第二届超大规模高性能计算机系统研制关键技术论坛

领域应用的定制化硬件加速方法

Hang Lu（路航）

副研究员、硕导、青促会会员、新百星

CCF HPC China 济南·2022

计算机体系结构国家重点实验室 State Key Laboratory of Computer Architecture, ICT, CAS

中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

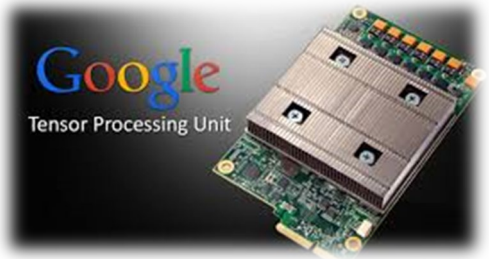
利用比特稀疏性的通用深度学习加速方法

计算机体系结构国家重点实验室
State Key Laboratory of Computer Architecture, ICT, CAS

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

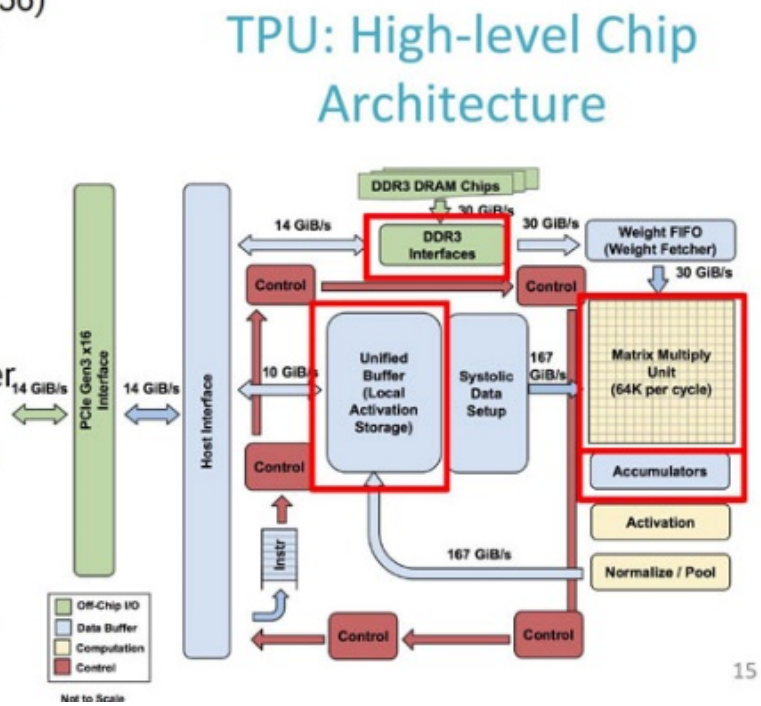
利用比特稀疏性的通用深度学习加速方法

- Deep learning tasks:
  - For computer vision – convolutions
- Deep learning accelerators:
  - performing MACs



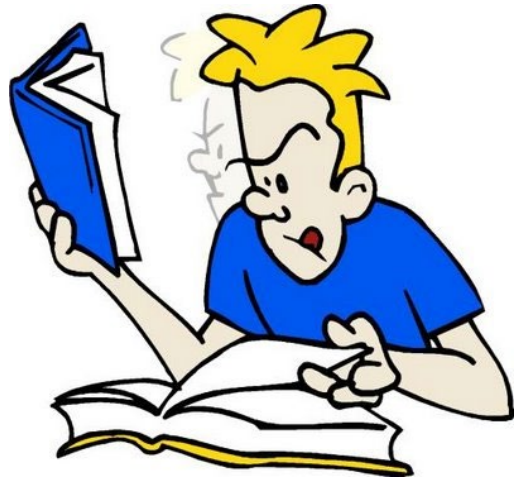
MAC is a representative of state-of-the-art ML accelerator performance

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
  - 65,536 * 2 * 700M
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory



利用比特稀疏性的通用深度学习加速方法

- 通用性不好
  - 支持模型种类单一，换场景就要换平台 ☹
  - 位宽范围小，大多(u)int8。fp32/16无法满足帧率要求 ☹

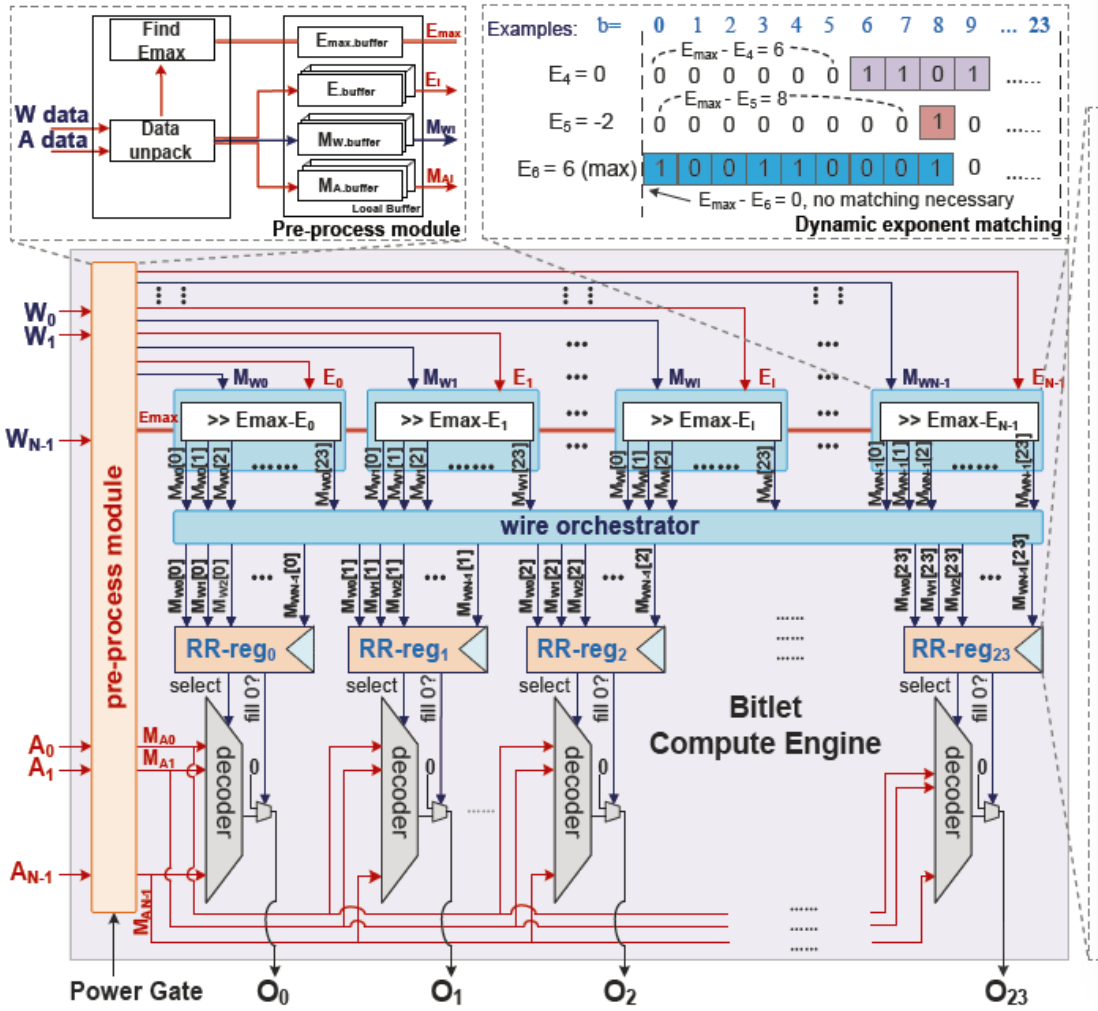


- Our solution
  - 灵活性
    - 支持数据位宽——定点数：1~24bit，浮点数：fp32/16，bfloat16
    - 在同一个计算引擎中实现混合精度计算（训练or推理）
    - 自动识别并充分利用比特级稀疏性进行加速

    → High TOPs/W ☺
  - 通用性
    - 定点、浮点加速集成于同一计算引擎，无需设计额外硬件支持所有AI场景
  - 搭载硬件剪枝

    → High TFLOPS/W ☺

利用比特稀疏性的通用深度学习加速方法



MICRO'21 | Oct 18-22nd, 体系结构领域顶会！清北中科院，天津大学与腾讯上榜！

原创 CS Conferences CS Conferences 2021-10-13 10:30

收录于话题
#近期顶会
66个 >

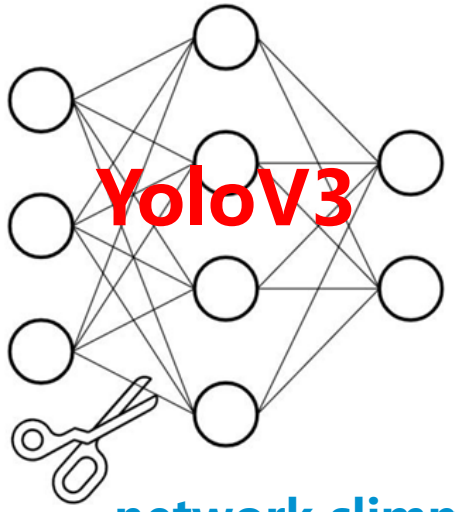
动动大拇指 快快关注哦~

The International Symposium on Microarchitecture (MICRO)，国际微架构研讨会(MICRO)，是介绍和讨论微体系结构、编译器、硬件/软件接口以及高级计算和通信系统设计的主要论坛。MICRO是CCF A类会议，H5指数45，Impact Score高达6.76，在体系结构领域具有极高的评价。MICRO的目标是将微体系结构、编译器和系统领域的研究人员聚集在一起进行技术交流。

利用比特稀疏性的通用深度学习加速方法

- The benefits of general purpose accelerators:
  - to fit for various DL tasks
  - that can be applied in both training and inference
- How to boost the performance - leveraging the sparsity of the operands
  - Software-based pruning
  - Quantization-aware training

传统加速方法——数值稀疏性



network slimming (ICCV' 17)

Weakness

2~3 days sparse training ☹

Time-consuming!

3~4 days retraining / fine-tuning ☹

Quantization

+1/-1 weights, Courbariaux 2016

Bineray, Courbariaux 2015

INQ, Zhou 2017

Ternary, Li 2016

.....

Weakness

Requires tedious re-training ☹

Suffering from accuracy loss ☹

Not precision tunable ☹

利用比特稀疏性的通用深度学习加速方法

- The headroom of value-level sparsity is very limited.
- **However, bit-level sparsity is inherently fertile.**

“比特”稀疏性

Model	Weight Sparity	Bit Sparity
DenseNet121	4.84%	48.64%
ResNet50	0.33%	48.64%
ResNet152	0.75%	48.64%
ResNext50_32x4d	0.37%	48.64%
ResNext101_32x8d	3.43%	48.65%
InceptionV3	0.05%	48.64%
MNASNet0.5	0.00%	48.60%
MNASNet1.0	8.07%	48.98%
MobileNetV2	0.01%	48.67%
ShuffleNetV2_x0_5	0.00%	48.36%
ShuffleNetV2_x1_0	1.53%	48.63%
SqueezeNet1_0	0.05%	48.64%
SqueezeNet1_1	0.02%	48.64%

非常有限！

Weight sparsity : the values below 10^{-5} over the total parameter size

天然大量暴露！

Bit sparsity : total bit 0s over the total “bit count” of the mantissas

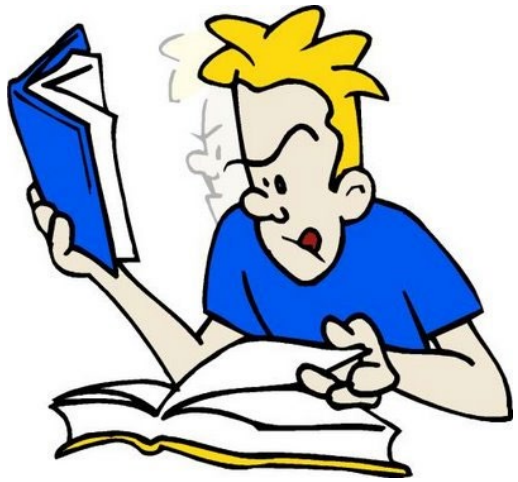
利用比特稀疏性的通用深度学习加速方法

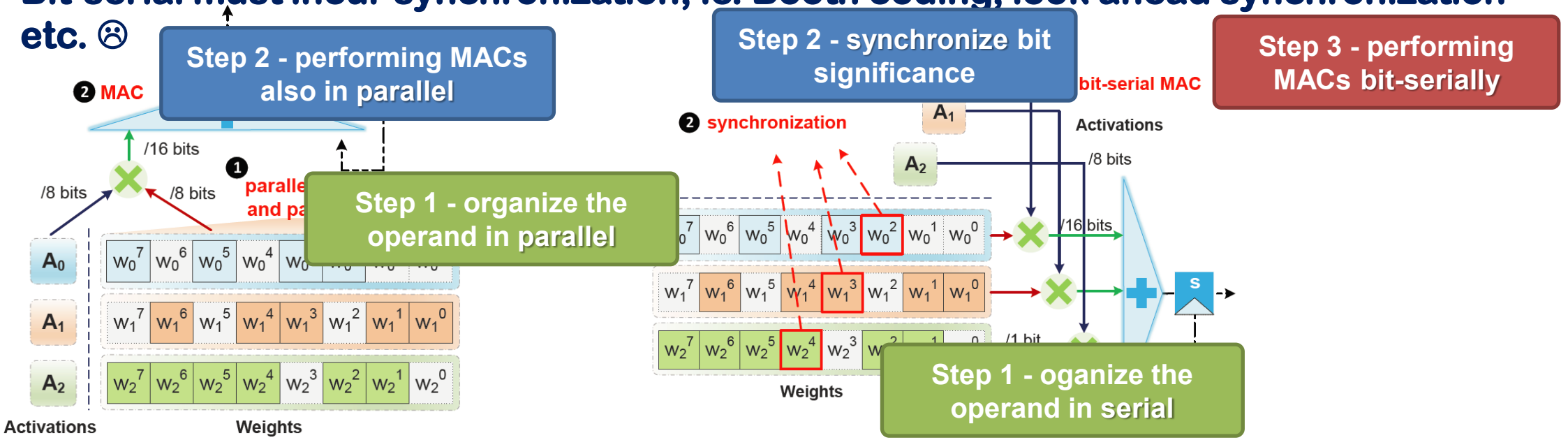
Any state-of-the-art solutions?

设计方式	加速器原型	Sparsity Exploited	精度位宽	支持训练?
Bit parallel (比特并行)	Eyeriss, DaDianNao	N/A	16b	No
	Cambricon-S, EIE	A- / W- value	16b	No
	SCNN	A- & W- value	16b	No
Bit serial (比特串行)	UNPU, Stripes	N/A	1~16b	No
	Bit Fusion	N/A	2,4,8,16b	No
	Pragmatic	A- / W- bit	1~16b	No
	Bit Tactical	A- bit & W-value	1~16b	No
	Laconic	A- & W- bit	1~16b	No
Bit interleaving (比特交织)	**Bitlet (Ours)**	**W- bit & W-value (or A- bit & A-value)**	**fp32/16, 1~24b bfloat16**	**Yes**

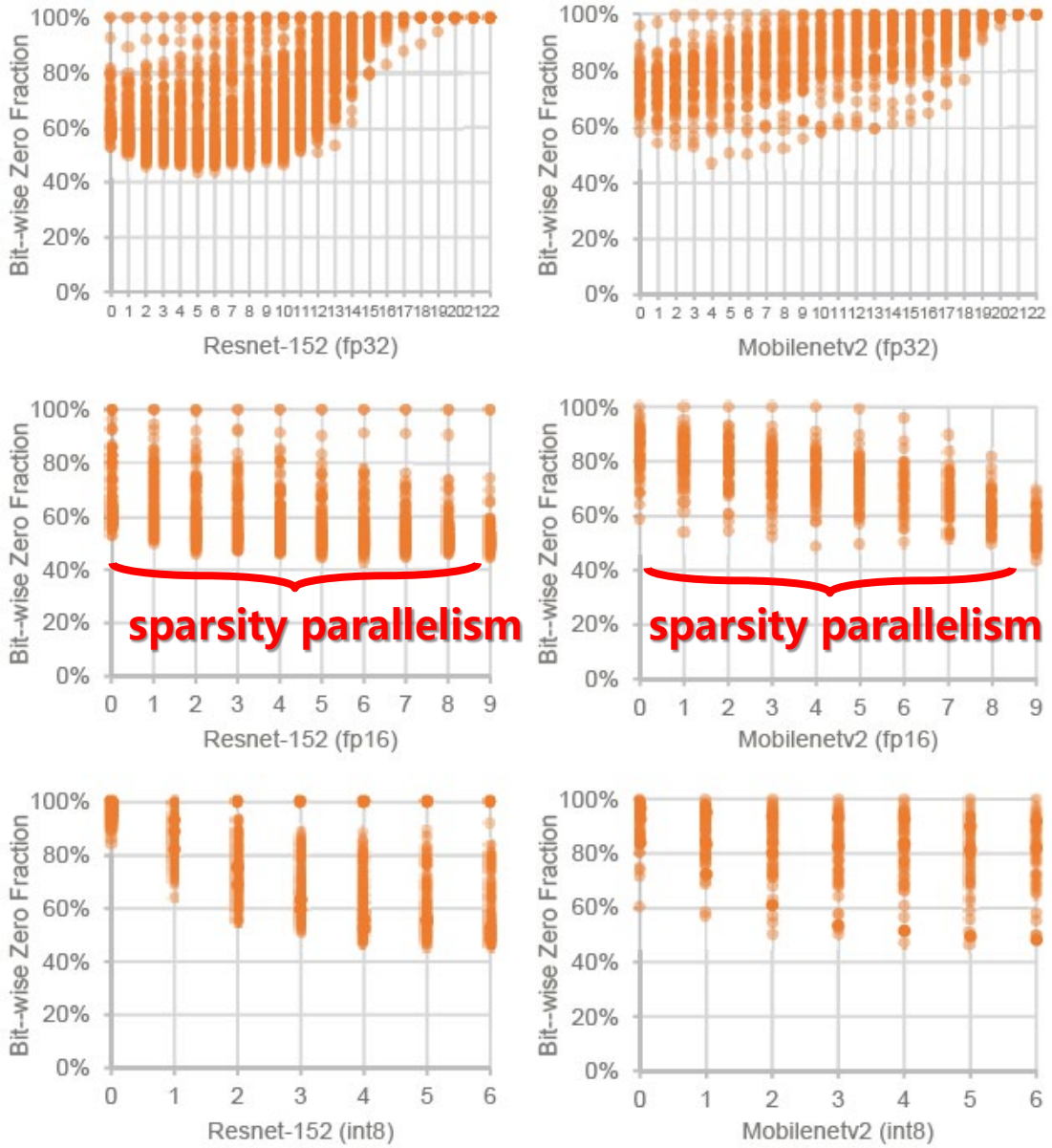
利用比特稀疏性的通用深度学习加速方法

- **The weaknesses of such design philosophy:**
  - **None of them are general-purpose!（不通用）**
    - Only use in inference ☹
    - Only support fixed-point arithmetic ☹
  - **Sub-optimal sparsity utilization（而且也不快！）**
    - Bit-parallel cannot leverage bit-level sparsity ☹
    - Bit-serial must incur synchronization, ie. Booth coding, look ahead synchronization etc. ☹





利用比特稀疏性的通用深度学习加速方法

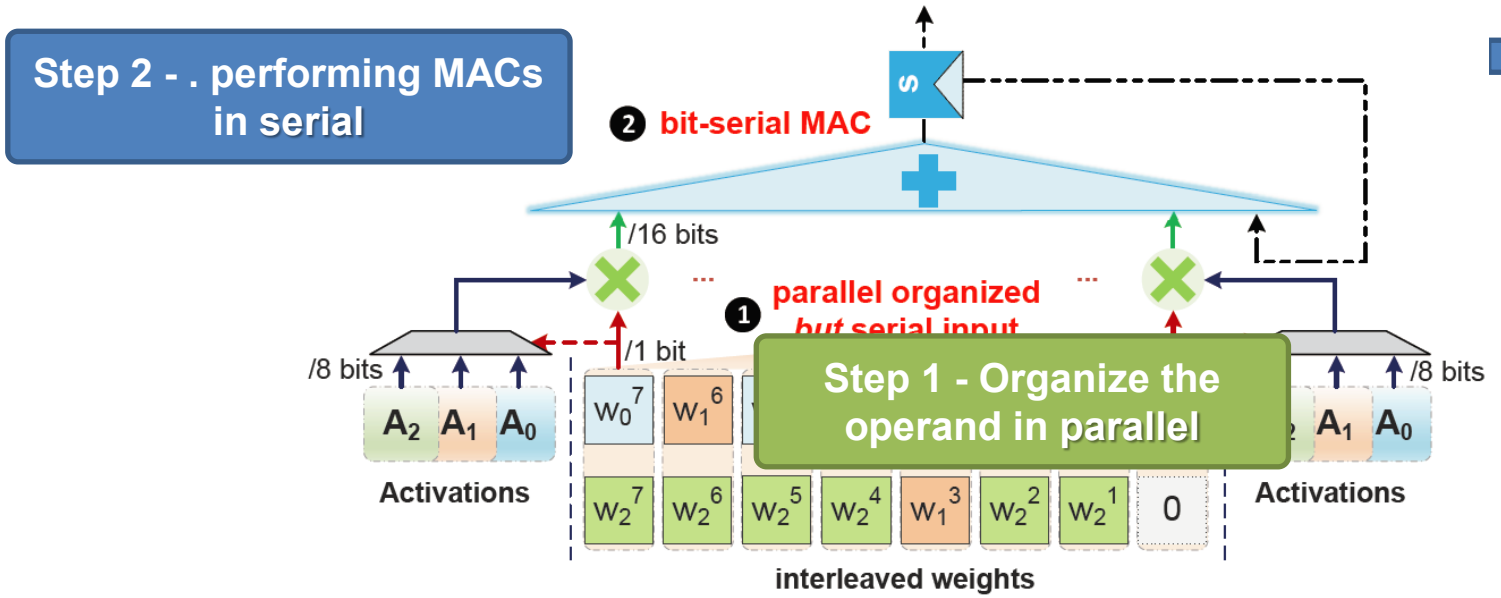


- **High sparsity percentage at each bit significance**
- **The sparsity is nearly uniform in terms of :**
  - Different precisions, including floating point, fixed point and integer
  - Different bit significances (for floating point, we focus on the mantissa)

利用比特稀疏性的通用深度学习加速方法

- **Our solution – bit interleaving** → High TOPs/W ☺
  - Can we obtain the benefits of them both?（变快！）
    - Efficient utilization of the sparsity
    - Avoid the complex synchronization procedures
  - Most importantly, design a general-purpose accelerator that can leverage the bit-level sparsity （变通用！）

→ High TFLOPS/W ☺



bit interleaving

利用比特稀疏性的通用深度学习加速方法

设计原理

Floating-point MAC decomposition:

浮点MAC可以无损转化为定点数计算

How we embark

$$\sum_{i=0}^{N-1} A_i \times W_i = \sum_{i=0}^{N-1} (-1)^{S_{W_i}} A_i \times M_{W_i} \times 2^{E_{W_i}}$$

Sign

Exponent

mantissa

详细推导请见MICRO’21 paper

$$\sum_{i=0}^{N-1} \sum_{b=E_i-E_{max}}^{E_i-E_{max}-23} \left[(-1)^{S_{W_i}} \oplus S_{A_i} \cdot \left(M_{A_i} \times M_{W_i}^{b} \right) \right] \times 2^{E_{max}+b}$$

Exponent matching

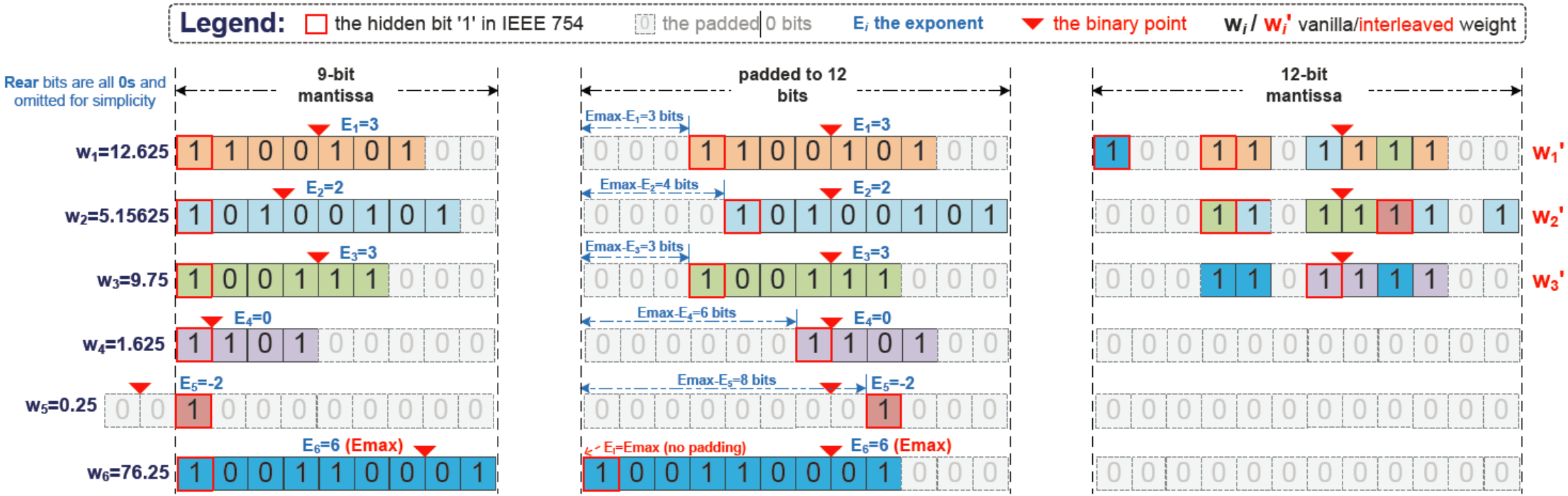
Bit-level arithmetic

Bit significance

Final sign

Maximum exponent

利用比特稀疏性的通用深度学习加速方法



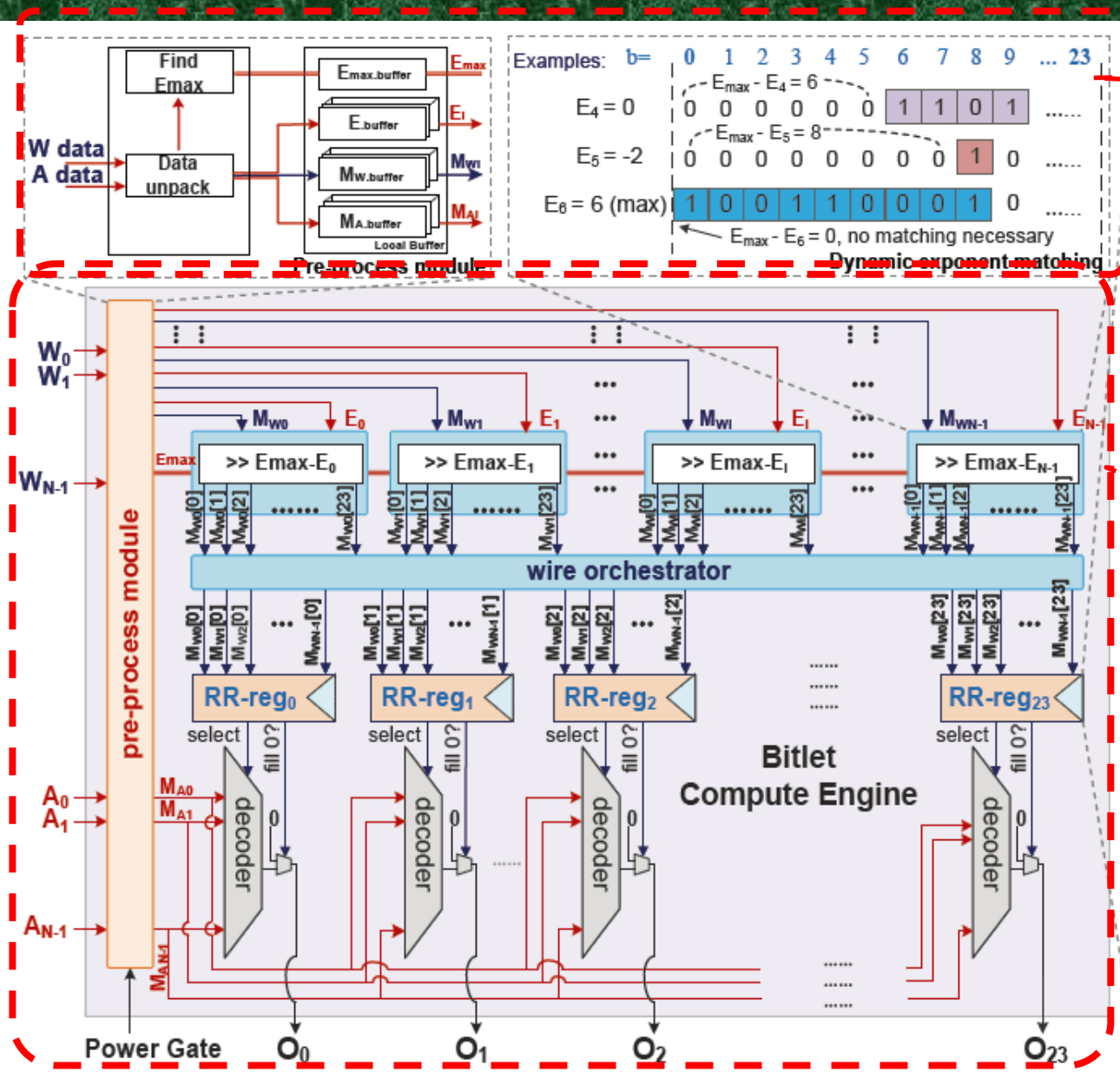
(a) **Step 1:** preprocessing the floating-point weights

(b) **Step 2:** dynamic exponent matching

(c) **Step 3:** bit distillation

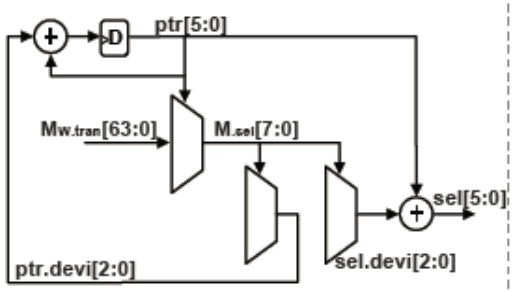
- **具体操作:**
  - **前处理**
  - **动态的、统一的对阶——减少浮点计算每次对阶带来的开销**
  - **比特蒸馏——利用稀疏性，减少不必要计算**

利用比特稀疏性的通用深度学习加速方法



High level example

Diagram of RR-reg:

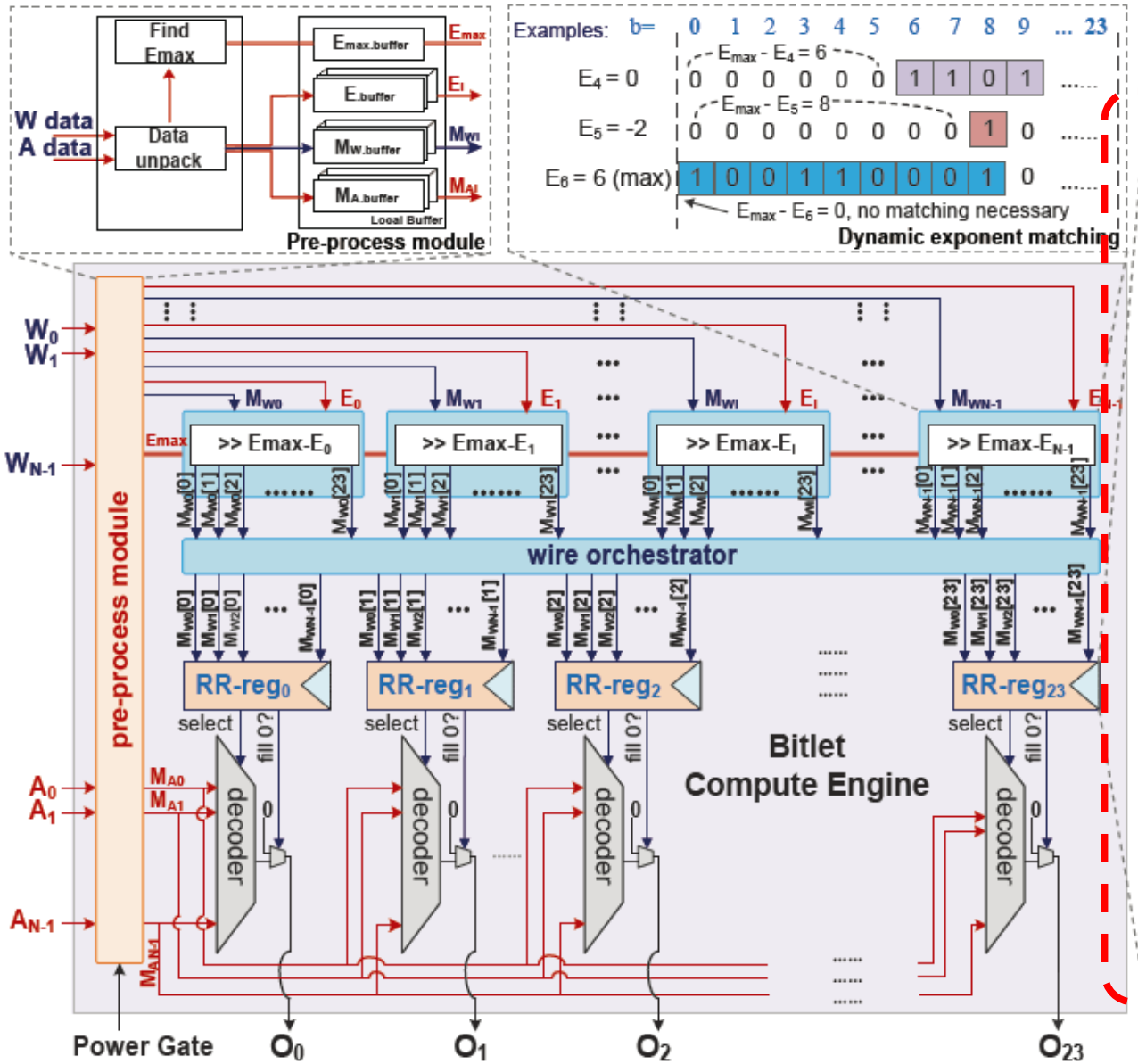


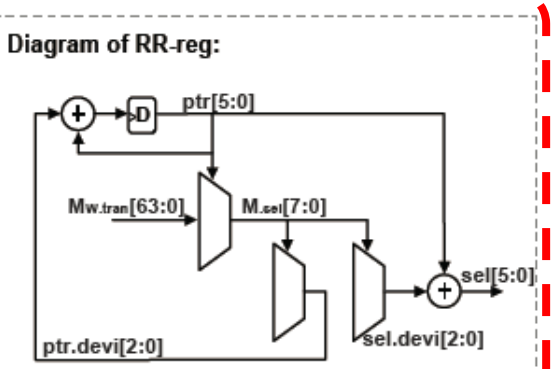
Logic in RR-reg:

```
i = 0, selector:
while(i<23):
    if  Mi[b] == 1
        selector.append[i]
if selector is empty:
    fill_0 = 1
else:
    item = selector.popfirst()
    select = item
    selector.delete[item]
    return select
```

Dynamic exponent matching: Emax is the maximum exponent, i.e. E6 in the above high level example

利用比特稀疏性的通用深度学习加速方法



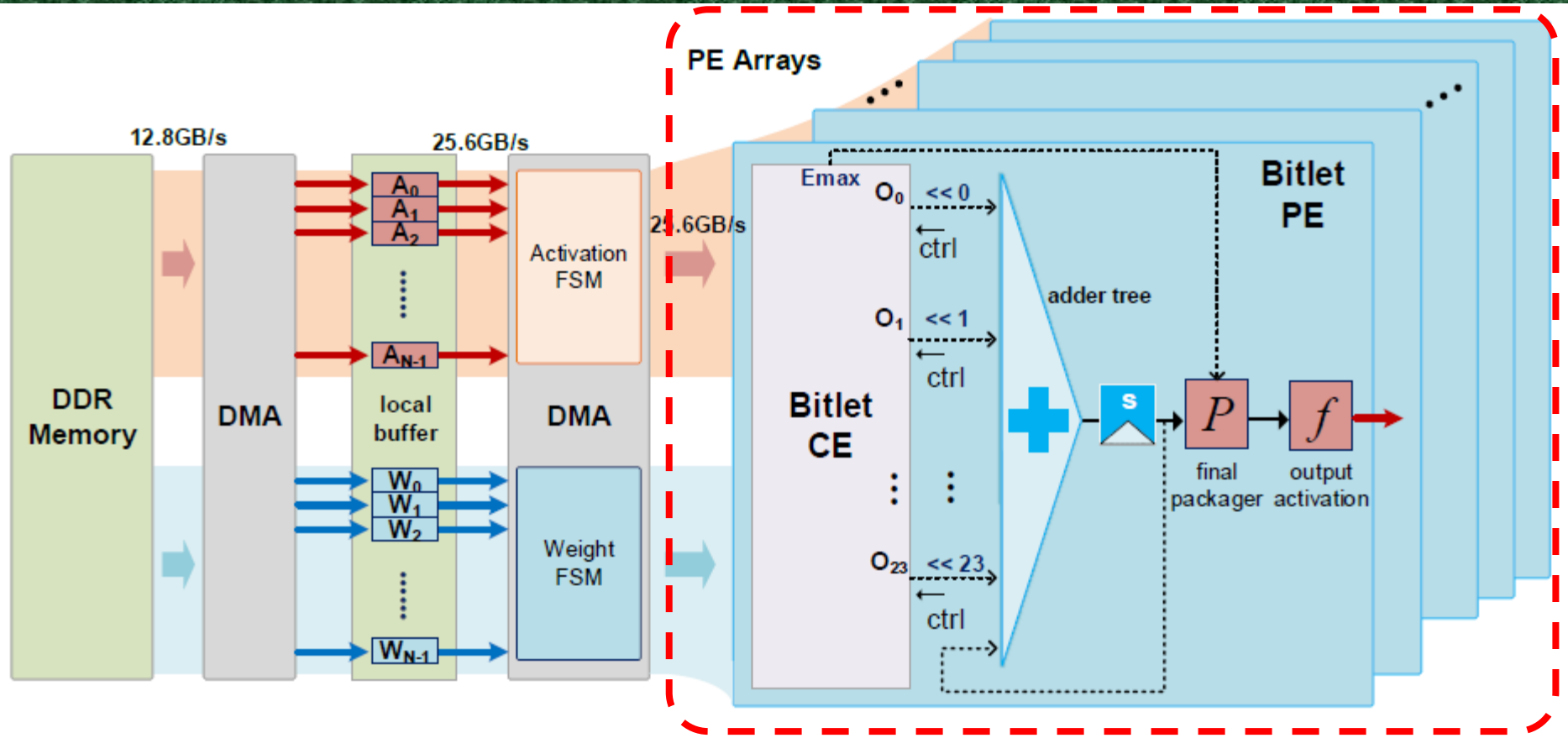


Logic in RR-reg:

```
i = 0, selector;
while(i<23):
    if  Mi[b] == 1
        selector.append[i]
if selector is empty:
    fill_0 = 1
else:
    item = selector.popfirst()
    select = item
    selector.delete[item]
    return select
```

Essential bit distillation: RR-reg is responsible for this operation, but the logic is very simple!

利用比特稀疏性的通用深度学习加速方法



Bitlet PE organization and the memory accesses. 每个PE共享访存带宽

- **整体加速器设计:**
  - 每个PE由一个BCE和后续adder tree构成
  - DDR3 is used on our FPGA Virtex-7 SoC platform.

利用比特稀疏性的通用深度学习加速方法

为了证明加速器的通用性，选取12个不同细分领域的AI task

Models	Type	Precision	Domain	Dataset	GFLOPS	Weights	W-bit Sparsity (%)
ResNet-50[18]	2D Convolution	8 bit	Image Classification	ILSVRC'12[3]	8.21	25.56M	70.15 (fixed point)
MobileNetV2[35]	2D Convolution	8 bit	Image Classification	ILSVRC'12[3]	0.615	3.49M	76.85 (fixed point)
YoloV3[34]	2D Convolution	8 bit	Object Detection	CoCo[1]	25.42	61.95M	77.78 (fixed point)
Multi-Pose[24]	2D Convolution	8 bit	Pose Estimation	CoCo[1]	97.55	59.59M	66.33 (fixed point)
lapSRN[25]	2D De-Convolution	16 bit	Image Super Resolution	SET14[4]	736.73	0.87M	74.31 (fixed point)
DCPDNet[45]	Encoder -Decoder	16 bit	Deraining /Dehazing	NYU-Depth[38]	254.37	66.9M	75.00 (fixed point)
DenseNet-161[20]	2D Convolution	16 bit	Image Classification	ILSVRC'12[3]	15.56	28.68M	68.92 (fixed point)
FCOS[39]	Feature Pyramid	16 bit	Object Detection	CoCo[1]	80.14	32.02M	70.83 (fixed point)
CartoonGAN[11]	GAN	float 32	Style Transfer	flickr[2]	108.98	11.69M	48.49 (floating point)
Transformer[41]	Seq2Seq	float 32	Word Embedding	wmt'14[6]	10.6	176M	45.75 (floating point)
C3D[40]	3D Convolution	float 32	Video Understanding	UCF101[5]	38.57	78.41M	45.83 (floating point)
D3DNet[44]	3D Deformable	float 32	Video Super Resolution	Vimeo-90k[42]	408.82	2.58M	47.69 (floating point)

In order to prove the general-purpose feature of bitlet, we use 12 domain-specific DL tasks with 8 of them quantized to 16bit and int8.

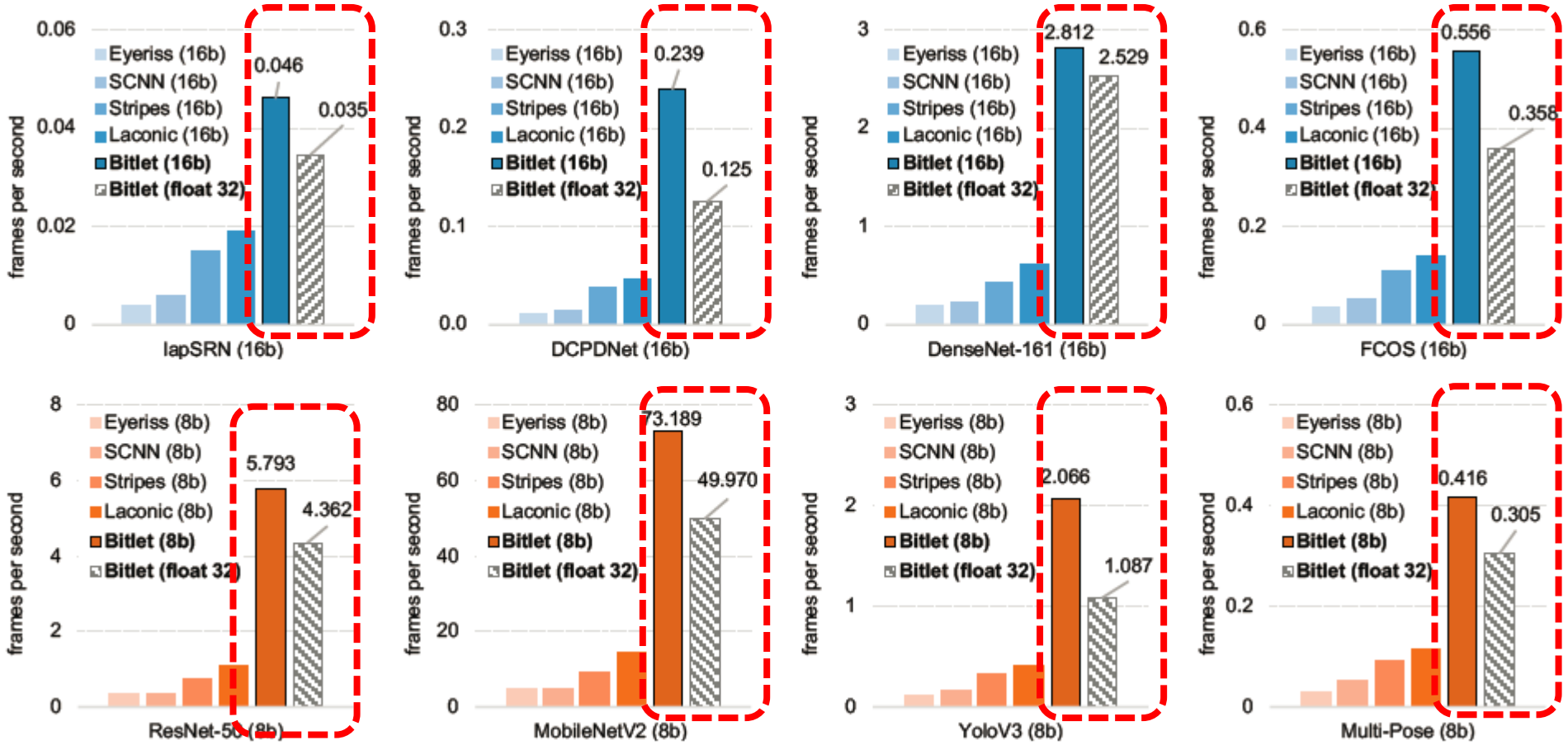
利用比特稀疏性的通用深度学习加速方法

具体测试指标：

	Accelerator ASICs						GPUs		
Chip	Eyeriss [14]	SCNN [32]	Stripes [22]	Laconic [36]	**Bitlet (Ours)**		Titan V	Titan Xp	Tegra X2
PEs/Cores	168	64	4096	192	**32**		5120	3840	256
Precision	16b	16b	1~16b	1~16b	**fp32/16, 1~24b**		fp32/16, 8b	fp32, 8b	fp32/16
Technology	65nm TSMC	16nm TSMC	65nm TSMC	65nm TSMC	**28nm TSMC**	**65nm TSMC**	12nm TSMC	16nm TSMC	16nm TSMC
Freq. (MHz)	250	1000	980	1000	**1000**		1455	1582	854
PEAK Performance (GOPs)	23.1	2000	–	–	**204.8 (fp32) 372.35 (16b) 744.7 (8b)**		14900 (fp32) 29800 (fp16)	12150 (fp32)	750.1(fp32) 1330 (fp16)
Power	278mW	–	–	–	**570mW(fp32) 432mW(16b) 366mW(8b)**	**1829mW(fp32) 1390mW(16b) 1199mW(8b)**	250W	250W	15W
PEAK Power Efficiency (GOPs/W)	83.09	–	–	441 (16b) 805 (8b)	**359.15 (fp32) 667.97(16b) 1335.93 (8b)**	**111.97 (fp32) 267.87 (16b) 621.10 (8b)**	59.6(fp32) 119.2(fp16)	48.6 (fp32)	50.0 (fp32) 88.7(fp16)
Area (mm^2)	12.25	7.9	122.1	1.59	**1.54**	**5.80**	–	–	–

Compared with SOTA accelerators, bitlet supports floating-point arithmetic with higher efficiency (GOPs/W)

利用比特稀疏性的通用深度学习加速方法



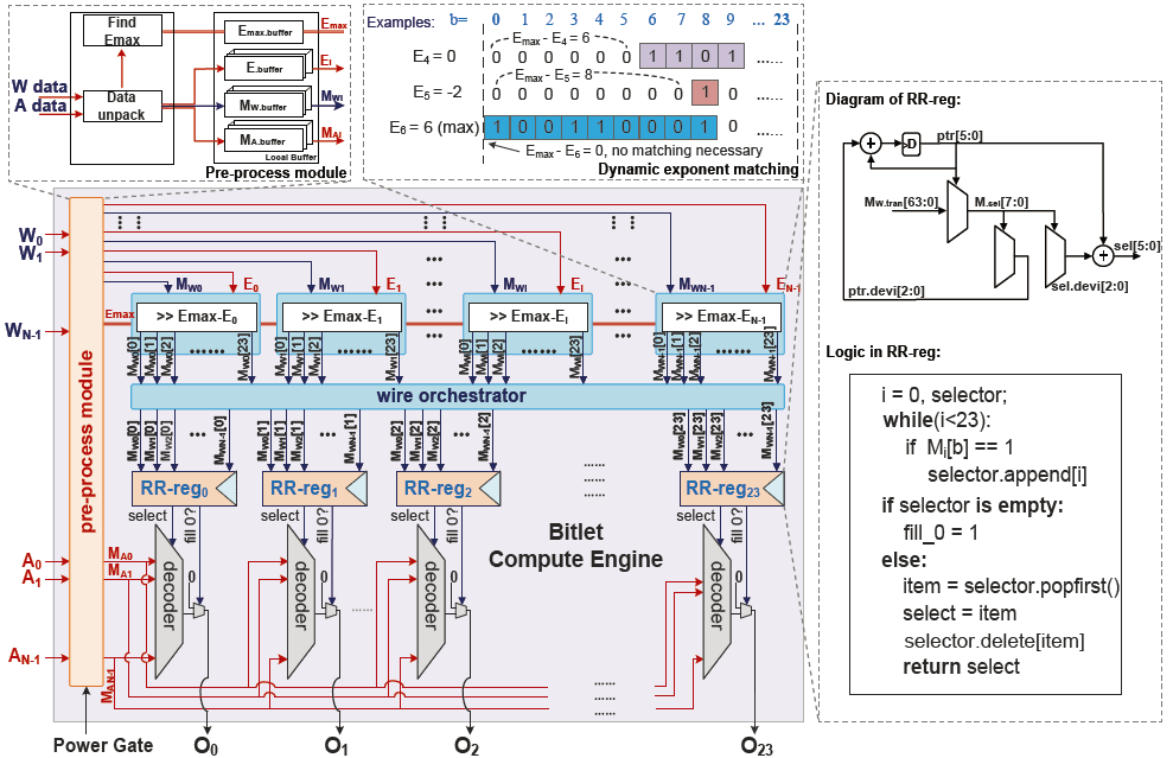
- We compare bitlet with fixed-point accelerators:
  - An interesting phenomenon is that bitlet-fp32 behaves even better than 16/8b fixed point accelerators!

利用比特稀疏性的通用深度学习加速方法

Instance	bare-m	*bitlet-fp32*			
Parameter	$N = 1$	$N = 32$		$N = 64$	
Ablation	w/o M w/o D	w/ M w/o D	w/ M w/ D	w/ M w/o D	w/ M w/ D
CartoonGAN	0.012	0.209	0.269	0.244	0.352
Transformer	0.126	2.152	2.879	2.509	3.763
C3D	0.035	0.590	0.771	0.670	0.976
D3DNet	0.003	0.056	0.068	0.065	0.084
Instance	bare-m	***Bitlet-16b***			
Parameter	$N = 1$	$N = 32$		$N = 64$	
Ablation	w/o M w/o D	w/o M w/o D	w/o M w/ D	w/o M w/o D	w/o M w/ D
lapSRN	0.004	0.033	0.040	0.038	0.046
DCPDNet	0.011	0.096	0.184	0.109	0.239
DenseNet-161	0.187	1.567	2.260	1.779	2.812
FCOS	0.036	0.304	0.455	0.345	0.556
Instance	bare-m	***Bitlet-8b***			
Parameter	$N = 1$	$N = 32$		$N = 64$	
Ablation	w/o M w/o D	w/o M w/o D	w/o M w/ D	w/o M w/o D	w/o M w/ D
ResNet-50	0.354	2.970	4.598	3.371	5.793
MobileNetV2	4.730	39.644	60.001	45.001	73.189
YoloV3	0.114	0.959	1.640	1.089	2.066
Multi-Pose	0.030	0.250	0.346	0.284	0.416

硬件消融实验：

- w/ or w/o M: with or without exponent Matching
- w/ or w/o D: with or without bit Distilation



利用比特稀疏性的通用深度学习加速方法

Item	*Bitlet*(float 32)		*Bitlet*(16b)	*Bitlet*(8b)
	Area (mm^2)	Power (mW)	Power (mW)	Power (mW)
Preprocess-ing Module	1.916 (33%)	1208.5 (66.1%)	1000.8 (71.9%)	1000.8 (83.5%)
Wire Orch. & Decoder	2.327 (40.1%)	164.1 (8.1%)	111.4 (8.0%)	55.7 (4.6%)
RR-Reg & Check Win.	0.7 (12.0%)	112.1 (7.2%)	74.8 (5.3%)	37.4 (2.9%)
Adder Tree	0.7 (12.0%)	293.3 (16.0%)	195.5 (14.1%)	97.8 (9.8%)
PostProcess-ing Module	0.2 (2.9%)	48.5 (2.7%)	7.4 (0.5%)	7.4 (0.6%)
Total	**5.8**	**1829.6**	**1390.0**	**1199.1**

TSMC 65nm

Item	*Bitlet*(float 32)		*Bitlet*(16b)	*Bitlet*(8b)
	Area (mm^2)	Power (mW)	Power (mW)	Power (mW)
Preprocess-ing Module	0.553 (35.8%)	356.8 (62.6%)	296.598 (68.6%)	296.598 (81.2%)
Wire Orch. & Decoder	0.570 (35.9%)	63.857 (11.2%)	40.28 (9.73%)	20.651 (5.54%)
RR-Reg & Check Win.	0.131 (9.6%)	27.37 (4.8%)	20.28 (4.56%)	10.128 (2.88%)
Adder Tree	0.244 (15.8%)	107.424 (18.8%)	71.616 (16.6%)	35.808 (9.80%)
PostProcess-ing Module	0.044 (2.9%)	14.88 (2.6%)	2.304 (0.53%)	2.304 (0.63%)
Total	**1.542**	**570.15**	**432.08**	**365.49**

TSMC 28nm

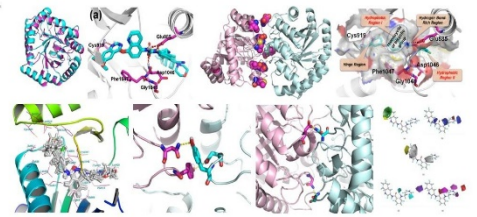
利用比特稀疏性的通用深度学习加速方法

- **FPGA上，本IP和赛灵思浮点数IP计算时间比较（64个浮点MAC）：**

资源消耗	本IP	赛灵思浮点乘法IP v12.0			
LUT	**2558**	35,773	34,083	33,270	33,419
FF	**2040**	8505	21,217	35,462	41,674
最大频率	300MHz	100MHz	200MHz	300MHz	400MHz
延迟	**40**	6	24	48	73



总结

云-边-端通用的加速方法（即将推出IP）

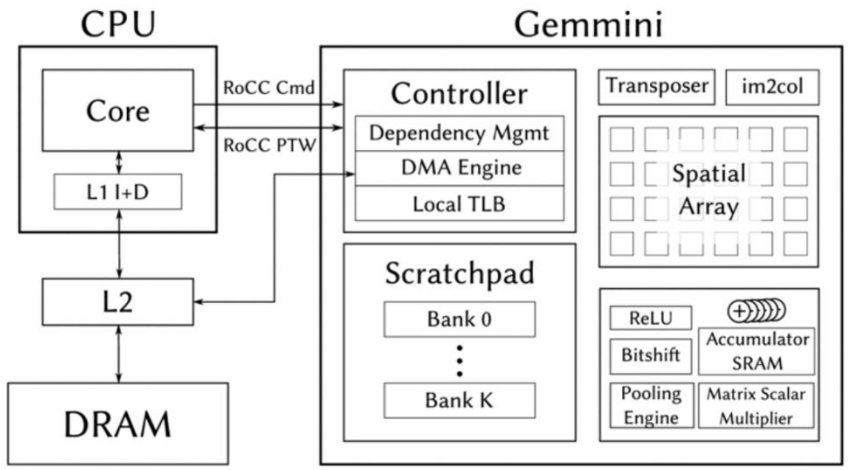
Features：一个硬件加速IP，云-边-端多场景使用。

- 通用性（General Purpose）——平衡能效和通用性，多种精度支持集成于同一个加速引擎，高度灵活可配置，无额外硬件设计开销。
- 多精度支持（multi-precision support）——支持浮点数（fp32/16，bfloat16），定点数（1~24bit位宽），实现混合精度计算；使用本IP在浮点数模式下的推理时间与普通加速IP的int8推理时间相当；

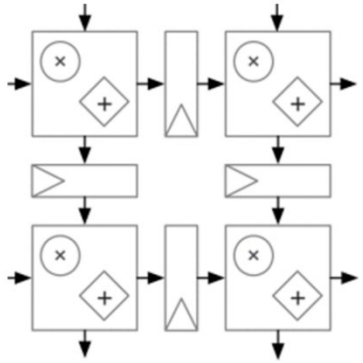
系统DEMO

感谢华为的支持！

HUAWEI

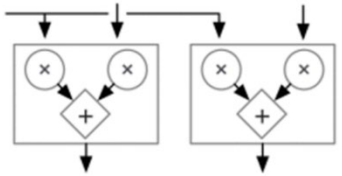


替换PE array



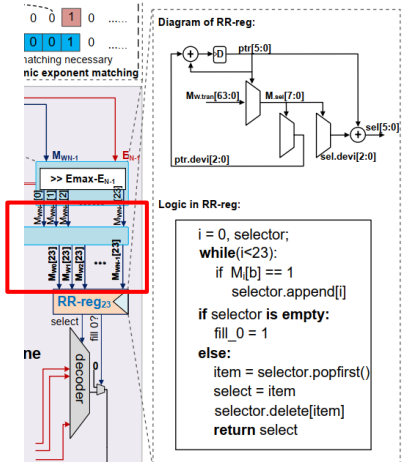
Systolic spatial array (TPU-like)

Spatial Array	Freq. (GHz)	Area at 500 MHz
Systolic	1.89	120K μm²
Vector	0.69	67K μm²



Parallel vector engines (NVDLA-like)

可配置的关键路径



同态加密软硬件协同加速方法

计算机体系结构国家重点实验室
State Key Laboratory of Computer Architecture, ICT, CAS

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

数据安全的重要性

数据安全上升为我国国家战略

数据面临着被窃取或篡改的潜在风险！

研究数据隐私保护技术对我国信息产业乃至国家安全都具有重要的意义，将会创造重大的经济效益和社会效益。

数据安全的重要性

隐私保护计算对企业具有重要意义

2018年Facebook的8700万用户数据被泄漏，公司股价一度蒸发590亿美元

2019年美国第七大商业银行CapitalOne的1.06亿用户数据被黑客通过攻破亚马逊的AWS云服务获取，严重损害用户隐私

2021年7月“滴滴出行”APP严重违法违规收集使用个人信息问题被我国网信办依法予以下架处理，并进一步上升为国家安全事件

隐私计算走红兴起，国内市场规模有望触达百亿元

北京日报客户端

值得一提的是，目前无论是BAT等大厂，或是初创型科技企业，都在纷纷接连入局隐私计算。蚂蚁金服、腾讯云、百度推出了各自的产品，阿里巴巴、微众银行、京东等企业也在各自的技术领域形成一定优势。此外，华控清交、富数科技、矩阵元、数牍科技、锘崴科技、光之树科技、零知识科技等一批专注于隐私计算产品化的初创企业也不断涌现。诸多区块链企业、数据安全企业、金融风控企业、电信企业等也纷纷拥抱隐私计算技术。

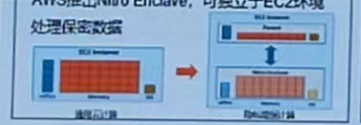
数据安全的重要性

云计算安全的未来是机密计算

- 云计算发展迅猛，包括金融、电信等强监管的业务都已经开始上云
  - 纳斯达克会分阶段把全部业务迁移到亚马逊云
  - 日本最大的电信运营商会把数据仓库上云
  - 美国富国银行的目标是10年后让其所有工作量都在公共云上运行
- 微软AZure云，谷歌云，亚马逊云，阿里云等持续布局机密计算
- 安全和隐私问题已经成为云计算发展的最大障碍
- 云计算进入密态计算才有可能最终解决客户的安全顾虑
- 国际主流CPU厂商都已推出机密计算方案

AWS、Google、微软、IBM等巨头持续布局机密计算

AWS推出Nitro Enclave，可独立于EC2环境处理保密数据



Google Cloud基于AMD二代霄龙处理器，推出Confidential VMs和Shielded VMs服务

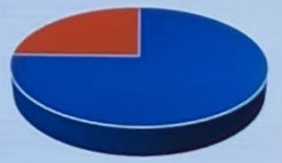
微软云推出Confidential Compution

支持Intel SGX、AMD SEV、可信虚拟机等多种产品模式

IBM推出基于Intel SGX的数据隐私增强产品：Data Shield，支持裸金属和容器云多种模式

中国将占~15%的市场份额

监管行业推动了75%的需求



同态加密——密码学的“圣杯”

同态加密是隐私保护计算的重要手段

Fully Homomorphic Encryption

Idea: Outsource computation w/o revealing inputs.

Enc(x)

Enc(f(x))

User: Has input x, Wants f(x)

Cloud: has f(·)

通信成本

安全成本

降低计算供应商数据安全方面的成本

信任成本

减少通信代价，平衡各方的计算代价

Homomorphic Encryption

吸引更多用户将数据和计算迁移至云端

丰富应用

实现数据存储加密,保障数据安全

打破数据孤岛，用于联邦学习

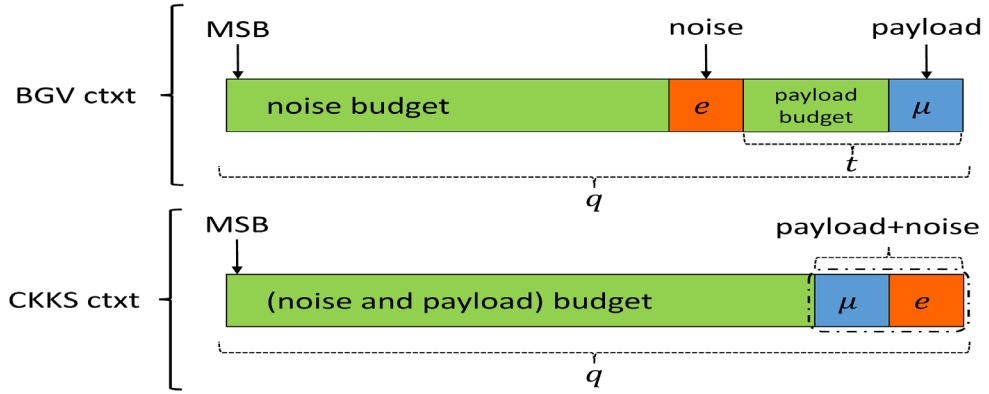
数据安全

同态加密可以在不交换明文数据的情况下依然进行业务计算

同态加密——密码学的“圣杯”

虽然同态加密在隐私计算领域有着重要的作用，但是巨大的计算开销阻碍了它的实际应用。如何在保证密文计算结果正确性的前提下，提高计算性能和能效是当前业界公认的难题。

应用	同态加密方式	明文数据长度	密文数据长度	明文计算时间	密文计算时间
联邦学习模型训练	CKKS	30.5 KB	**1.33 GB (5.2万倍)**	0.145 秒	**390.6 秒 (2694倍)**
神经网络推理	CKKS	29.9 MB	**546.8 GB (1.9万倍)**	212.16 秒	**112,026.16 秒 (528倍)**
保密数据库查询	BGV	0.71 KB	**1.58 GB (40万倍)**	0.053 毫秒	**7401.49秒 (1.4 亿倍)**



现存挑战

迫切需要软硬件协同加速方法，提升性能和可用性

性能距离实际应用需要仍有显著差距

加速器结构无法支持**多样化加密方案**

缺少面向实际应用的**系统化解决方案**

国内外研究现状

国内外研究现状

体系结构顶级会议近年来较为活跃

国外代表性研究机构	代表性成果（论文、专利、标准）	成果描述
加州大学圣地亚哥分校 微软研究院	HEAX: An Architecture for Computing on Encrypted Data **(ASPLOS 2020)**	为CKKS全同态加密方案提出了一种新颖的FHE硬件架构：HEAX，基于可重构硬件实现了多种加密参数的硬件加速器，**相较于传统软件实现获得了164-268倍的性能提升。**
比利时微电子研究中心 伯明翰大学	FPGA-Based High-Performance Parallel Architecture for Homomorphic Computing on Encrypted Data **(HPCA 2019)**	基于ARM+FPGA架构，为BFV全同态加密方案的计算密集型算子设计了定制的协处理器。在Xilinx Zynq UltraScale+ MPSoC ZCU102上进行测试，在200 MHz时钟频率下，相对于在**1.8 GHz下运行的Intel i5处理器上的BFV的软件实现，获得了13倍以上的加速**。
纽约大学 Facebook AI研究院 首尔国立大学 哈佛大学 Facebook虚拟现实实验室	Cheetah: Optimizing and Accelerating Homomorphic Encryption for Private Inference **(HPCA 2021)**	针对云服务器上同态加密下的深度学习推理速度缓慢的问题，提出了用于服务器端的同态加密深度学习推理的算法和硬件优化。在软件层面，通过参数调整和算子调度优化，获得了79倍的加速；同时，通过对各种加速器微架构进行设计空间探索，评估得出**理论上ResNet50的推理可以接近实时。**
麻省理工学院 密歇根大学	F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption **（MICRO 2021）**	提出一个通用可编程的针对FHE的全同态加速器ASIC，针对FHE的算法特点进行加速，构建了宽矢量处理器F1，支持各种FHE算子包括模乘，模加，NTT等，并通过编译器设计和架构设计减少数据搬运，**实现了5400x和17000x的性能提升。**
首尔国立大学 KAIST	BTS: An Accelerator for Bootstrappable Fully Homomorphic Encryption **（ISCA 2022）**	主要解决bootstrap的加速问题。针对bootstrap的计算和访存量大的问题，提出BTS方案，通过NoC对不同的FHE计算核进行互连。由于支持bootstrap，**可进行神经网络模型的密文计算，在ResNet20上速度提升为5556x，在LR上提升为1306x。**

国内外研究现状

国内外研究现状

国内针对同态加密加速方法的研究总体落后于国外

国内代表性研究机构	代表性成果（论文、专利、标准）	成果描述
中国科学院计算技术研究所	Poseidon: Practical Homomorphic Encryption Accelerator （High Performance Computer Architecture, HPCA 2023）	对CKKS的基本操作进行细粒度的拆分，包括：keyswitch，升降模数，模加模乘，rotation，rescale和bootstrap。拆分后重用算子，包括NTT、MA、MM、NTT和Automorphism，取模操作也是复用的。**性能相比于CPU提升370倍，相比于ASIC提升10.6倍。**
香港科技大学	FPGA-Based Hardware Accelerator of Homomorphic Encryption for Efficient Federated Learning (Computer Science Cryptography and Security 2020)	为了减小基于同态加密的联邦学习中巨大的时间开销，设计了一种用于Paillier同态加密方案的FPGA加密框架。实验结果表明，**将各种联邦学习模型训练过程中的加密时间减少了71%。**
复旦大学	A Multi-Layer Parallel Hardware Architecture for Homomorphic Computation in Machine Learning (ISCAS 2021)	针对同态加密下的AI算法的推理速度慢的问题，提出了一种用于机器学习中同态计算的基于FPGA的多级并行硬件加速器。在ZCU102开发板上的实验结果表明该加速器优于以前的工作，并且相较于软件实现获得了一个数量级的加速。
香港科技大学	HAFLO: GPU-Based Acceleration for Federated Logistic Regression (Computer Science Machine Learning 2021)	为了提高同态加密下的联邦逻辑回归的计算速度，提出了**一种基于GPU的解决方案**：HAFLO。实验结果表明，在流行的联邦学习框架 FATE上，异构逻辑回归计算实现了49.9倍的加速，**同构逻辑回归计算实现了88.4倍的加速。**
	Efficient Comparison and Addition for FHE with Weighted	为了解决基于同态加密的算法的密文计算与明文计算过程的不一致问题，**提出了一种评估模型用于定量分析同态乘**

加速器计算性能距离实际应用相差较远，面向实际应用的系统化解决方案几乎没有

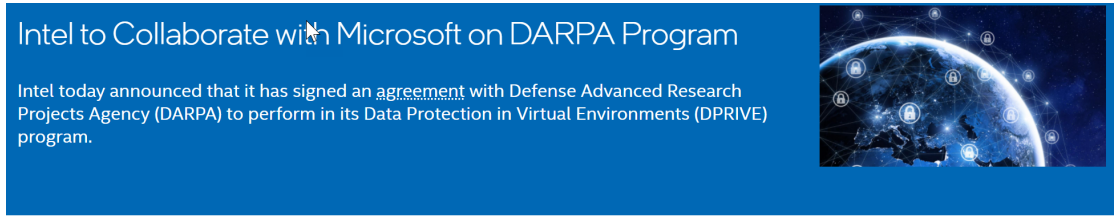
西方国家已全方位布局同态加密芯片技术

美国

军方研究单位DARPA于2019年陆续启动项目，研究新一代软硬件隔离的安全数据流通计算架构、数据隐私处理芯片和硬件加速器，特别针对同态加密的硬件加速，总预算额约1亿美元，微软、IBM、英特尔、通用电气等参与芯片设计和软件库开发。

欧盟

批准HEAT项目，旨在研究加密的方式处理敏感数据，保护个人隐私信息的新型软硬件系统架构。



News

- March 8, 2021
- Contact Intel PR

More Security News →

Tags

Security

What's New: Intel today announced that it has signed an agreement with Defense Advanced Research Projects Agency (DARPA) to perform in its Data Protection in Virtual Environments (DPRIVE) program. The program aims to develop an accelerator for fully homomorphic encryption (FHE). Microsoft is the key cloud ecosystem and homomorphic encryption partner leading the commercial adoption of the technology once developed by testing it in its cloud offerings, including Microsoft Azure and the Microsoft JEDI cloud, with the U.S. government. The multiyear program represents a cross-team effort across multiple Intel groups, including Intel Labs, the Design Engineering Group and the Data Platforms Group, to tackle "the final frontier" in data privacy, which is computing on fully encrypted data without access to decryption keys.

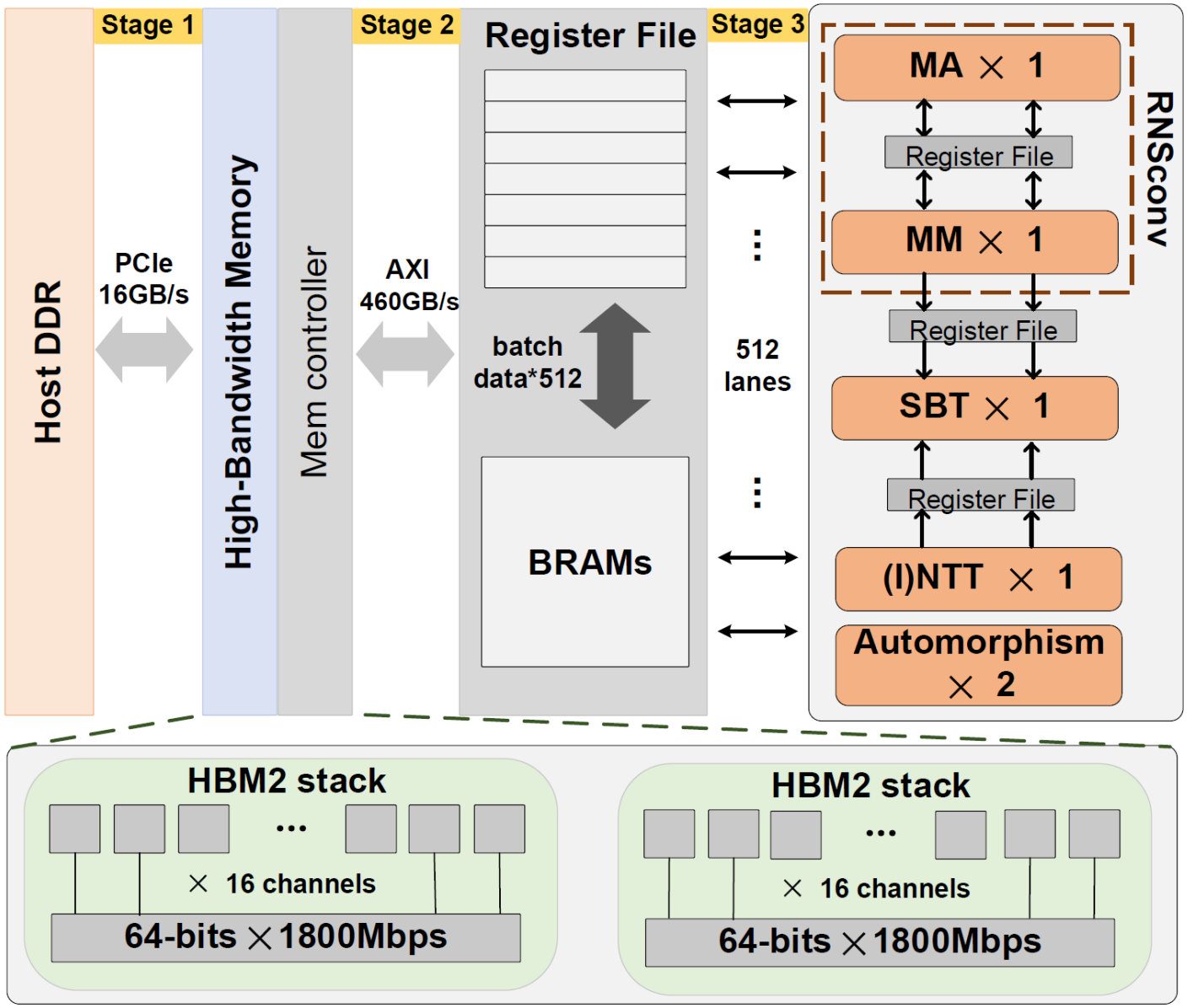
同态加密算法的特性

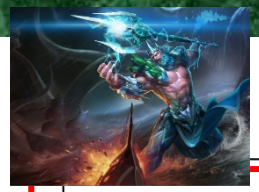
应用	总存储器访问（未加密 / 加密）	理论带宽需求	Runtime（未加密，无内存访问限制）	Runtime（加密，无内存访问限制）
线性回归（训练）	48.63MB / 538.5GB	**3.37 TB/s**	0.145 s	**390.6 s (2,693.8x)**
线性回归（测试）	26.19MB / 28.6GB	**1.99 TB/s**	13.48 ms	**2.44 s (181.4x)**
MNIST 推理	19.25GB / 62,408.5GB	**273.28 GB/s**	212.16 s	**112,026.16 s (528.0x)**
矩阵相乘	0.565 GB / 1,683.52 GB	**1.49 TB/s**	1.1 s	**1,838.5 s (1,111.76x)**

实验参数配置： 10-core / 20-thread, 2.2GHz Xeon Silver E4114 CPU×2; DDR4, 2666 MHz DRAM; 4TB SATA III hard drive

DDR4 2666MHz峰值带宽： 21.3 GB/s

Poseidon——算子重用的同态加密加速器





细粒度算子

	MA	MM	NTT	Automorphism	SBT
Modup	✓	✓			✓
Moddown	✓	✓			✓
HAdd	✓				
PMult	✓	✓	✓		✓
CMult	✓	✓	✓		✓
Rotation	✓	✓	✓	✓	✓
Keyswitch	✓	✓	✓		✓
Rescale	✓	✓	✓		✓
Bootstrapping	✓	✓	✓	✓	✓

③ **Rescale**: this operation scales down the `PMult` or `CMult` result by the scaling factor Δ. The process can not be achieved by simple division in the RNS-based FHE schemes, because the large integers, including the modulus and polynomial coefficients, have already been decomposed into multiple small-integers, a.k.a., the RNS components, by the Chinese Remainder Theorem.

Therefore, the RNS-based `Rescale` operation is described formally as $ct = (ct^{(j)} = (c_0^{(j)}, c_1^{(j)})_{0 \le j \le l})$ into $ct' = (ct'^{(j)} = (c_0'^{(j)}, c_1'^{(j)})_{0 \le j \le l-1})$, where $c_r'^{(j)} = q_l^{-1} \cdot (c_r^{(j)} - c_r^{(l)}) \mod q_j$ for $r = 0, 1$ and $0 \le j \le l-1$. The $ct^{(j)}$ is one of the RNS components of a ciphertext, and q_l^{-1} is the inverse of q_l under q_j, which satisfies $q_l^{-1} \cdot q_l \mod q_j = 1$.

Operators: the `Rescale` operation is fulfilled by a series of ModAdd and ModMult. Therefore, it contains three lower-level operators as well: **MA**, **MM**, and **NTT/INTT**.

将同态加密的基本操作进行算子拆分

取私钥(1 , s)； 公钥(b = − a · s + e , a)

其中s和a向量，e是一个随机数，b 是一个数，

这里就由RLWE问题确保了一件事：仅使用公钥的话很难解出私钥)

密文(c_0 ,c_1) = r (b , a) + (m + e_1 , e_2) = (rb + m + e_1 , ra + e_2)，其中r是随机整数，e_1 , e_2是随机的多项式（也可以理解为向量）。

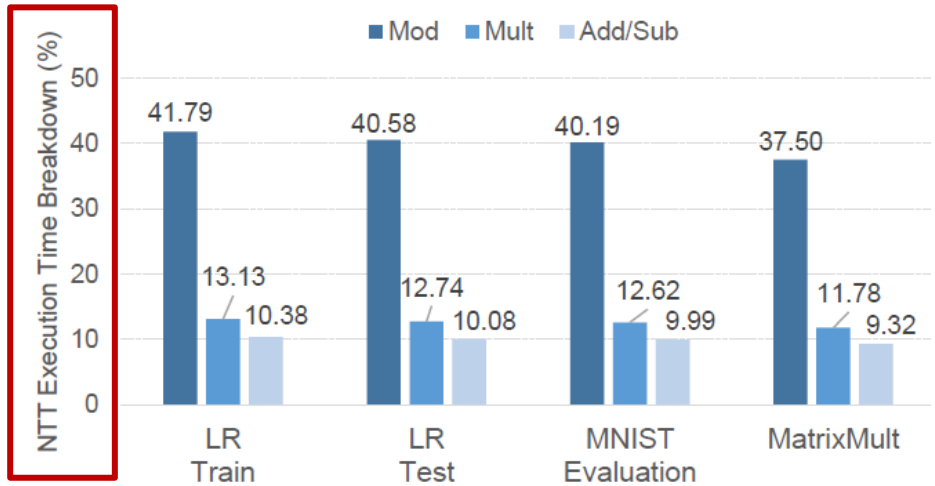
解密：m(X) <- (c_0(X), c_1(X)) $c_0 + c_1 s = rb + m + e_1 + ra \cdot s + e_2 s = m + e_1 + e_2 s - er$

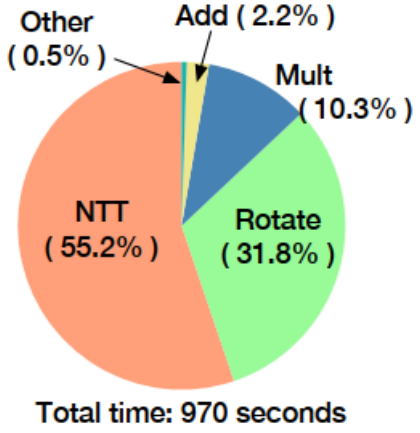
$$(c_{11} + c_{12}s) \times (c_{21} + c_{22}s) = c_{11}c_{21} + (c_{21}c_{12} + c_{11}c_{22})s + c_{12}c_{22}s^2$$

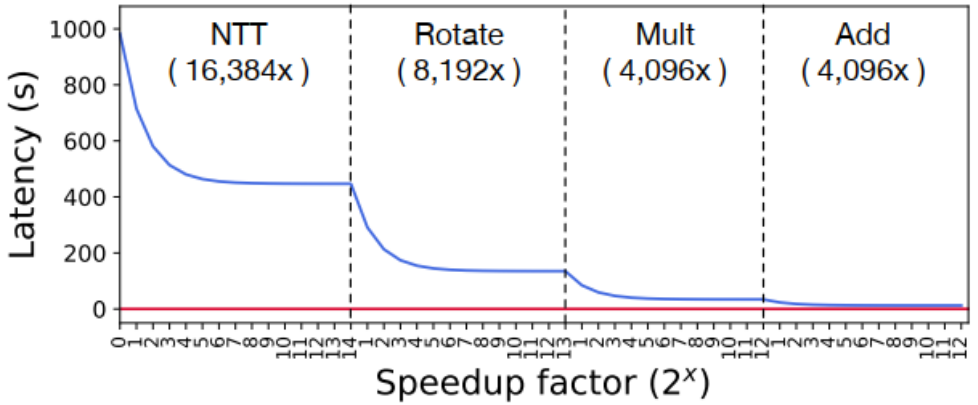
算子分析与复杂度优化

- 核心算子计算时间分解
  - **NTT**
  - Add
  - Mult
  - Keyswitch
  - Rotation
  - Conjugate
  - Rescale

基本算子里Mod占据绝大部分计算时间







Algorithm 1 Optimized Modular Mult. | MulRed(x, y, y', p)

Input: $x, y \in \mathbb{Z}_p$, $p < 2^{w-2}$, and $y' = \lfloor y \cdot 2^w / p \rfloor$
Output: $z \leftarrow x \cdot y \pmod p$

1: $z \leftarrow x \cdot y \pmod{2^w}$ ▷ the lower word of the product
2: $t \leftarrow \lfloor x \cdot y' / 2^w \rfloor$ ▷ the upper word of the product
3: $z_\epsilon \leftarrow t \cdot p \pmod{2^w}$ ▷ the lower word of the product
4: $z \leftarrow z - z_\epsilon$ ▷ single-word subtraction
5: **if** $z \geq p$ **then**
6: $z \leftarrow z - p$
7: **end if**

模乘算法

Algorithm 2.14 Barrett reduction

INPUT: p, $b \geq 3$, $k = \lfloor \log_b p \rfloor + 1$, $0 \leq z < b^{2k}$, and $\mu = \lfloor b^{2k} / p \rfloor$.
OUTPUT: $z \bmod p$.

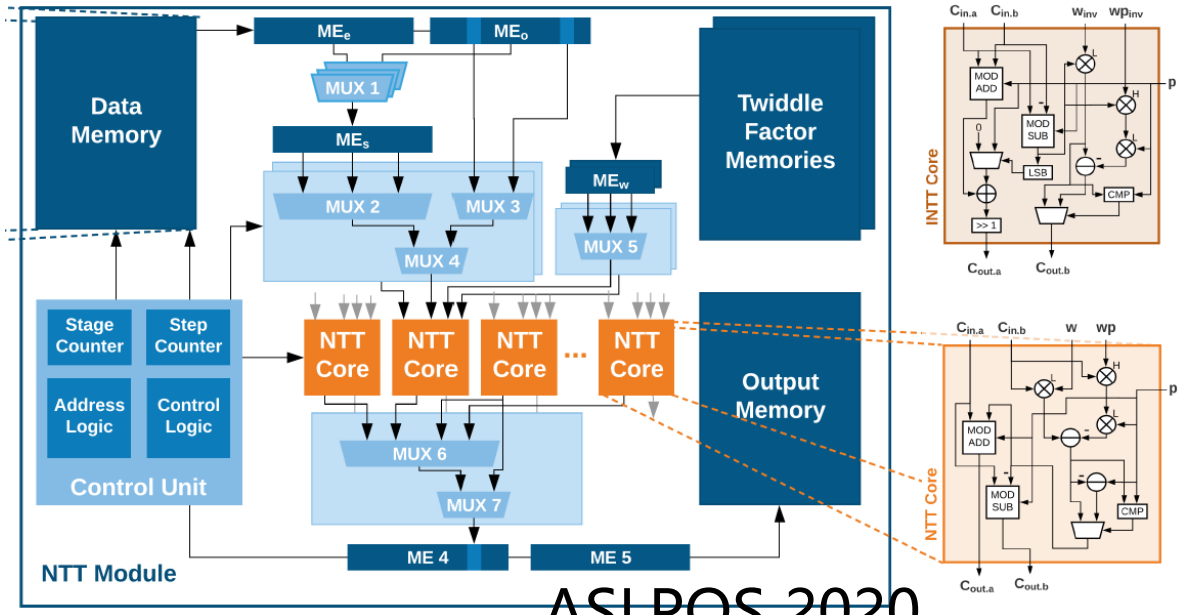
1. $\widehat{q} \leftarrow \left\lfloor \lfloor z / b^{k-1} \rfloor \cdot \mu / b^{k+1} \right\rfloor$.
2. $r \leftarrow (z \bmod b^{k+1}) - (\widehat{q} \cdot p \bmod b^{k+1})$.
3. If $r < 0$ then $r \leftarrow r + b^{k+1}$.
4. While $r \geq p$ do: $r \leftarrow r - p$.
5. Return(r).

求模算法

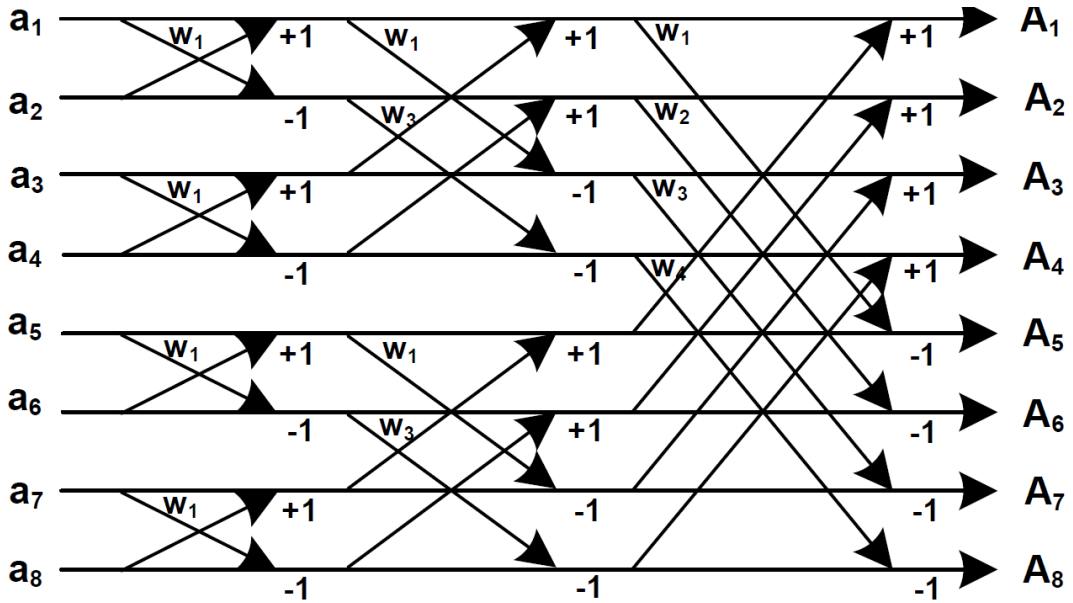
算子分析与复杂度优化

NTT的计算开销

- 计算方式：**TAM（Twiddle, Accumulation and Modulo)**
- 旋转因子：g^((p-1)/n)，g为原根，p为模数，可取998244353。
- 同时进行NTT的**多项式维度**：4096 ~ 65536个



大“计算量”以及大“并行度”



引理：另 X1 =a1+a2 · w1, X2 =a3+a4 · w2 , X3 =[X1 mod q+(X2 mod q) · w3] mod q,

因此，X3 = (X1+X2 · w3) mod q.

算子分析与复杂度优化

NTT 融合

定理：

$A1 = [a1 + a2 \cdot w1 + (a3 + a4 \cdot w1) \cdot w1] \bmod q$

$A2 = [a1 - a2 \cdot w1 + (a3 - a4 \cdot w1) \cdot w2] \bmod q$

$A3 = [a1 + a2 \cdot w1 - (a3 + a4 \cdot w1) \cdot w1] \bmod q$

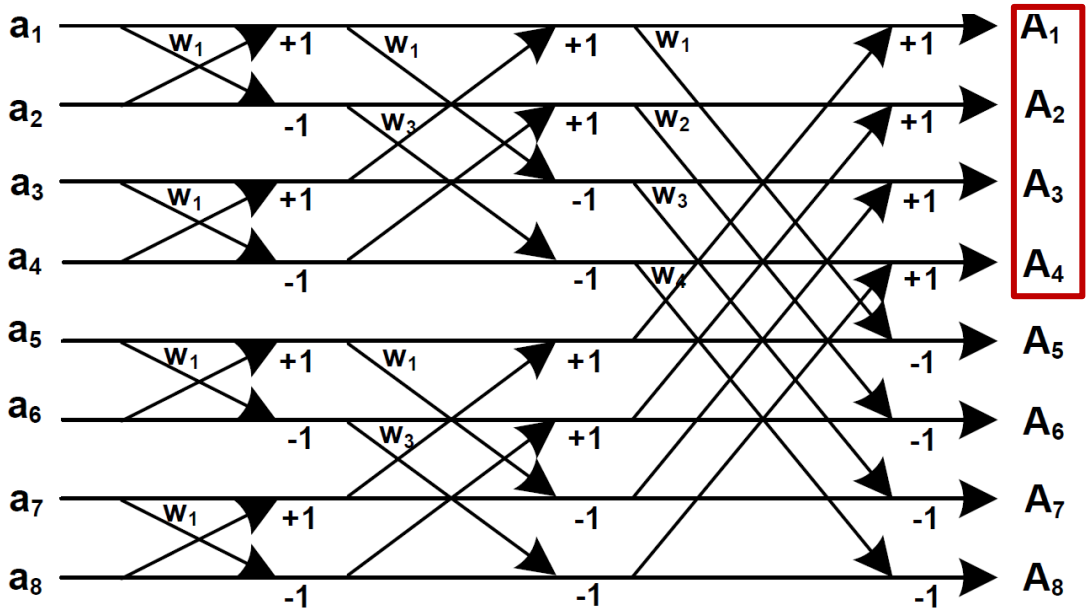
$A4 = [a1 - a2 \cdot w1 - (a3 - a4 \cdot w1) \cdot w2] \bmod q$

……

Example: 4096维度的多项式，计算NTT操作需要12个phase，每个phase需要做4096次TAM；

融合之后：4个phase，每个phase 4096次TAM

大“计算量”以及大“并行度”



算子分析与复杂度优化

密文计算下的性能提升

应用	线性回归（训练）	线性回归（测试）	MNIST推理	矩阵相乘
未融合（10^9）	266	0.828	31900	428
NTT融合（10^9）	**102**	**0.276**	**12300**	**165**

访存优化

应用	总存储器访问（未加密 / 加密）	理论带宽需求	Runtime（未加密，无内存访问限制）	Runtime（加密，无内存访问限制）
线性回归（训练）	48.63MB / 538.5GB	**3.37 TB/s**	0.145 s	**390.6 s (2,693.8x)**
线性回归（测试）	26.19MB / 28.6GB	**1.99 TB/s**	13.48 ms	**2.44 s (181.4x)**
MNIST 推理	19.25GB / 62,408.5GB	**273.28 GB/s**	212.16 s	**112,026.16 s (528.0x)**
矩阵相乘	0.565 GB / 1,683.52 GB	**1.49 TB/s**	1.1 s	**1,838.5 s (1,111.76x)**

实验参数配置： 10-core / 20-thread, 2.2GHz Xeon Silver E4114 CPU×2; DDR4, 2666 MHz DRAM; 4TB SATA III hard drive

DDR4 2666MHz峰值带宽： 21.3 GB/s

访存优化

片上提升访存效率，片外增加带宽

: the 1st 8 input data for the NTT core
: the 4th 8 input data for the NTT core
: the 8th 8 input data for the NTT core

0	15	22	29	36	43	50	57
1	8	23	30	37	44	51	58
2	9	16	31	38	45	52	59
3	10	17	24	39	46	53	60
4	11	18	25	32	47	54	61
5	12	19	26	33	40	55	62
6	13	20	27	34	41	48	63
7	14	21	28	35	42	49	56

Iter1: 512 (=4096/8)

Offset = 1

0	8	⋯	4080	4088
1	9	⋯	4081	4089
⋮	⋮		⋮	⋮
6	14	⋯	4086	4094
7	15	⋯	4087	4095

Iter2: 64 (=4096/64)

Offset = 8

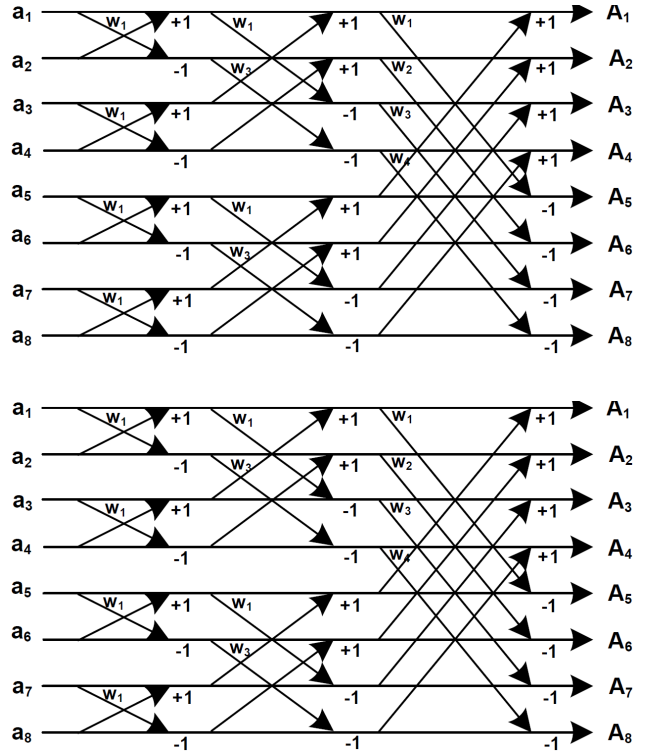
0 ~ 3 ~ 7	64 ~ 71	⋯	3968 ~ 3975	4032 ~ 4039
8 ~ 11 ~ 15	72 ~ 79	⋯	3976 ~ 3983	4040 ~ 4047
⋮	⋮		⋮	⋮
48 ~ 51 ~ 55	112~119	⋯	4016 ~ 4032	4080 ~ 4087
56 ~ 59 ~ 63	120~127	⋯	4024 ~ 4021	4088 ~ 4095

Iter3: 8 (=4096/512)

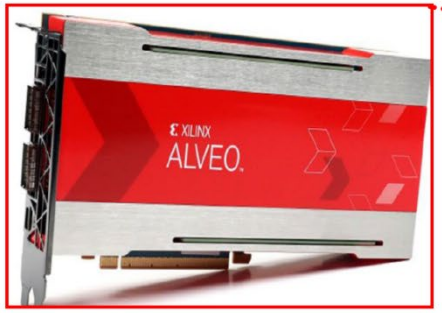
Offset = 64

0 ~ 63	512 ~ 575	⋯	3072 ~ 3135	3584 ~ 3647
64 ~ 127	576 ~ 639	⋯	3136 ~ 3199	3648 ~ 3711
⋮	⋮		⋮	⋮
384 ~ 447	896 ~ 959	⋯	3456 ~ 3519	3968 ~ 4031
448 ~ 511	960 ~ 1023	⋯	3520 ~ 3583	4032 ~ 4095

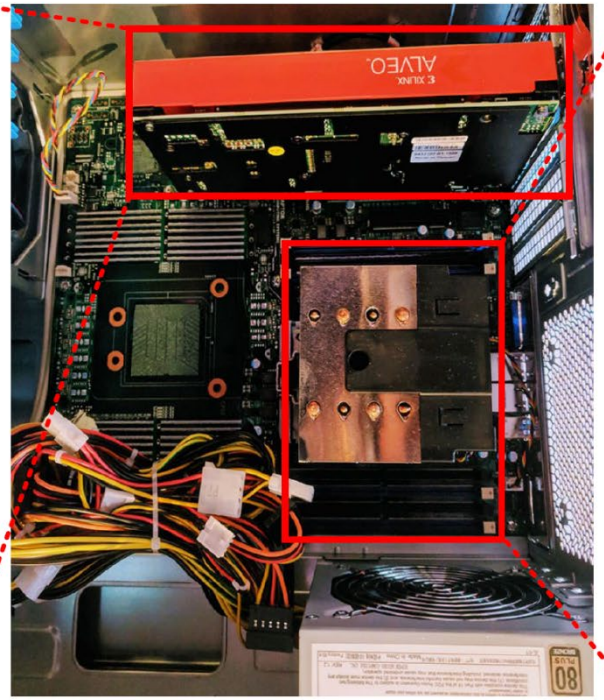
采用融合的NTT需要对片上访存模式重新设计。



Poseidon的系统实现



U280	
Storage	
HBM	8 GB
DRAM	32 GB
FPGA	
Device	xcvu37p
LUT	1304 k
FF	2607 k
DSP	9024
BRAM	70.9 Mb



CPU	
Core:	10-core Intel Xeon Silver 4114
Frequency:	2.2GHz
Threads:	20
TDP:	85W
DRAM	
Capacity:	64 GB x 4
Interface:	DDR4
Frequency:	2666 MHz
SSD	
Capacity:	256 GB
Interface :	SATA 3.0
HDD	
Capacity:	4 TB
Interface:	SATA 3.0

	CPU (Xeon)	Over 100x (GPU) [21]	HEAX (FPGA) [32]	Poseidon (FPGA)	speedup
HAdd	35.56	4807	4,161	13,310	**374**×
PMult	38.14	7,407	4,161	13,310	**349**×
CMult	0.38	57	119	273	**718**×
NTT	9.25	/	237	12,474	**1,348**×
Keyswitch	0.4	/	104	312	**780**×
Rotation	0.39	61	/	302	**774**×
Rescale	6.9	1,574	/	3,948	**572**×

	LR [19]	LSTM [27]	ResNet-20 [28]	Packed Boot-strapping [30]
F1+ (ASIC)	639	2,573	2,693	58.3
CraterLake (ASIC)	119.52	138.0	249.45	3.91
BTS-1 (ASIC)	39.9	/	1,910	/
BTS-2 (ASIC)	28.4	/	2,020	/
BTS-3 (ASIC)	43.5	/	3,090	/
ARK (ASIC)	7.717	/	294	/
over100x (GPU)	775	/	/	/
Poseidon (FPGA)	**72.98**	**1,848.89**	**2,661.23**	**127.45**

Poseidon的系统实现

全系统性能和带宽、逻辑资源使用情况

HPCA 2023 accept!

	HBM Capacity / Bandwidth (GB / TB/s)	Scratchpad Capacity / Bandwidth (MB / TB/s)	Running Fre. (GHz)
F1+ [35], [36] (ASIC)	16 / 1	256 / 29	1
CraterLake [36] (ASIC)	16 / 1	256 / 29	1
BTS [24] (ASIC)	16 / 1	512 / 38.4	1.2
ARK [23] (ASIC)	32 / 2	512 / 20	1
Poseidon (FPGA)	**8 / 0.45**	**8.6 / 3.4**	**0.45**

为了支持高速计算，其他工作在片上需要256~512M不等的ScratchPad

	LR [19]	LSTM [27]	ResNet-20 [28]	Packed Bootstrapping [30]
HAdd (%)	97.79	97.69	97.76	63.29
PMult (%)	97.65	97.15	97.48	97.48
CMult (%)	44.72	55.55	50.15	72.35
Keyswitch (%)	36.8	47.47	42.05	63.29
Rotation (%)	65	32.39	58.67	48.67
Rescale (%)	26.16	29.98	26.83	26.83
Bootstrapping (%)	46.39	56.43	52.18	/
Average (%)	**42.78**	**51.99**	**48.08**	**59.07**

	LUT (k)	FF (k)	DSP	BRAM	Latency (cycles)
MA (×1)	50	68	0	0	3
MM (×1)	170	160	1536	0	5
NTT (×1)	358	344	4032	1024	21
Automorphism (×2)	52	2	0	1024	517
SBT (×1)	98	403	3072	0	11

模加比模乘的带宽利用率高

总结

云计算安全的未来是机密计算

- 云计算发展迅猛，包括金融、电信等强监管的业务都已经开始上云
  - 纳斯达克会分阶段把全部业务迁移到亚马逊云
  - 日本最大的电信运营商会把数据仓库上云
  - 美国富国银行的目标是10年后让其所有工作量都在公共云上运行
- 微软AZure云，谷歌云，亚马逊云，阿里云等持续布局机密计算
- 安全和隐私问题已经成为云计算发展的最大障碍
- 云计算进入密态计算才有可能最终解决客户的安全顾虑
- 国际主流CPU厂商都已推出机密计算方案

AWS、Google、微软、IBM等巨头持续布局机密计算

AWS推出Nitro Enclave，可独立于EC2环境处理保密数据

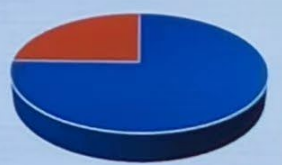
Google Cloud基于AMD二代霄龙处理器，推出Confidential VMs和Shielded VMs服务

微软云推出Confidential Compution支持Intel SGX、AMD SEV、可信虚拟机等多种产品模式

IBM推出基于Intel SGX的数据隐私增强产品：Data Shield，支持裸金属和容器云多种模式

中国将占~15%的市场份额

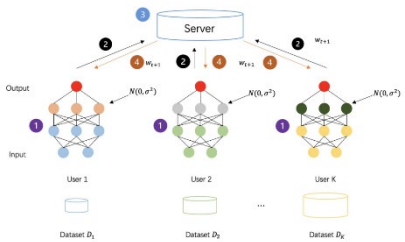
监管行业推动了75%的需求



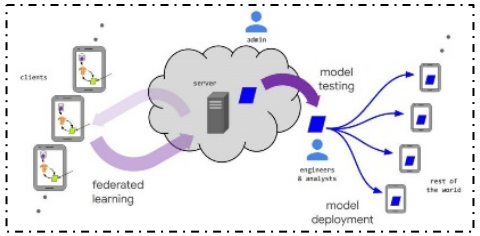
如何破局?

指令集==》编译==》框架==》API

配套软件栈



⇦ **示范性应用** ⇨



“十四五”数字经济发展规划

（一）加快建设信息网络基础设施。建设**高速泛在**、天地一体、云网融合、智能敏捷、绿色低碳、**安全可控**的智能化综合性**数字信息基础设施**。

为数字经济提供隐私计算信息基础设施！

自主知识产权的全同态加密处理器芯片，与软件方式相比达到10,000倍的性能提升。

处理器核心芯片+硬件加速卡	**全同态处理器芯片与全域可信硬件加速卡**
全域隐私保护软硬件一体机系统	**全域可信软硬件一体机系统**
全域可信数据基座应用验证	**面向全域可信的数据要素流通重大行业应用**

感谢聆听！

更多信息请见：https://luhang-CCL.github.io

Hang Lu（路航）

副研究员、硕导、青促会会员、新百星