

半導体プログラム特別講義I

ポストムーア時代の HPC・AI向けプロセッサ

牧野淳一郎

Preferred Networks/ 神戸大学

話の構成

- これまでの計算機の進歩と現状の困難
- 「ポストムーア」
- 計算機の進歩の方向
- **AI** 向けプロセッサの方向
- **MN-Core** について
- シミュレーション用計算機は？
- アプリケーションは？
- まとめ

牧野はどういう人か？

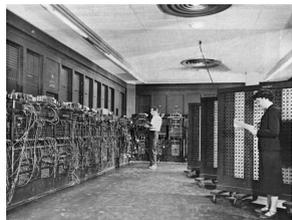
- 元々、銀河系や星団の、コンピュータシミュレーションを使った研究をしていた(今もしている)
- 「シミュレーション専用計算機」の開発を **1990** 年くらいから始める
- **2004** 年から、汎用並列計算機開発にもかかわる
- **2011** 年から、理研の「ポスト京」プロジェクトにも
- **2016** 年から、**AI** ベンチャーの **Preferred Networks** と共同で深層学習向けプロセッサを開発
- **2020** 年 **6** 月、完成したプロセッサ **MN-Core** を使った **MN-3** が「**Green500**」ランキングで世界一位に
- **2023** 年 **11** 月から **Preferred Networks** 兼任(クロスアポイントメント)

これまでの計算機の進歩

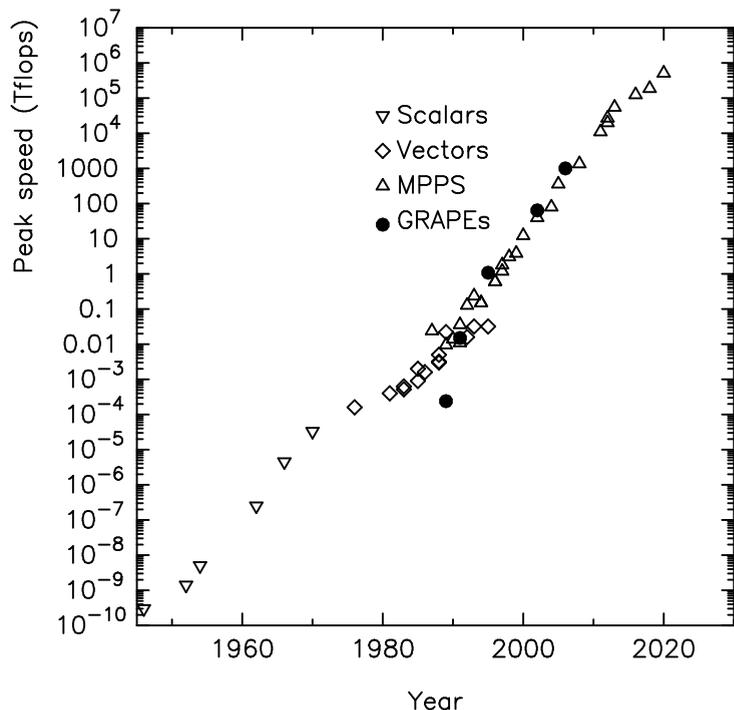
1940年代から2010年代までの70年間、ほぼ10年で100倍。何故そのような指数関数的進歩を長期に続けたのか？

基本的な理由：

- 使うスイッチ素子が高速になった
- 使うスイッチ素子が小型、低消費電力になって、沢山使えるようになった
- 使うスイッチ素子が安くなって、沢山使えるようになった(スパコンの物理的大きさは70年代が最小。そこまで段々小さくなって、そこからまた大きくなった)



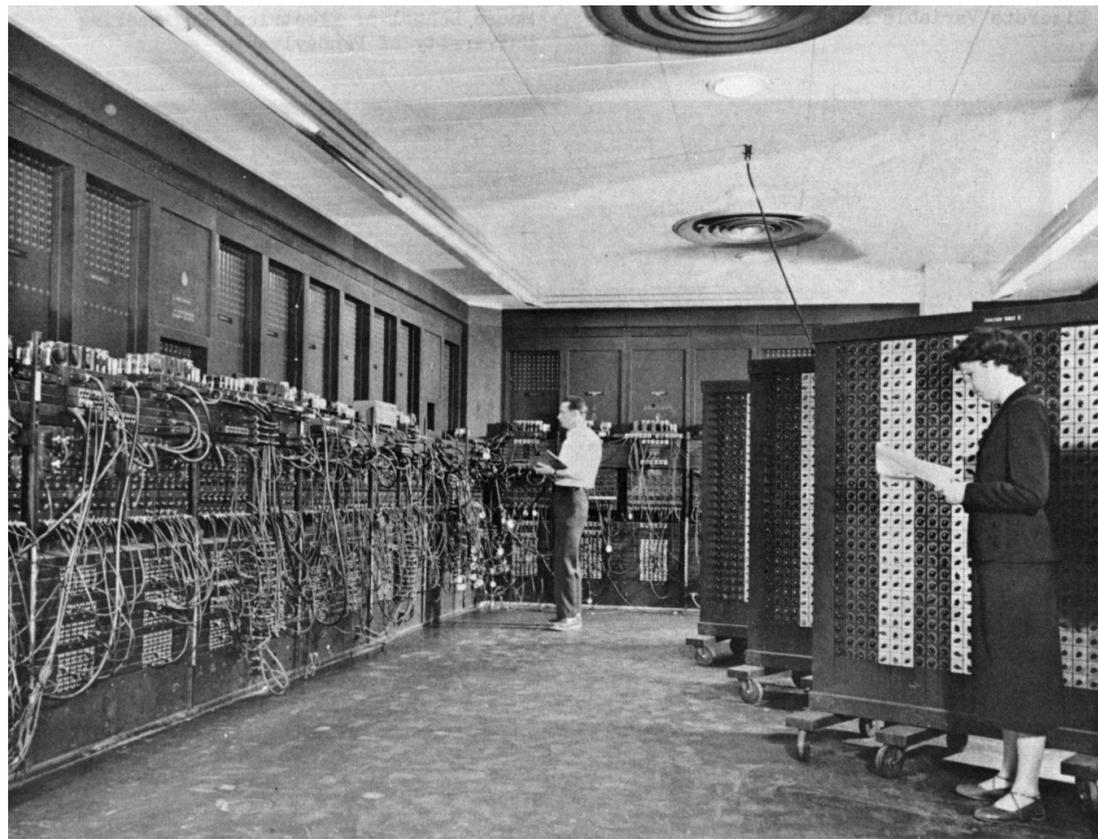
スパコンの進歩



- スパコン: 単純には、「その時代で最高速クラスの計算機」
- **80年間で16桁弱速くなった=ほぼ10年で100倍**
- 何故これほど進歩したか? は「スパコンとは何か」そのもの

(但し、現状では **Top500** の数字があんまり意味がなくなっているかも、、、 **GPT** とかの **LLM** 学習に使われているシステムのほうが大きい)

1940年代の「スパコン」

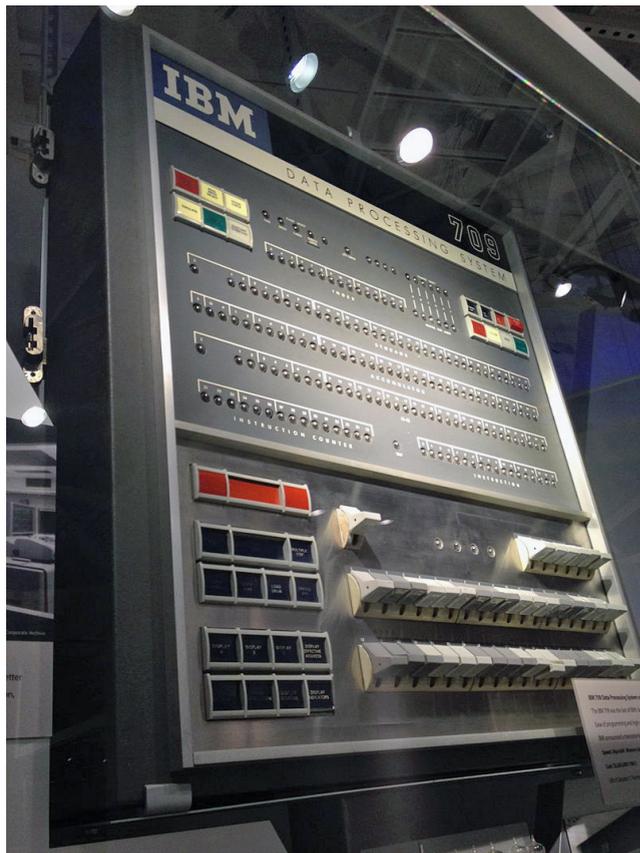


ENIAC (Photo:Wikipedia)

ENIAC

- 第二次世界大戦中に、アメリカ陸軍が弾道計算のために開発。完成は1946年
- 18,000本の真空管、消費電力150KW、10進法10桁の数が基本
- 「プログラム」はスイッチ、ケーブルの変更で行う
- 加減算が1秒5000回、乗算が400回くらい(らしい)

1950年代の「スパコン」



**IBM 709 — IBM 最後の真空管
計算機**

1960年代の「スパコン」



CC BY <https://collection.sciencemuseumgroup.org.uk/objects/co62625/control-data-cdc-6600-super-computer-super-compute>

CDC 6600 1Mflops

CDC6600

- 1964 年から出荷。シリコントランジスタを利用
- 10MHz クロック、1Mflops 程度の性能
- 「スカラー計算機」1つの命令は演算やデータ移動1つを記述
- 近代的計算機アーキテクチャの原型
 - ロード・ストアアーキテクチャ
 - 複数の演算器の「アウト・オブ・オーダー」実行
- 演算器はパイプライン化されていない。されるのは CDC7600 から。
- シーモア・クレイが設計

1970年代の「スパコン」



CRI Cray-1 160Mflops

Cray-1

- 1976 年から出荷。IC (ECL 素子による小規模 IC) を使用
- 80MHz クロック、160Mflops
- 最初に成功したベクトル計算機
 - ベクトルレジスタ
 - 半導体メモリ
- シーモア・クレイが設計

ベクトル計算機って？

- 「スカラー計算機」と「ベクトル計算機」と分ける
- 「ベクトル処理」をするかどうか
- スカラー計算機(あるいは普通の計算機の基本命令): 命令1つで演算1つ。例えば掛け算1つとか
- ベクトル計算機: 1命令で複数の演算を処理。ベクトルの、要素同士の四則演算が基本。それだけでは足りないので色々追加機能がある
- 必ずしも並列処理するわけではない。Cray-1では「パイプライン処理」。長さ n のベクトル命令の実行に 定数 + n クロックかかる

ベクトル計算機の利点

- 演算器の有効利用ができる。クロックサイクル毎に結果を得ることが可能
- 普通のスカラー計算機だと、演算以外の色々な処理が必要なので、複雑な「スーパースカラー」実行をする仕掛けが必要。ベクトル計算機は単純なハードウェアで演算器を有効利用できる

欠点:

- メモリとベクトルレジスタの間のデータ移動が高速であることが前提: 低レイテンシ・高バンド幅なメモリがないと性能がでない

1980年代の「スパコン」



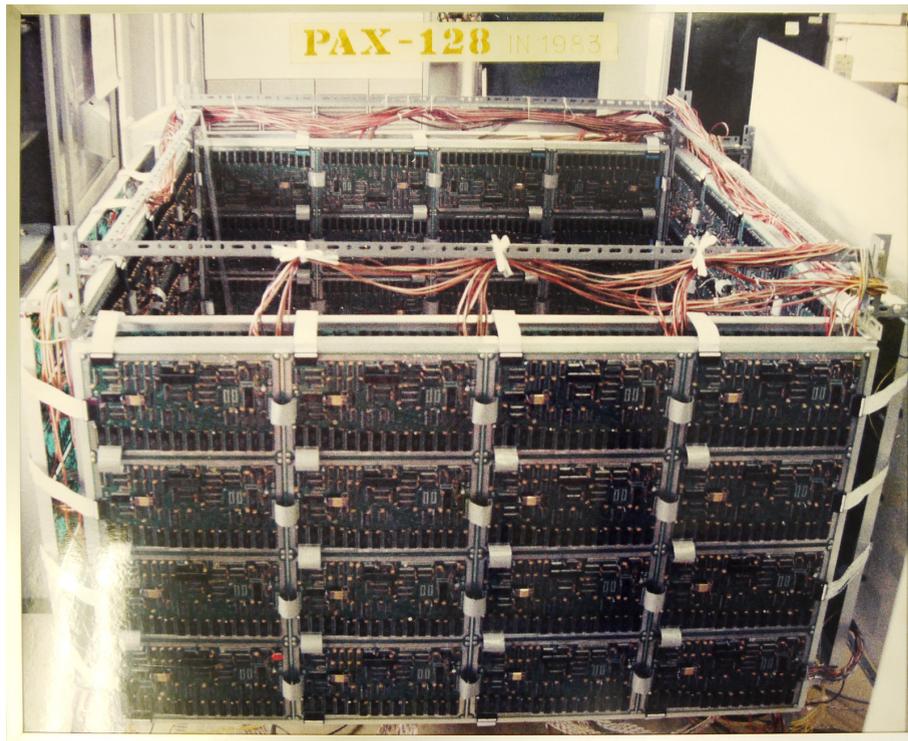
(出典: 情報処理学会コンピュータ博物館)

NEC SX-2 1.3Gflops

NEC SX-2

- 1985 年から出荷。CML 論理素子による LSI を使用。1000 ゲート程度 (Cray-1 の IC の 100 倍程度)
- 6ns クロック、掛け算、加減算パイプラインをそれぞれ 4 本
- 富士通、日立が同様なマシンを出荷、NEC は最後
- Cray は共有メモリマルチプロセッサに (Cray XMP, YMP)

1980年代にはこんなのも



出典:https://www.ccs.tsukuba.ac.jp/research_project/ha-pacs/rekishi/

PAX-128 4Mflops。筑波大学、星野ら

1990年代の「スパコン」



出典: <https://en.wikipedia.org/wiki/ASCI.Red> public domain

ASCI-Red, Intel Pentium Pro, 1.8TF

ASCI-Red

- 1996年に完成。200MHz Pentium Pro プロセッサ 9216台 = スパコン専用プロセッサではない、「普通の」マイクロプロセッサ
- 3次元メッシュネットワーク(ほぼ2次元、、、)
- 核爆弾のシミュレーション用

2000年代の「スパコン」



出典 <https://commons.wikimedia.org/wiki/File:20080831-R0012506.JPG> public domain

IBM BG/L 360TF

IBM BG/L

- 2004年完成、カスタムプロセッサを最大 131072 個使用。3次元トーラスネットワーク
- プロセッサ自体は PowerPC 440 から浮動小数点演算ユニットを強化したもの。
- ピーク性能 360TF (もっと大きい構成もあった模様)
- 後継の BG/P, アーキテクチャを一新した BG/Q の開発後、プロジェクト解散

2010年代の「スパコン」



神戸市・ポートアイランドの「京」コンピュータ、10PF (牧野撮影)

「京」

- 2012年完成、カスタムプロセッサを8万個使用。6次元トーラスネットワーク
- ピーク性能11PF
- 商用版富士通FX10, 「FX100」が開発された。2020年「富岳」への入れ換えが完了
- 「富岳」は「京」の約50倍のスピード
- (牧野は開発プロジェクト副プロジェクトリーダーの1人)

現在(2020年以降)の普通のスパコン

- **NVIDIA** の **GPU** を使用
- 沢山並べる
- 高速ネットワークを使う、あるいは **Cray** 社のシステムを買う
- 独自プロセッサとかはあまり使われない
- あんまり芸がないので写真は省略

過去のスパコンの進化

何の話をしたかったかというと：何故計算機はどんどん速くなったのか？

基本的な理由：

- 使うスイッチ素子が高速になった
- 使うスイッチ素子が小型、低消費電力になって、沢山使えるようになった
- 使うスイッチ素子が安くなって、沢山使えるようになった(スパコンの物理的大きさは70年代が最小。そこまで段々小さくなって、そこからまた大きくなった)

それらとアーキテクチャの関係は？ 今後は？ という話を以下で。

素子の高速化

- といっても、真空管でもそれなりに速かった
- スイッチング速度が重要でないわけではないが、配線を信号が伝搬する速度のほうが昔から重要
- 昔は信号はほぼ光の速さでつたわった
- 最近の LSI 上の配線は非常に細く (抵抗が大きく)、キャパシタンスを充電しないといけないことによる RC(抵抗とキャパシタンス) 遅延のため、信号が伝わる速度は光速度よりはるかに低い
- 太い配線に大電流を流せば速いが、大量の電力消費になる

素子の小型化

- 真空管 → トランジスタ → IC という進化は 1970 年代までは重要
- サイズだけでなく、消費電力が下がることが重要
- 80 年代から重要になったのは CMOS LSI の微細化。10 年でサイズが 1/10 になる
- CMOS 素子では(2000 年くらいまでは)微細化すると電圧を下げる
ことができ、消費電力が下がり、速度は向上した。いわゆる CMOS
スケーリング。
- 2000 年頃からは電圧が下がらないので、電力はちょっと下がるが
速度は上がらなくなった。いわゆる CMOS スケーリングの終焉。
- そろそろ微細化も困難になってきた。また、トランジスタの構造・
製造工程が複雑になり、微細化するとかえって価格上昇するよう
になった。いわゆるムーアの法則の終焉。

CMOS スケーリングって何？

「Dennard Scaling」とも

トランジスタのサイズ (3次元的にすべて)、電源電圧を $1/k$ にし、不純物濃度を k 倍にすると、トランジスタの速度は k 倍、スイッチングあたりの消費電力は $1/k^3$ になる

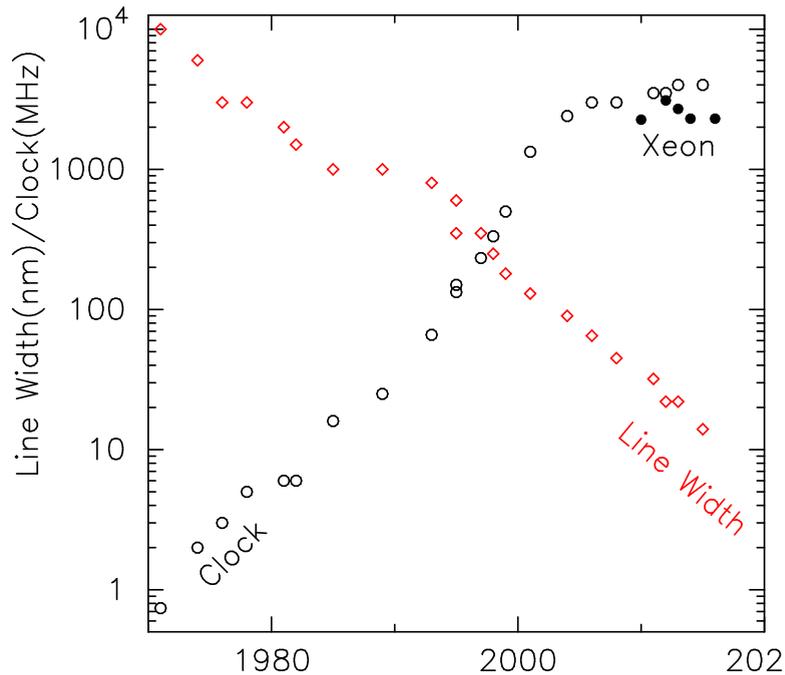
これが成り立つなら、

動作周波数 $\propto 1/\text{線幅}$

動作電圧 $\propto \text{線幅}$

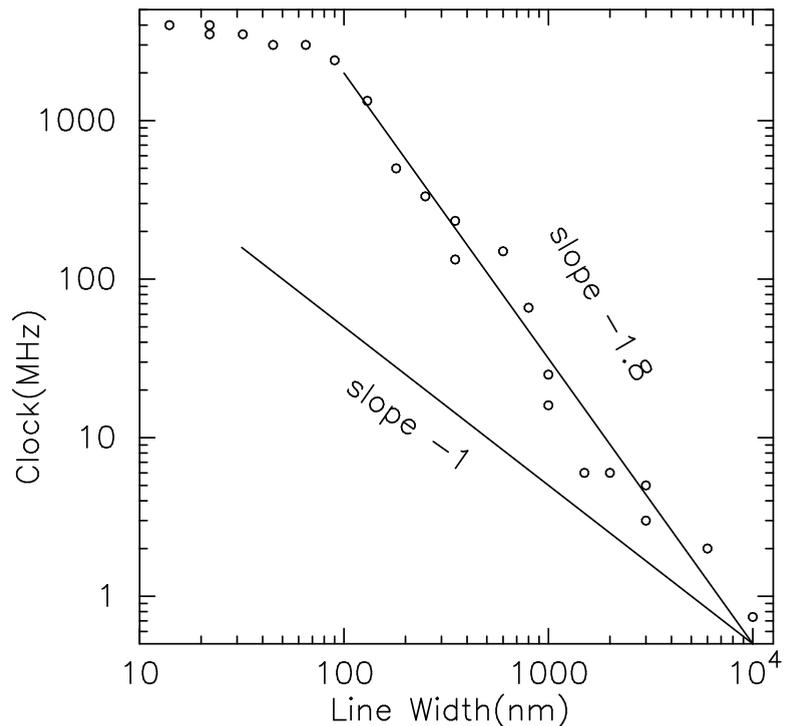
性能 $\propto 1/\text{線幅}^3$ (電力、面積一定で)

現実には



40年間の Intel マイクロプロセッサの線幅(トランジスタサイズ)と動作クロック
線幅は**40年間で3桁縮小**。クロックは**25年間で3桁上昇**、そのあと停止。(コア数の多い **Xeon** は段々クロック下がる)

線幅とクロック



線幅とクロックの関係

90nm までは

クロック \propto 線幅^{-1.8}

そのあとはほぼクロック一定

(Xeon は下がる、、、)

90nm までのクロック上昇は理屈より速い=無理していた。

つまり

- ムーアの法則(トランジスタサイズは時間の指数関数)は**2015**年くらいまでは成り立っていた(今は成り立たってない)
- いわゆる **CMOS** スケーリング(速度が線幅に反比例)は**2000**年くらいまでしか成り立たってない

何が起こったのか？

- **90nm** までは電圧をあまり下げず、パイプライン段数を増やすことでクロックをあげた
- これは消費電力増やすので、**1チップ100W** になったところで限界
- そこからは、コア数やコア内演算器数を増やすことで性能向上
- 最近はそれも難しくなってきたので再びクロック上げる方向も

計算機アーキテクチャの進化を振り返ってみる

- 1976 年まで: スカラー計算機。最後は **CDC 7600**
- 1976 年から 1992 年まで: ベクトル(共有メモリ並列含む)計算機。**Cray-1** から **C-90** まで
- 1993 年から 2008 年まで: (マルチコア含む) マイクロプロセッサの分散メモリ並列計算機 **Cray T3D** から、、、 **Red Storm** あたりまで
- 2008 年から: **GPU** アーキテクチャのアクセラレータとマイクロプロセッサの組合せ: **IBM RoadRunner (Cell** なので **GPU** じゃないけど)

大体 15 年毎にアーキテクチャが大きく変わった。GPU の次はまだでてきてない

アーキテクチャの変化が必要な理由

基本的な理由: そのアーキテクチャでは、半導体の進歩を有効に利用できなくなる

- 半導体技術の変化
- アーキテクチャのスケーラビリティの限界

スカラー計算機 ベクトル計算機

- スカラー計算機の進歩: 増えるトランジスタを「1つの演算器を速くする」に (CDC 6600 は複数のパイプライン化されていない乗算器をもっていたが、7600 でパイプライン演算器に)
- 主記憶は磁気コアで、半導体よりはるかに遅い
- **CDC 7600 (S. Cray の設計)** あたりが最後
- **S. Cray** はこの後、4 プロセッサ並列の **CDC 8600** を開発していたが、完成前に **CDC** やめて **Cray Research** を設立。ベクトルの **Cray-1** を開発する

スカラー計算機では

- **IC** の開発で使えるゲート数が1演算器を超えて大きくなった
- 半導体メモリの開発でメモリが非常に高速になった

ことを生かせなかった

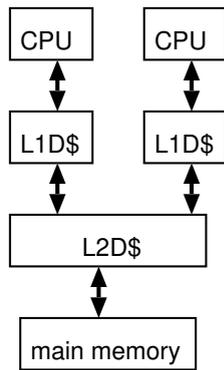
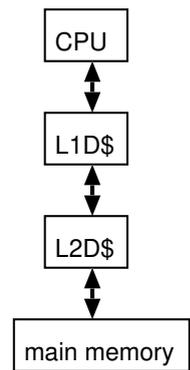
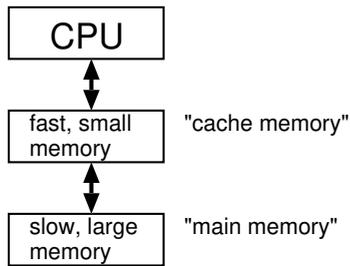
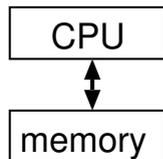
ベクトル計算機 分散メモリ並列計算機

- ベクトル計算機の進歩: 1 プロセッサのパイプライン数や主記憶を共有するプロセッサの数を増やす
- パイプライン数、プロセッサ数に比例以上にメモリへの配線の数が増える。ある程度 (**64**パイプラインあたり) で限界
- パイプライン化した1プロセッサが1チップにはいるようになる (**Intel 80860**)
- 1演算器でも、プロセッサチップと外付けメモリの間のバンド幅が不足すようになり、キャッシュが必須になる。
- 多数の小さなプロセッサとメモリの組合せの間を細いネットワークでつなぐ構成が有利になる。ベクトル計算機よりはるかに少ない配線で高い性能になる。
- 初期の代表: **Cray T3D**。但しネットワークが太い設計でそのうち破綻。

マイクロプロセッサ超並列 GPU

- 1チップの中に沢山プロセッサがはいるようになる
- これは実はシステムレベルでのベクトル計算機の限界と同じ
- マルチ(メニー)コアプロセッサでは、キャッシュコヒーレンシを維持した階層
キャッシュとオフチップの共有メモリ
- キャッシュコヒーレンシの維持のための電力が支配的になる
- **GPU** ではキャッシュコヒーレンシを緩和したり、捨てたりで若干改善
- 本質的な解決ではない

キャッシュとは？

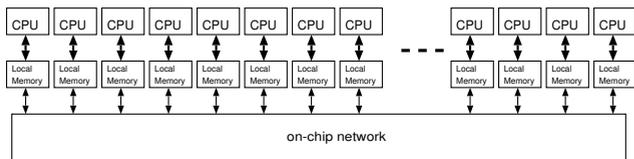
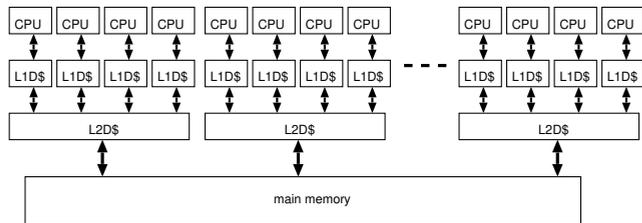


一時的にデータを置く小さいメモリ
何を置くかはハードウェアが勝手に
決める=ソフトウェアからみるとメ
インメモリをアクセスしているだけ。
データがキャッシュにあると速い
複数コアになると非常に複雑

ある **CPU** が自分のキャッシュに書いたものが他の **CPU** が元々のそのデータのアドレスを読んだ時にちゃんと読める必要がある (キャッシュコヒーレンシ)

GPU ???

- コヒーレンシを諦めても、オフチップメモリや階層キャッシュで物理的に遠くにデータ移動させること自体が性能の制限になる
- と書いた以上、階層キャッシュを諦めて、なるべく物理的に遠くにデータ移動させないようにすることが必要になるはず
- つまり、1チップの中でも、多数の独立なプロセッサに分かれた、チップ内分散メモリにいく必要がある



何故チップ内の横方向のデータ移動が制約になるか？

配線遅延・配線消費電力の問題

微細化しても、「単位長さ辺りのキャパシタンス」はかわらない、ところが、「単位長さあたりの抵抗」は配線幅の二乗に反比例して増える:

- 配線長がスケールしても配線遅延は減らない
- 配線長一定だと遅延は二乗で増える。電力は減らない

(最近だと、トランジスタ密度は上がっても配線はそんなに細くならないので配線が短くなれば遅延・電力が減る。長いと電力も遅延も減らない)

数字をいれてみると

- **10cm** ($10^5\mu\text{m}$) の配線は大体 **20pF** のキャパシタンスがある。これは微細化してもかわらない(線と平面の間のキャパシタンス)。電圧 **1V** だと、この線は **10pJ/bit** を消費する
- **DDR5** メモリは **20pJ/bit** くらい。電圧 **1.3V** とか。
- **LPDDR5** と **GDDR6x**: **10 pJ/bit** くらい。配線が **DDR5** のモジュールより短くて電圧も低い
- **HBMx**: **3-4 pJ/bit**。概ね **25mm** くらいまで配線短くなっている

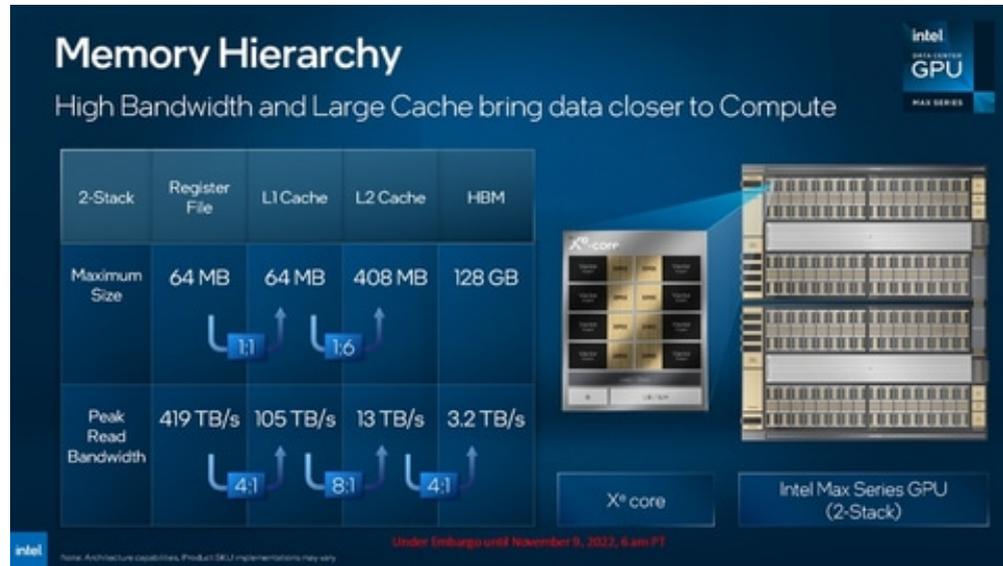
最先端の **GPU** だと、メモリアクセスだけで消費電力の半分くらいになる

64bit のデータを **10 cm** 動かすと **640pJ**。最新の **GPU** だと **64** ビット浮動小数点演算 **1** つに **20pJ** くらい。データの横方向移動の消費電力はものすごく大きい。

Intel GPU の例

B/F: 1 演算の間に何バイトのデータを移動できるか

B/F の値



レジスタ: 8? (普通 16 いる。行列乗算器だと 8 はありえる)

L1: 2

L2: 0.25

メモリ 0.06

非常に小さい

それでも電力消費が大きい原因になっている

設計目標は **1.6GHz** 動作だったが、熱の問題で **900MHz** 程度の動作になった模様。

最新の GPU の状況

	FP64 性能 (TF)	L2 バンド幅 (B/F)	メモリバンド幅 (B/F)
V100	7.8	0.32	0.12
A100	19.5	0.25	0.08
AMD MI250X	47.9	0.27	0.07
H100	34	0.17	0.10
Intel MAX	52(31)	0.25	0.06

- L2 バンド幅が極端に小さくなっている。0.2 前後だと L2 にのるアプリケーションでもメモリアクセスリミットなら実行効率は 5% 程度。
- CPU に比べてメモリバンド幅・演算性能の絶対値は高いが、キャッシュでも B/F は低く、アプリケーション実行効率が高くない。

ポスト GPU アーキテクチャ

ポスト GPU アーキテクチャでは

- チップ内部の演算器が物理的に共有するオフチップメモリはもたない
- チップ内部の演算器が物理的に共有するキャッシュはもたない

必要がある。これは **1990** 年代前半の共有メモリベクトル並列から分散メモリ超並列への移行と同じ。

つまり、チップ内で分散メモリ超並列を実現すればよい。

但し、普通にオフチップ **DRAM** をつけたのでは駄目 (バンド幅が足りない)

システムレベルアーキテクチャとチップレベルアーキテクチャの進化の類似

システム	チップ	
CDC 7600 まで	Intel 80860 まで	1パイプ
Cray 1 から T90	Intel KNL まで	複数パイプ共有メモリ
Stanford DASH/SGI Altex	GPUs? NUMA CPUs?	分散共有メモリ
Cray T3D から	???	分散メモリ

分散メモリシステムへのチップレベル対応物がまだない。

ここはどうなるか？の話の前に深層学習向けプロセッサと **MN-Core** の話を。

深層学習向けの計算機

- ニューラルネットワークの1層は「行列ベクトル積」
- 画像認識で主流の「畳み込みネットワーク」では「行列行列積」
- 計算精度はあんまりいらぬ。最近は8ビット表現、さらに4ビット表現も使われる
- トランスフォーマーでも「行列行列積」が主要な計算

現在の最先端

- **NVIDIA H100 (2022年発表)**
- **16ビットで 4×4 の行列同士の乗算の専用回路、8ビットだとその2倍の性能になるプロセッサ**
- **1チップで 1 Pflops 程度を実現。** (「富岳」の1チップは 11Tflops なので、100倍近く速い)
- **通常の64ビットだと 50Tflops 程度。**
- **今年 B300 がでた。性能4.5倍だがプロセッサダイが2個はいついて電力も2倍近い。**

以下、Preferred Networks でやっている話を

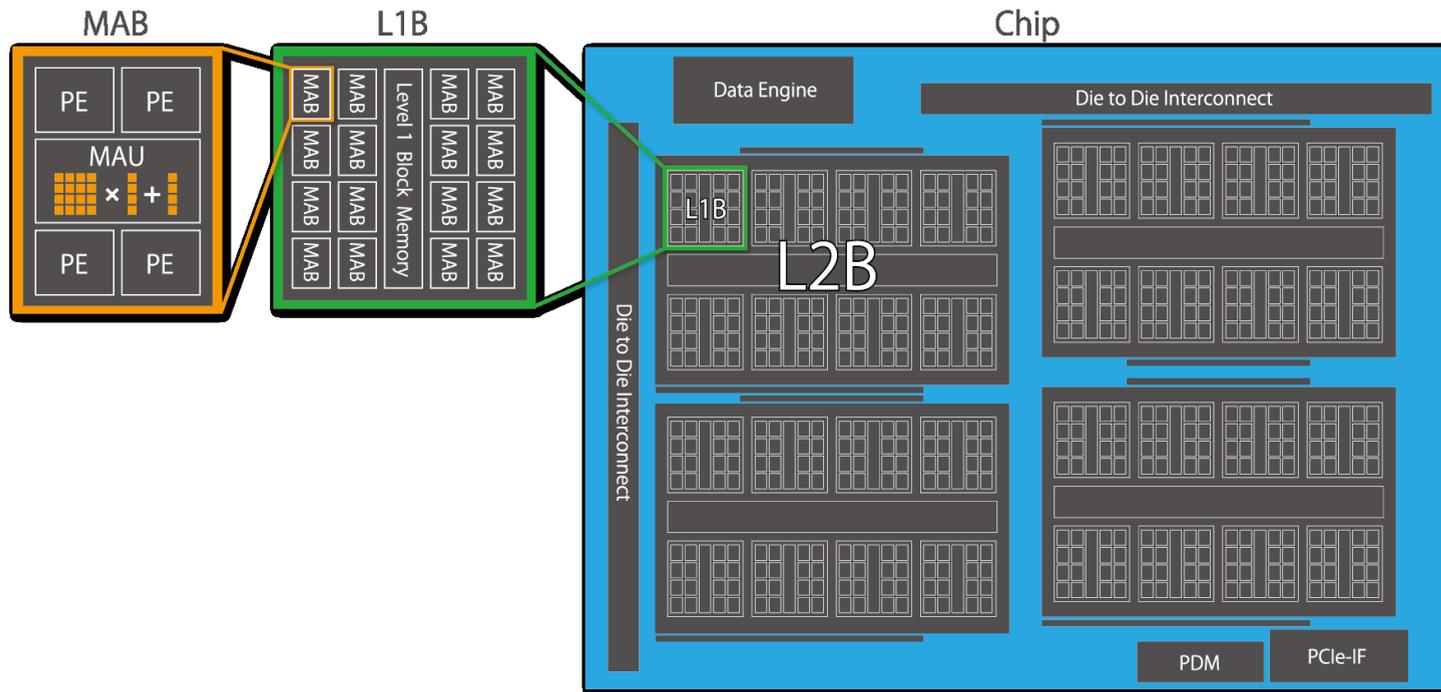
MN-Core

- **PFN** と神戸大学で開発した深層学習向けプロセッサ
- 深層ニューラルネットワーク (**DNN**) の学習で世界最高の電力あたり性能を実現
- **2020** 年に完成
- **FP16** (ライク) フォーマットでピーク **524TF**
- 電力消費 (チップレベル) **500W** 以下、**1.2TF/W**
- 同じ半導体技術の **NVIDIA V100** の **2.5** 倍以上の電力あたり性能を実現した

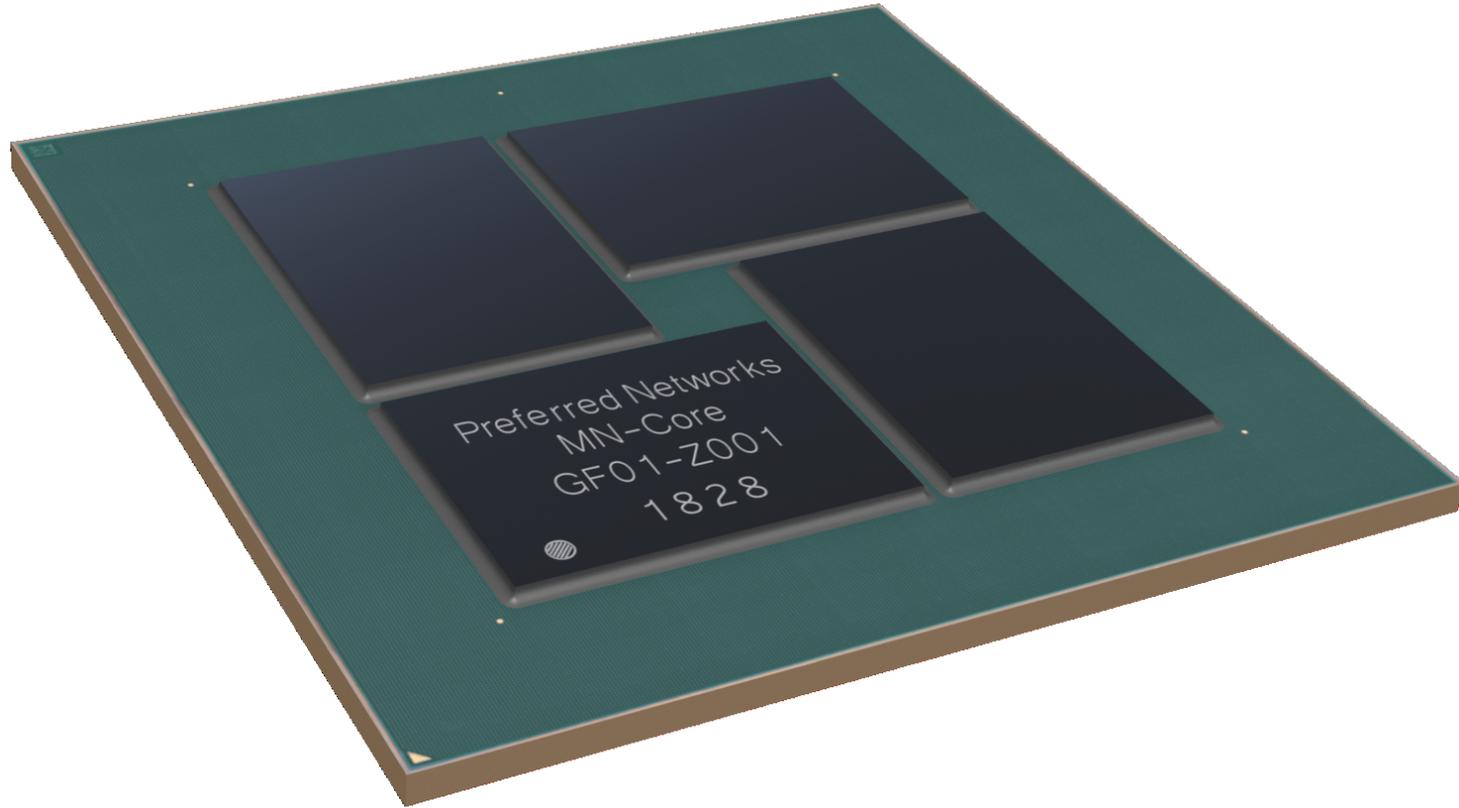
MN-Core 概要

- 1カード: 1「モジュール」
- 1モジュール: 4ダイ MCM
- 1チップ: PCIe (gen4, x16), xDDRx メモリ (内緒) 4 “Level-2 放送ブロック” (L2Bs)
- 1 L2B: 8 L1Bs
- 1 L1B: 16 MABs (Matrix Arithmetic Blocks)
- 1 MAB: 4 Processor Elements。これに対して1つ「行列乗算ユニット」
- FP64:FP32:FP16 性能比は 1:4:16
- カード全体が SIMD で動く。命令ストリームは1つ。

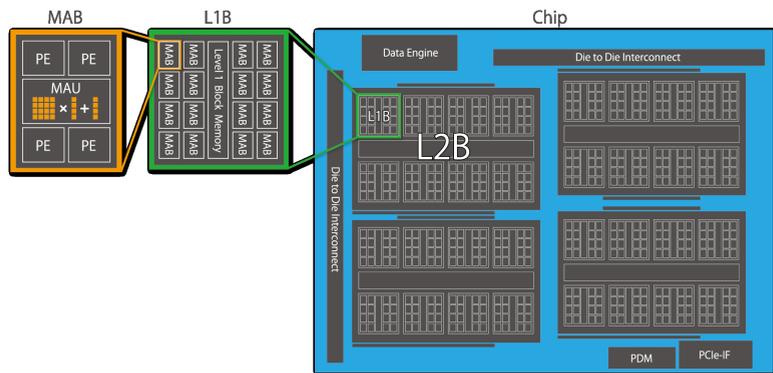
MN-Core構成図



MN-Core



チップ階層構造



- **トップレベル:** 4チップ内に4個(合計16個)の**L2B**(レベル2放送ブロック)
- **L2B**の中: 8個の**L1B**(レベル1放送ブロック)
- **L1B**の中: 16個の**MAB**(行列演算ブロック)

- **MAB**の中: 1つの**MAU**(行列演算ユニット)と4個の**PE**(プロセッシングエレメント)

MN-Core: 1ボードに**PE 8192**個。**MAB 2048**個。ポスト富岳の現在の案は**PE**あたりの性能を**2倍**に強化、**クロック 5倍**、**PE数 1.5倍**くらい。

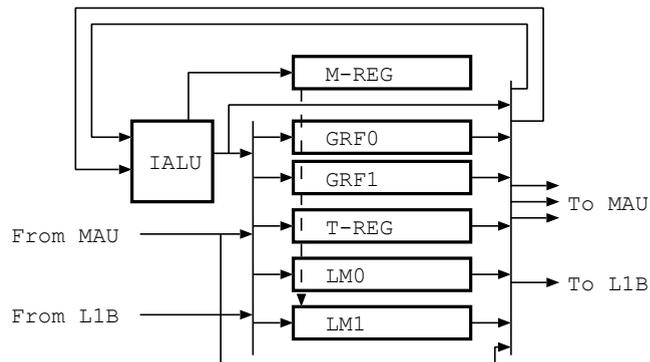
チップ階層構造続き

- 概念的には全体が同期して単一の命令流を処理する。 **L2B** 以下は実際にクロック同期。チップ間を含めて **L2B** 間は (周波数は同じだが) 命令実行にずれがあることを許す。
- チップ毎に **DRAM** ブロックがある。 **DRAM** は **L2B** とブロック転送でデータ交換できる。放送、個別転送、チップ内分配等いくつかのモードがある。

詳細構造

- PE(プロセッシングエレメント)
- L1B(レベル1 放送ブロック)
- L2B(レベル2 放送ブロック)
- 全体

PE(プロセッシングエレメント)



- 演算器は **IALU** と **MAU**
- **MAU** は倍精度、単精度、半精度精度の行列ベクトル積を実行
- **L2B** の中: **8** 個の **L1B**(レベル1 放送ブロック)
- **L1B** の中: **16** 個の **MAB**(行列演算ブロック)

- バスみたいに書いてますが実際の回路はマルチプレクサ
- **GRF** は (**MN-Core** では) **1R1W 2** ポート。 **LMx** はシングルポート。
- **T-reg**: 補助レジスタ。 **1R1W** で **1** ベクトル分
- **LMx** (ローカルメモリ) は **2048 64** ビット語 x **2**、 **GRF** は **512** 語
- ベクトル長 **4** の固定長ベクトル命令。

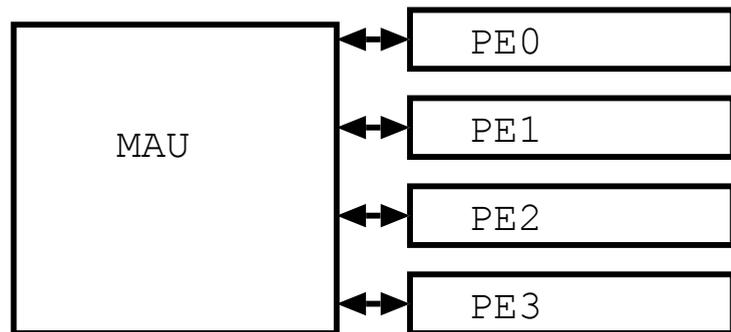
PE(プロセッシングエレメント) 続き

- 直前の命令の演算結果を入力オペランドにとることが可能(フィードバック)
- ベクトル命令でのアドレッシングは固定、連続アドレス、ストライドアクセスが可能(ベクトルレジスタもストライドアクセス可能)
- T-レジスタを使った間接アドレスも可能
- アドレスはベースアドレスレジスタで修飾
- **IALU** の演算結果から大小比較、一致比較等のフラグベクトルを生成して **M** レジスタに格納可能
- **M** レジスタの値でメモリ、レジスタに書き込むかどうかを制御

PE(プロセッシングエレメント) 続き

- ロードストアアーキテクチャではなく、**LMx** の読み出し結果を直接演算器の入力にできる。また、演算器の出力をメモリに直接格納できる。複数のユニットに格納もできる。
- **B/F** が極端に大きい。(ベクトル演算モードに対して **8** ある)。このため、ローカルメモリにデータがあればほぼ理論ピーク性能がでる。
- この機能と、演算結果フィードバック、**T**レジスタを最大限利用することで、通常のアーキテクチャに比べてレジスタファイルのハードウェア規模、消費電力を大きく削減している。また、スーパースカラー実行制御、**OoO** 実行のための回路も不要となる。=高い電力性能の実現に大きく貢献

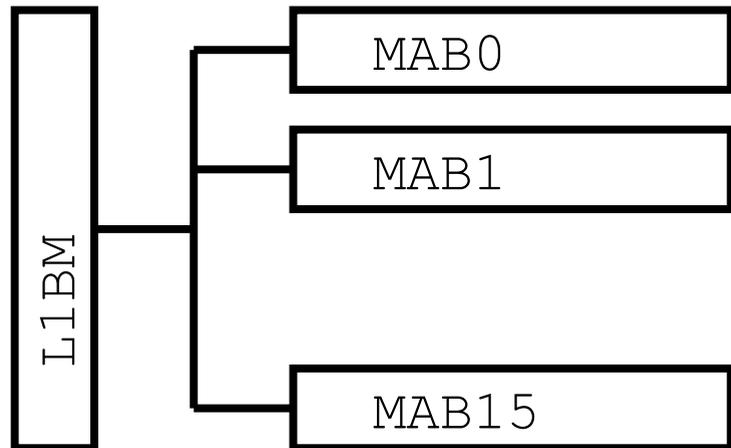
MAB(行列演算ブロック)



- **PE4** 個が 1 つの行列乗算器を共有
- サイクル毎に行列ベクトル積を実行。
 $d = A * b + c$
- **MAU** は行列レジスタをもつ

- 行列レジスタは複数あり、演算中に別の面への書き込みが可能
- 転置しながら格納も可能
- 倍精度、単精度、半精度行列に対応。サイクルあたり演算数は **32, 128, 512**
- ベクトル演算 (**PE** から見ると **64** ビットスカラー演算) モードもある
- 倍精度、単精度、半精度行列乗算器は同一の乗算器の構成変更で実現 (特許取得済): 他社の回路に比べると同一性能で回路規模が半分以下

L1B(レベル1放送ブロック)



- **MAB 16** 個が1つの**L1BM**(レベル1放送メモリ)に接続。
 - **L1BM** から読み出されたデータは放送される(各**PE/MAB**での書き込み制御で単一の**PE/MAB**に送ることもできる)。
 - 分配モードも一応ある(あんまり速くない)
-
- 各**PE**から読み出したデータは、縮約して**L1BM**に書き込むことができる。単一の**PE/MAB**を指定した読出しもできる。分配の逆操作(結合)も一応ある。
 - **PE**間の直接結合はない。

L1Bの特色

- 明示的な放送/縮約モードがあるため、複数 **PE** を使った細粒度並列処理を低オーバーヘッドで実行できる。
- **GPU** が非常に苦手とする **reduction** をオーバーヘッドを気にしないで使うことができる。
- 例えば比較的小さな行列乗算を **MAB** に分散させることができる。総和が遅いと、行列行列積 **A*B** で **B** を縦に切る並列化をすると、各 **MAB** で **A** は全体が必要になる。総和が速いと、**B** を横に切る並列化ができ、**A** を分散させられる。また、単一の行列ベクトル積の低オーバーヘッドの並列化ができる。推論(**LLM**でも)で非常に効率をあげやすくなる。
- (**PE** 命令と同期した、4サイクル毎のデータ転送命令で制御)

L2B(レベル2放送ブロック)

- **L1B** 8個に対して **L2BM**(レベル2放送メモリ)が1つ。
- **L1B** 内と同様な放送・縮約・分配・結合・個別読出しを **L1B** に対して行う。
- データ幅は広くしている。
- いくつかの **L1B** 同士の直接データ交換モードがある。
- **PE** 命令と同期した、4サイクル毎のデータ転送命令で制御。

全体

チップ4個を全体としてみると

- **L2B 16個、DRAM インターフェース4個、PCIe インターフェース4個がある。**
- **PCIe インターフェースはバッファメモリ (PDM) を持つ**
- **PDM-DRAM、PDM-L2B、DRAM-L2B で、放送・縮約を伴うデータ転送を実行可能。**
- **このレベルのデータ転送命令は非同期で実行される。PE 命令とはお互いにタグ待ちができる。**

PE 命令の概要

PE 命令は

- 各演算の動作モード、入力セレクト、各メモリユニットのアドレス、**LM** では **read/write** モード、入力セレクト、各メモリユニットの入力マスクレジスタ番号、書き込みマスクレジスタ番号等
- **L1BM-PE, L2BM-L1BM** の転送モード、転送アドレス等

を指定する。

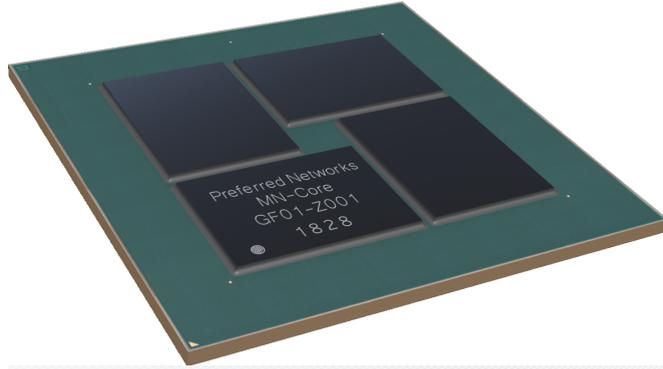
通常のアーキテクチャとの違い(1)

- キャッシュ階層がない、ローカルメモリだけのアーキテクチャ(**DRAM** はオフチップでデータ転送命令でアクセス): メモリアクセスの電力消費が小さい
- ロード・ストアアーキテクチャではなく、メモリとレジスタの区別はポート数だけ: 演算に対する「無駄な」メモリアクセスを削減
- 複数のメモリ要素を **ISA** として見せる: 面積・電力の大きいマルチポートレジスタファイルをなくす
- 大規模 **SIMD** アーキテクチャを採用: コア間の同期オーバーヘッドをなくし、細粒度並列化を可能に
- 放送・縮約をチップ内ネットワークレベルでサポート: 細粒度並列の性能向上に大きく貢献

通常のアーキテクチャとの違い(2)

- 現在の通常の **CPU** では、レジスタリネーミングと **OoO** 実行によって、最内側ループに本来ある並列性を実行時に回復する必要がある。これは、通常はアーキテクチャレジスタの数が不足し、コンパイラによるアンロールだけではパイプラインを埋められないからである(「京」は例外)
- **MN-Core** ではアーキテクチャレジスタ数の制限がないため、レジスタリネーミングが不要となる。
- さらに、固定長ベクトル命令の採用により、**OoO** 実行自体が不要になっている。直前の命令の実行結果が必ず利用可能だからである。
- かなり単位の大きな **SIMD** 実行をすることで、非常に長い命令語を許しており、レジスタのアドレス長等の制限が不要になっている。

MN-Core/MN-3 system



The
GREEN
500 CERTIFICATE

MN-3 - MN-Core Server, Xeon 8260M 24C 2.4GHz, MN-Core, RoCEv2/MN-Core

DirectConnect

Preferred Networks, Japan

is ranked

No. 1 in the Green500

among the World's TOP500 Supercomputers

with 21.11 GFlops/Watt Linpack Power-Efficiency

on the Green500 List published at ISC 2020 Digital Conference, June 22nd, 2020

Congratulations from the Green500 Editors


Wu-chon Feng
Virginia Tech


Kirk Cameron
Virginia Tech

PFNのスパコン「MN-3」が世界1位に、消費電力性能ランキングのGreen500で

岡林 遼太郎 日経クロステック/日経コンピュータ

2020.06.23



[PR]

Preferred Networks (PFN) のスーパーコンピューター「MN-3」が2020年6月22日（欧州時間）、スーパーコンピューターの消費電力性能ランキング「Green500」で世界1位を獲得した。HPC（ハイ・パフォーマンス・コンピューティング）に関する国際会議「ISC 2020 Digital」が同日ランキングを発表した。



PFNのスーパーコンピューター「MN-3」

(出所: PFN)
(画像のクリックで拡大表示)

MN-3はPFNが独自開発した深層学習用プロセッサ「MN-Core」を使ったスーパーコンピューターだ。PFNのスーパーコンピューター「MN-2」の後継機で、2020年5月に運用を始めた。160個のMN-Coreを搭載し、1ワット当たり21.11ギ

MN-Core 2 と後継

- **MN-Core2** 2023 年完成。販売開始した。
- **TSMC 7nm**。比較的小さいチップで **400TF** と **MN-Core** 並みの性能を実現
- 後継の開発もスタート
 - **Samsung 2nm**。完成時点で世界最高レベルの性能を目標
 - **LLM 推論**向けの開発も開始
- **ポスト富岳**向け検討もしてきたが、直近のものには採用されていない。

MN-Core 2 のソフトウェア

- **MNSDK: 機械学習向け**
 - **PyTorch** 記述から **ONNX** 経由でコード生成
 - 既存の **PyTorch** コードから小さい修正で動作。ネットワークの記述等は同じ
- **HPCSDK: 汎用 HPC 向け**
 - **OpneCL** 方言(メモリモデルが違う)、**OpenACC** 方言を用意

MN-Core 2 のアプリケーション性能

	MN-Core 2	A100
GCN(PFN internal use)	5.41TF(FP32)	3.17TF
ResNet50 training	77TF(FP16)	33.2TF(BF16)
ResNet50 Inference	154TF(FP16)	33.7TF(BF16)
HIMENO benchmark	9.03TF(FP32)	0.634TF
OpenFDTD	0.655TF(FP32)	0.488TF

- AI ワークロードでは同程度のピーク性能の **A100** の **1.5-5** 倍の性能。並列化効率が高いことを実証。
- 差分法系のアプリケーションやベンチマークでも高い性能。但し、**OpenFDTD** は **temporal blocking** を実装して **DRAM** アクセスを減らした成果。姫野ベンチはオンチップメモリにはいるサイズ。

MN-Core の特徴

- チップ内の演算コアがそれぞれメモリをもち、コア間がツリーネットワークでつながる「チップ内分散メモリアーキテクチャ」
- キャッシュはない。外部 **DRAM** は共有される形で接続されるが、各コアがアドレスをだすのではなくデータ移動命令でツリーノードとの間でブロック転送
- チップ内分散メモリアーキテクチャと従来のアーキテクチャの中間
- 従来型プロセッサより高い性能と実アプリケーションでの高い実行効率を実現
 - **SIMD** 動作と、総和、放送をサポートする低レイテンシチップ内ネットワークによるオーバーヘッドの小さい並列化
 - 大きな **PE** ローカルメモリによる効率のよい **DRAM** アクセスの低減
 - 高い **PE** ローカルメモバンド幅 (ベクトル演算に対して **B/F=8**) によるカーネル部分の高効率実行

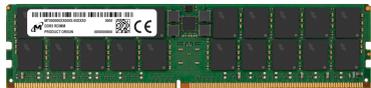
とはいえ、、、

- 大容量の **DRAM** は依然としてチップ外部。このバンド幅はあまり大きくない。
- **DRAM** 転送バンド幅をあまり必要としないアルゴリズム・アプリケーションでないと性能がでない。 **CNN** はそうだが **LLM** はそうではない。
- 「真の」オンチップ分散メモリとはいいいがたい。

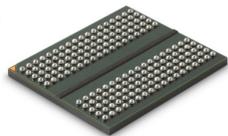
「真の」オンチップ分散メモリは？

- **HBM** は **DRAM** の平面的実装としてはほぼ究極。プロセッサダイのすぐ隣に **DRAM** ダイがある。
- それでも、電力消費はプロセッサダイと **DRAM** ダイの中を横に動く分、下のインターポーザの中を横に動く分が大きい。また、どうしても共有メモリアーキテクチャになる。
- 分散メモリにするは、**DRAM** ダイをプロセッサダイの上にのせて、さらに面的な配線で演算器とその直上(下?)の **DRAM** メモリブロックを接続すればよい。

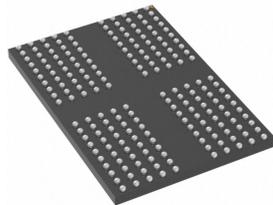
メモリアーキテクチャの変化



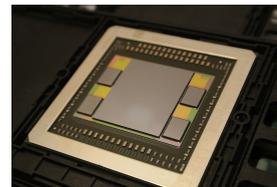
DDRx(2000 ~)



GDDRx(2003 ~)



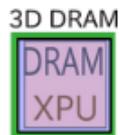
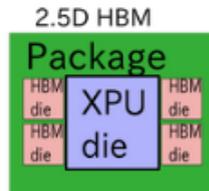
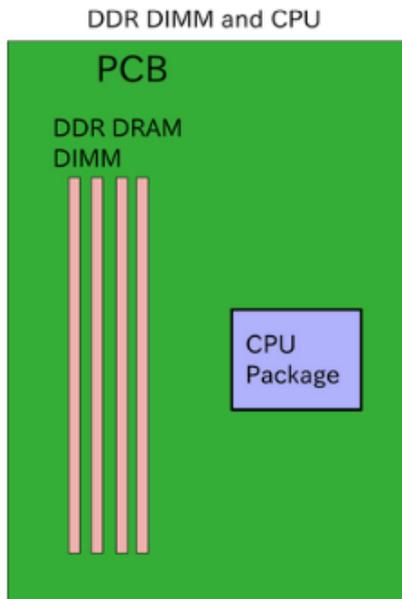
LPDDRx(2008 ~)



HBMx(2015 ~)

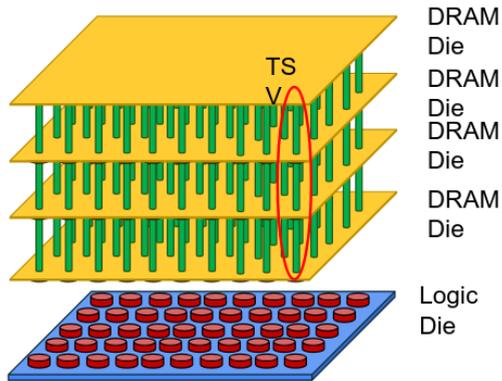
- **DRAM** メモリセル自体の進化はあんまりない(キャパシタンスを小さくしたいが、するとリフレッシュ間隔が短くなりすぎる)
- 主な進化は配線長の短縮
- アクセスエネルギー: **DDRx: 20pJ/bit, HBMx: 4pJ/bit**。大きく減ったがそれでも大きい。

次はどうするべきか？



3D DRAM、つまり、**DRAM** をプロセッサダイの上に載せる(か、下に置く)必要がある。
メモリ階層を変える必要もある。

3D 積層メモリ



複数の **DRAM** ダイとロジックダイを非常に多数の **TSV** とパッドでつなぐ

マイクロバンプか「ハイブリッドボンディング」が使われる

いくつかの (3大ベンダでない、、、) **DRAM** ベンダが技術開発している。

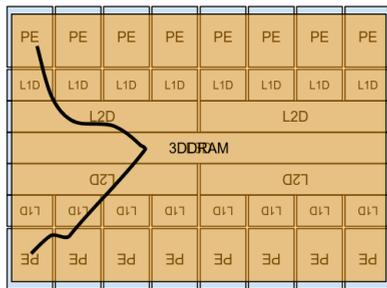
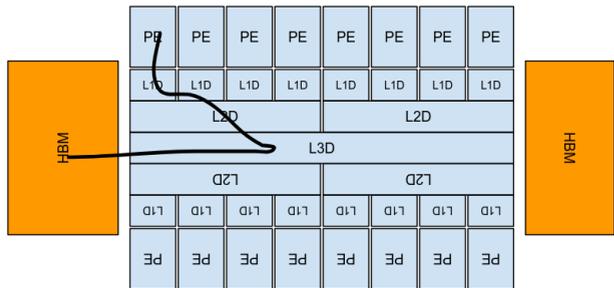
台湾: **Powerchip, Winbond, Nanya**

中国: **CXMT, YMTC, XMC.**

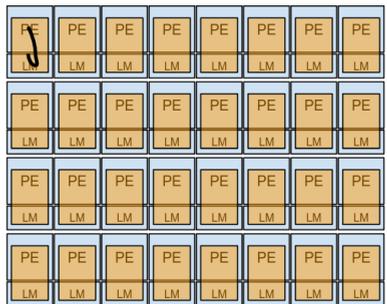
コンピュータアーキテクチャの歴史で、主記憶バンド幅が大きくジャンプする
2回目の(最後かもしれない)機会

最初: 磁気コアメモリから半導体メモリへの変化。

積層メモリと共有メモリ、分散メモリ



共有メモリだとデータ移動距離があんまり減らない。



分散メモリだとデータ移動距離を最小化できる。

つまり: 意味がある「オンチップ分散メモリアーキテクチャ」を考えられる。

システムレベルアーキテクチャとチップレベルアーキテクチャの進化の類似

システム	チップ	
CDC 7600 まで	Intel 80860 まで	1パイプ
Cray 1 から T90	Intel KNL まで	複数パイプ共有メモリ
Stanford DASH/SGI Altex	GPUs? NUMA CPUs?	分散共有メモリ
Cray T3D から	オンチップ分散メモリ プロセッサ	分散メモリ

分散メモリシステムへのチップレベル対応物が存在する(かも)

接合技術の現状

基礎となる技術:

- 接合: マイクロバンプ or ハイブリッドボンディング
- TSV: TSV-middle 等のファインピッチ技術

マイクロバンプ

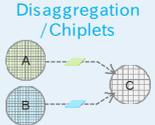
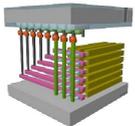
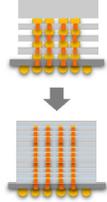
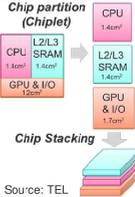
- 基本的にハンダボールや **C4** バンプの延長の技術
- 現状で利用可能なピッチ: $\sim 40\mu\text{m}$ 、**Intel MAX GPU** では $37\mu\text{m}$ 。
- 将来的には $10\mu\text{m}$ を切るといわれている。
- **HBM3e** までの各社の **HBM** で採用
- 現状では 1mm^2 あたり **600** 本程度なので、 800mm^2 のチップだと全面において **50** 万本。 **50TB/s** 程度までなら実現可能
- 複数積層だと **TSV** があるエリアを集中させたいのもう **1** 桁程度高密度にしたい
- ダイ間を樹脂で充填するので、熱抵抗が大きい。(現在の **HBM** で大きな問題)
- **CoW/CoC** なので、イールドの問題は少ないが加工費用が高価

ハイブリッドボンディング

- ダイ表面の **Cu** パッドを直接接合。
- 最初の実用化はソニーの センサー
- 現状で多くのベンダで利用可能なピッチ: $\sim 5\mu\text{m}$
- 将来的には $1\mu\text{m}$ を切るといわれている。
- 現状では 1mm^2 あたり 4万本程度なので、 800mm^2 のチップだとエリアの **2.5%** でも **100万本**。 **1Gbps** で **1Pbit/s**、つまり概ね **100TB/s** が実現可能。
- ダイ間に樹脂がはいらないので、熱抵抗が小さい。
- **Wafer-on-Wafer** なので加工費用は安価 (**BSPDN** でも同様な技術が使われる) が、イールドの問題はある。 **1Gbit DRAM** のイールドは **90-95%** といわれており、それを前提にした冗長設計が必須になる。

技術の現状

貼り合わせ接合適用例

Application	CIS	NAND	DRAM	Logic		Advanced Package	
Stacking Device	BSI Pixel + Peripheral + Logic	3D NAND Cell + Peripheral	3D DRAM Cell + Peripheral	Backside PDN Logic + Bare Si	Sequential CFET Logic + Logic	HBM DRAM (↔) ⋮ DRAM (↔) + Logic	Disaggregation / Chiplets 
Bonding	W-W (Ox Fusion) Cu Hybrid	W-W Cu Hybrid	W-W Cu Hybrid	W-W Ox Fusion	W-W Ox Fusion	D-W u Bump → Cu Hybrid	D-W / D-D Cu Hybrid
3D I/O Pitch	3um → 1um	1um → 0.5um	1um → 0.5um	Sub um (nTSV)	Sub um (nTSV)	40um → 25um	10um → 1um
Structure			 <small>Source: TEL 3D DRAM structure</small>				 <small>Chip partition (Chiplet) CPU 1.4cm² L2/L3 CPU SRAM 1.6cm² L2/L3 SRAM 1.4cm² GPU & I/O 1.0cm² GPU & I/O 1.7cm² Chip Stacking Source: TEL</small>
Status	HVM	R&D~HVM	R&D	R&D	R&D	R&D	R&D~HVM

次世代デバイスへ向けて、貼り合わせ接合技術の導入が拡大

原理的な限界は？

- **HBM** 等でも、**DRAM**側の現状の設計は **DDRx** の **DRAM** と同等。クロックがはいり、**FIFO** やセレクタを通してデータがでてくる。部分書き込みのためのデータ保持回路等もある。
- **DRAM** チップ内でみても、実際の **DRAM** セルやセンスアンプの消費電力は結構小さい (**O'Connor et al. Fine-Grained DRAM: Energy-Efficient DRAM for Extreme Bandwidth Systems, MICRO-50**)
- ハイブリッドボンディングでは非常に大きなパッド数が可能であるので、回路構成から再検討が可能。
- **0.1-0.2 pJ/bit** くらいまではそのうちいけそう。

実装上の懸案事項

一杯ある。

- **HBM** みたいに **DRAM** スタックできるし、**TSV middle** とかですごく沢山パッドだせるんだけど、**DRAM** スタック上につんで冷却できるか？
- **face up** でロジックダイ使って電源とか信号線引き出せるか？
- **DRAM** スタック下においたら、**DRAM** スタック通して信号とか電源だせるか？

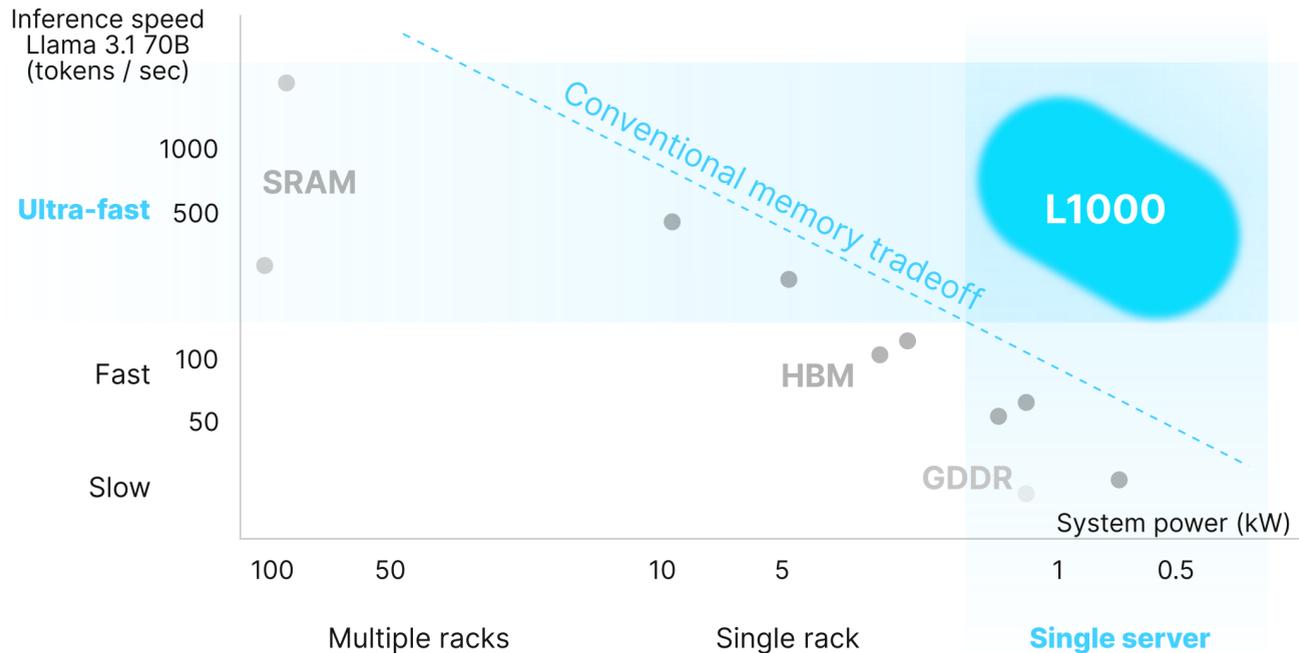
とはいえ、解決できない問題という感じでもない。物理的に金属配線でつながるので微細化できれば数が増えてバンド幅があがる。(磁界結論や容量結合はスケールしない)

MN-Core アーキテクチャと 3D DRAM

- **MN-Core** アーキテクチャはチップ全体が **SIMD** アレイ。それぞれの **PE** がローカルメモリをもって、階層的なネットワークで結合。
- **3D DRAM** は、基本的には各 **PE** に **DRAM** ブロックがつく。例えば **16MB** とか。
- **1980** 年代風の超並列 **SIMD** プロセッサとほぼ同じ。ネットワークはさぼっている。概ね同じようにプログラミングもできる。
- つまり: ほぼ **HPF (OpenACC** ともいう) でプログラムできる。(但し、**DRAM** とオンチップメモリの違いを意識する必要がある)

PFN の計画

2026 年に、現在の GPU 等と比べて同じ電力で 10 倍の LLM 推論性能をもつ MN-Core L1000 を利用可能にする
電力性能だけでなく価格性能比でも世界最高を実現する



<https://mn-core.com>

他社の AI むけプロセッサは？

- 昨年の **HotChips 24**: 口頭発表 **24** 個のうち **14** 個が **AI** アクセラレータ。
- **Tenstorrent Blackhole**、**SK Hynix PiM**、**Blackwell**、**Sambanova SN40L**、**Intel Gaudi 3**、**AMD MI300X**、**FuriosaAI RNGD**、**AMD(Xilinx) Versal**、**Onyx** (Stanford の研究)、**Meta MTIA**、**Tesla Dojo**、**Cerebras CS-2**、**MS MAIA**、**PFN MN-Core 2**
- **Blackhole**、**SN40L**、**Onyx**、**MTIA**、**Dojo**、**Cerebras**、**MAIA**: **2D** オンチップネットワークに **MIMD** コア+行列演算ユニット。
- **Gaudi**、**RNGD**: 大きな行列乗算器もつ独自アーキテクチャ(**Groq** も)
- **SK Hynix** は **GDDRx** メモリチップの中に演算器いれるもの
- **Versal** は **FPGA**、あとは **Nvidia**、**AMD**、**PFN**

アーキテクチャの特徴

- 階層キャッシュはもうないというのは **Nvidia** と **AMD** の **GPU** 以外では共通。
- **AI** プロセッサアーキテクチャはほとんどが **MIMD 2D** メッシュ。メッシュの辺のところに **HBM** つける。
- どうやってプログラムを書くのか?というとりあえずとても大変そう。
- **AI** 用プロセッサは **FP64** ない。 **FP32** も絶滅しつつある。 **Nvidia**、 **AMD** の **GPU** も **FP64** の性能が相対的に下がってきている。(**MN-Core** は異なる精度間で演算器を共有しているので比較的 **FP32**, **FP64** の性能を維持しやすい)
- **HBM** なのはみんな同じ。次世代では **HBM4** ではなくカスタム **HBM**(チップ間接続に **UCIe** 等を使う) がでてくるかも。 **3D DRAM** には時間がかかりそう。

まとめ

- 計算機の進歩について、「ポストムーア」は今度は本当。シリコン半導体技術の限界は確実にきている。
- しかし、計算機設計という観点から見ると、問題はそこではなくて、メニーコア **CPU** や **GPU** の設計が半導体技術の進歩を上手く利用できなくなっていること。これは**30**年前にシステムレベルでの共有メモリ計算機が無理になってきたのと同じ。
- **GPU** アーキテクチャは階層キャッシュ構造から離れるのは難しいように見える。
- **AI** 向けプロセッサはキャッシュなし、ローカルメモリの方向。
- **MN-Core** は、さらに大規模 **SIMD** と階層的オンチップネットワークで高いピーク性能と **AI** でも汎用 **HPC** でも高い実行効率を実現。
- さらに、**3D DRAM** を有効に利用して **DRAM** バンド幅をあげることが重要になる。我々はこれにも取り組んできている。