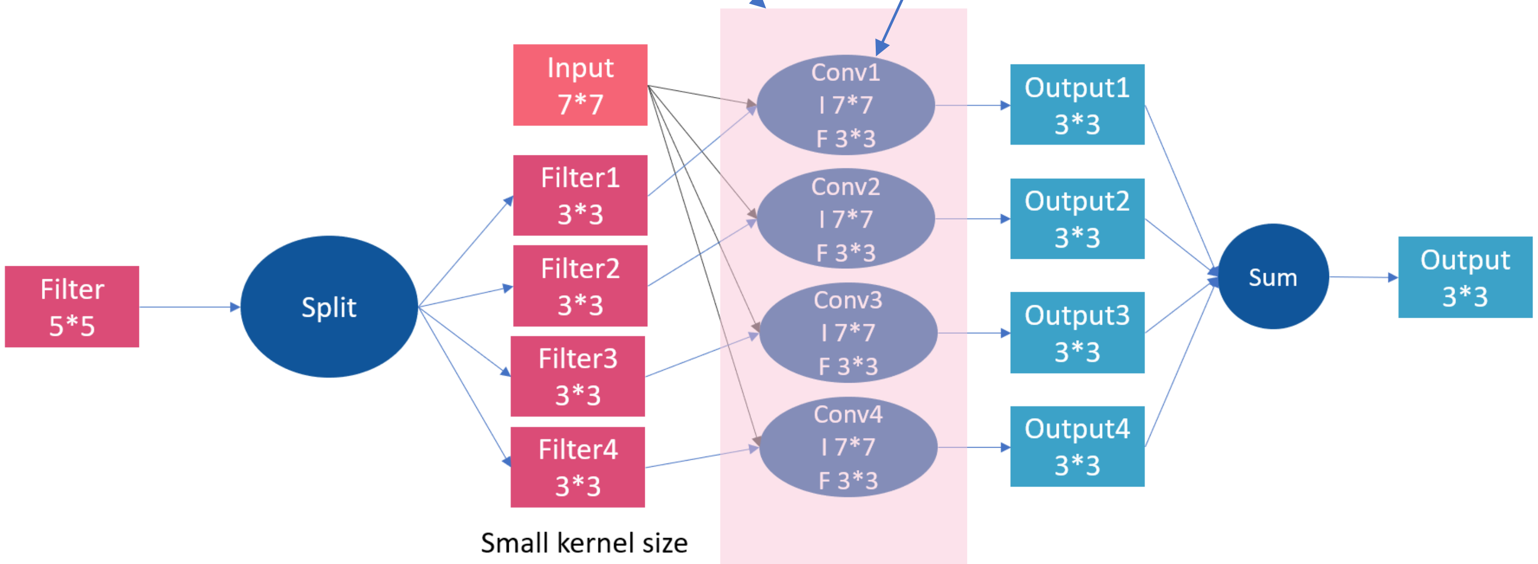
$$
\begin{aligned}
&F_{3\_i\_j}(a,b) = \sum_{m=0}^{2}\sum_{l=0}^{2} f(3i+l,3j+m)\times I_i(a+3i+l,b+3j+m) \\
&0 \le i < K-1;\ 0 \le k < K-1;
\end{aligned}
\tag{6}
$$

MAC Unit

```
#hardware execution
def conv_3x3(image, filter):
    kernel_size = 3
    acc = 0
    for i in range(kernel_size):
        for j in range(kernel_size):
            Image = image[i][j]
            Kernel = filter[i][j]
            multiple = Image*Kernel
            acc = acc + multiple
    return acc
```



The MAC unit provide hardware support



To Implement the Filter Computation

$$
\begin{aligned}
F_{3K}(a,b) &= \sum_{i=0}^{3K-1}\sum_{j=o}^{3K-1} f(i,j)\times I_i(a+i,b+j) \\
&= \sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{l=0}^{2}\sum_{m=0}^{2} f(3i+l,3j+m)\times I_i(a+3i+l,b+3j+m) \\
&= \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} F_{3\_i\_j}(a+3i,b+3j) \qquad (5)
\end{aligned}
$$

$$
F_{3\_i\_j}(a,b) = \sum_{m=0}^{2}\sum_{l=0}^{2} f(3i+l,3j+m)\times I_i(a+3i+l,b+3j+m)
$$

$$
0 \le i < K-1; 0 \le k < K-1; \qquad (6)
$$

MAC Unit

```
#hardware execution
def conv_3x3(image, filter):
    kernel_size = 3
    acc = 0
    for i in range(kernel_size):
        for j in range(kernel_size):
            Image = image[i][j]
            Kernel = filter[i][j]
            multiple = Image*Kernel
            acc = acc + multiple
    return acc
```

```
output = []
for a in range(output_h):
    for b in range(output_w):
        acc=0
        for i in range(K):
            for j in range(K):
                #
                curr_region = image_with_zero_padding[a+3*i+0 : a+3*i+3,  b+3*j+0 : b+3*j+3]
                curr_filter = filter_with_zero_padding[3*i+0 :   3*i+3,   3*j+0 :   3*j+3]
                # hardware execution
                result = conv_3x3(curr_region, curr_filter)
                acc = acc + result

        output.append(acc)
print(output)
output = array(output)
output = output.reshape((output_h, output_w))
```

Appendix

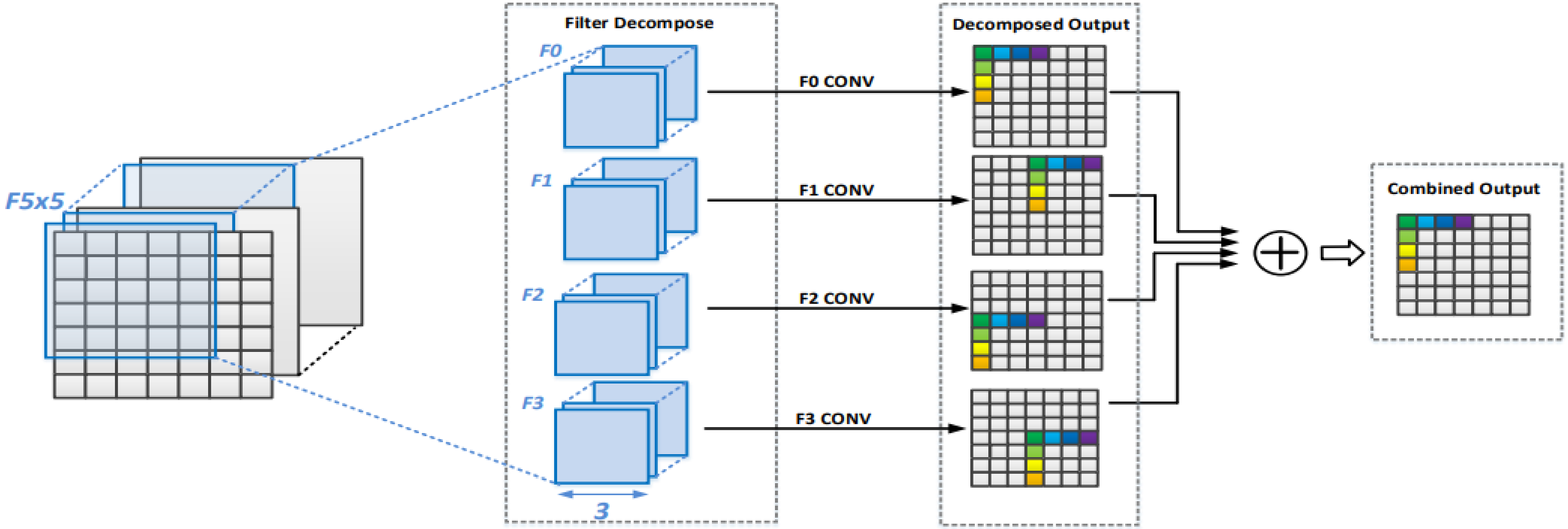


Fig. 6 Filter decomposition technique to compute a 5x5 filter on the 7x7 image. The Filter is decomposed into F0, F1, F2, F3, generating four sub-images. The sub-images are summed based on their filter's shift address. Same color's pixels will be added together to generate the corresponding pixels in the output image.