IP softcore for a bubbling convolutional accelerator in a neural network

By Yingxiu Chang, Ying Zhang, Liwei Liao, Jian Cao and Dunshan Yu, Peking University, Beijing

The development of neural network algorithms has gathered significant pace these days, as they become widely implemented in a variety of real-time image- and pattern-recognition applications.

Neural networks tend to use field-programmable gate arrays (FPGA), and to design our algorithm accelerator we turned to these devices first, but found some problems with them. For a start, they offer fewer pipelines for convolution, which results in low performance. To avoid computational bottlenecks and fit different size convolutional kernels, they use smaller processing elements (PEs), but these further delay the convolutional operations and reduce system peak performance.

In 2017, FPGAs were placed at the heart of a design for frequency-domain acceleration of a convolutional neural network (CNN) by implementing an 'overlap and add' (OaA) convolver, which consists of fixed-size FFTs that require zero padding during the convolutional procedure.

Now, we've devised a 2D 'sliding window of convolutional kernel' (SWCK) IP softcore, and formed an accelerator of a bubbling convolutional layer (ABCL). ABCL contains n convolution kernels, each made of SWCKs to create depths for the input feature map.

The Structure

Our solution's IP softcore is shown in Figure 1; it mainly consists of two shift registers with several multipliers (MUX) and adders. The MUX module is a rectifier linear unit (ReLU) activation function f(x) = max (0, x). Image pixel data in the RAM module enters the cache of the feature- or image-maps as a data stream. CC is the number of columns and CR the

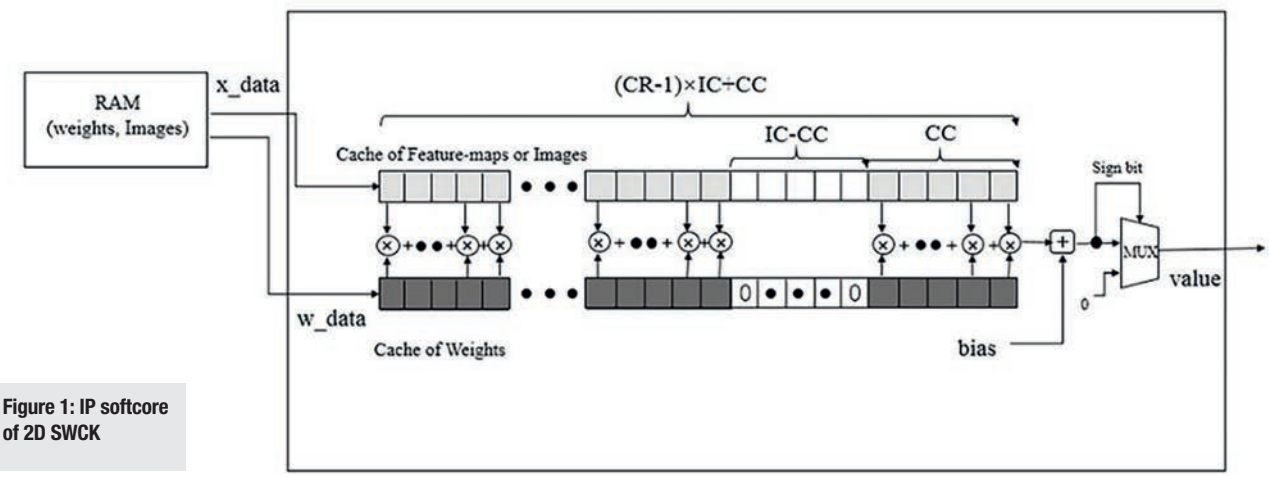

Figure 1: IP softcore of 2D SWCK

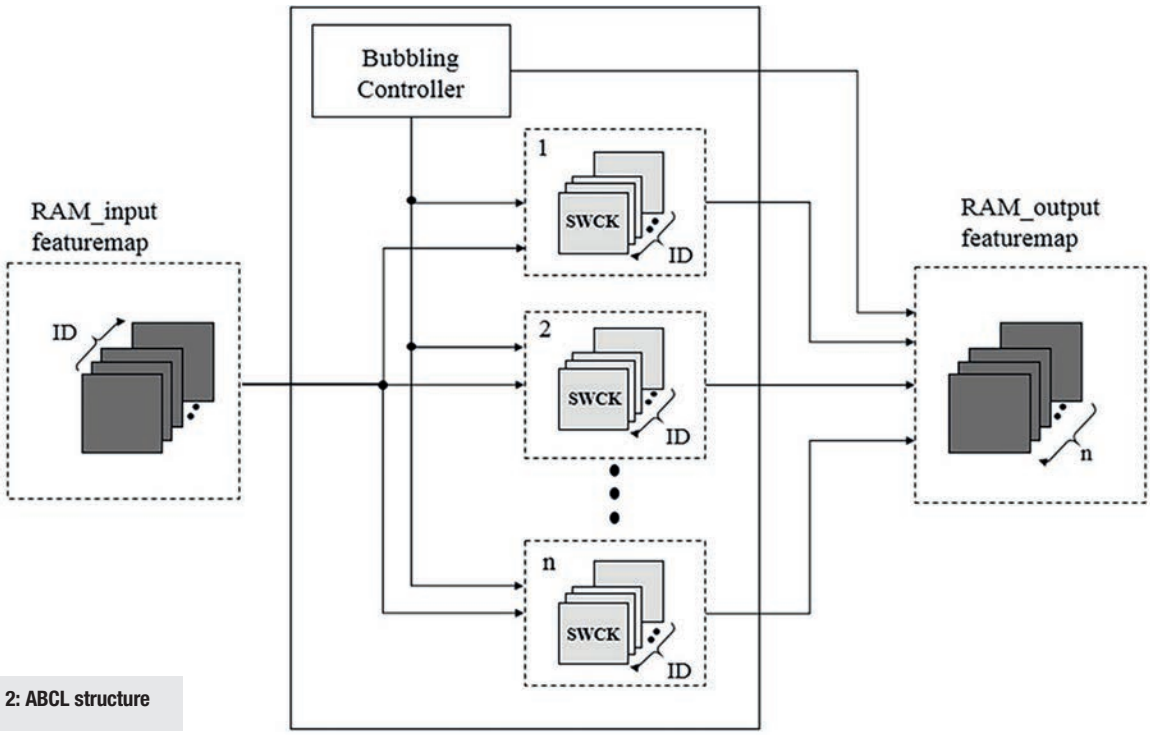

Figure 2: ABCL structure

number of rows in the convolutional kernel; see the light-shaded portion of the cache in Figure 1.

Two adjacent light-shaded portions are separated by IC-CC unit registers, and the cache of feature- or image-maps consists of (CR-1) × IC + CC unit registers. IC and CR are the convolutional kernel's columns and rows for the 2D feature map, respectively.

Similarly, weight parameters stored in the RAM are entered as a data stream into the cache of weights. This cache's structure is the same as that of the cache of feature- or image-maps; there are IC-CC zeros that follow IC valid weights in the RAM block, read sequentially during the convolutional process. The cache of weights is fixed after data is stored, whereas the cache of feature- or image-maps still shifts the data stream because of the input feature maps. The input feature map is a 3D matrix with depth ID ≥ 1.

The caches are connected by multiply-add units at corresponding positions, forming an abstract 2D SWCK. Modifying the depth of the two caches consist of modifying the IC to match different convolutional layers and various sizes of the input feature maps. The convolutional process begins by reading the first data of the input feature maps into the corresponding cache and ends with the last data entry.

The ABCL Structure

The ABCL structure is shown in Figure 2. The depth of the input feature map in the RAM is ID. ABCL is made of n convolutional kernels, each kernel shown in Figure 2 as a structure in a dashed box.

Stacking ID SWCKs into a complete 3D convolutional kernel achieves deep parallel operation of convolution. The SWCKs in each convolutional kernel are fed by a 2D input feature map of corresponding depth; the SWCKs do simultaneous calculations.

The ABCL structure increases the number of channels of the input feature map, despite stacking SWCKs to improve the performance.

Bubbling Convolutional Layer Timing Graph

The timing graph of the bubbling convolutional layer is shown in Figure 3, which also shows the timing sequences for reading the feature map and weights data and the convolution process, respectively.

To clarify the term "bubbling": If we think of the convolution process as liquid (with valuable results represented by a 'Val' period) and that liquid discharges excess air (in this case synonymous with the 'Inval' period), after several iterations the 'Inval' periods are disposed of and only 'Val' periods remain. This is why the method is referred to as "bubbling".

The waiting period of the first two sequences in Figure 3a schedules the timing for the weights to enter the cache. Shorter than this period and the results of the convolution become invalid; Equation 1 helps determine the operation's delay.

The holding period in the second sequence means to stop and hold reading the weights, which lasts as long as the whole shaded block in the first sequence is covered (this block keeps the data stream flowing through to the cache in the SWCK and is the period of the convolution process).

Meanwhile, during the convolution operations, the period

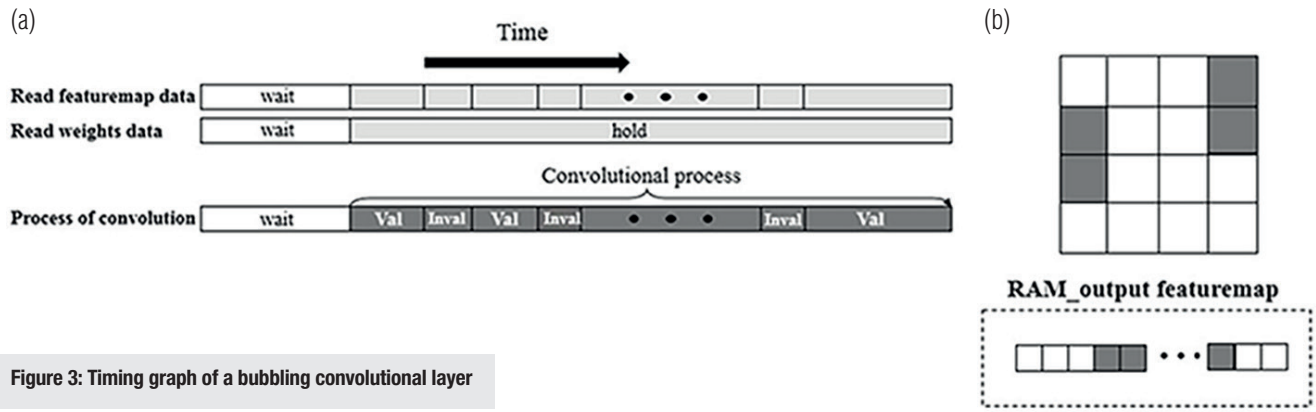

Figure 3: Timing graph of a bubbling convolutional layer

Layer	1	2	3	4	5
ID	1	3	3	6	-
n	3	1	6	1	-
IC=IR	28	24	12	8	-
CC=CR	5	2	5	2	-
Stride	1	2	1	2	-
State	Conv	Max-pooling	Conv	Max-pooling	Fully-connected

Table 1: CNN structure

'Val' in the third sequence indicates that data of the output feature map is valid; in other words, an output of valid results when the cross-row is non-convoluted. The delay for this period can be determined with Equation 2.

In contrast, the invaluable period 'Inval' in the third sequence indicates that data of the output feature map is invalid; it corresponds to the invalid data generated by the convolutional kernel cross-row in Figure 3b. The delay for this period can be determined with Equation 3.

Figure 3a shows the timing logic for ABCL. The entire delay of the convolution is expressed by Equation 4.

$$clk_{wait} = (CR-1) \times IC + CC - 1 \quad (1)$$

$$clk_{Val} = IC - CC + 1 \quad (2)$$

$$clk_{Inval} = CC - 1 \quad (3)$$

$$\begin{aligned} clk_{Total} &= clk_{wait} + Convolutional\ process \\ &= clk_{wait} \\ &\quad + \{[(IC-CC+1)+(CC-1)] \times (IR-CR) + (IC-CC+1)\} \\ &= IC \times IR \end{aligned} \quad (4)$$

IR and IC are the number of rows and columns of the input feature map, respectively, whereas CR and CC are the number of rows and columns of the 2D convolutional kernel.

Platform	Our work	GPU	CPU
Layer 1 (occupancy rate)	7.84×10^{-6}s(31.0%)	8.52×10^{-3}s(31.4%)	1.13×10^{-2}s(30.4%)
Layer 2 (occupancy rate)	5.76×10^{-6}s(22.8%)	6.63×10^{-3}s(24.4%)	1.25×10^{-2}s(33.7%)
Layer 3 (occupancy rate)	1.44×10^{-6}s(5.7%)	9.00×10^{-3}s(33.2%)	0.74×10^{-2}s(19.9%)
Layer 4 (occupancy rate)	0.64×10^{-6}s(2.5%)	2.58×10^{-3}s(9.5%)	0.56×10^{-2}s(15.1%)
Layer 5 (occupancy rate)	9.6×10^{-6}s(38.0%)	0.40×10^{-3}s(1.5%)	0.032×10^{-2}s(0.9%)
Total	25.28×10^{-6}s(100%)	27.13×10^{-3}s(100%)	3.712×10^{-2}s(100%)
Speed increase	1.00x	1073.18x	1468.35x

Table 2: Comparison of execution times for different platforms

	FPGA	
Fix point MACs	Time	GMACs
Layer 1	7.84 × 10-6s(31.0%)	7.5
Layer 2	5.76 × 10-6s(22.8%)	0
Layer 3	1.44 × 10-6s(5.7%)	45
Layer 4	0.64 × 10-6s(2.5%)	0
Layer 5	9.6 × 10-6s(38.0%)	0.1
Total	25.28 × 10-6s(100%)	--
Overall GMACs/s	4.93	
Overall GOPs/s	9.86	

Table 3: FPGA performance

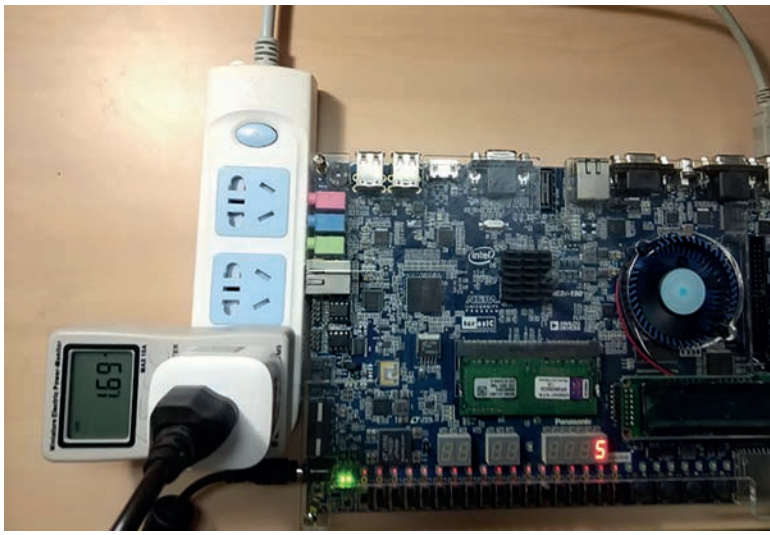

Figure 4: Power measurement of the FPGA during operation

	Primary Server		Our work
	CPU	GPU	
Power (W)	99.68	140.42	1.69
Comparison	58.98x	83.09x	1.00x
Energy (J)	3.70	3.89	4.27 × 10-5
Comparison	5.15 × 10⁴x	5.42 × 10⁴x	1.00x
Efficiency (GOPs/s/W)	67.37 × 10-6	65.43 × 10-6	5.83

Table 4: Comparison of power consumption and efficiency between different devices

Experiments and Results

Using the Modified National Institute of Standards and Technology (MNIST) database as a verification dataset, the overall structure of the convolutional neural network (CNN) is shown in Table 1. We tested the latency, performance and efficiency of the image recognition using an Intel Xeon E3-1230 V2 CPU, NVIDIA Quadro 4000 GPU and DE2i-150 FPGA, respectively.

Table 2 shows a comparison of execution times for different platforms, with performance evaluations shown in Table 3. Our design improves the convolution process by a certain amount, reaching 9.86GOPs/s.

	Resource	Ratio
LUTs	81,195/149760	54
FFs	25,102/149760	17
Embedded 9-bit multipliers	720/720	100
BRAM	75,661/6,635,520	1

Table 5: FPGA resources needed

Table 4 compares the power consumption and efficiency of different devices. For a single image recognition, our work draws 1.69W with efficiency of 5.83GOPs/s/W, compared to 99.68W and 140.42W for a CPU and GPU, respectively.

Figure 4 shows the setup for taking the FPGA's power measurements during the execution of a single image recognition.

Table 5 shows the resources needed by the FPGA; the LUTs and multipliers take up a large portion, as high as 54% and 100% respectively, whereas the block RAM (BRAM) accounts for only 1%.

Pros and Cons

We analysed the pros and cons of different hardware accelerators for a CNN used to avoid redundant padding operations of different sizes for the input feature maps. We then suggested an IP softcore of 2D SWCK which can be mapped in different layers. The ABCL made of SWCKs offers significant performance and latency improvements for the same convolutional layer. Hence, we propose the bubbling convolutional layer to reduce the complexity of the convolution process.

We also determine that implementing ABCL in a CNN for image recognition is rather important. EW