# Unit 1

**(lec 1 - 10)**

Parallel computing has arisen because of the hard limit of speeding up chips.

There are 2 general ways of 2 threads or processes communicating with each other. Either a shared memory or message passing (there are other types). Nowadays it is a mix of both.

A few things to follow while programming with parallel concepts:

- Understand the model: The restrictions on the features of the application etc.
- Design accordingly. This is in terms of both theoretical model as well as the bare metal model.
- Treat each thread as an enemy of another. This is cuz we should know how to chose one thread over another.
- Set standards for communication and memory sharing.
- Employ high level constructs like letting certain libraries/ compiler etc handle a few parts of your code. This is to ease debugging.

---

## Shared Memory Architecture

Processes access the same global memory. This is divided into 2. Uniform Memory Architecture (UMA) and Non Uniform Memory Architecture (NUMA). NUMA is a way to give priority to cores that are closer to memory chips. Essentially each core has its own "local" memory it is responsible for.

| Advantages | Disadvantages |
|---|---|
| Easier to program | Harder to scale (adding CPUs increases traffic) |
| Faster memory access | Programmer initiated memory access. (hard to sync i guess) |

---

## Distributed Memory Architecture

Processors have only localized memory. They can access other core's memories through a robust communication network. Sync is programmer defined.

| Advantages | Disadvantages |
|---|---|
| Memory is easier to scale | Programs are more complex |
| Local access is faster (no cache) | Data communication is not easy to manage. |
| Cheap | |

## Parallel Computing models

There are a few models. Shared memory, message passing, threads,

### Shared Memory:

We will use locks and semaphores. Data can be cached locally.

### Message Passing:

Nothing new. (see distributed mem)

### Threads:

Independent processes with both local and shared data. Usually a mix of memory and shared.

### Data parallel:

Only one program is written but it is run parallely on different sets of data.

### Task Parallel:

This is basically threads but is the knuckles to data parallel's sonic.

### Pipeline:

This is like a chain (think chain of responsibility). Each processor passes data after some operation.

$$speedup_n = \frac{one\_processor\_time}{n\_processor\_time(Time_n)}$$
$$efficiency = \frac{S_n}{n}$$
$$cost = n * Time_n$$

---

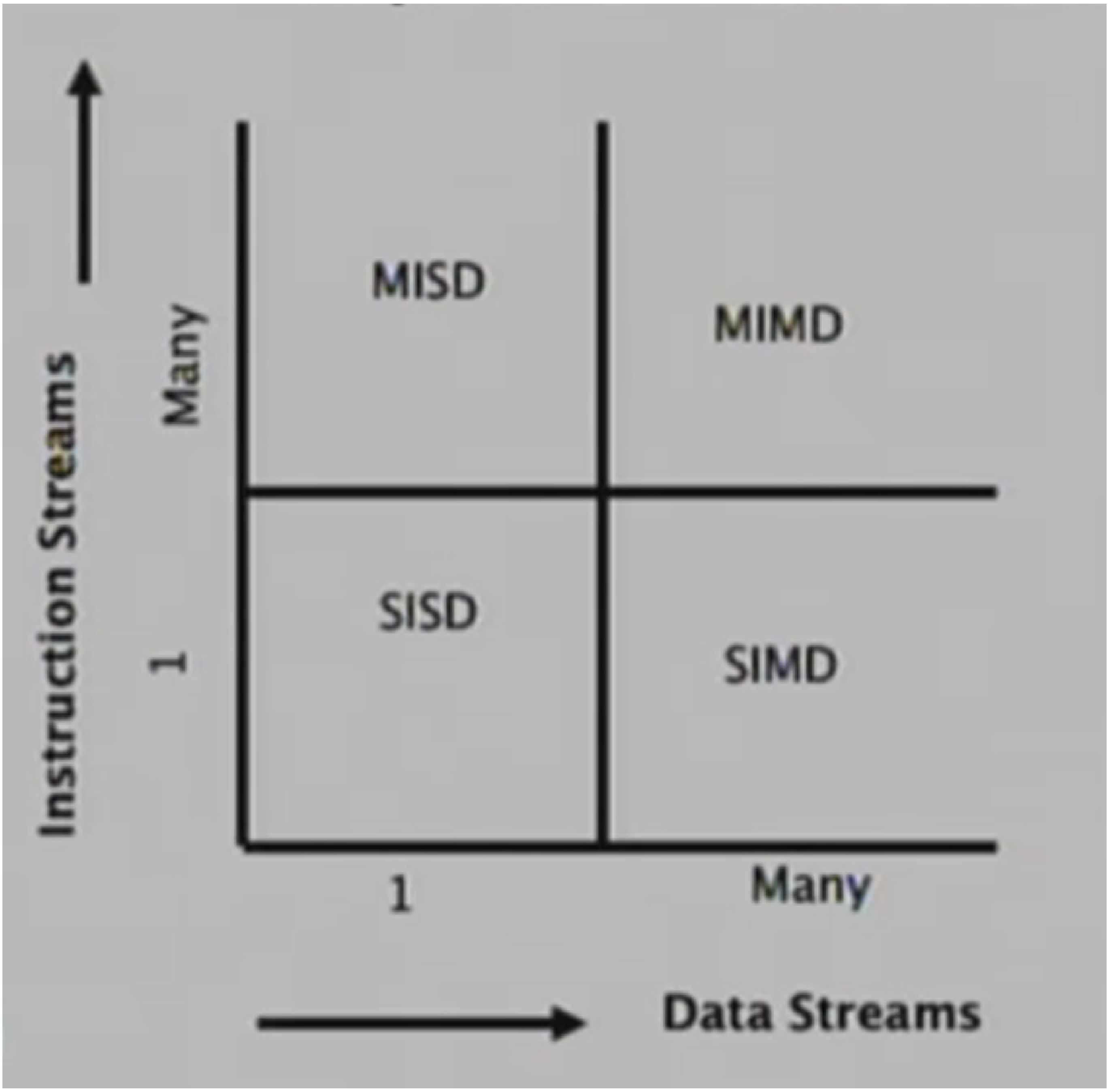
## Amdahl's Law

$$f = fraction\ of\ the\ problem\ that\ is\ sequential$$
$$1 - f = amount\ of\ code\ that\ is\ parallelizable$$
$$p = number\ of\ processors$$
$$Best\ time : T_p = T_1(f + \frac{1-f}{p})$$
$$Speedup : S_p = \frac{1}{f + \frac{1-f}{p}}$$

This is a way to see the maximum speedup paralellization can bring.

---

### Flynn's Taxonomy



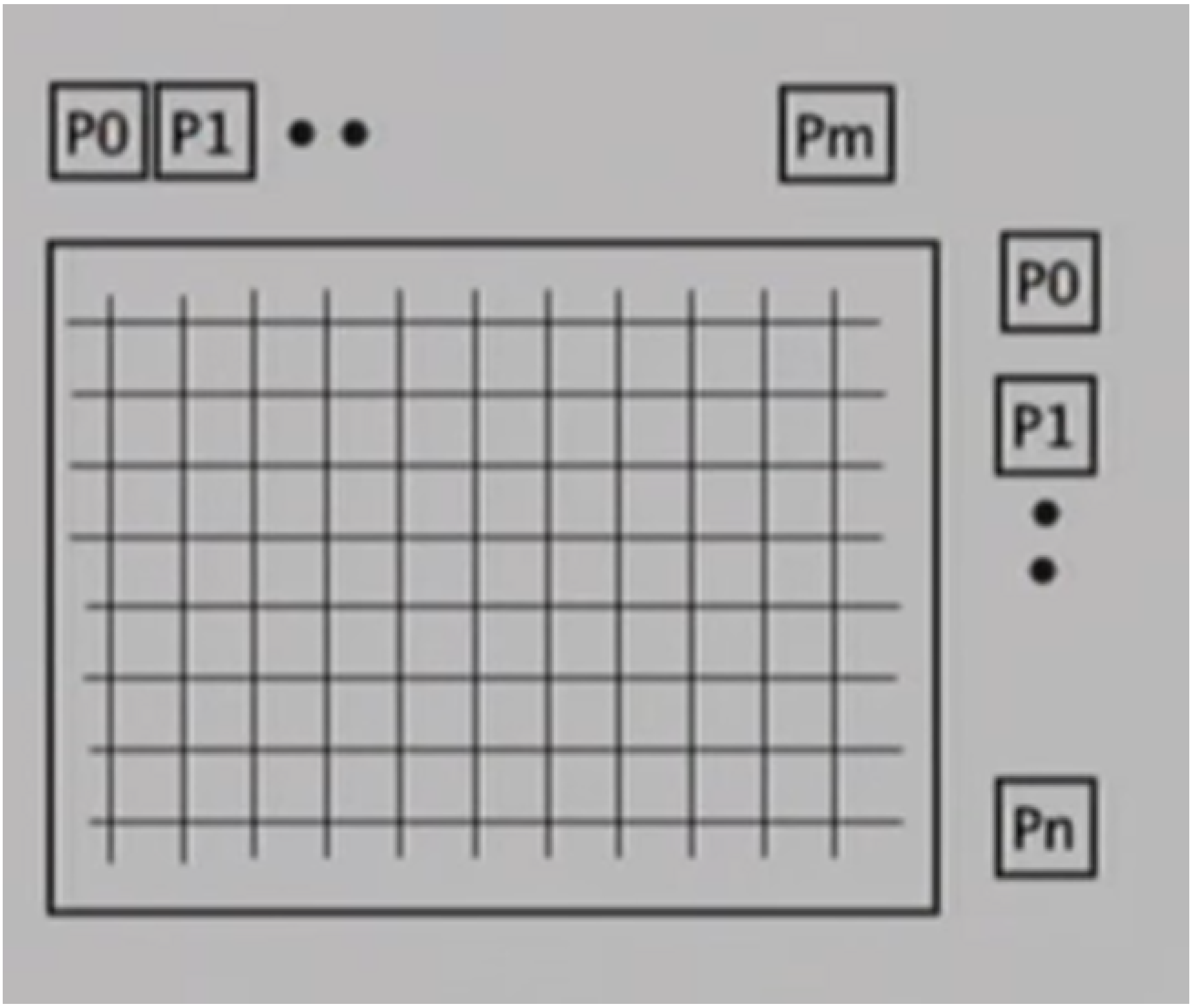
This is a basic way to categorize parallel architectures. MISD (Multiple Instruction Single Data) is kinda "hidden" and not very common. SIMD is pipeline (i think). SIMD is cheaper and more power efficient and smaller. SIMD should preferably be condition free.

## Interconnect

2 basic network types: direct networks and indirect. Direct is when we have direct processor to processor. Indirect will have a switch or something in between.

The connections can be made using buses. This can be in any form. Direct circuit lines or even ethernet.

Another way is crossbars. This mess below.

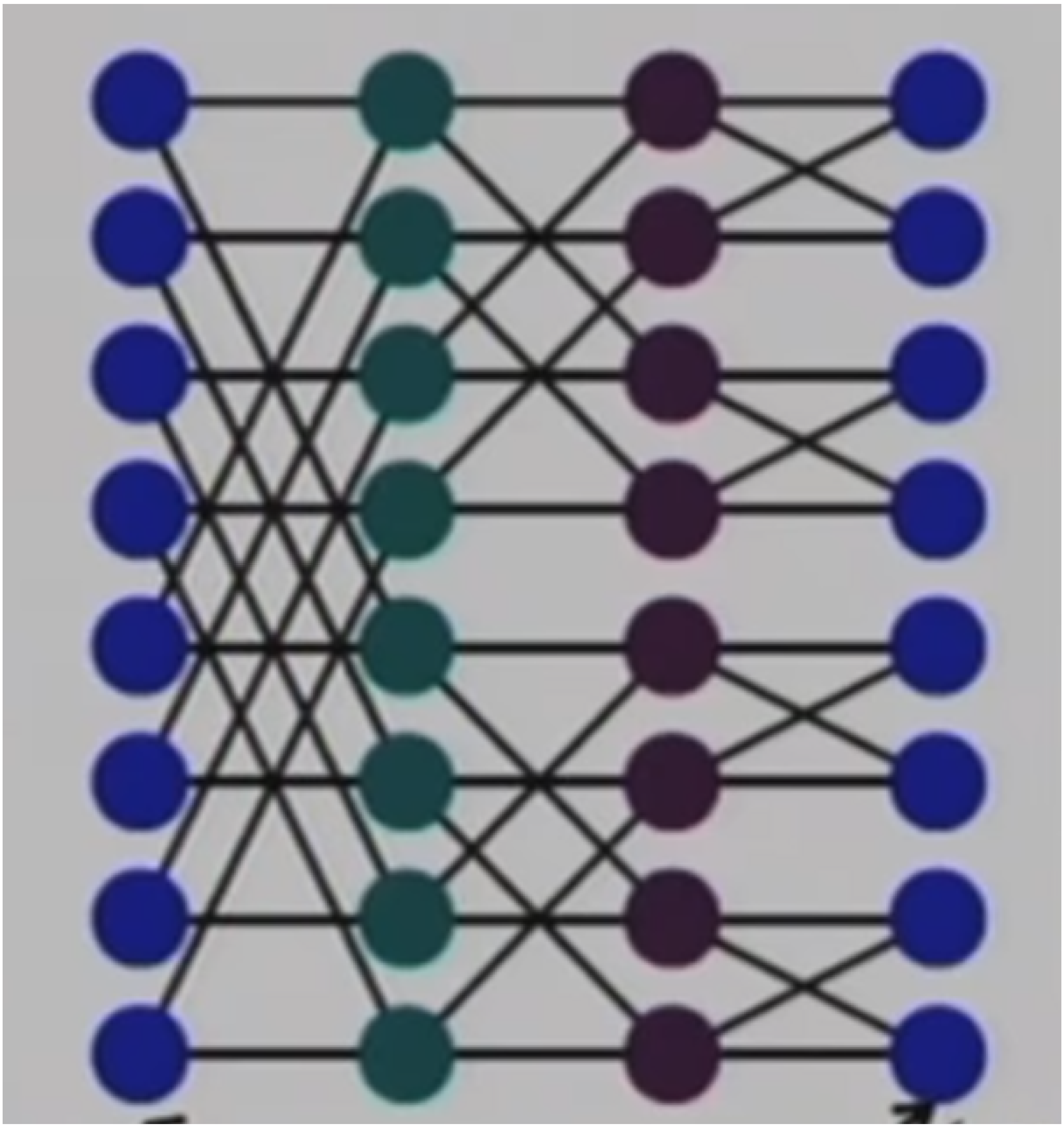


Usually it is a mix of crossbar and bus.

---

## History

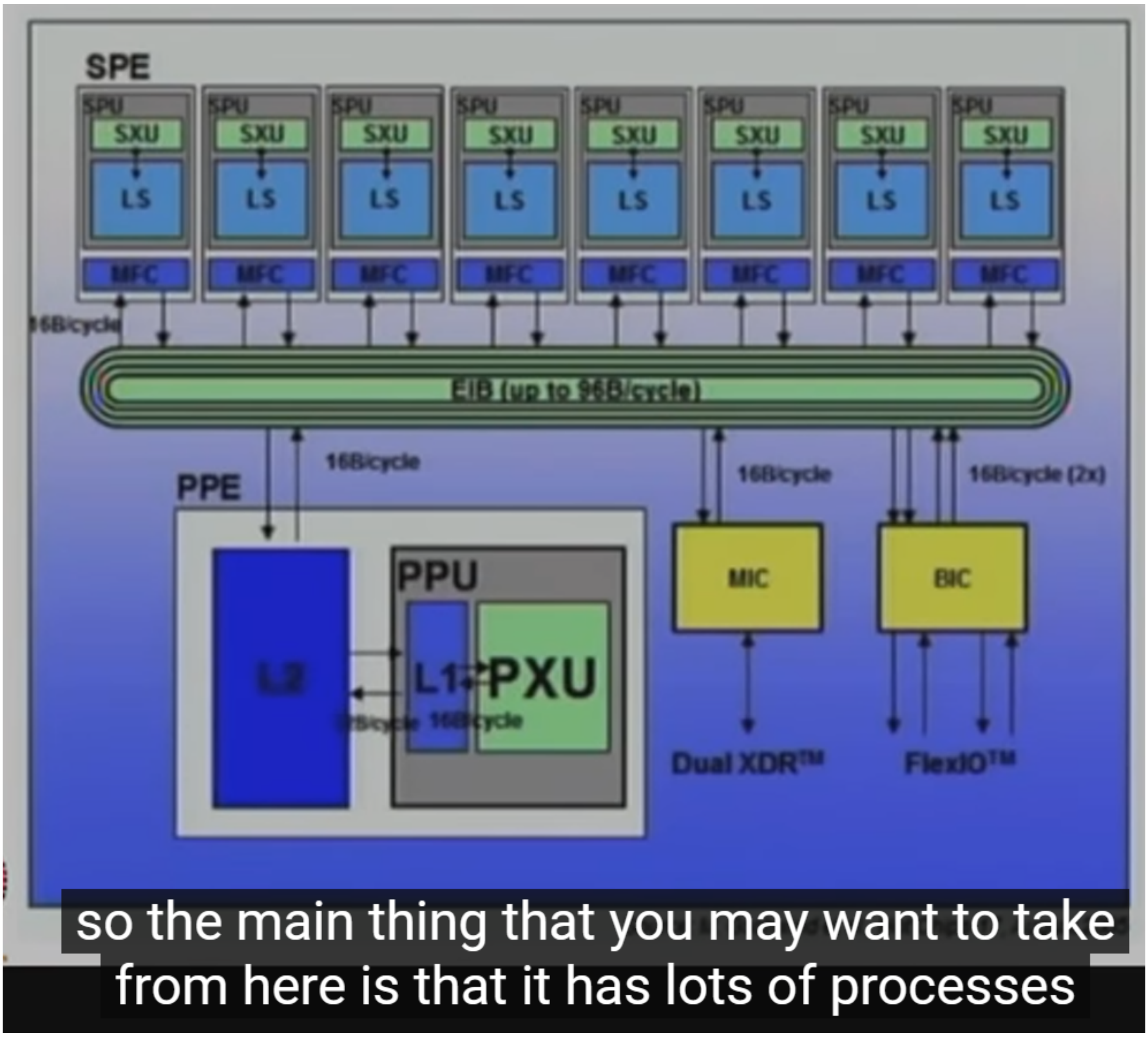
One of the earliest supercomputers used a crossbar/ mesh. Torus connections is the donut shape(circle). Hypercube is 3D mesh. Some other network style is a tree. FAT tree is when higher nodes have higher number of wires or higher bandwidth. So root node has fat thangs. Butterfly is this nonsense.



Essentially its a bit swap thing. First set is first bit swap and so on. So make groups and swap.

- Cray is an old processor manufacturer. It was 80MHz and $9 million. 0_0. Nowadays 3GHz is available for $300
- CM-2, 5 … this was a harward group. They push SIMD.
- NCube was a hypercube processor.
- Maspar used a crossbar.
- Roadrunner was the 2008's best supercomputer? Its in petaflops now. (not petaHz). This was one of the last non cluster type. This is using a cell

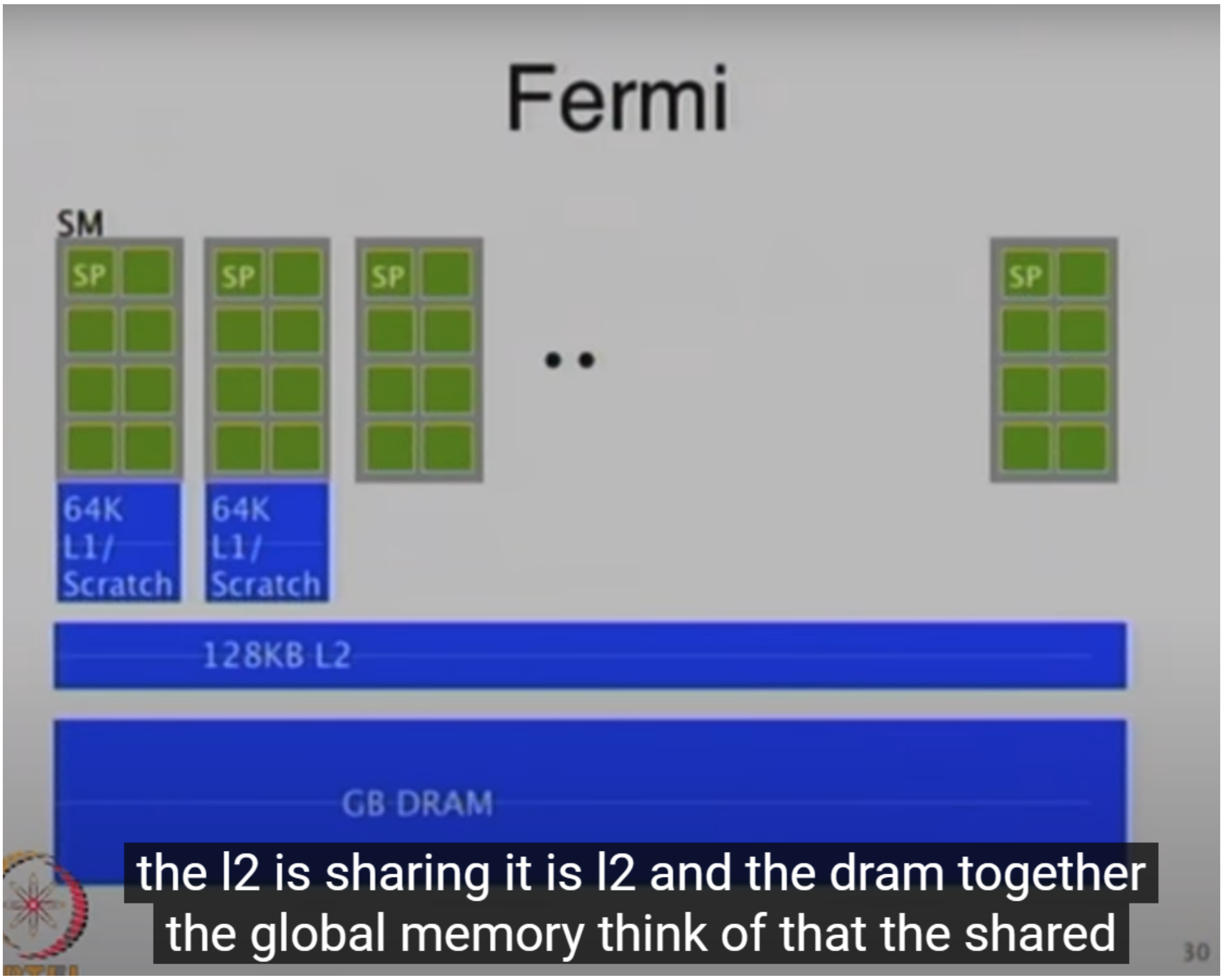
processor (from ibm). This is the piece of shit that the ps3 uses.

- 

  so the main thing that you may want to take from here is that it has lots of processes

  WTH is this? A crap architecture that didn't catch on. IBM can suck it.

- Nvidia made a GPU called Tesla in 2000 something 2008 i guess. These are REALLY OLD NOW (2023). Fermi came out in 2010.



Fermi

the l2 is sharing it is l2 and the dram together the global memory think of that the shared

Oh hey good architecture appeared!