WUST

学习总结

Mode by : 董 勇

目录
CONTENT

PART 01

题目：S. Cao et al., “Efficient and Effective Sparse LSTM on FPGA with Bank-Balanced Sparsity,” in Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside CA USA, Feb. 2019, pp. 63–72, doi: 10.1145/3289602.3293898.

01 Paper

motivation

- 在不受限制的稀疏LSTM中，不规则的计算和内存发访问限制了可实现的并行性

contribution

- 提出了Bank-Balanced Sparsity
- 设计了一个基于FPGA的BBS加速器

01 Paper

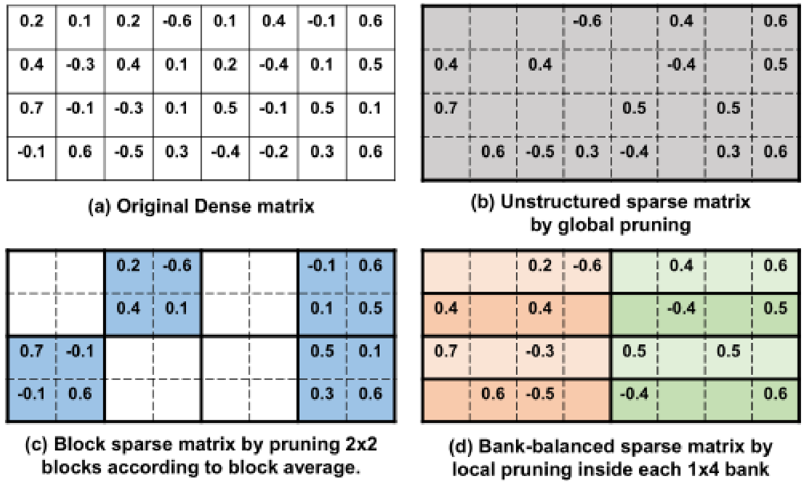

Figure 1: Comparing BBS with unstructured sparsity and block sparsity by pruning a dense matrix with a sparsity ratio of 50%.

D中的bank-balance稀疏矩阵保留了与b中的非结构花稀疏矩阵相似的较大的权值，但c中的块稀疏矩阵去掉了一些较大的权值，但保留了一些较小的权值。

01 Paper

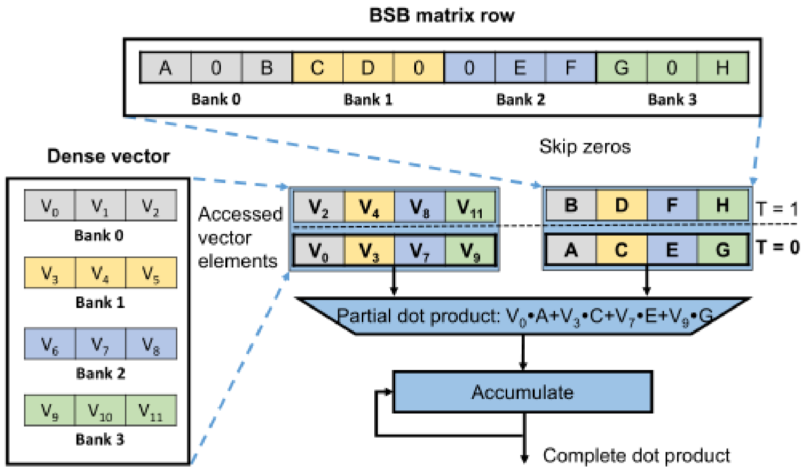

Figure 3: Exploiting inter-bank parallelism in dot product computation of one BBS matrix row and the dense vector.

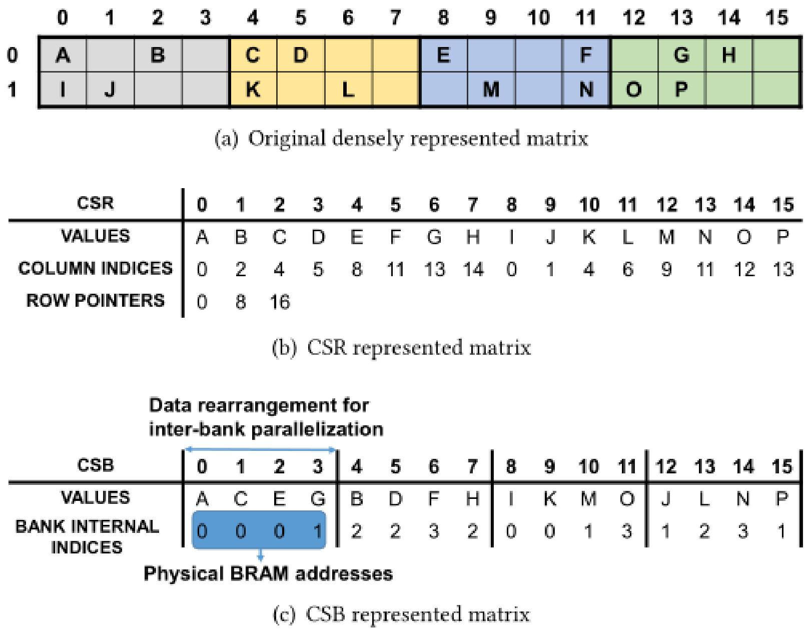

Figure 4: The comparison between CSR and CSB.

这种数据重排的目的是显示的暴露bank间的并行性，因此CSB中的每一个连续的N个元素都可以直接并行的获取和计算

01 Paper

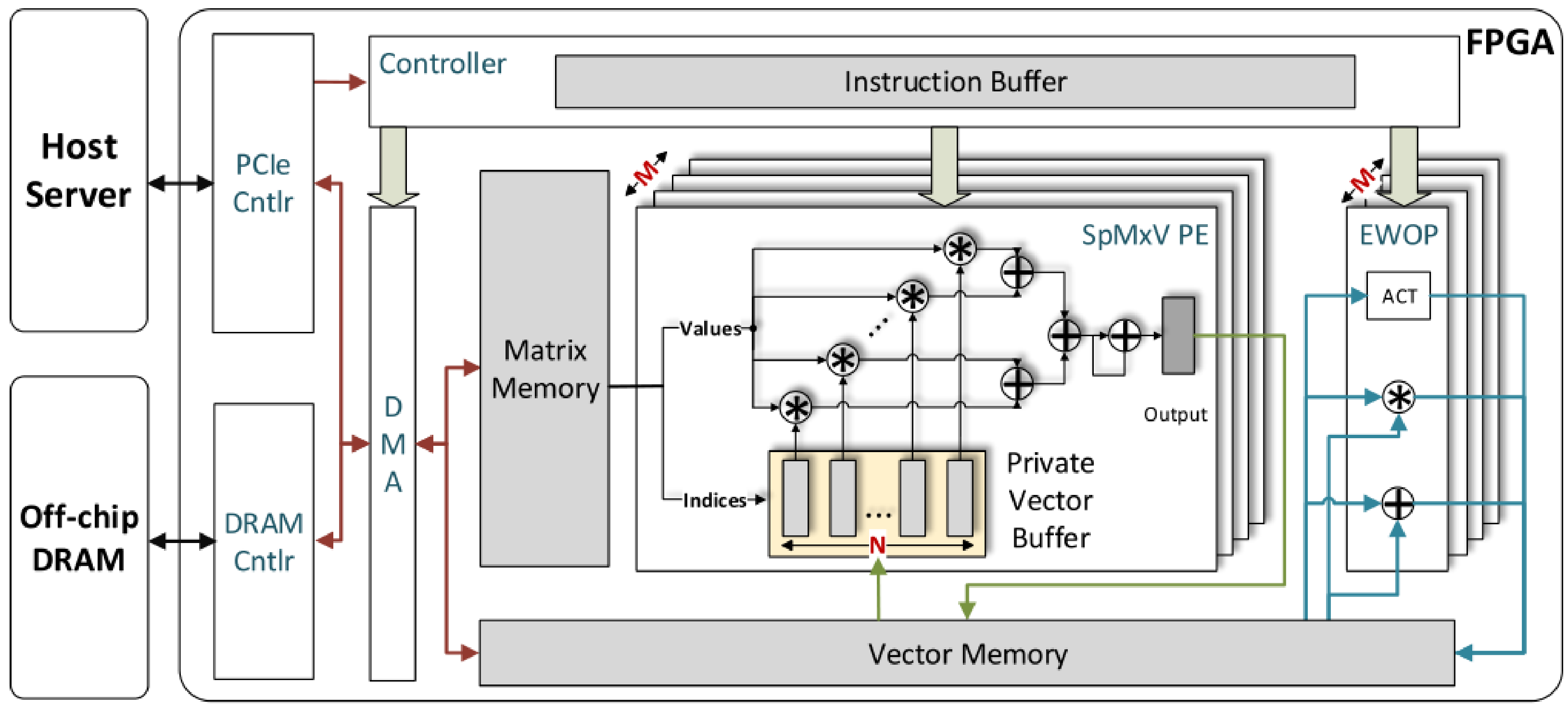

Figure 5: Overall architecture.

稀疏矩阵向量乘法单元
Element-wise vector operation单元

01 Paper

Table 6: Speedup comparison with state-of-the-art LSTM accelerators

	ESE[8]	C-LSTM[21]	DeltaRNN[6]	Ours
Platform	XCKU060	Virtex-7	XC7Z100	Arria 10 GX1150
Frequency (MHz)	200	200	125	200
Sparsity (%)	88.7	87.5	-	87.5
Quantization	fixed-12	fixed-16	fixed-16	fixed-16
Accuracy Degradation	0.30%	0.32%	-	0.25%
Throughput (GOPS)	282.2	131.1	192.0	304.1
Power (W)	41.0	22.0	7.3	19.1
Energy Efficiency (GOPS/W)	6.9	6.0	26.3	15.9
Latency(us)	82.7	16.7	-	**2.4**
Throughput at batch 1 (GOPS)	8.8	43.7	192.0	**304.1**
Effective Throughput at batch 1 (GOPS)	79.2	349.6	1198.0	**2432.8**

Conclusion:与采用不同压缩技术的LSTM FPGA加速器相比，BBS加速器的能量效率提高了2.3 ~3.7倍，延迟降低了7.0 ~34.4倍，模型精度的损失可以忽略不计

PART 02

- P3-p8

02/ Video

	LAS	CTC	RNN-T
Decoder	dependent	independent	dependent
Alignment	not explicit (soft alignment)	Yes	Yes
Training	just train it	sum over alignment	sum over alignment
On-line	No	Yes	Yes

PART

03

- Network
- Compression

03/ Experiment

```
nn.Sequential(
    # Depthwise Convolution
    nn.Conv2d(bandwith[0], bandwidth[0], 3, 1, 1, groups=bandwidth[0]),
    # batch Normalization
    nn.BatchNorm2d(bandwidth[0]),
    # Relu6 是限制Neuron最小只会到0， 最大只会到6， MobileNet系列都是使用ReLu6
    # 使用ReLu6的原因是如果数字太大，会不好压到float16 or futher qunatization 因此才会给限制
    nn.Relu6(),
    nn.Conv2d(bandwidth[0], bandwidth[1], 1),
    #过完Pointwise Convolution不需要再做ReLu，经验上Pointwise+ReLu效果会变差
    nn.MaxPool2d(2, 2, 0),
    #每过完一个Block就Dwon Sampling

),
```