Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks(2016 ISCA)
Eyeriss:一种用于深度神经网络的可重构加速器
Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks(2015 FPGA 619)
基于FPGA的深度卷积神经网络加速器优化设计

1.Abstract

提出了一种新颖的Row Stationary(RS)数据流来最小化数据移动所带来的能量消耗

使用AlexNet，其他数据流的在卷积层和全连接
层消耗的能量分别达到了 RS数据流的(1.4-2.5)和(1.3-）倍

2.**Inroduction**

本文的主要贡献：1. 对现有CNN dataflow进行分类

2. 基于row stationary的spatial architecture

3. 对不同CNN dataflow的量化的分析框架

2. **EXISTING CNN DATAFLOWS**

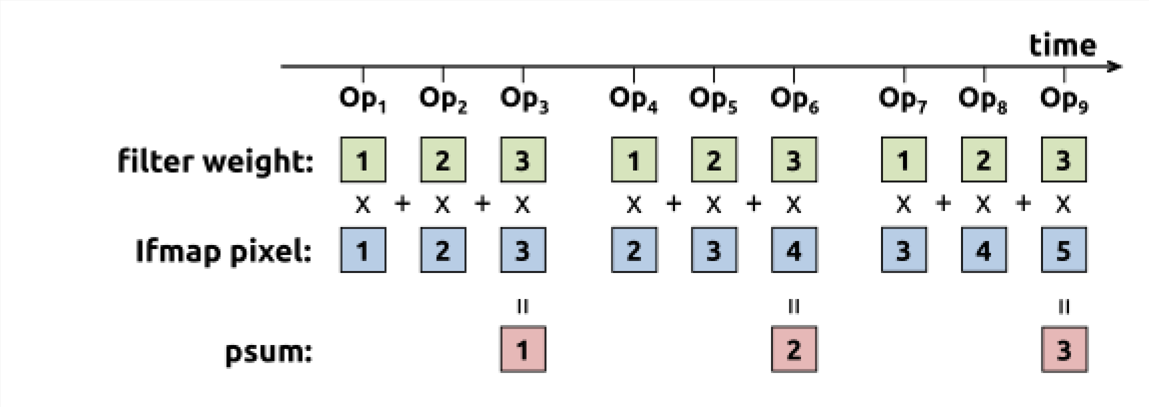
1. weight stationary(WS) dataflow 每个过滤器权值存在Register File(RF)

2. output stationary(OS) dataflow 卷积操作时产生的部分和存在同样的RF中

3. no local reuse(NLR) dataflow 通过Possess Element(PE)之间的内部通信

3. **ENERGY-EFFICIENT DATAFLOW: ROW STATIONARY**

a. 1D Convolution Primitives



3. **ENERGY-EFFICIENT DATAFLOW: ROW STATIONARY**

b. two-step primitive mapping

逻辑映射：每个1D primitive首先映射到PE

物理映射：同一物理PE上的将执行多个来自不同PEs的1D primitive

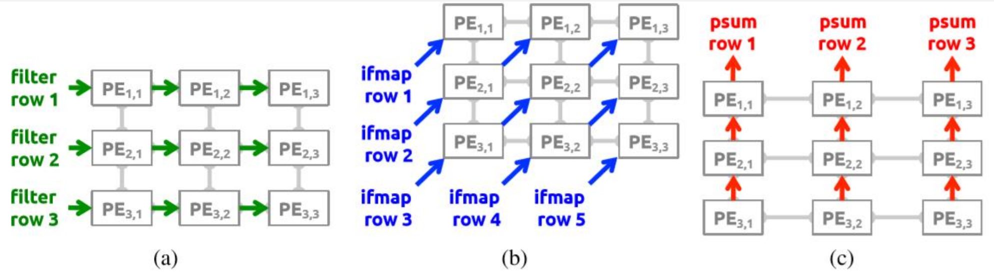


Figure 6. The dataflow in a logical PE set to process a 2D convolution. (a) rows of filter weight are reused across PEs horizontally. (b) rows of ifmap pixel are reused across PEs diagonally. (c) rows of psum are accumulated across PEs vertically. In this example, $R = 3$ and $H = 5$.

3. **ENERGY-EFFICIENT DATAFLOW: ROW STATIONARY**

c. Energy-Efficient Data Handling

RF: filter and ifmap reuse

Array (inter-PE communication):

Global Buffer：存储第二次折叠后的 filter ifmap psum

d. Support for Different Layer Types

FC layer: 跟卷积很像

Pool layer: 将mac替换成max

4. **Experiment**

6个数据流在卷积层每次操作的标准能耗

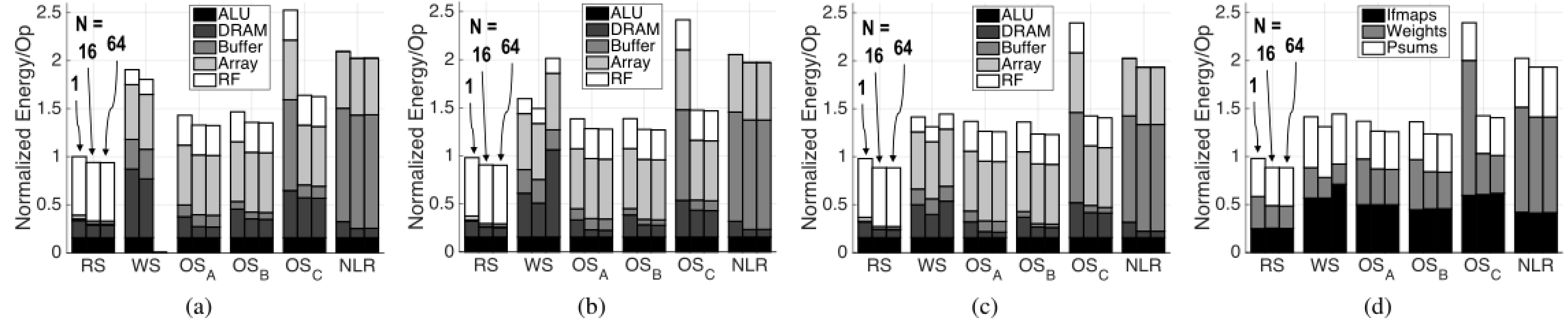


Figure 12. Energy consumption of the six dataflows in CONV layers of AlexNet under PE array size of (a) 256, (b) 512 and (c) 1024. (d) is the same as (c) but with energy breakdown in data types. The energy is normalized to that of RS at array size of 256 and batch size of 1. The RS dataflow is 1.4× to 2.5× more energy efficient than other dataflows.

[redacted]

Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks

Abstract

针对计算吞吐量可能和FPGA平台提供的内存带宽不匹配的问题，作者提出了一种新的roofline模型设计方案。

Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks

1.Introducion & Background

Contributions:

- 在设计空间中使用roofline模型识别了所有可能的解决方案
- 提出了一种CNN加速器设计，在不同的卷积层上使用同一的循环展开因子

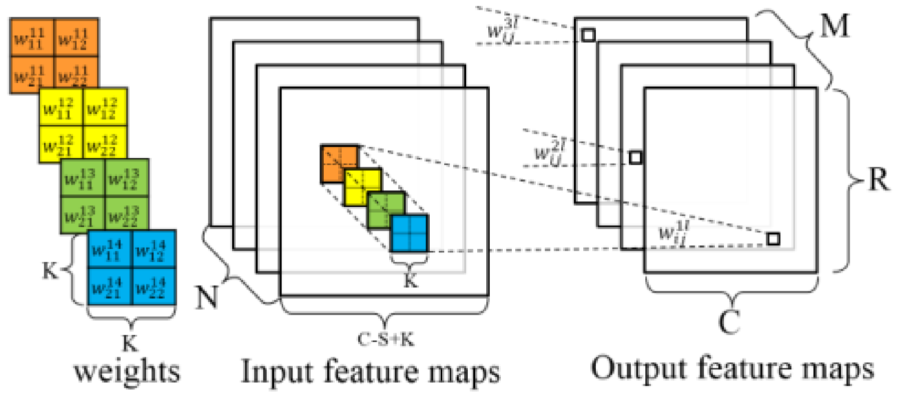


Figure 1: Graph of a convolutional layer

```
for(row=0; row<R; row++) {
 for(col=0; col<C; col++) {
  for(to=0; to<M; to++) {
   for(ti=0; ti<N; ti++) {
    for(i=0; i<K; i++) {
     for(j=0; j<K; j++) {
   L:   output_fm[to][row][col] +=
           weights[to][ti][i][j]*
           input_fm[ti][S*row+i][S*col+j];
} } } } } }
```

Code 1: Pseudo code of a convolutional layer

2. ACCELERATOR DESIGN EXPLORATION

2.1 Computation Optimization

a. Loop Unrolling
b. Loop pipeline

Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks

2.2 Memory Access Optimization

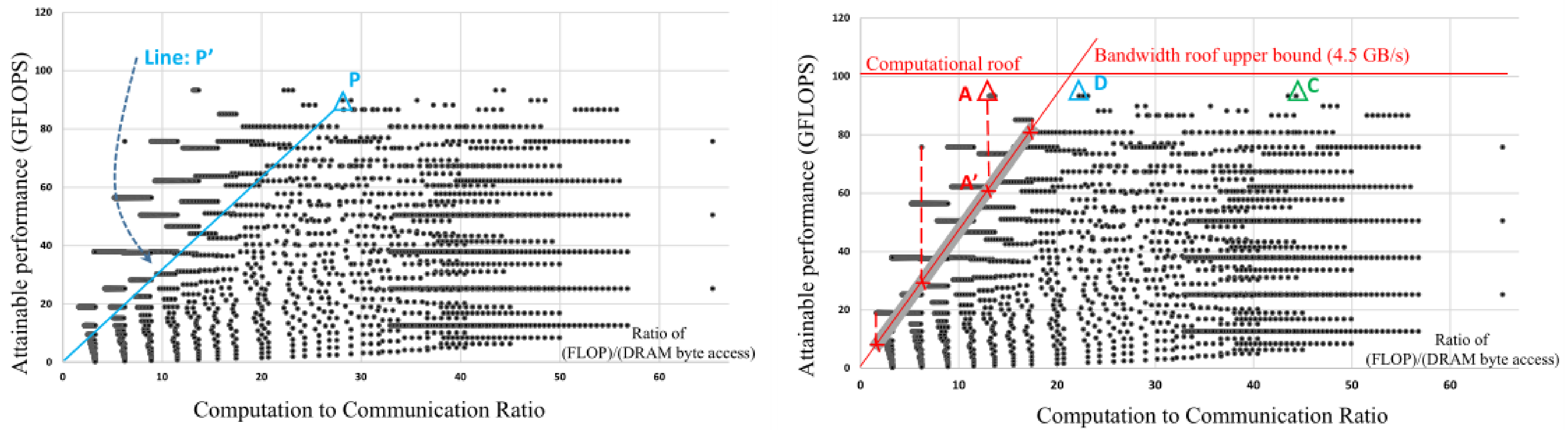
a. Local Memory Promotion
b. Loop Transformations for Data Reuse
(使用基于多面体的优化框架来识别所有的合法循环转换)

```
for(row=0; row<R; row+=Tr) {
 for(col=0; col<C; col+=Tc) {
  for(to=0; to<M; to+=Tm) {
   for(ti=0; ti<N; ti+=Tn) {
   //load output feature maps
   //load weights
   //load input feature maps

       L:  foo(output_fm(to,row,col),
               weights(to,ti),
               input_fm(ti,row,col));
   //store output feature maps
   }

  } } }
```

Figure 9: Local memory promotion for CNN

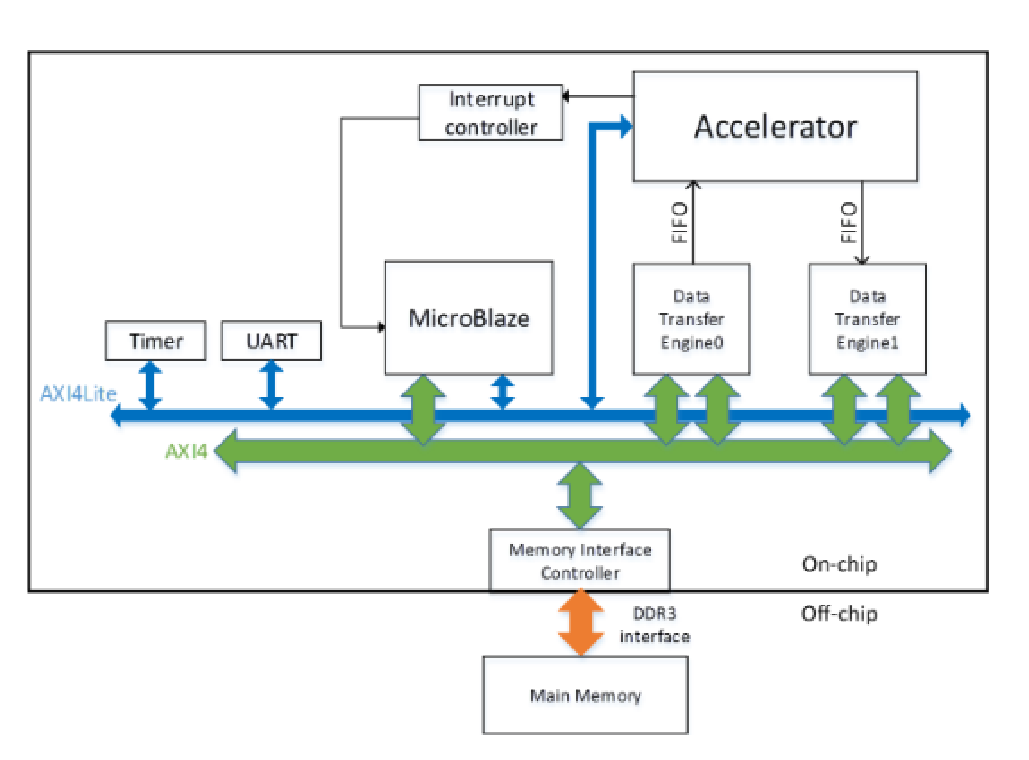
2.3 Design Space Exploration

枚举所有可能的循环次序和分块尺寸可以产生一系列计算性能和计算-通信比对



(a) Design space of all possible designs (b) Design space of platform-supported designs

3. IMPLEMENTATION DETAILS



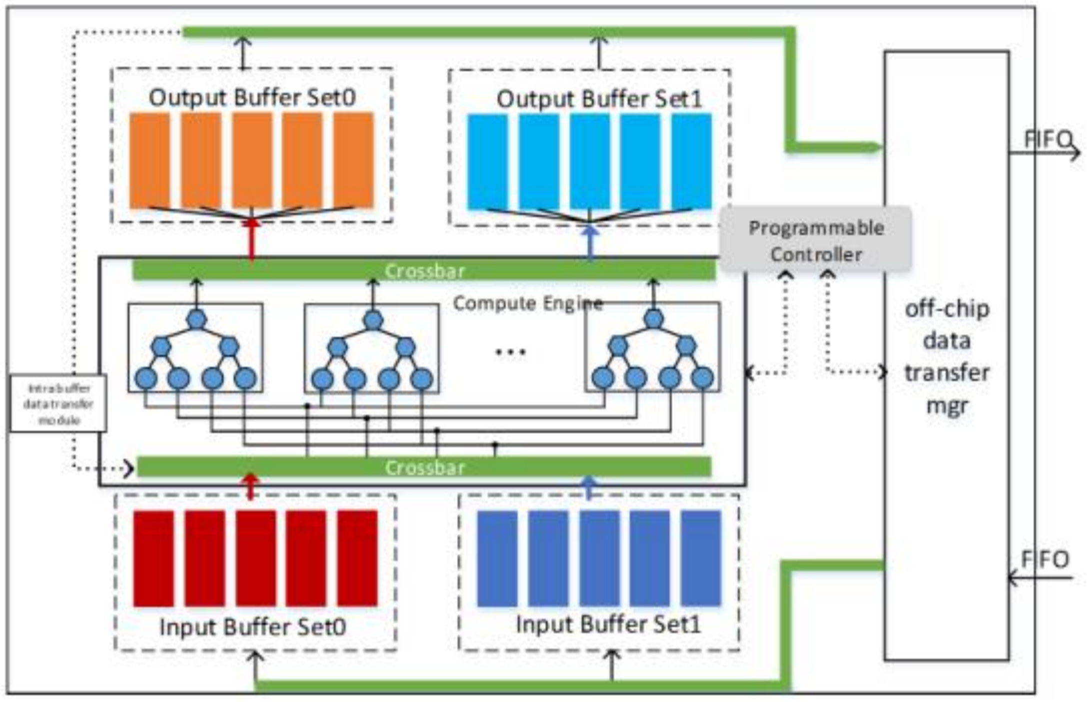


Figure 11: Block diagram of proposed accelerator

Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks

4. Experiment

Table 5: Comparison to previous implementations

	ICCD2013 [12]	ASAP2009 [14]	FPL2009 [6]	FPL2009 [6]	PACT2010 [2]	ISCA2010 [3]	Our Impl.
Precision	fixed point	16bits fixed	48bits fixed	48bits fixed	fixed point	48bits fixed	32bits float
Frequency	150 MHz	115 MHz	125 MHz	125 MHz	125 MHz	200 MHz	100 MHz
FPGA chip	Virtex6 VLX240T	Virtex5 LX330T	Spartan-3A DSP3400	Virtex4 SX35	Virtex5 SX240T	Virtex5 SX240T	Virtex7 VX485T
FPGA capacity	37,680 slices 768 DSP	51,840 slices 192 DSP	23,872 slices 126 DSP	15,360 slices 192 DSP	37,440 slices 1056 DSP	37,440 slices 1056 DSP	75,900 slices 2800 DSP
LUT type	6-input LUT	6-input LUT	4-input LUT	4-input LUT	6-input LUT	6-input LUT	6-input LUT
CNN Size	2.74 GMAC	0.53 GMAC	0.26 GMAC	0.26 GMAC	0.53 GMAC	0.26 GMAC	1.33 GFLOP
Performance	8.5 GMACS	3.37 GMACS	2.6 GMACS	2.6 GMACS	3.5 GMACS	8 GMACS	61.62 GFLOPS
	17 GOPS	6.74 GOPS	5.25 GOPS	5.25 GOPS	7.0 GOPS	16 GOPS	61.62 GOPS
Performance Density	4.5E-04 GOPs/Slice	1.3E-04 GOPs/Slice	2.2E-04 GOPs/Slice	3.42E-04 GOPs/Slice	1.9E-04 GOPs/Slice	4.3E-04 GOPs/Slice	8.12E-04 GOPS/Slice

Table 7: Performance comparison to CPU

float	CPU 2.20GHz (ms)		FPGA	
32 bit	1thd -O3	16thd -O3	(ms)	GFLOPS
layer 1	98.18	19.36	7.67	27.50
layer 2	94.66	27.00	5.35	83.79
layer 3	77.38	24.30	3.79	78.81
layer 4	65.58	18.64	2.88	77.94
layer 5	40.70	14.18	1.93	77.61
Total	376.50	103.48	21.61	-
Overall GFLOPS	**3.54**	**12.87**	**61.62**	
Speedup	**1.00x**	**3.64x**	**17.42x**	