Distilling Bit-level Sparsity Parallelism for General Purpose Deep Learning Acceleration

0. Abstract (1)

- DL 모델은 나날이 발전함
  - Complexity 증가로 인해 High-performance가 요구된다.
- 대부분의 가속기는 Training 은 배제되는 경향이 있음
  - Bitlet 방식은 Inference 뿐 만 아니라, 학습 시에도 사용 가능함.

0. Abstract (2)

- ‘**비트 인터리빙(Bit Interleaving)**’ 제시
  - Bit-level sparsity 를 사용하는 새로운 방식
- **Bitlet** – 비트 인터리빙을 활용한 새로운 범용 가속기(General-purpose)
  - 부동 소수점 FP32/16 둘 다 가능함.
  - 고정 소수점 1bit ~ 24bit
- **고성능(High-performance + Efficiency)**
  - x15 performance
  - x8 Efficiency

1. Introduction

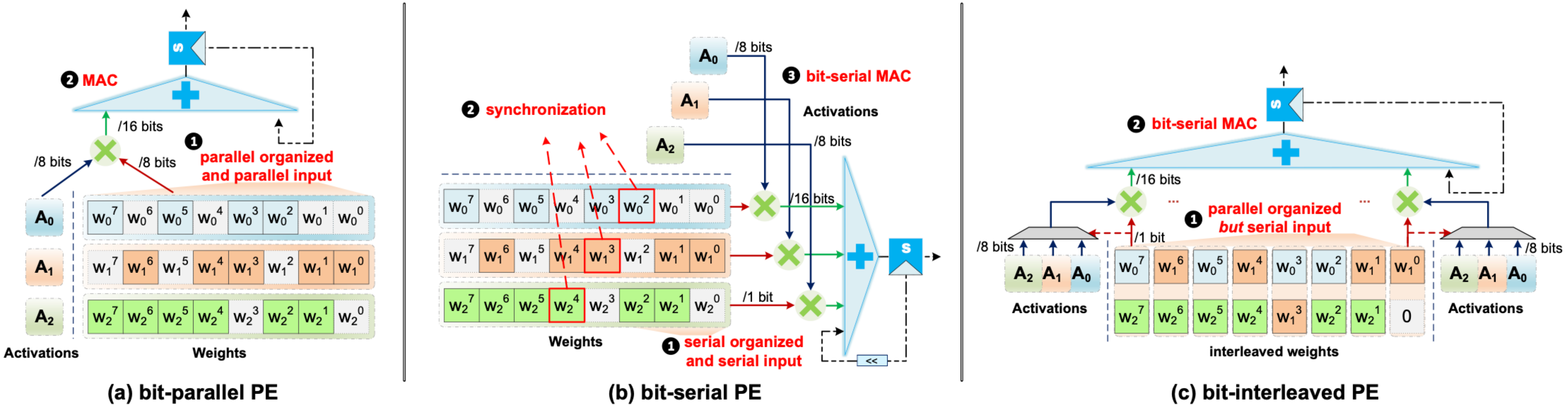


Figure 1: High-level step-by-step example comparing the bit-interleaved PE with prior bit-parallel/serial PE in the fixed-point mode. The $w_i{}^j$ marked in grey is the non-essential bit (0 bit). In (a) bit-parallel PE, Step ❶ organizes the weights for MAC in parallel; Step ❷ issues MAC. In (b) bit-serial PE, Step ❶ organizes the weights in serial; Step ❷ synchronizes the significance of the essential bits; Step ❸ issues the "bit-serial" MAC. In (c) bit-interleaved PE, Step ❶ organizes the weights in parallel, but Step ❷ issues the bit-serial MAC along each bit significance, excluding the synchronization operation. Note that bit interleaving also supports the floating-point MAC, as will be specified in Section 3.

1. Introduction

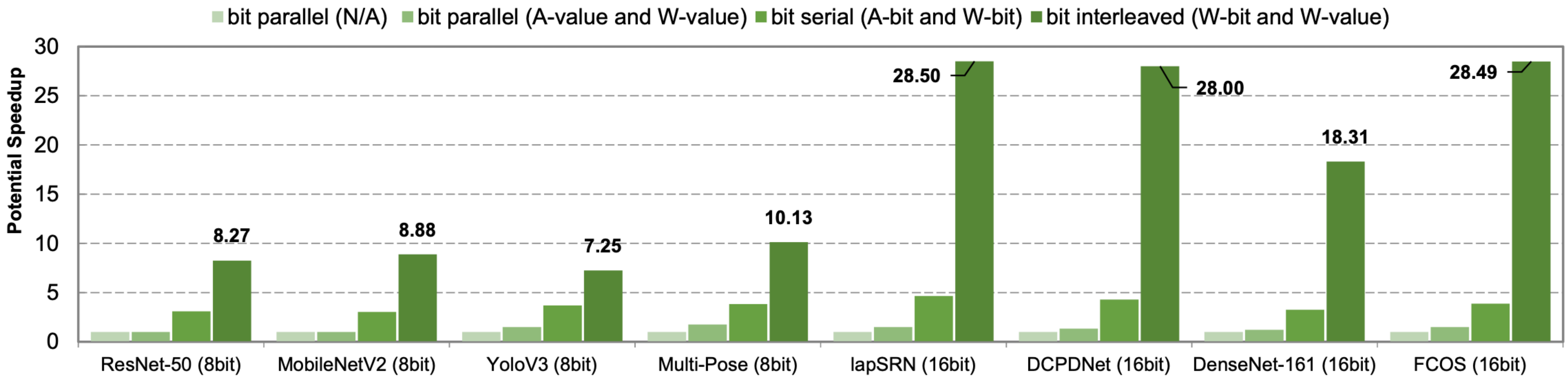


Figure 2: Potentials of bit interleaving. The baseline design is "bit parallel (N/A)". Most existing sparsity-aware accelerators only target fixed-point precision, so we only compare 16b and 8b DNNs in Table 2. In Section 5, we will evaluate the floating-point applications over GPUs.

1. Introduction

Table 1: Accelerator design philosophies.

Philos.	**Design**	**Sparsity Exploited**	**Preci. V.**	**Training Support**
bit parallel	Eyeriss[10], DaDianNao[12]	N/A	16b	No
	Cambricon -S[47], EIE [17]	A-/W-value	16b	No
	SCNN[32]	A-&W-value	16b	No
bit serial	UNPU[27], Stripes[22]	N/A	$1 \sim 16$b	No
	Bit Fusion [37]	N/A	2,4,8,16b	No
	Pragmatic [8]	A-/W-bit	$1 \sim 16$b	No
	Bit Tactical[26]	A-bit&W-value	$1 \sim 16$b	No
	Laconic[36]	A-&W-bit	$1 \sim 16$b	No
bit inter -leaving	**Bitlet (this work)**	W-bit &W-value, (or A-bit&A-value)	$fp32/16$, $1 \sim 24$b	Yes

2. Related Works

- **이전의 것들(Bit-parallel, Bit-serial)은 general-purpose가 아님.**
  - 추론(Inference) 시에만 가속기가 적용되고, FP precision에도 제약이 존재.
- **최적화 되어 있지 못한 Sparsity의 활용**
  - Bit-parallel : bit-level sparsity 를 활용할 수 없음.
  - Bit-serial : synchronization이 필요함.

2. Related Works

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%

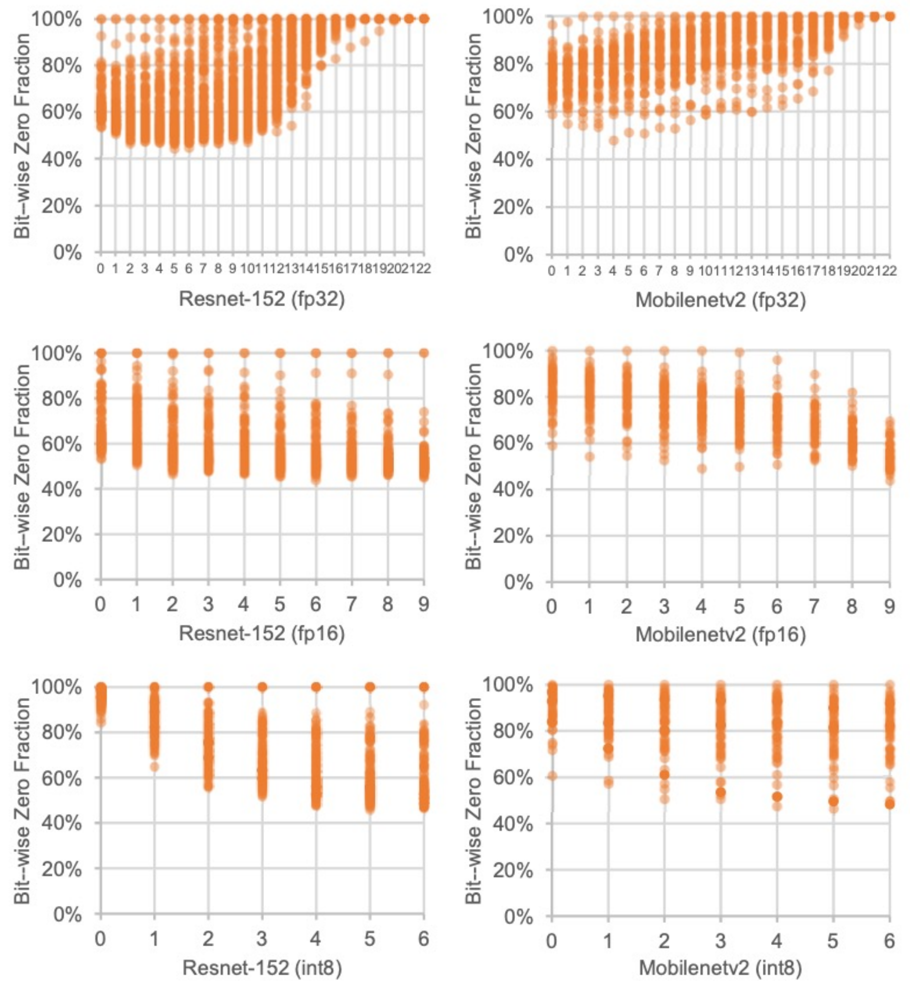


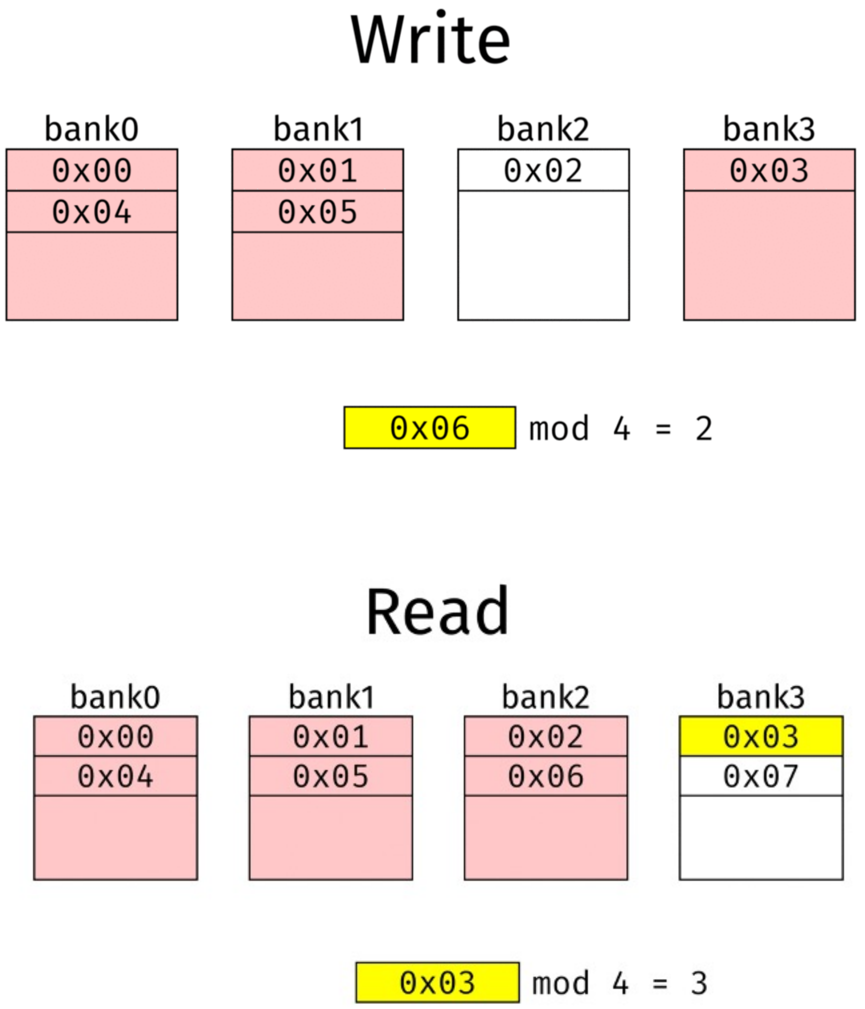
Figure 3: Sparsity parallelism. Each dot indicates the fraction of zeros on this bit lane across all the weights of this kernel. It shows ~ 50% bits are 0s for all kernels. On X-axis in the figure, the sparsity only entails the mantissa (23/10 bits for float 32/16), and 7 significant bits excluding the sign bit for int8 precision. We do not need to consider the sparsity of the exponential bits.

2. Related Works

- **각 bit significance 에서 높은 Sparsity 를 보임.**
- **균일한 Sparsity**
- **병렬 처리에서 발생하는 좋은 점들**
  - 동기화 필요 없음.
  - Bit-level 산술 연산은 독립적으로 수행 된다.

3. Methods – Bit Interleaving

- Bit-level sparsity를 활용할 수 있는 GP Accelerator 설계하기.
- 장점
  - 복잡한 Synchronization 필요 없음.
  - Sparsity를 효율적으로 이용할 수 있음.



3. Methods – Bit Interleaving

- FP32 – 1Sig + 8Exp + 23Man
- FP16 – 1Sig + 5Exp + 10Man **(S, M, E 로 구성된다.)**
- fp=(−1)^S*1.M×2^E-127

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

Detailed deduction is in the paper

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

Exponent matching

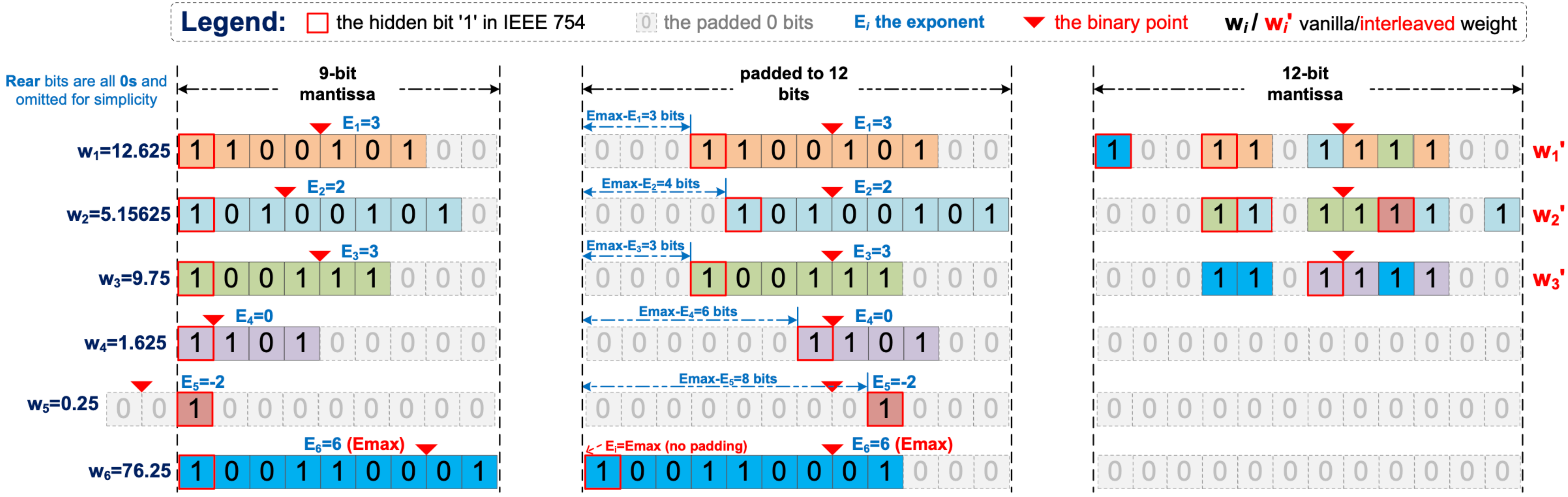
Bit-level arithmetic

Bit significance

Final sign

Maximum exponent

4. Bitlet Accelerator



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

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

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

Figure 4: Core concept of "*bit interleaving*". Vanilla fp32 weights are exemplified in Step ❶. It pro-processes the weights by interpreting out the exponent E_i and mantissa M_i. According to the maximum exponent (E_{max}), the weights are shifted and zero padded in Step ❷. Note that the exponent matching is only allowed to the right hand side in case of severe precision loss as standardized in IEEE 754. Step ❸ is responsible for bit distillation. Only 3 interleaved weights are finally involved in the accelerator.

4. Bitlet Accelerator

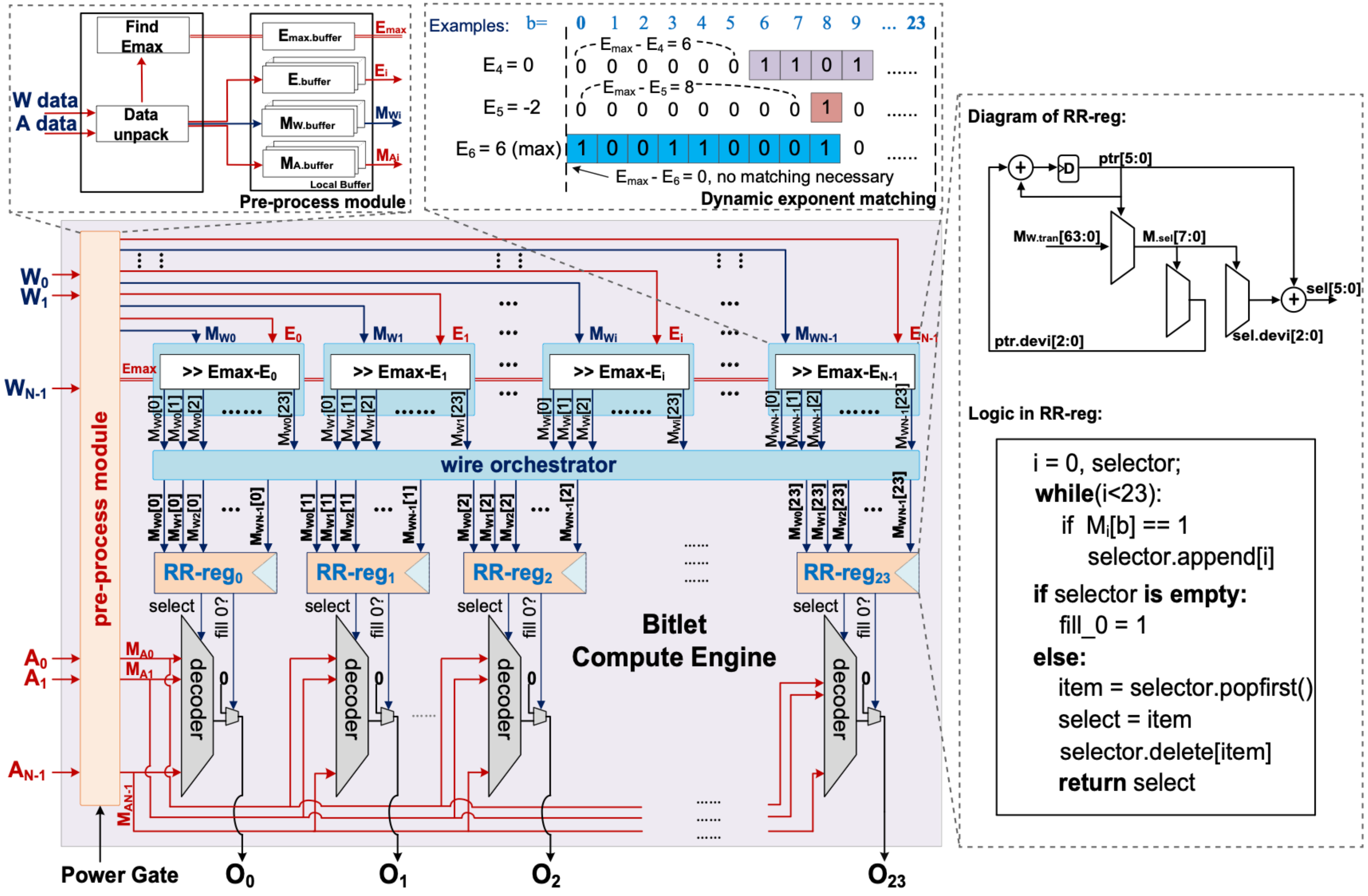


Figure 5: Microarchitecture of the core module – *Bitlet Compute Engine (BCE).*

4. Bitlet Accelerator

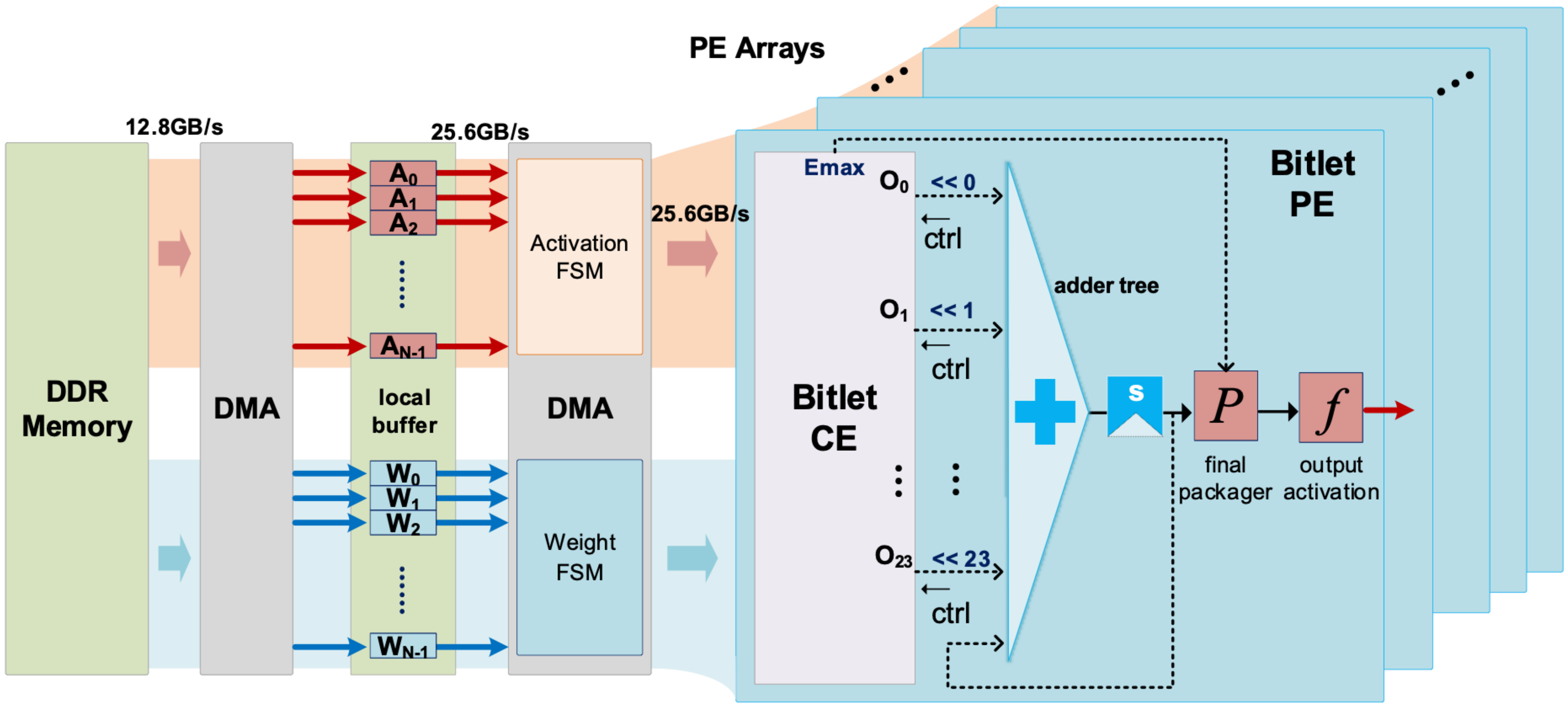


Figure 6: *Bitlet* accelerator. Each *Bitlet PE* is comprised of one BCE and a series of adders used for computing the output activations. *Bitlet* is versatile: for the floating point, Emax is dynamic, while for the fixed-point precisions, Emax is fixed to the target precision (*i.e.*, 16 or 8).

5. Evaluation

Table 2: Benchmark DNNs and their specs, which are used for motivating bit interleaving and the evaluations.

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)

Table 4: Quantitative comparison for average computation performance (cycles/MAC).

	16b				8b			
	lapSRN	DCPDNet	DenseNet-161	FCOS	ResNet-50	MobileNetV2	YoloV3	Multi-Pose
Eyeriss	339.34	357.39	343.68	346.62	344.08	343.77	345.08	341.71
SCNN	226.22	262.09	281.87	231.08	344.08	343.77	230.05	197.14
Stripes	90.49	106.25	150.86	115.54	158.39	174.07	119.94	110.23
Laconic	71.44	83.64	105.18	89.77	111.13	113.44	93.22	89.92
Bitlet	29.51	16.45	22.85	22.44	21.03	22.22	19.04	24.64

5. Evaluation - FPS

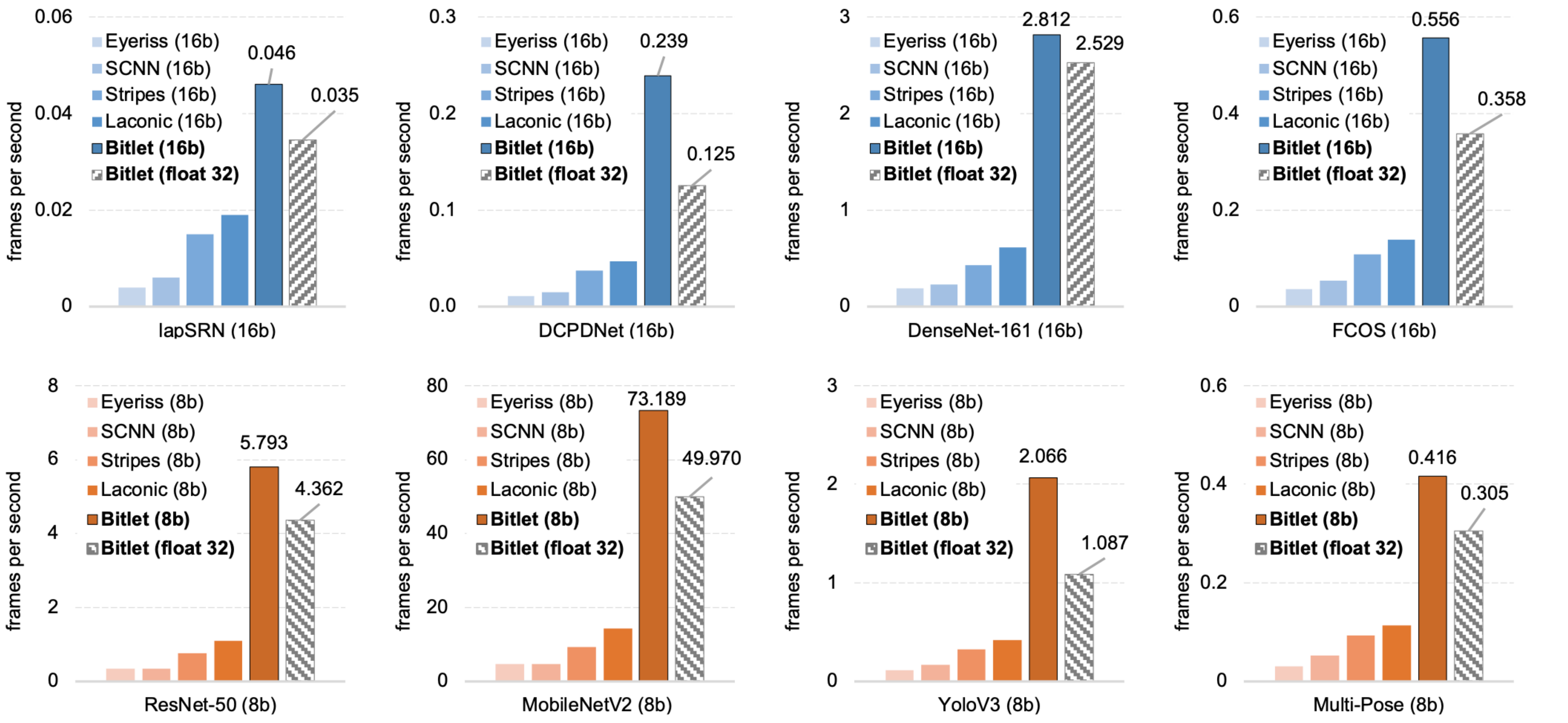


Figure 7: Speedup results. The upper row denotes the 16b DNN benchmarks and the bottom row denotes the 8b benchmarks. We also run the float-32 version on *bitlet* for reference. All the results are real values in frames per second (fps). Higher is better.

5. Evaluation – Energy Consumption

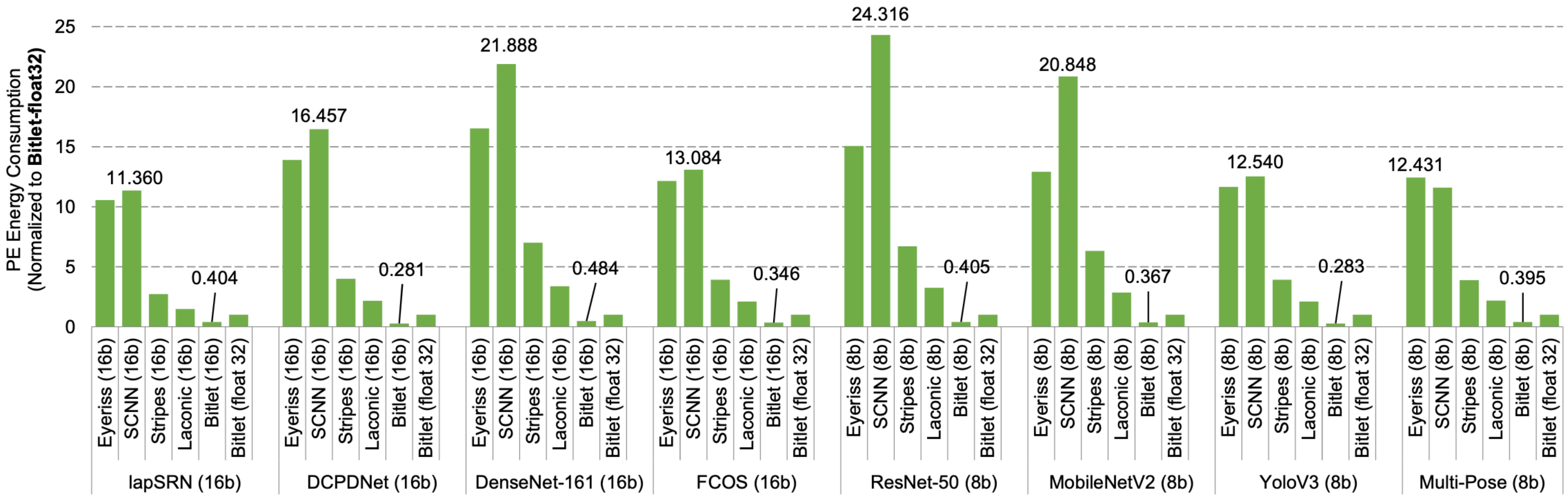


Figure 8: Energy Consumption. We also report the energy result of the float-32 inference, and all the fixed-point results are normalized to it. Lower is better.