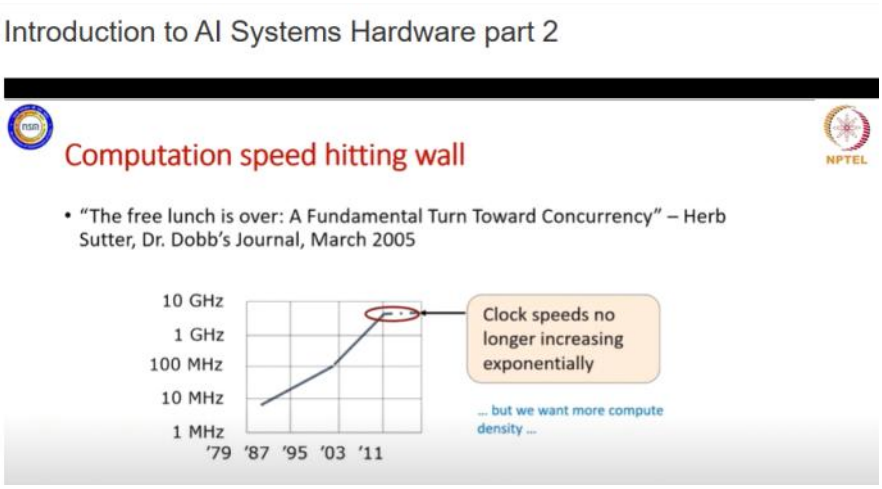
AI ACCELERATORS

21 July 2025 11:04



How we will get much more computation density ? In terms of speed as well as computation.

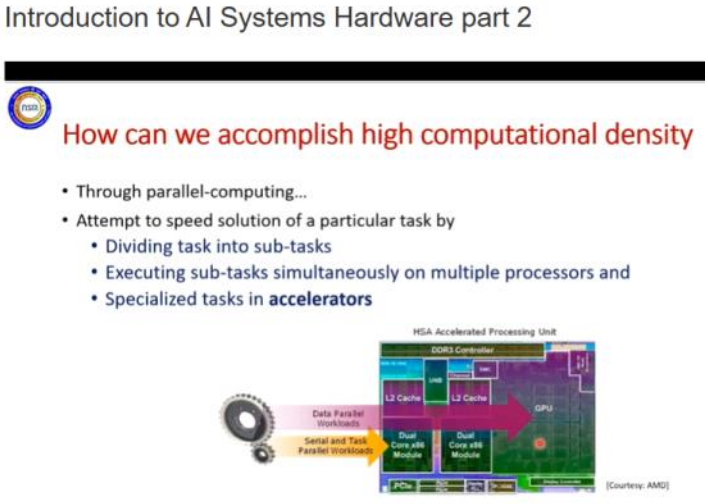
- to have flexibility of including more numbers of transistors in a chip
  - The clock speed also cannot increase
  - So compute density cannot increase after a certain limit

Modern systems + traditional systems

HETEOGENEOUS COMPUTING

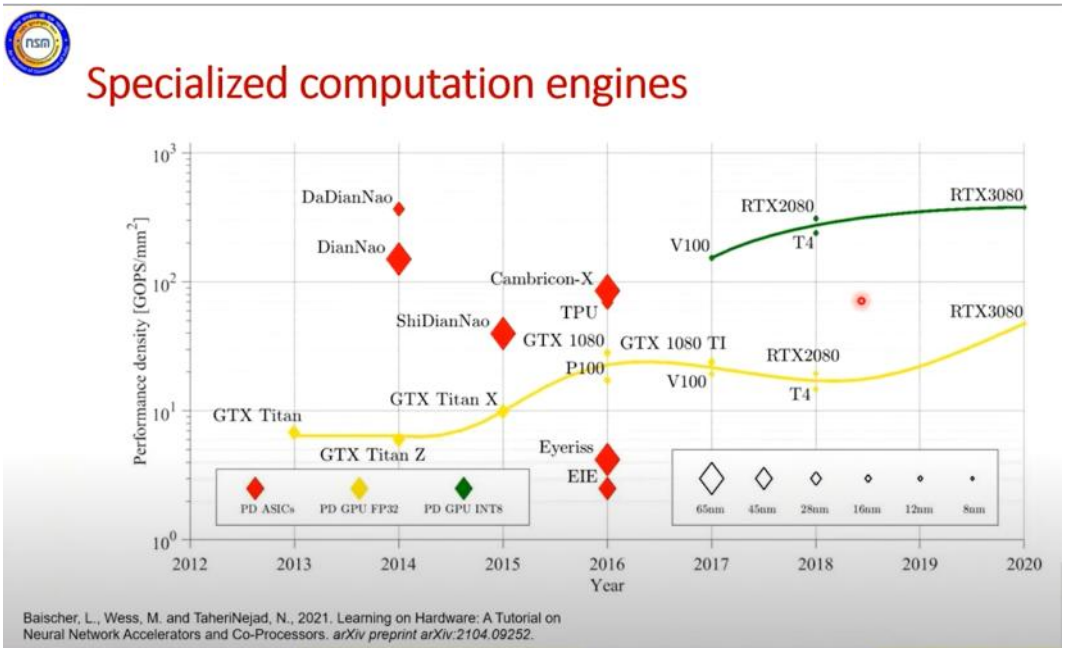
- SERIAL TASKS OR MOSTLY PARALLEL WORKLOADS which are not much data intensive they ar being executed by processor itself
- Single core/muilti core processor

  DATA PARALLEL WORKLOADS execution specialised computing engines called accelerators
- Several accelerators are used particularly for ai benchmarks
- One such specialized system or accelerator is called cerebroas wafer engine
- Here we have accelerators along with the processors



ACCELERATORS AVAILABLE NOWA DAYS

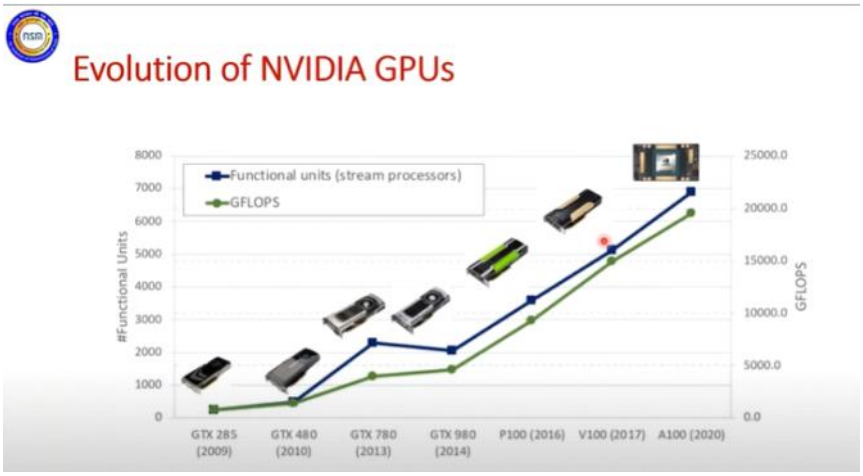
ASICS - highly specific for computations ai based (here)
GPUS - video /graphics redenring /as well as ai generalised

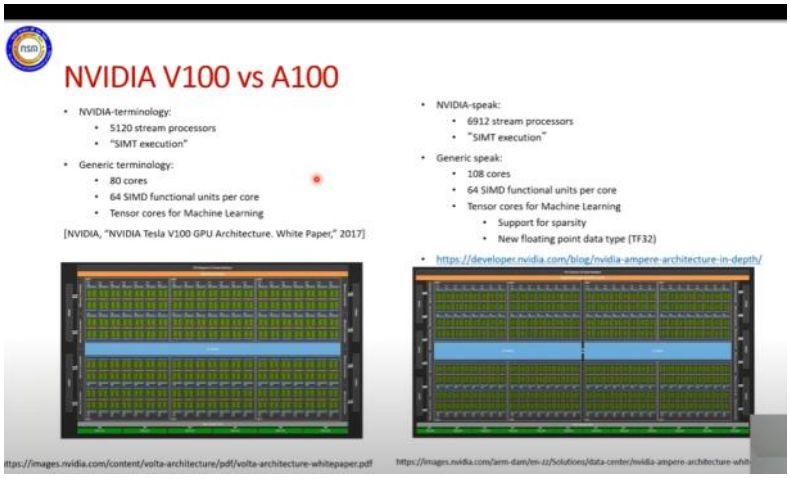


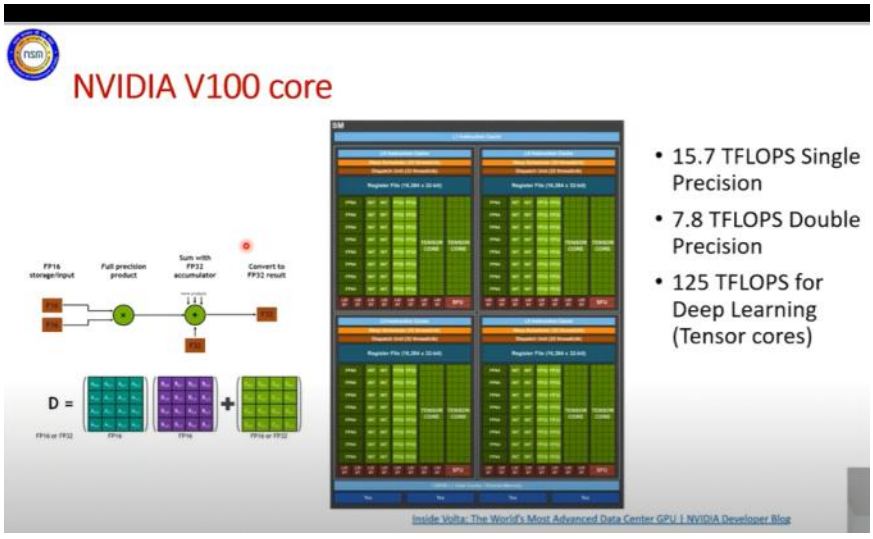
The more better the precision the higher will be the accuracy

How to get higher computing density ?

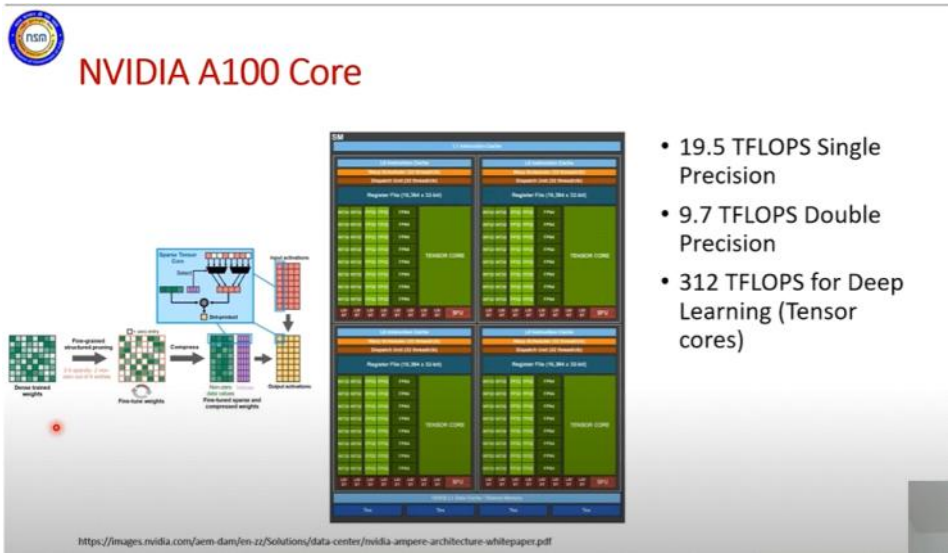
- Reducing size of feature maps
- We need much more gops per mm^sq
- i.e 7-8 nm cmos technology used to manufacture chips
- Nvidia V100 series GPU
- 28 nm da diannao (asic based version of diannano) 2014
- 2016 tpu 1st version asi based google
- Cambricon , eyeriss mit used several feature map size
- Decreasing feature maps means more units fit inside chip -> increase performance density (gops per mm^2)
- Cannot go beyond certain limit feature map size cannot increase beyond 1nm
- So theres a certain level or limit of performance one can achieve





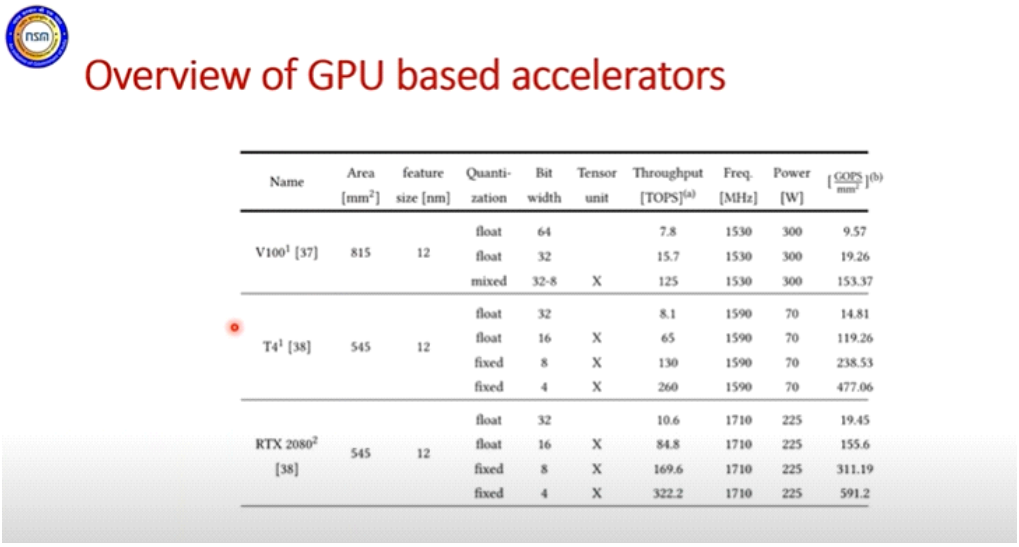


Tensor cpres for mat muiltiplications



It supports all different precisions of computitions
All these fp32 int8 etc units used for -> graphics. Video processing acceleration
All tensor cores ->accelerating ai benchmarks
Tensor core in ampere series is much more flexible then earlier series,
It has something called - sparsity (we can have muiltiple weights and params which are very close to zero

& can be interpretted as 0 but it say like let 4x4 matrix muiltiplication half of data is zero
Then we don't need to do mul for those particular half number of muiltiplications It will be manifold as 2d matmul --> increase efficiency,throughput



Overview of GPU based accelerators

Name	Area [mm²]	feature size [nm]	Quanti-zation	Bit width	Tensor unit	Throughput [TOPS][a]	Freq. [MHz]	Power [W]	[GOPS/mm²][b]
V100[1] [37]	815	12	float	64		7.8	1530	300	9.57
			float	32		15.7	1530	300	19.26
			mixed	32-8	X	125	1530	300	153.37
T4[1] [38]	545	12	float	32		8.1	1590	70	14.81
			float	16	X	65	1590	70	119.26
			fixed	8	X	130	1590	70	238.53
			fixed	4	X	260	1590	70	477.06
RTX 2080[2] [38]	545	12	float	32		10.6	1710	225	19.45
			float	16	X	84.8	1710	225	155.6
			fixed	8	X	169.6	1710	225	311.19
			fixed	4	X	322.2	1710	225	591.2

Mixed precision is a technique in deep learning where **different numerical precisions** (like 16-bit and 32-bit floating-point) are used **within the same model** during training or inference to improve performance and reduce memory usage **without significantly compromising accuracy**.

🔢 What does "precision" mean here?

Precision refers to the number of bits used to represent floating-point numbers:

Precision Type	Format	Bits	Example
FP32 (Single)	float32	32	Default in most DL
FP16 (Half)	float16	16	Lower memory and faster
BF16	bfloat16	16	Similar to FP32 range, better for training

🏵 How Mixed Precision Works

Instead of using 32-bit floats (FP32) everywhere:

- Most of the **calculations** are done in **FP16** or **BF16**.
- Some key parts (like **loss scaling**, **gradient accumulation**, **model updates**) are still done in **FP32** to preserve **numerical stability**.

Think of it like using cheaper materials for most of a building, but reinforcing the critical load-bearing structures with stronger materials.

🧠 Why Use Mixed Precision?

Benefit	Description
☑ Speed	FP16 operations are faster, especially on GPUs with Tensor Cores (e.g. NVIDIA V100, A100).
☑ Memory	Uses ~50% less memory—train bigger models or use larger batch sizes.
☑ Scalability	Enables training models on hardware with limited memory (e.g. consumer GPUs).
⚠ Stability	Requires techniques like **loss scaling** to avoid underflow/overflow.

🚀 When to Use It?

Mixed precision is especially useful for:

- Training large models (e.g. Transformers, CNNs)
- Running on GPUs with native support (e.g. NVIDIA Volta, Turing, Ampere architectures)
- Inference optimization on edge or cloud

🛠️ How to Use It (Example: PyTorch)

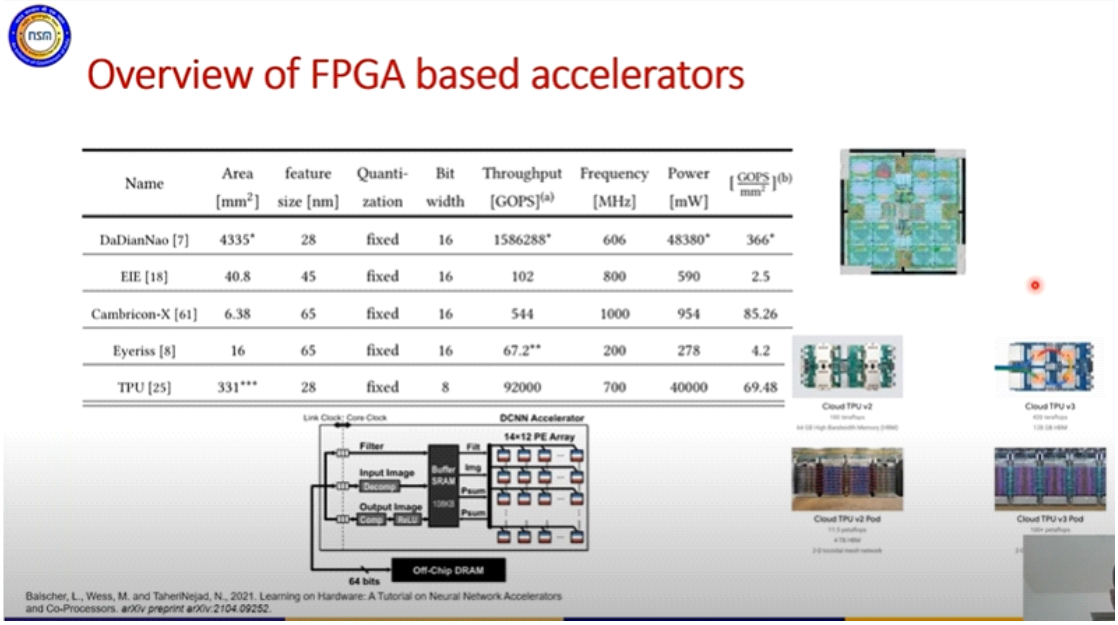
```
from torch.cuda.amp import GradScaler, autocast
scaler = GradScaler()
for data, target in dataloader:
   optimizer.zero_grad()
   with autocast():
      output = model(data)
      loss = loss_fn(output, target)
   scaler.scale(loss).backward()
   scaler.step(optimizer)
   scaler.update()
```

🔍 Summary

Feature	Mixed Precision
Uses FP16/FP32	☑
Faster training	☑
Lower memory	☑
Needs tuning	⚠ Yes (e.g. loss scaling)

Would you like a comparison with full FP32 or details for a speci

---



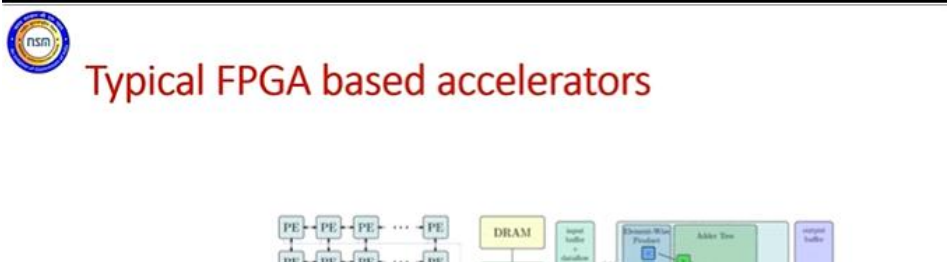
fic framework (like TensorFlow or JAX)?

FPGA based accelerators

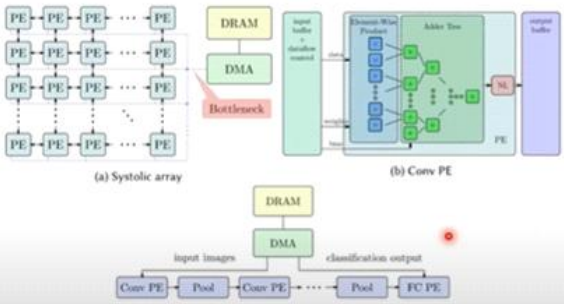
-fully reprogrammable field of programmable gate arrays mostly bit level configurable devices
In fpgas we can employ different asic level accelerators bocoz its fully congiurable
Examples - tpu (google) v1 earlier published for inferencing but now a days v2,v3 other versions also used
- all these computation engines are essentially are synthetic array based engines

- Power consumption in here is in terms of milliwats
- Whereas in gpu based accelerators it is in terms of watts coz they are more generalized whereas asic based are more specialized

---



Typical FPGA based accelerators



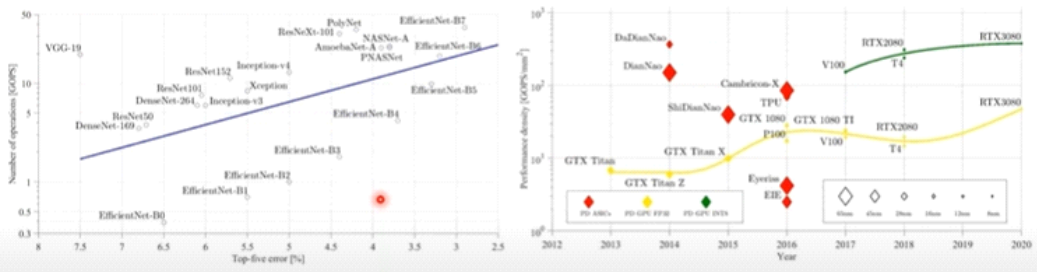
Baischer, L., Wess, M. and TaheriNejad, N., 2021. Learning on Hardware: A Tutorial on Neural Network Accelerators and Co-Processors. *arXiv preprint arXiv:2104.09252*.

Market share of different technologies

- As of the third quarter of 2017
  - GPU
    - Nvidia represented 72.8%
    - with the rest by AMD
  - FPGA
    - Xilinx 53%
    - Altera 36%
    - Microsemi 7%
    - Lattice Semiconductor (3%)

Hwang, Tim. "Computational power and the social impact of artificial intelligence." *Available at SSRN 3147971* (2018).

The Gap...



To get particular level of accuracy that is linear level of accuracy we need to exponentially increase computational density