しくじりフェロー

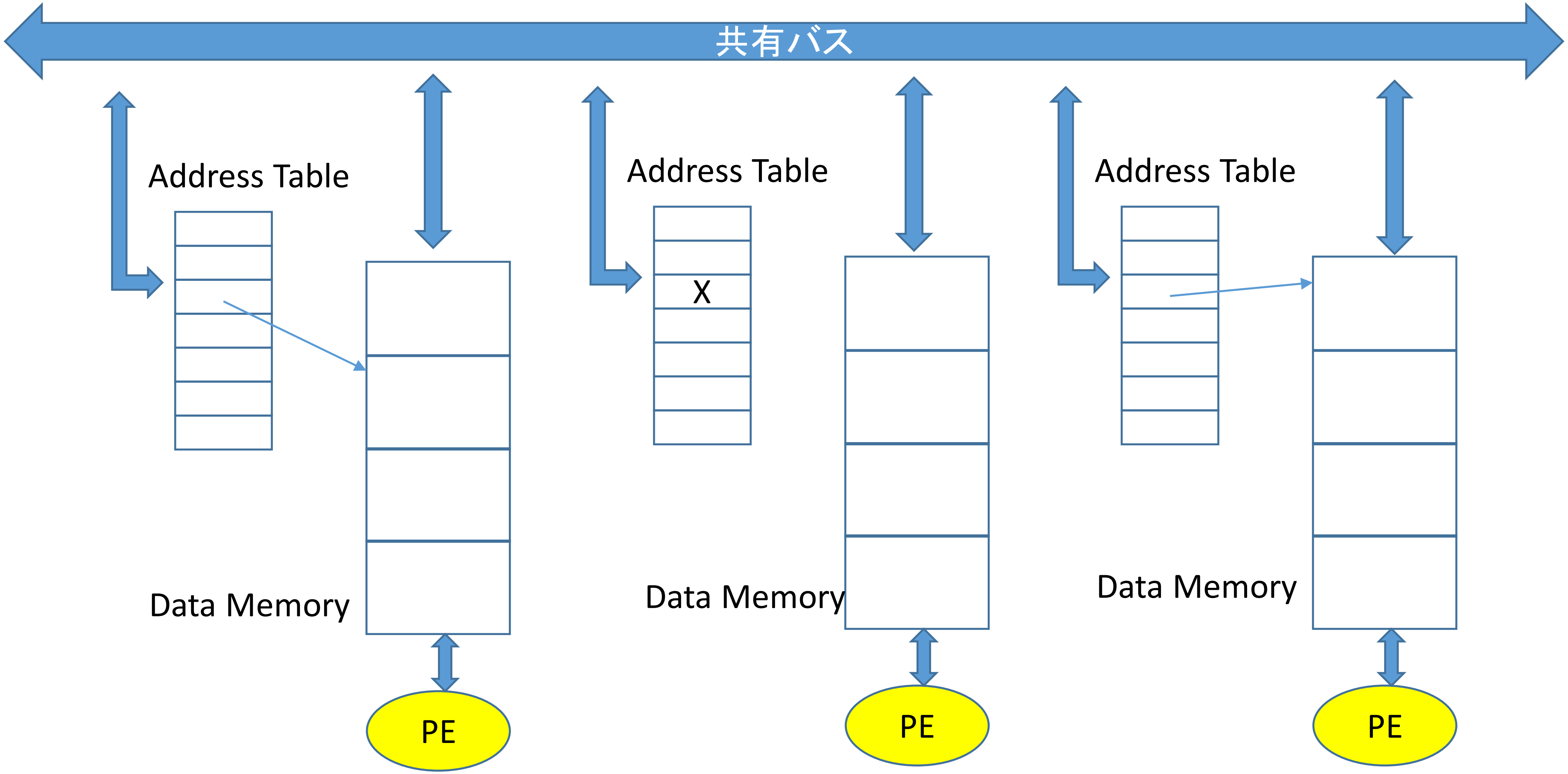
慶應義塾大学　天野英晴

ハードルが上がってしまった！

天野は何をやってきたのか？

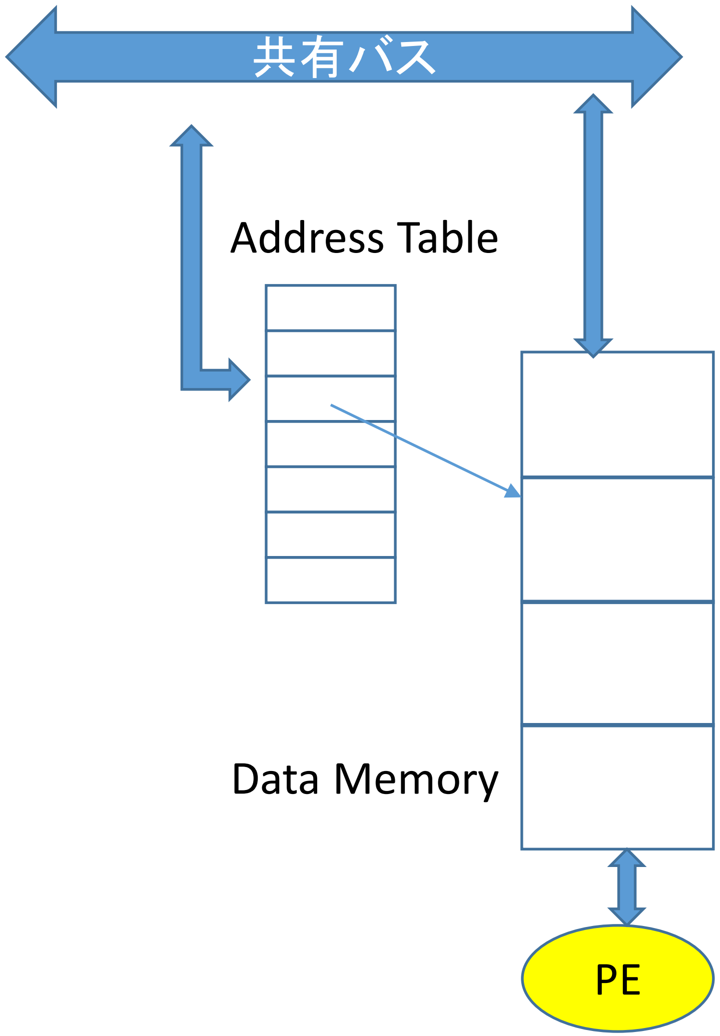
- 主に並列計算機を研究してきた
  - CPU自体は、MBP-lightに至るまで作ったことがなかった。
    - 学生の頃趣味でオリジナルCPUを作ったが、モニタがなくてろくに動かなかった。
    - CPU作ってもOSが作れないので動かせない。
  - 交信用メモリ
  - 結合網
  - リコンフィギャラブルシステム
- ハードウェアを作ってきた
  - コアをLSIチップで実装
  - システム全体を作る
- ずっと慶應に居た

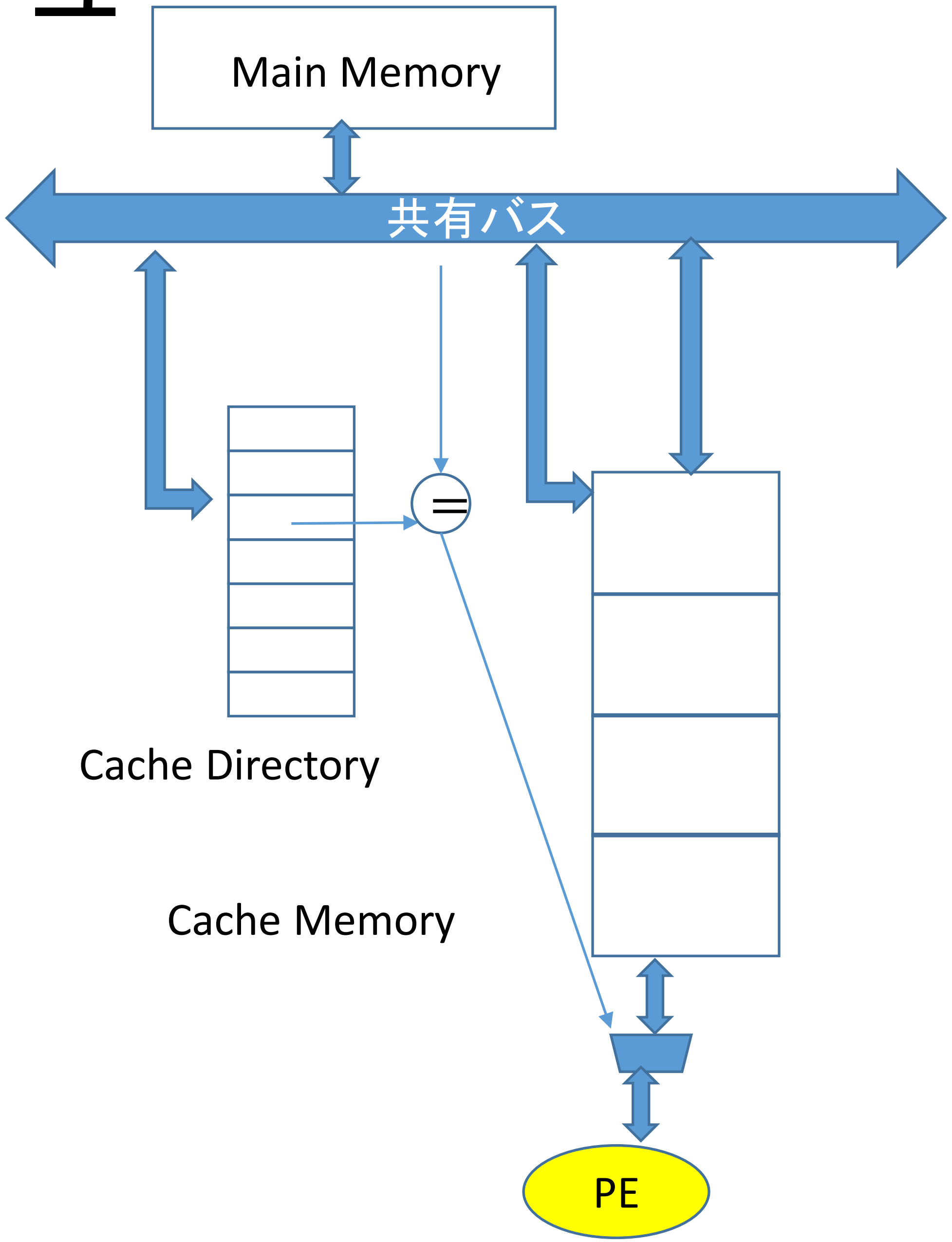
マルチプロセッサの交信用メモリRSM(1983)



- 各PEの交信用メモリが分散的に共有バスのアドレスをチェックし、必要なデータを独立な領域に取り入れる
- RSM(Receiver Selectable Multicast) (SM)2 ISCA83, (SM)2-II ISCA85

RSM vs.スヌープキャッシュ





なぜRSMはスヌープキャッシュまで行けなかったのか？

- 発想は類似していた
- 1983年のISCA（スウェーデンのストックホルムでやった）ではGoodmanがスヌープキャッシュの元のアイディアであるGoodman Cacheの発表をやっていた
- この時、RSMをWrite Update型スヌープキャッシュに進めることができなかったのか？
- 奴のは実装ができないが、俺のは実装ができると思ってしまった。

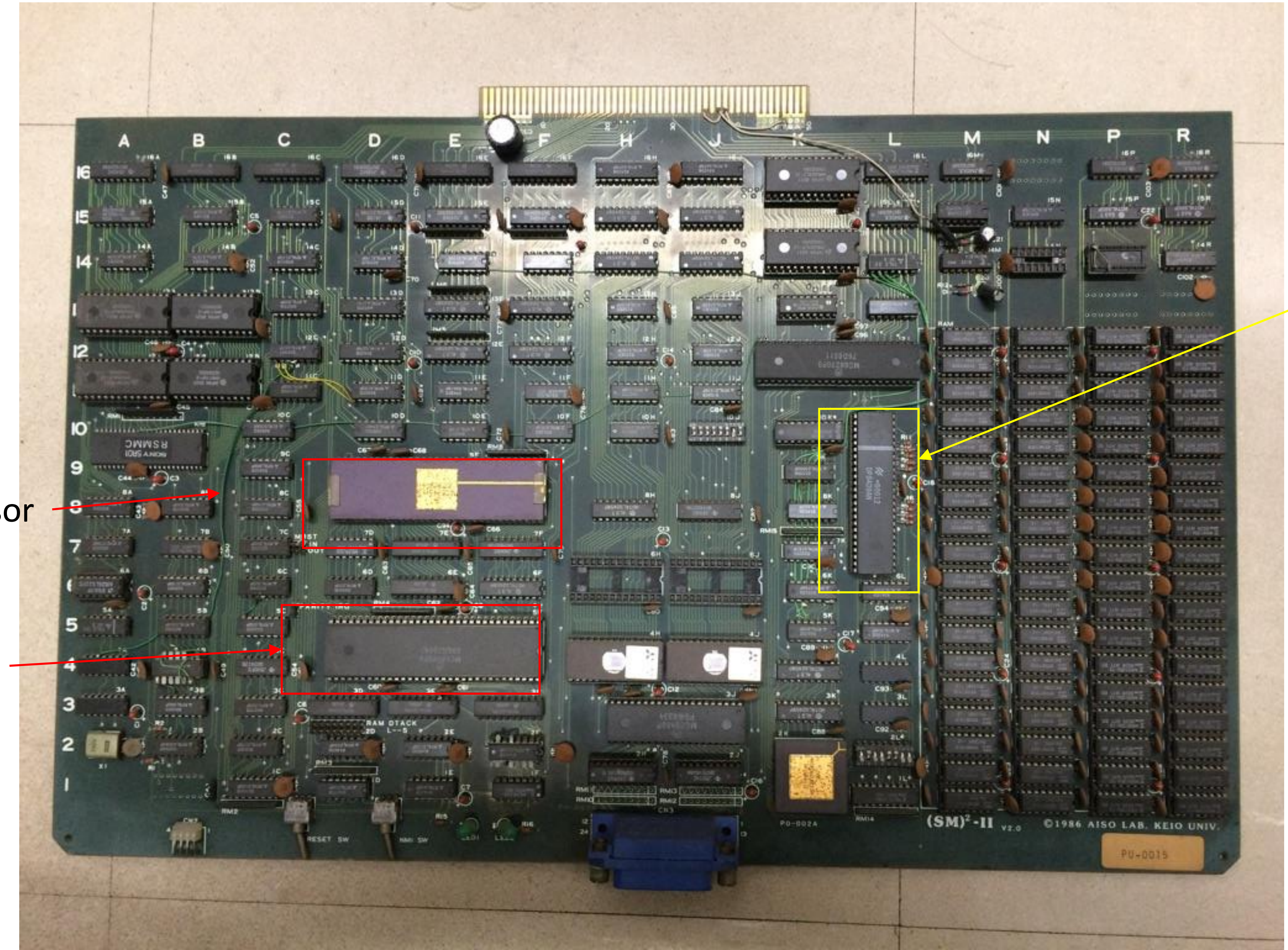
⇩

その時の自分の実装能力の限界が発想の限界となる

確かに、実装できた：(SM)2-II

Co-processor

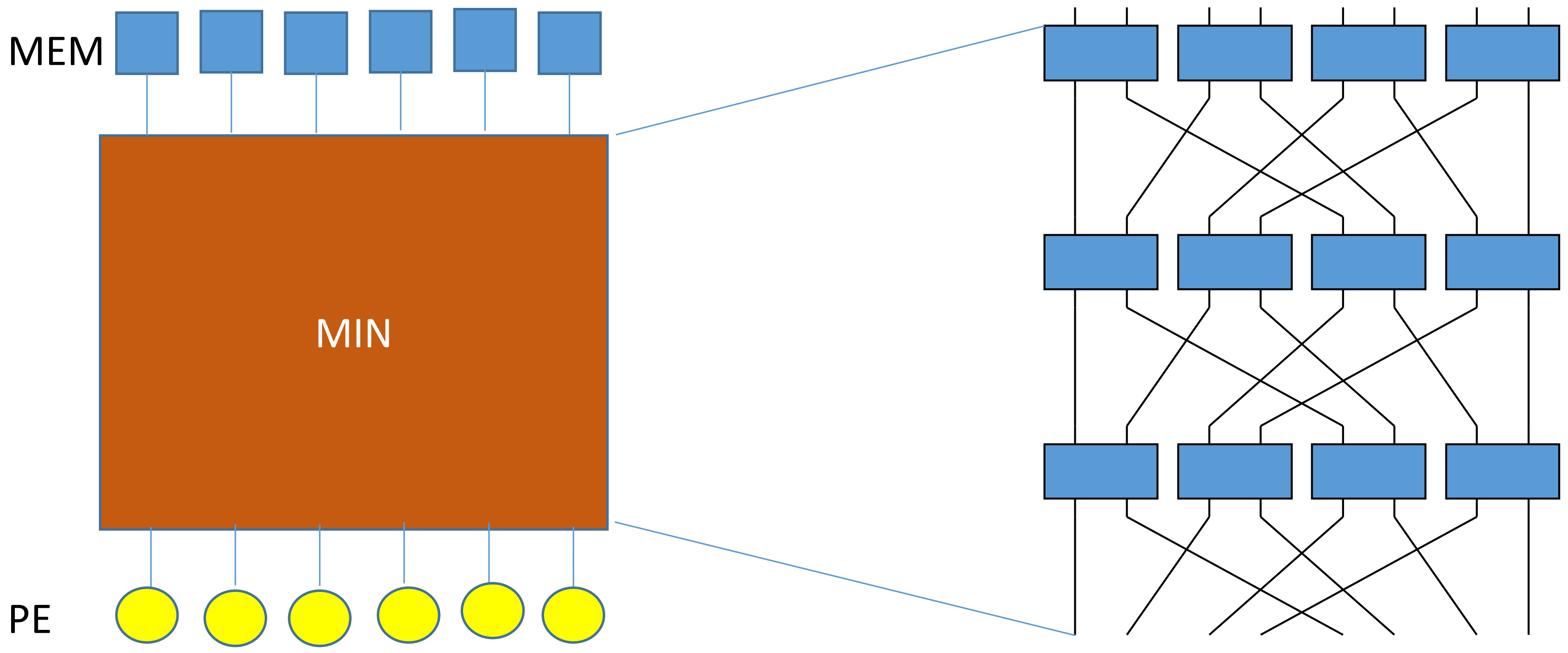
68030



RSMController
0.6μm Gate Array

ISCAが東京に来た
時にデモをやった

SSS-MIN (Simple Serial Synchronized) Multistage Interconnection Network 1989

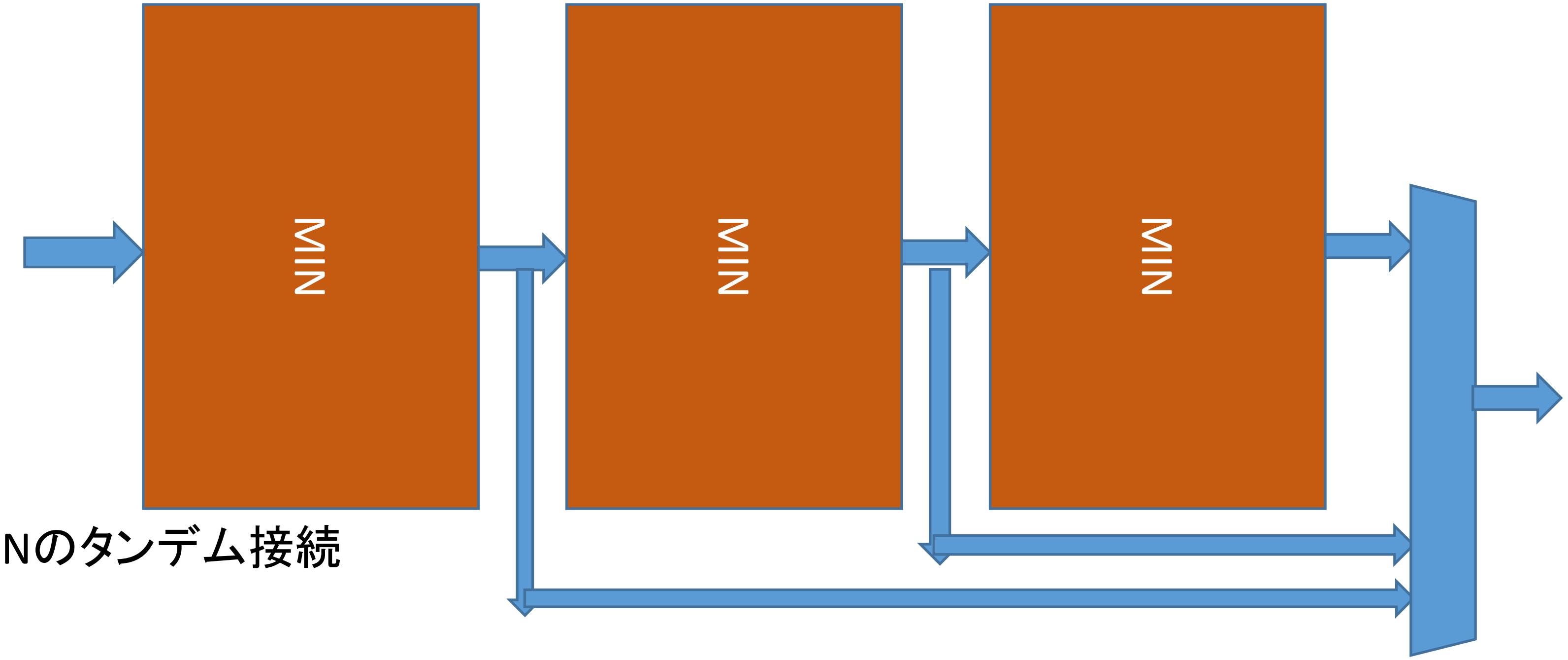


核となるアイディア

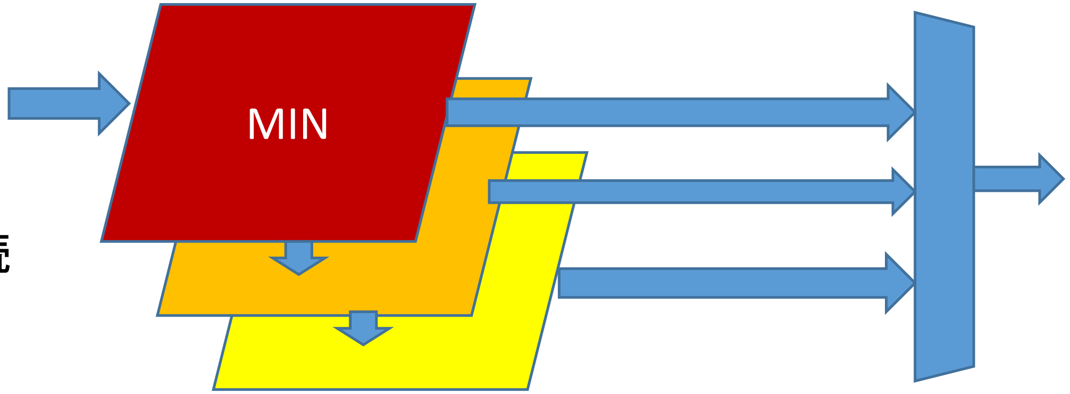
- パケットを数ビットのシリアルで送る
- 各スイッチングエレメントで全体の転送をパイプライン化する
- 各スイッチングエレメントは数ビットのバッファしか持たない
- 場合によってはステージを飛ばす
- パケットの通過経路を利用してACKを返す
- 全パケットを同期する

上記により、スイッチングエレメントを簡単にし、スイッチを高速化する

TBSF（Tandem Banyan Switching Fabrics)と PBSF(Piled Banyan Switching Fabrics)



TBSFはMINのタンデム接続



PBSFはMINの三次元接続

SSS-MIN　vs. Wormholeルーチング

- ATM交換機の影響→思想は類似
- スイッチングエレメントの構成、動作は類似

- SSS-MIN
  - パケットの同期
  - パケットのCombining
  - Virtual Channelが思いつかなかった
  - MINのみの方式
- Wormholeルーチング
  - パケットは独立に動作
  - どのような結合網にも利用可
  - Virtual Channelとの組み合わせ
  - 一般的パケットのフローコントロールへ

なぜSSS-MINはWormholeルーチングまで行けなかったのか？

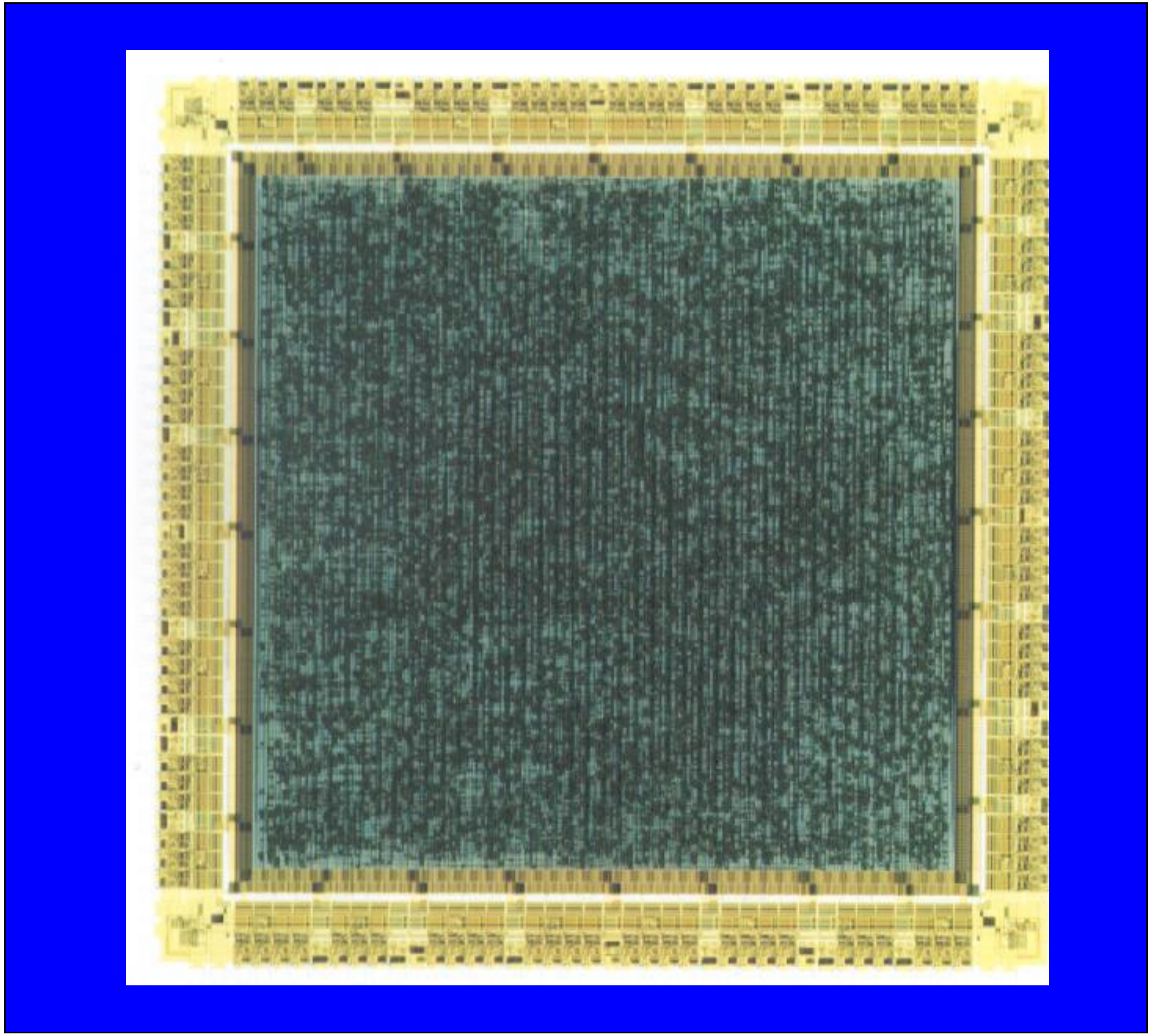
- 発想は類似していた
- MINとパケット同期にこだわった
  - 同期しないと実装がムリだ、と思った。
- パケット衝突、混雑をPacket Combiningで解決しようとしていた

→　Virtual Channelが思い浮かばなかった

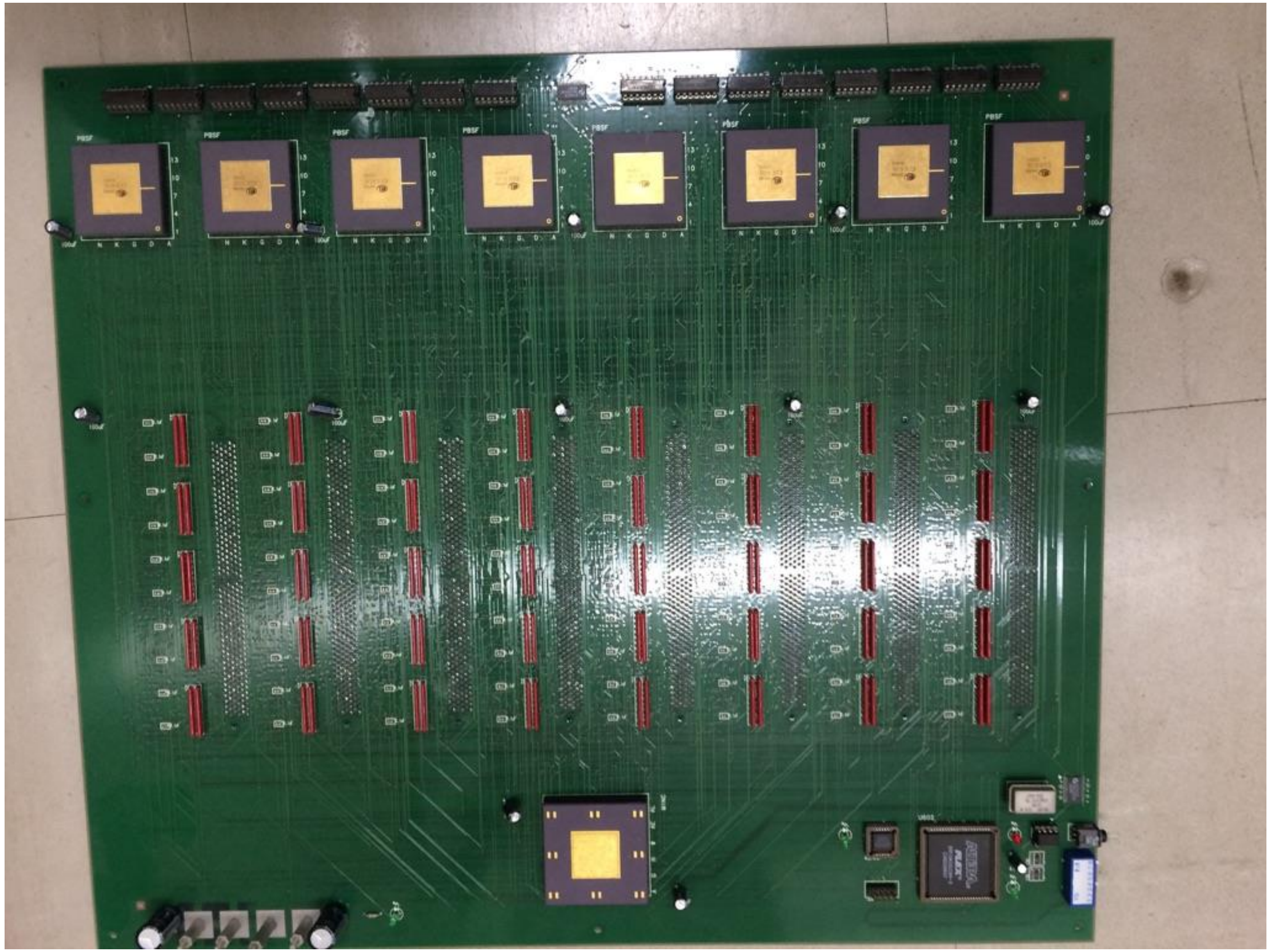
その時の自分の実装能力の限界が発想の限界となる

PBSFチップ

0.6µm
LPGA(Laser Programmable Gate Array)といってレーザビームで加工するGate Arrayを使っていた。



SNAIL-II



SNAIL-IIのマザーボード

PBSFチップ

教訓1

- 自分の実装能力が発想の限界になってはダメ
  - 自分が実装できそうもないからといって他人もできないとは限らない。
  - 汎用を目指そう。困難な実装も可能になる。
- 研究面では、大胆、高飛車、傲慢になろう
  - 専用に閉じこもった
  - 実装できそうなものしかしなかった
  - 行動面では大胆、高飛車、傲慢だった
    - ISCAなんか通って当たり前と思っていた
    - IEEE Trans. on Computersに通せばD論なんていい加減でいいと思っていた
  - 逆にするか、両方共高飛車の方がよかった

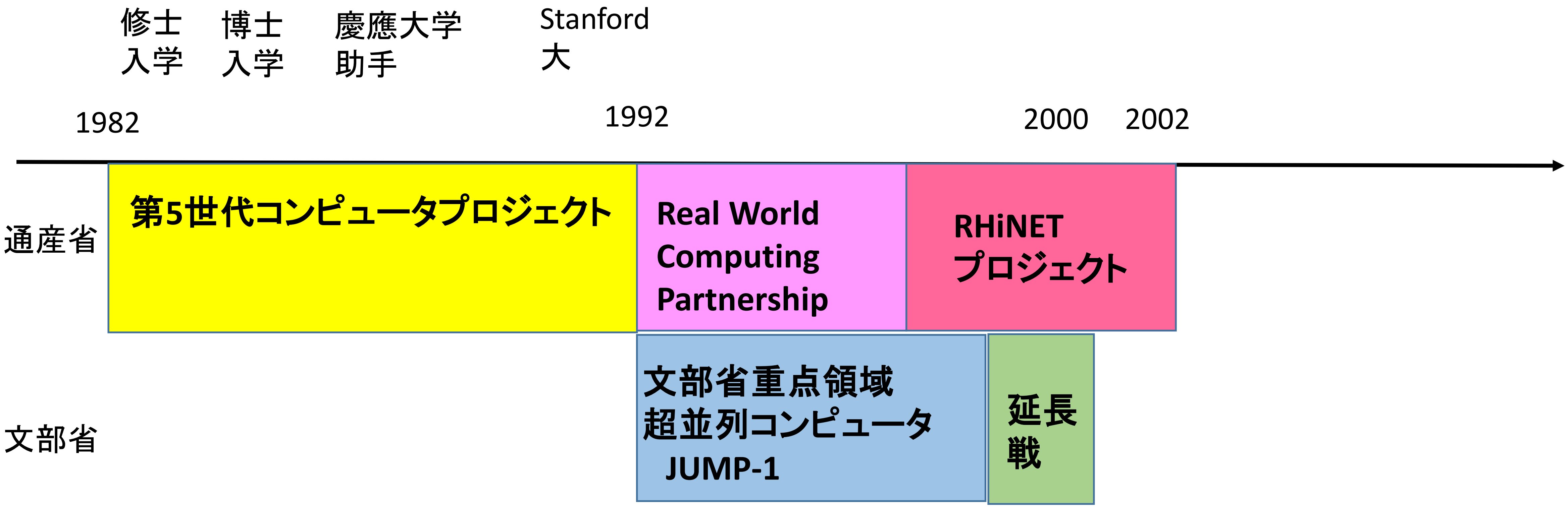
で、どうしたか？

- 一人でやっていると限界がある
- 発想の限界、資金の限界

そうだ、大型プロジェクトに参加しよう！

大型プロジェクトの時代

国内計算機産業の育成からの大転換



第5世代コンピュータプロジェクト

- 背景
  - IBM
- 目
  - タ(当時、LSIを 使
  - 論理型
- 手法
  - 言語、OS、アプリケーション、アーキテクチャの総合プロジェクト
  - 海外、国内大学への高い開放性
- 1982年から10年570億

論理型言語を
受け付けなかった
GHC講習会から脱走した

文部省重点領域超並列計算機プロジェクト

総括：田中英彦（東大）

- A領域：アプリケーション
- B領域：言語
- C領域：オペレーティングシステム
- D領域：アーキテクチャ、ハードウェア

  7大学共同で超並列計算機JUMP-1を開発
  - 東大：メモリコントロール用プロセッサ
  - 京大：キャッシュシステム
  - 慶應大、東京工科大：結合網
  - 神戸大、岡山理科大：I/O
  - 九工大：デバッグシステム

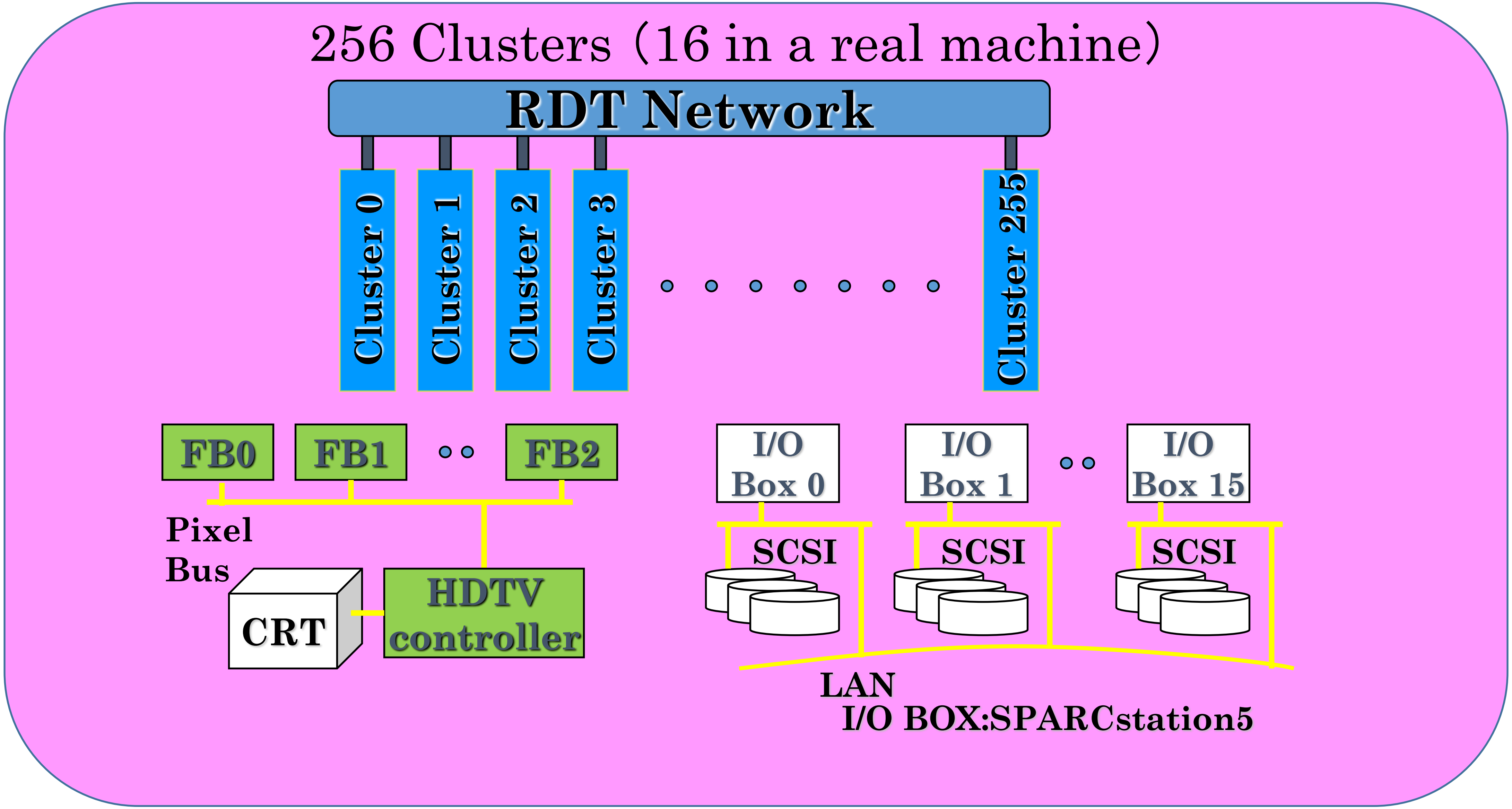
JUMP-1

- CC-NUMA（Cache Coh … …テクチャを探る
- Stanford DA…
- 研究・… …テクチャを恐れ…
- 問題点…
  - 予算が… …ない（6億円くらい）
  - 水平統合だったので、…

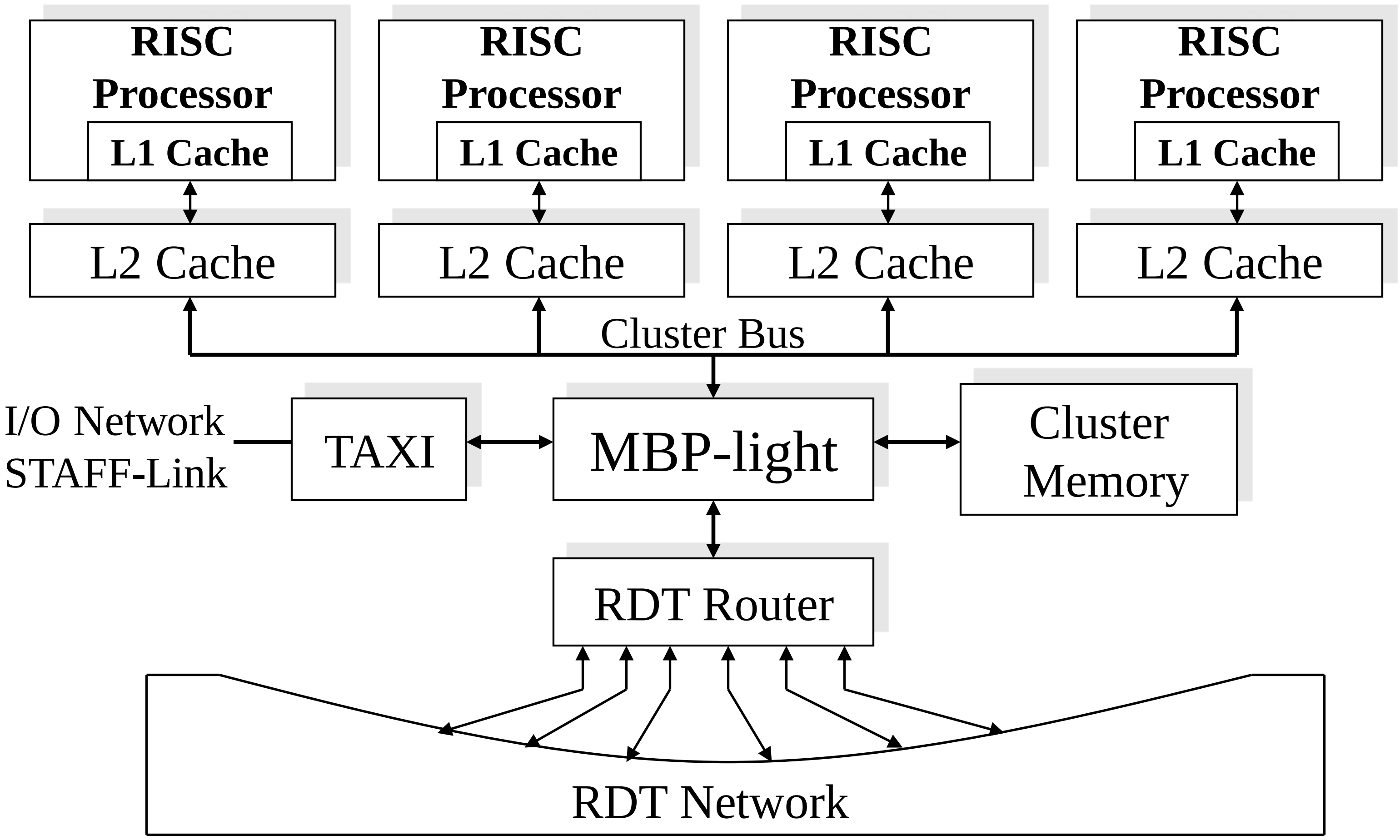
自分では実装できないと思ったが、結合網部分だけだからいいか、と思った

JUMP-1: massively parallel machine

CC-NUMA



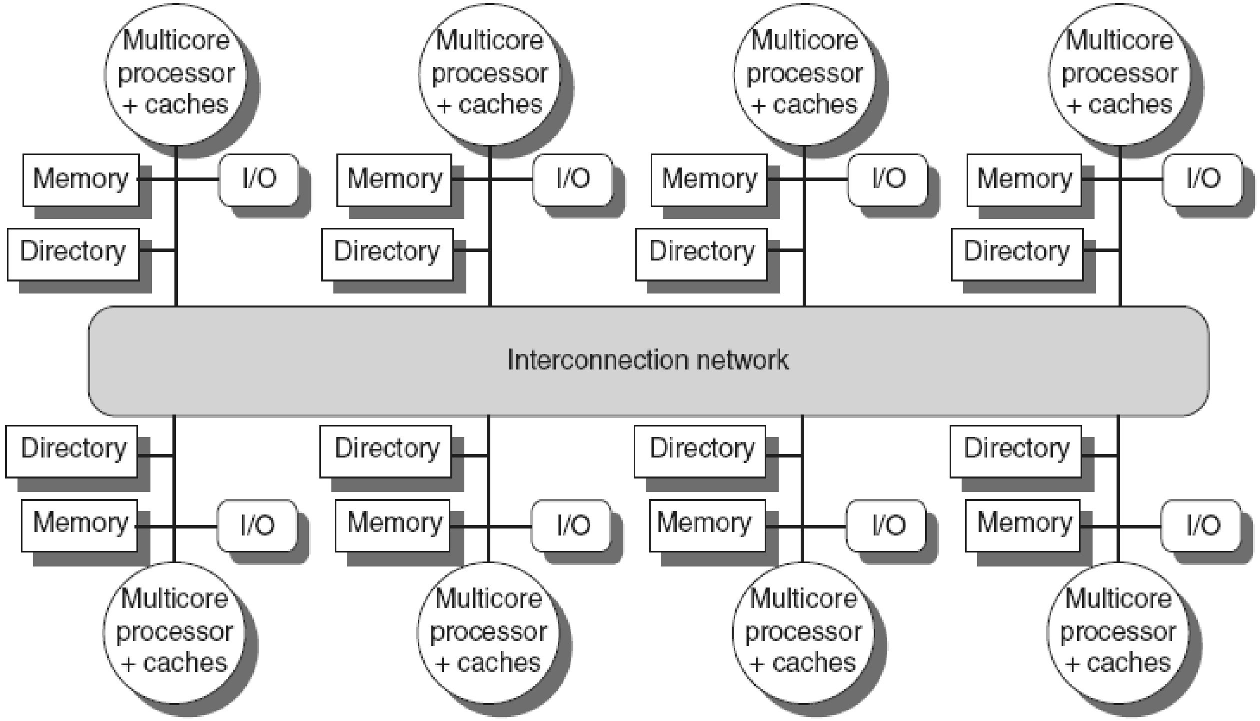
A cluster of JUMP-1



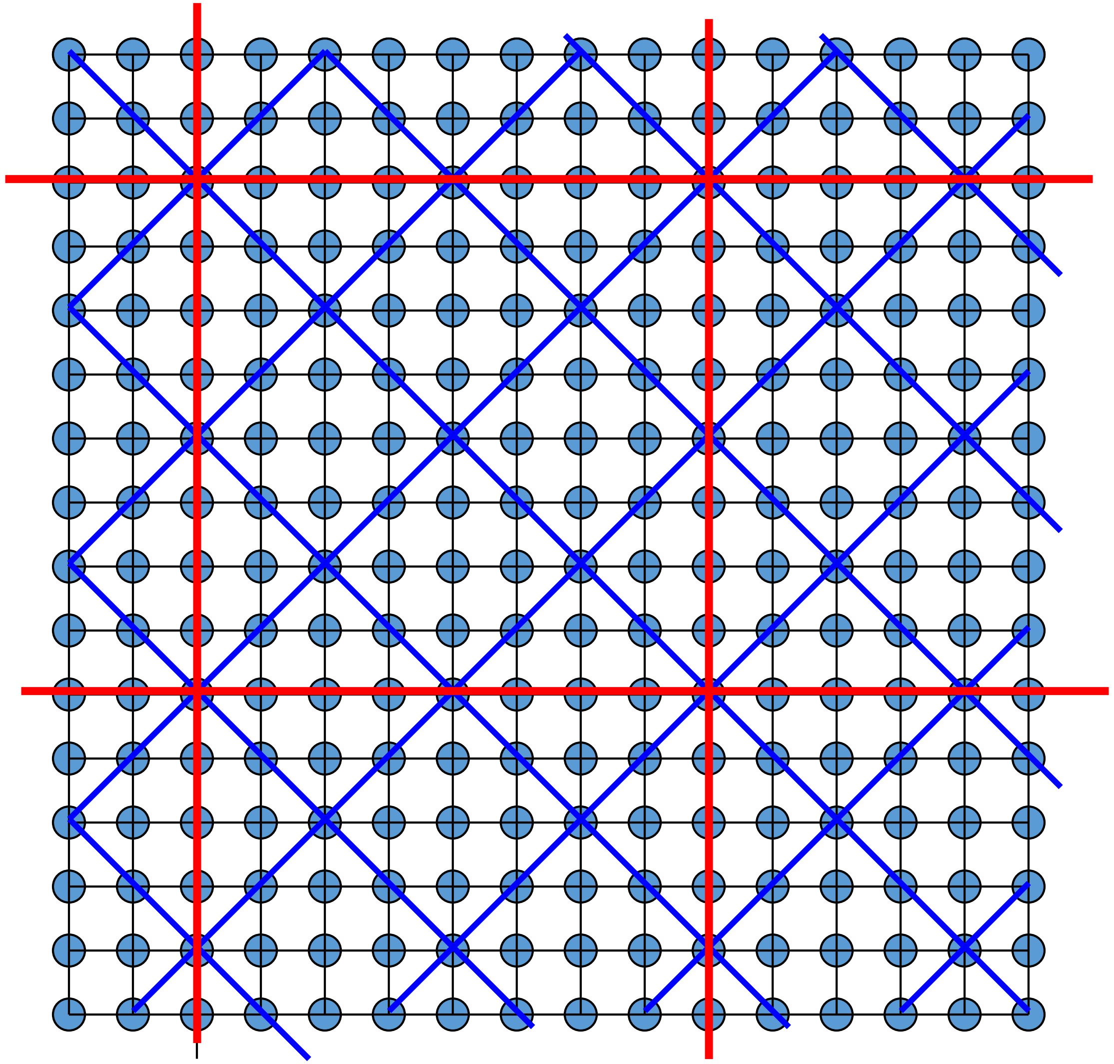
Multicore Based systems

- 現在のサーバーはほぼ同じアーキテクチャと言える

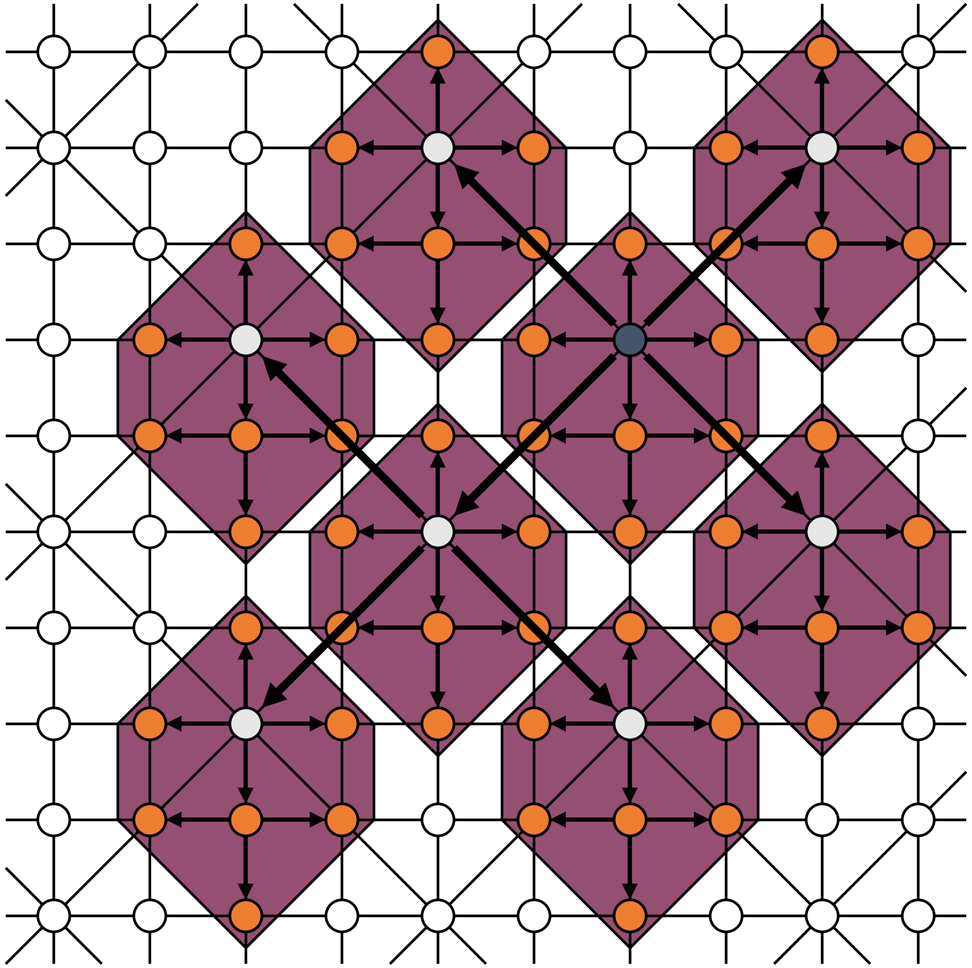
IBM Power 7
AMD Opteron 8430



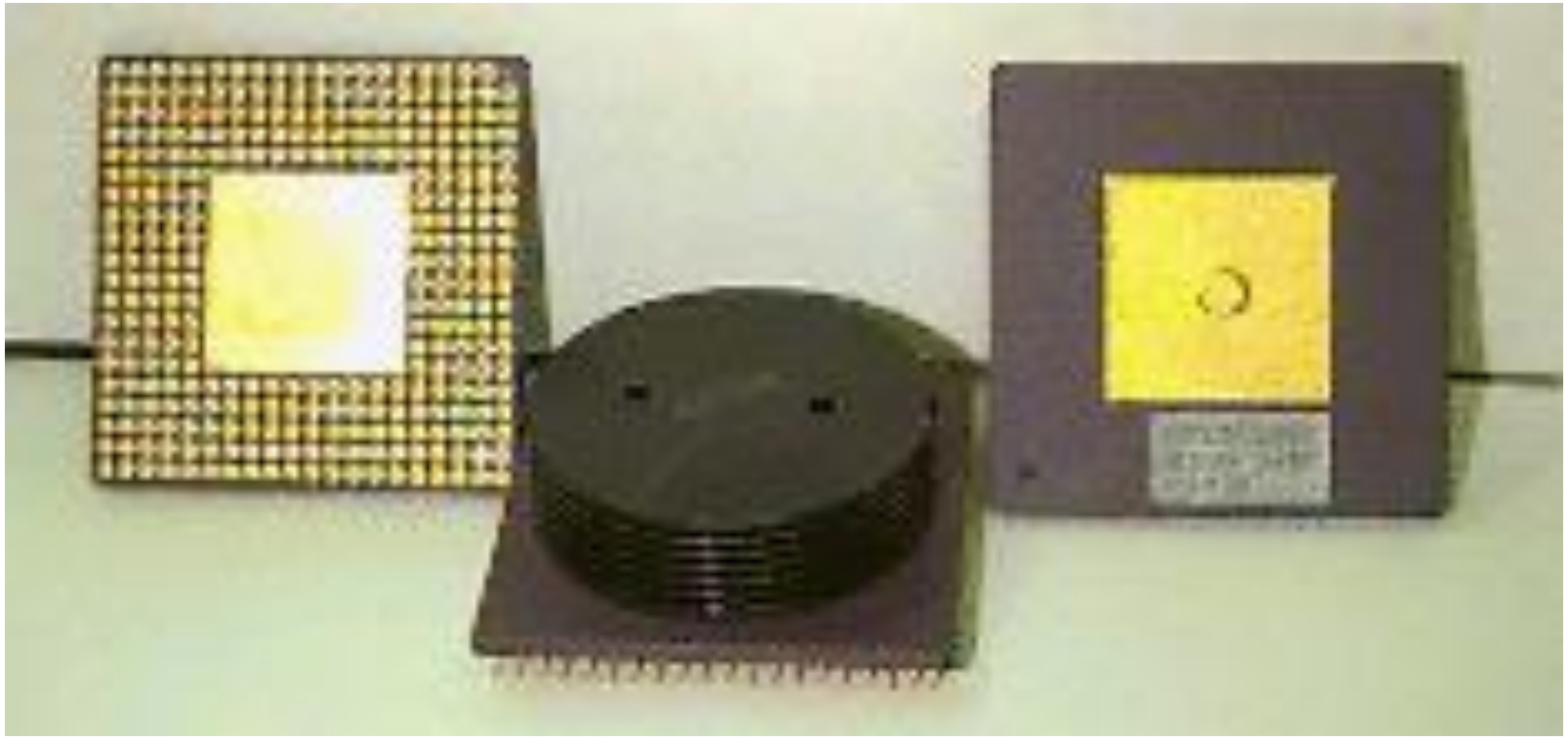
RDT(Recursive Diagonal Torus)



Multicasting on the RDT

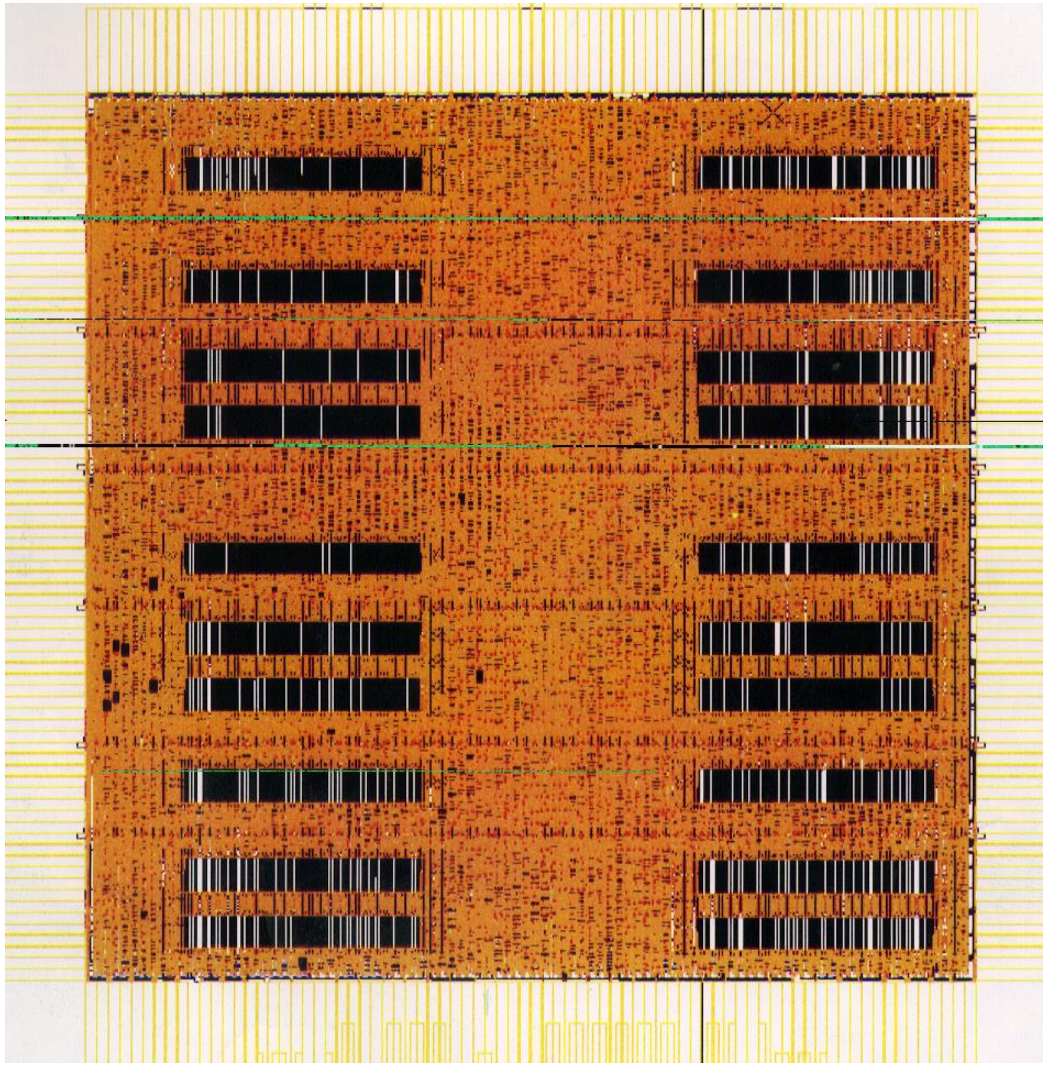


RDTルータチップパッケージ



VHDLとスケマティックの混合設計→　はじめてHDLを使ってチップを作った
日立のBiCMOSを利用→ファンアウトの大きい部分をバイポーラで駆動
比較的簡単に動作した。1995年3月にJump-1/2のデモをやった

ルータチップ内部レイアウト図



パケットバッファ用のメモリのIPを利用
他の部分はゲートアレイ

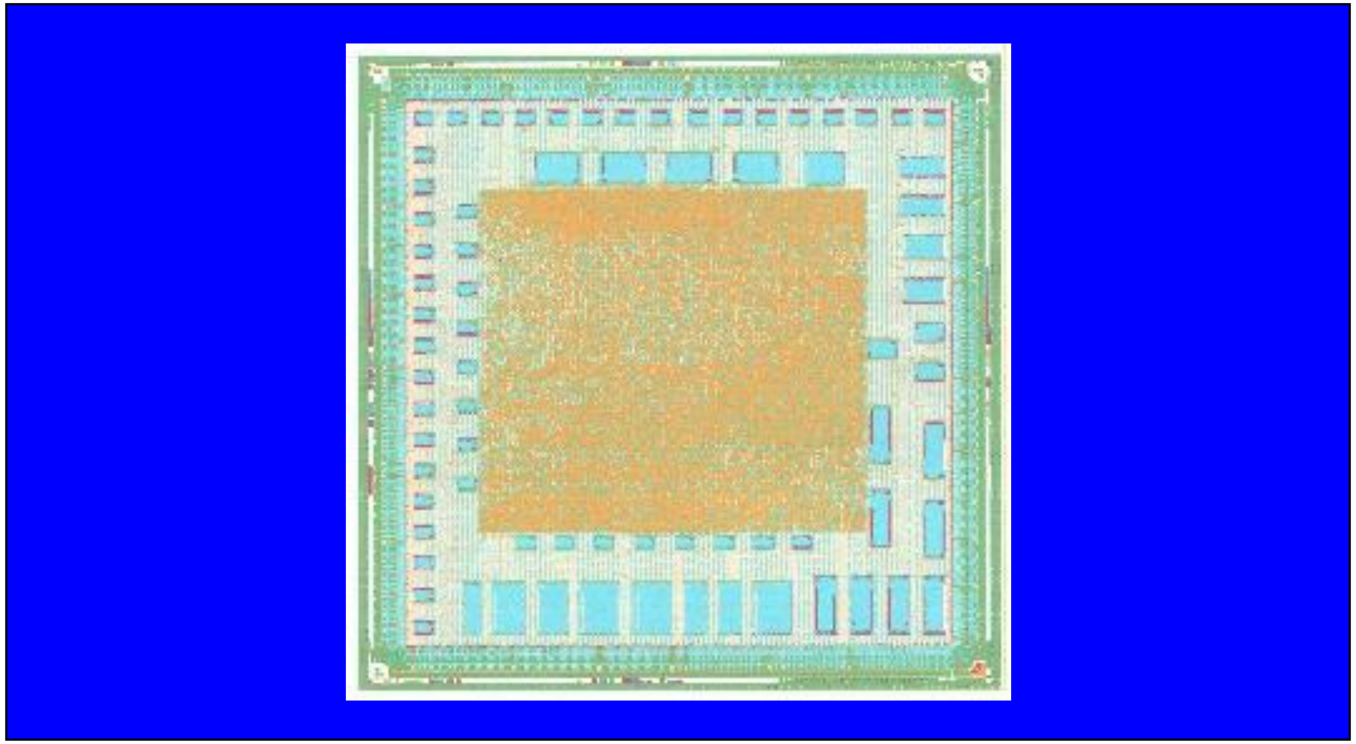
Embedded Array方式を利用

パケットバッファを入力毎に配置

JUMP-1開発の遅れ

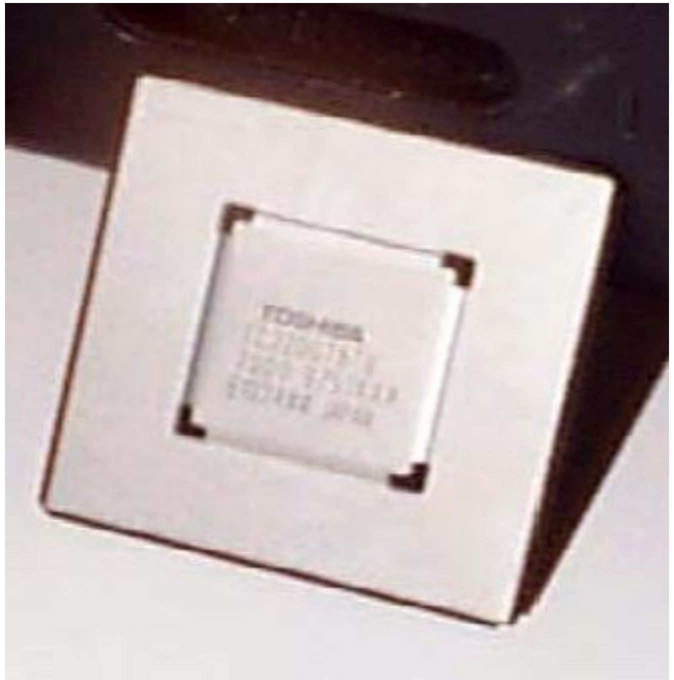
- RDTルータチップをあっさり稼働させてヒマになったが、MBPとキャッシュは手に負えないと思い、静観していた
- しかし、本当にいつまでたってもできなかった
- 複雑性が高すぎるのが問題だろう
  - 大幅に簡単化したMBP-lightを設計することにした
- 本当の開発の遅れの原因を認識していなかった
  - 恐ろしい魔物が控えていた　その名はVLCAD
- 東芝自社製CADと格闘して2年が経過、ようやくテープアウト
  - 後に攻略本を東芝に寄贈したが、数ヵ月後にこのCADは使われなくなった

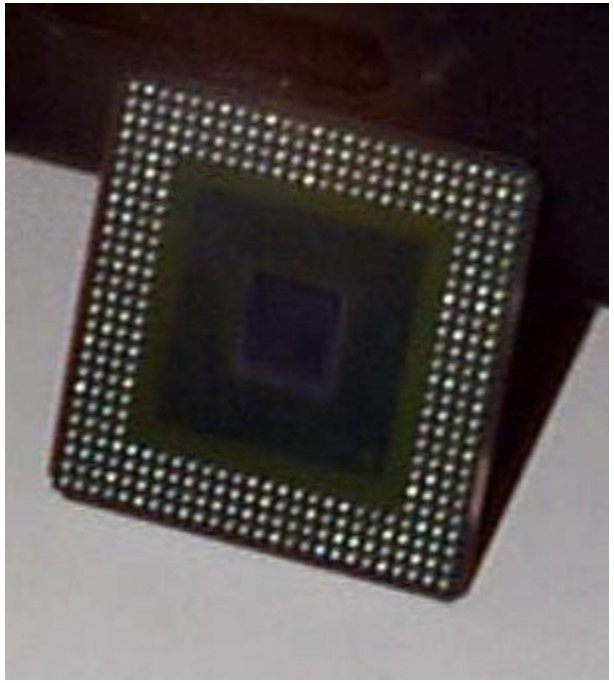
MBP-lightのレイアウト



これもEmbedded Array方式
MBP-lightはキャッシュ制御用バッファを無闇に使ったため配置ができず、やむなく周囲に置き、ランダムロジックを中央にした。

MBP-lightの外観





Ball Grid Arrayを使った。基板屋さんが慣れておらず、一部の半田ボールが浮いて非常にやっかいなバグが出た
チップ自体は非常に優秀だった

完成したJUMP-1の勇姿 2000年3月

A system with 4 clusters
(Keio Univ.)

A system with 16 clusters
(Kyoto Univ.)

当時、ムーアの法則は18ヶ月で2倍を越えて作用した。結果として出来た頃はその辺のWSの方が性能が高かった。

JUMP-1に漂う失敗感

- 出来上がった頃はハードウェアが完全に時代遅れになってしまった
  - SuperSPARC+　1992年に寄付されたのを使った
  - 動作周波数は50MHz-100MHz
  - 微妙なバグが残っていて、64台動作は中々大変
- そうは言っても方向性は正しく、グランドデザインは良かったので、もっとプロジェクトの成功と成果のアピールをすれば良かった
  - マルチコア、メニーコア時代への足がかりがもっとできた
  - 遅れたこと、稼働時の性能が低かったことにより首脳部が失敗感をいだく

    →黒歴史化

    →抹殺
  - 論文はたくさん通したのだが、、IEEE Trans. Parallel & Distributed Systems、ICPP他
- 1900年代のユニプロセッサの性能向上を目の当たりにした日本の並列計算機アーキテクトの自信喪失
  - もう3年がんばれば、Intelが方針転換したのに、、、

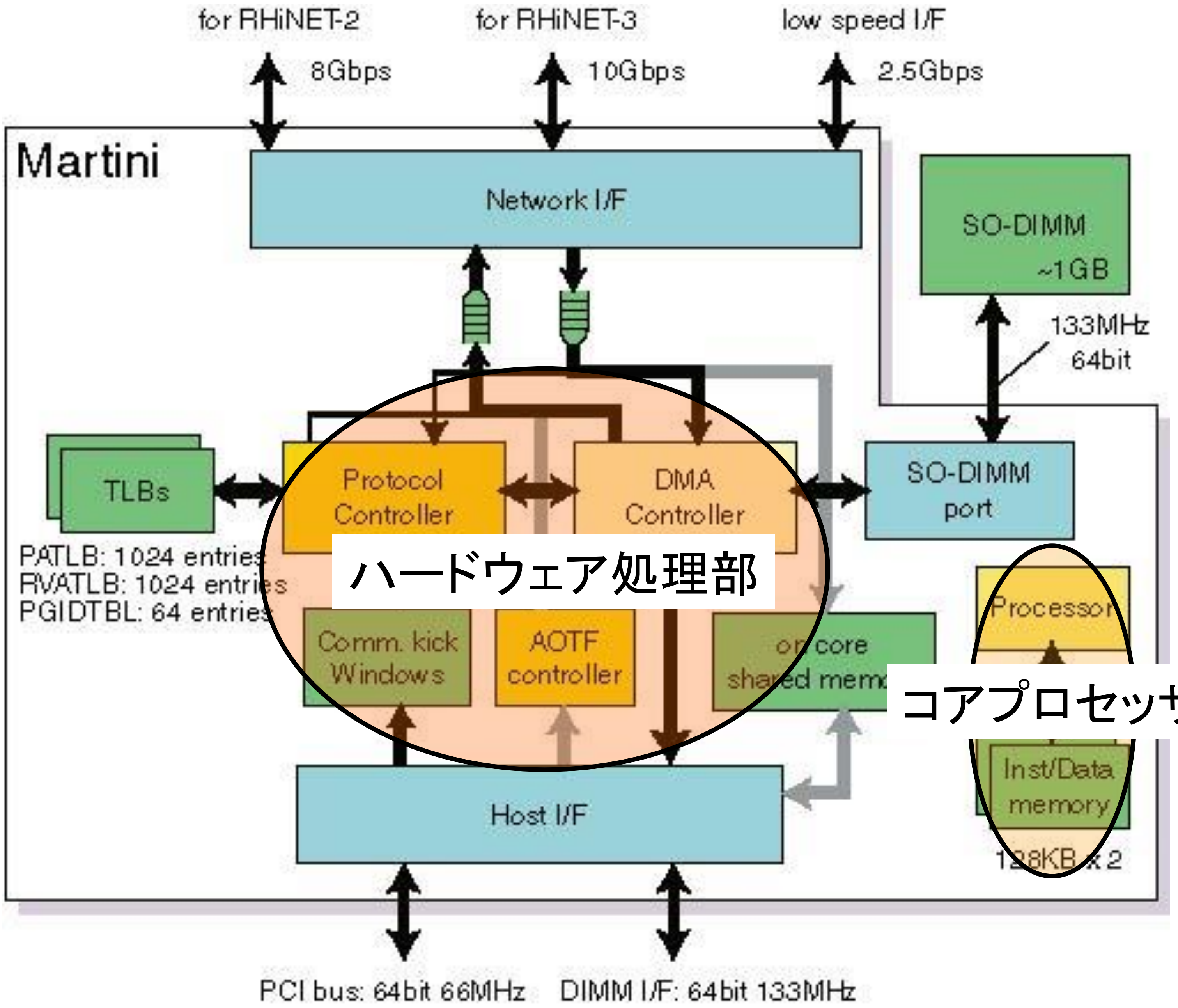
教訓

研究用プロトタイプはスピードが命。首脳部が黒歴史化する前に動かそう。意地で実装を続けるのは周囲の迷惑だから止めよう

- ベターな選択
  1. もっと早くMBP-lightの設計に取り掛かる
  2. VLCADの実態が分かった時点でプロジェクトを終了

RHiNETプロジェクト

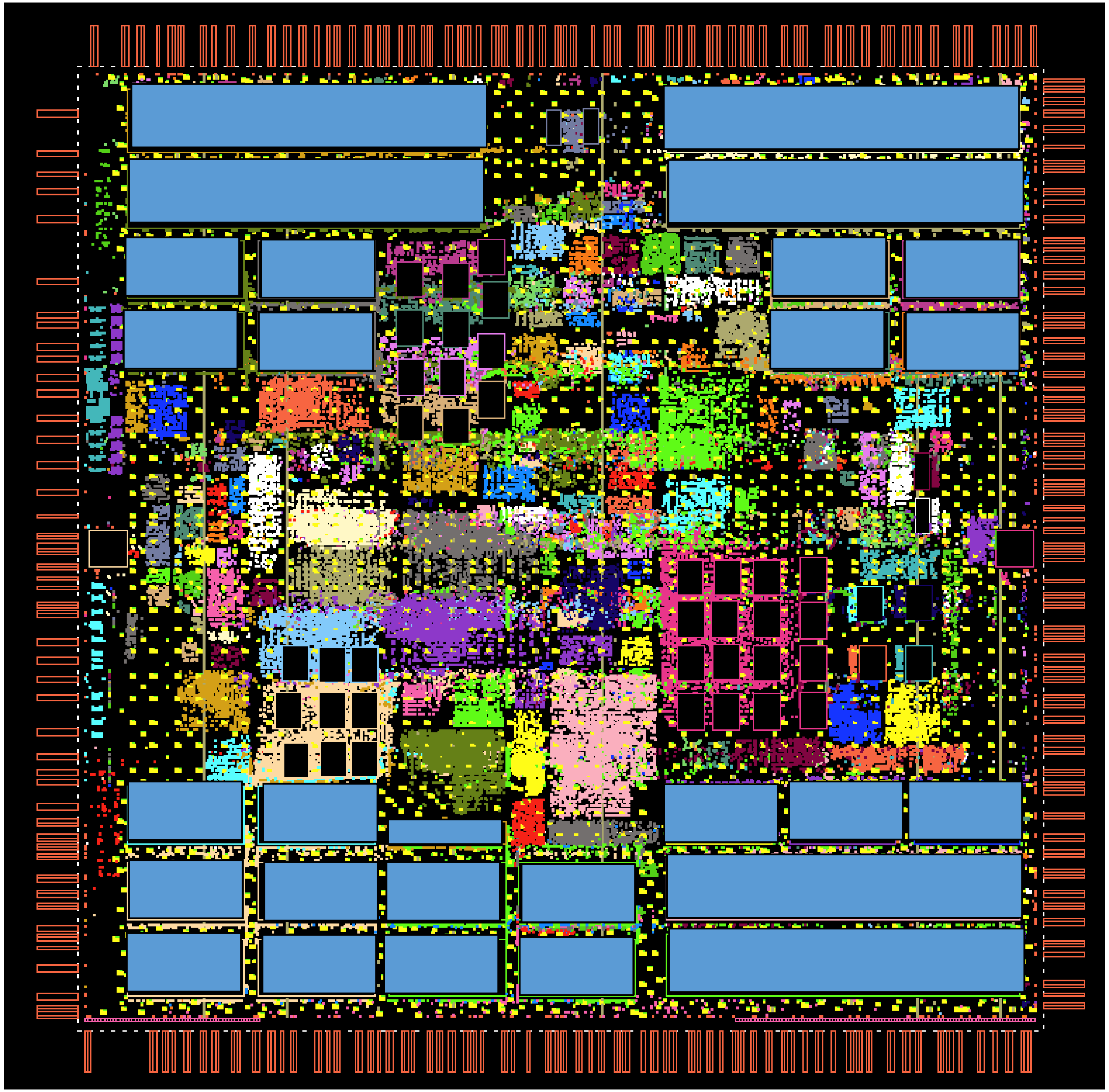
- RWCP（Real World Computer Partnership)
  - 超並列計算機RWC-1を作っていたが、、、
  - JUMP-1の苦闘を横目にあっさり諦め、クラスタコンピューティングに方向転換
  - RHiNETプロジェクト
    - コモディティからなるノード
    - 光ネットワーク
    - 専用HW＋プロトコルコントローラからなるNetwork Interface
  - 研究の方向性、グランドデザインは正しい
  - 設計環境も悪くない：企業の独自CADから標準CADへ



for RHiNET-2
8Gbps
for RHiNET-3
10Gbps
low speed I/F
2.5Gbps
Martini
Network I/F
SO-DIMM
~1GB
133MHz
64bit
TLBs
Protocol Controller
DMA Controller
SO-DIMM port
PATLB: 1024 entries
RVATLB: 1024 entries
PGIDTBL: 64 entries
ハードウェア処理部
Processor
Comm. kick Windows
AOTF controller
on core shared memory
コアプロセッサ
Inst/Data memory
128KB x 2
Host I/F
PCI bus: 64bit 66MHz
DIMM I/F: 64bit 133MHz

Martini の諸元

デザインルール		0.14μm
ダイサイズ		272.91mm^2
メモリ総量		538KB
I/O 伝送周波数		
	RHiNET-2,3/SW	800MHz
	OIP-SW	250MHz
内部動作周波数		
	コア部	66MHz
	DIMM ホストI/F	133MHz
	スイッチI/F	125MHz
パッケージ		784 BGA



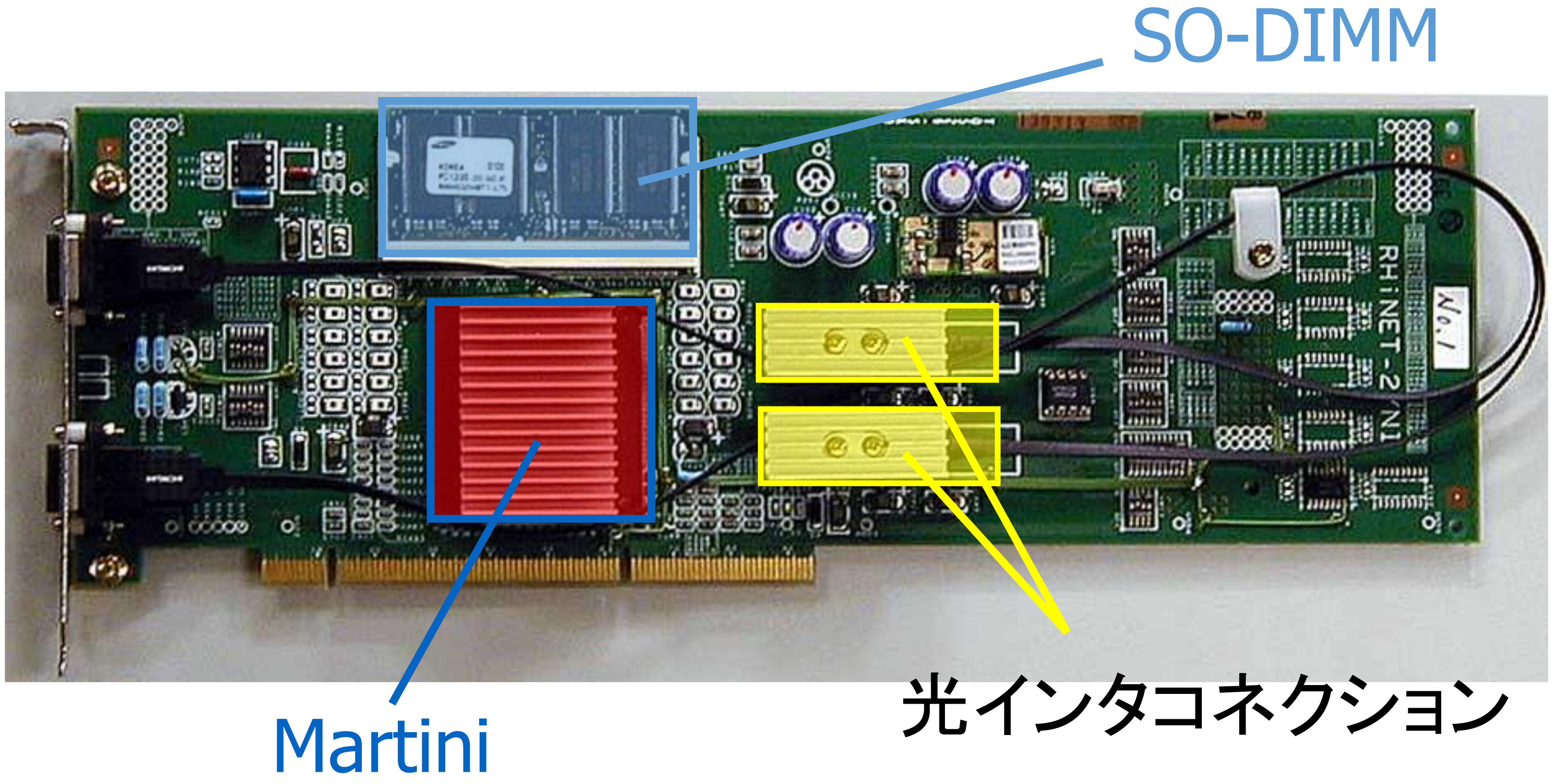
多分Embedded Array方式の最後の時期

MartiniもバッファRの塊なので、メモリのIPが多数

なぜかカラフルなのは日立デバイスセンターの秘密保持のため

ASIC版 NIC

- コントローラに専用ASICを開発
- PCI bus/ DIMM slot



ASICを用いて製作したPCクラスタRHiNET

RHiNET-1/2

- RHiNET-1/2 SWは高速動作を実現（世界最速を称していた）、高速光リンクと共に高く評価された
- 遅れはしたが、JUMP-1ほどではなかった
  - RHiNET-2は稼動に成功
  - しかしバグが出て動作が不安定に（これは僕が出したバグ、すいません）
- 論文はやはりたくさん通した
- プロジェクトとしては成功だったのでは？

問題点：

光リンクのコストが高すぎた　ノードのPC自体を上回る

MartiniのEmbedded Arrayの性能がいまひとつ出ない

ライバルのMyrinet、Infiniband, GbitEtherを使ったクラスタに勝てない

教訓

汎用化、国際標準を狙い王道を行き過ぎるとライバルが強すぎる

2002年頃、大型プロジェクトの時代も終わり
深刻な悩みが訪れた

このままやってちゃ
まずいんじゃないかい？？？

このままやってていいんだろうか？

- 今までの研究、開発の仕方
  - CPUにはCOTを使う（68000シリーズ、SPARC、Pentium 3）
  - 交信用メモリコントローラ（RSM、MBP-light、Martini）、スイッチ、ルータ（TPSF、PBSF、RDT）をGate Array、Embedded Arrayで開発、ここにOriginalityを入れる
  - 大規模汎用高性能システムを指向する→実際はあまり高性能ではなかった
- Intel/AMDの動作周波数は止めどなく上がっていった
  - いつかは止まると思っていたが、当時確信は持てなかった
  - 高速化に付いて行けない→高性能システムを手作りするのがムリだ！
  - PCIバスに繋いだり、メモリに繋ぐ（DIMMnet）のが大変だ。
- Embedded Arrayは性能が出ない。もう作れない。HDLテープアウトはできなくなった。
- いままで、色々作った。一流学会に論文をたくさん通した
  - でも企業の製品とは結びつかない
  - 研究の流れを作ってないのでは？？

戦略の変更

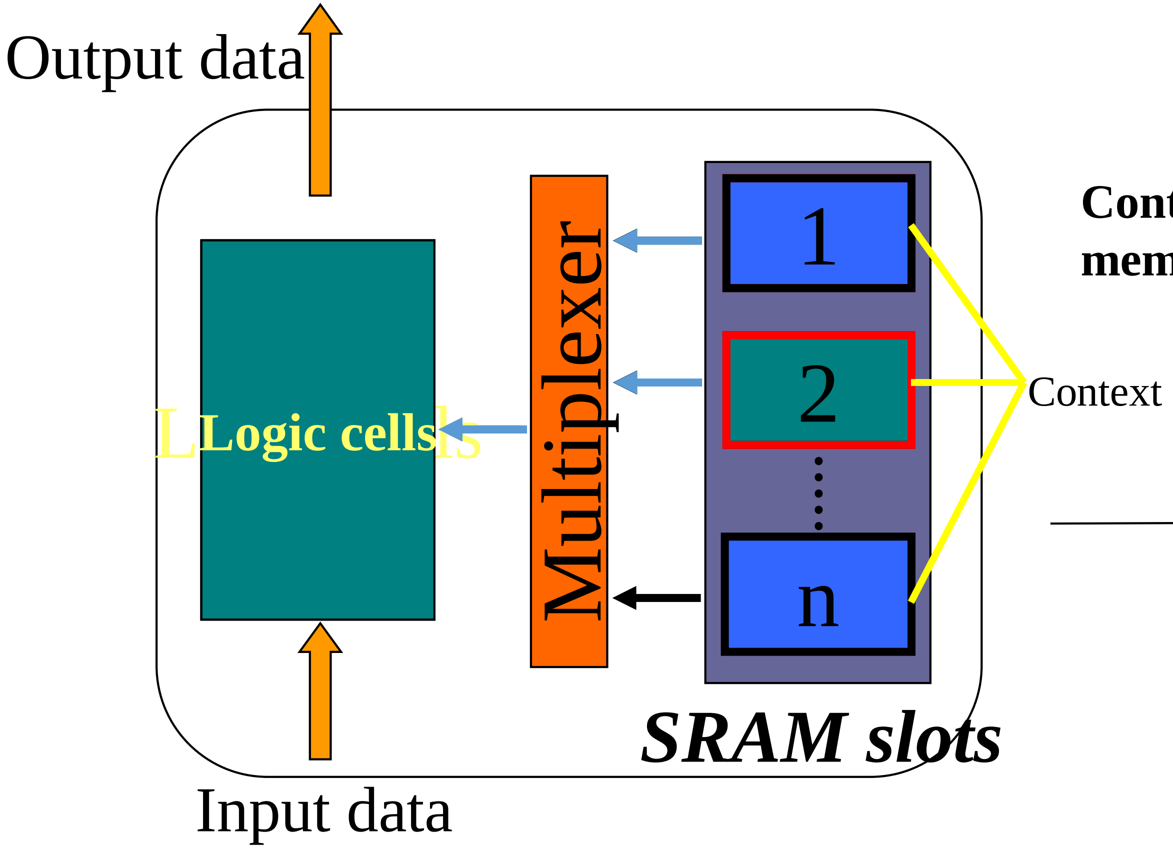
- 組み込みシステム用小規模、低電力指向
- リコンフィギャラブルシステム、動的リコンフィギャラブルシステム
- しばらく自分でチップを作るのは止める
  - VDECにより自分でレイアウトして作れるように技術を磨く
- FPGAボードは作っていく
  - ReCSiPプロジェクト
- 企業と組む
  - NEC（ルネサス）のDRL、DRPプロジェクト
- アーキテクチャを越えた範囲での連携（他人頼み）
  - 細粒度PG、誘導結合TCI、SOTB

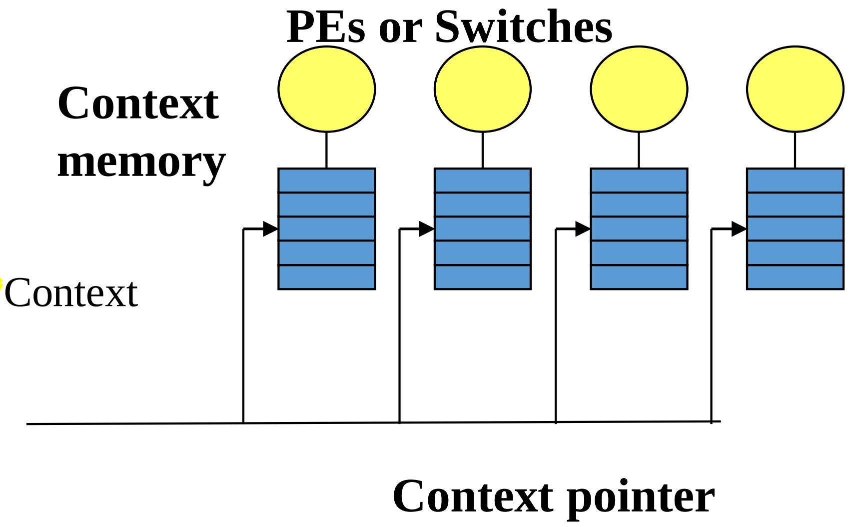
WASMII (What a Stupid architecture It is!　1991)

マルチコンテキスト構成

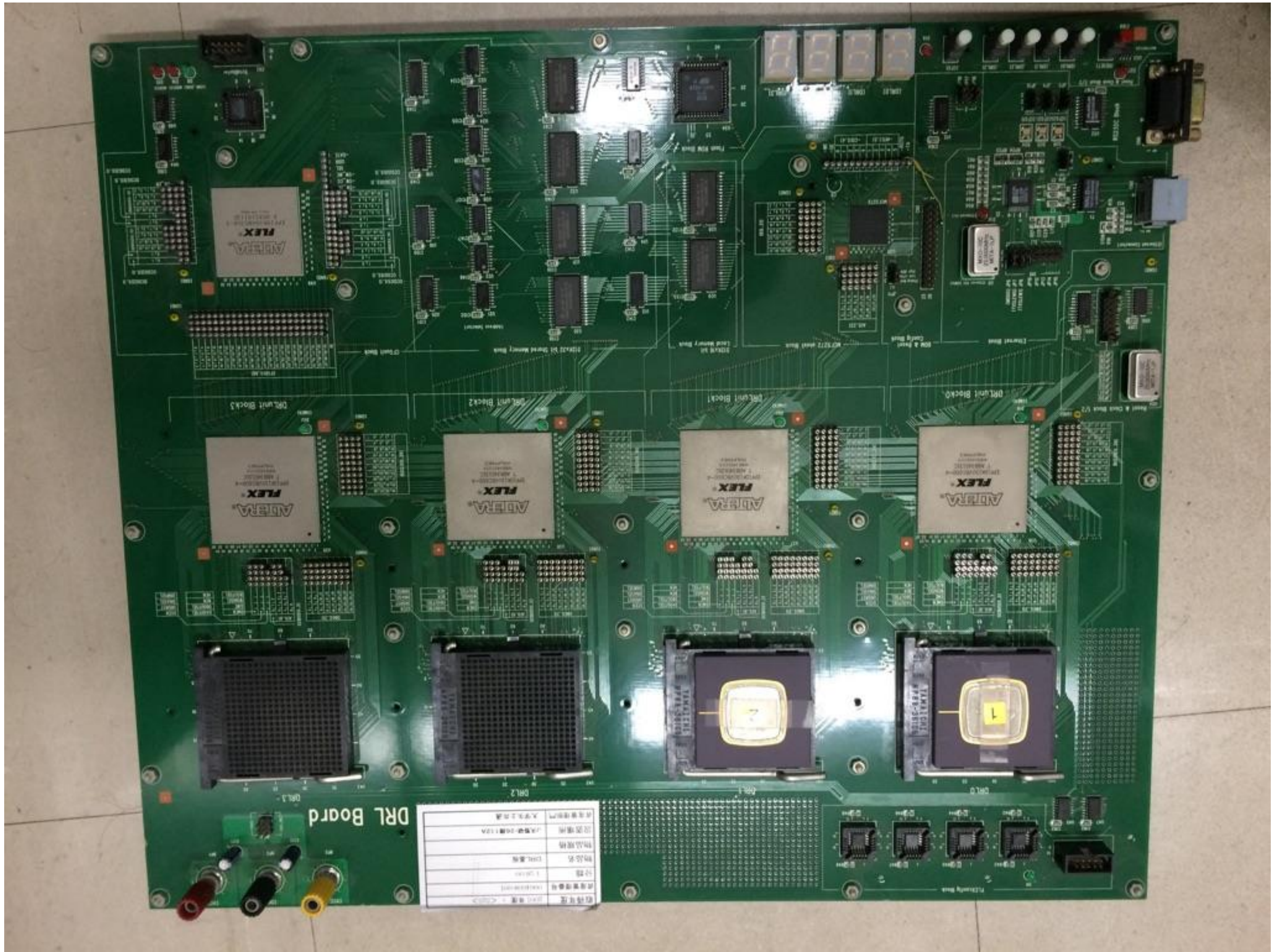
データフロー制御に基づくハードウェアコンテキストの切り替え

Xilinxの特許(1997年)に相当先んじる→先行発表、技術自由化戦略を採用





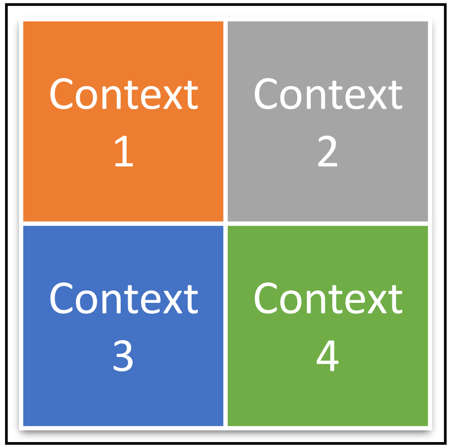
NECのDRL(Dynamically Reconfigurable Logic)と連携



NECのDRLチップを使った
WASMIIのエミュレータ

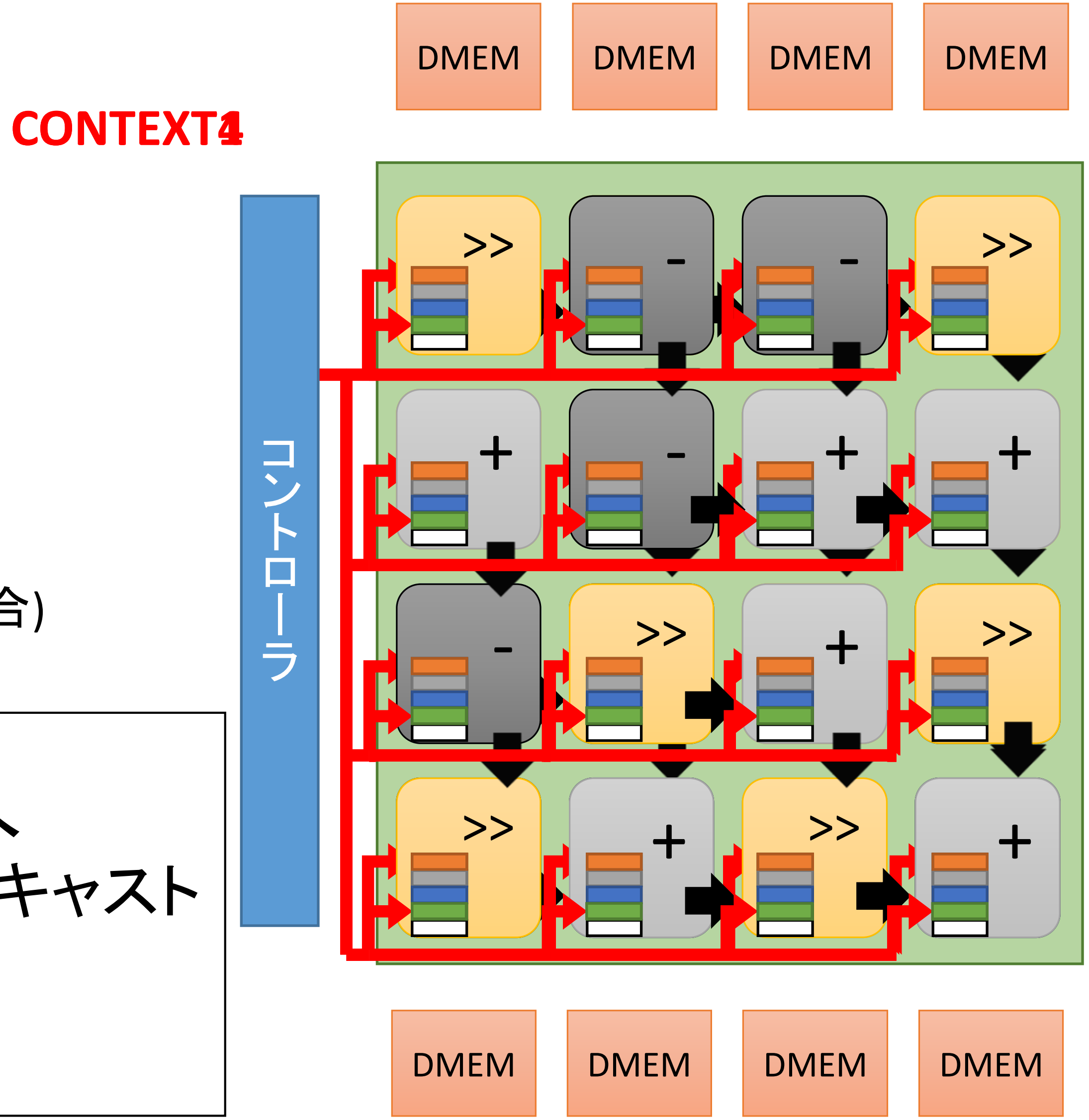
しかし、ここで細粒度は
ダメなんじゃないか、という
ことになった。
面積効率＋特許問題
Coarse GrainのDRPへ

動的再構成プロセッサ



専用ハードウェアの機能を分割

コンテキスト(回路の構成情報の集合)



1. コントローラが全てのPE(Processing Element)へContext Pointerをマルチキャスト
2. 構成情報を読み出す
3. 構成情報に従い再構成

動的再構成プロセッサのプロジェクト (2009 ASP-DAC Panel)

Product	Vendor	Context	Data	PE
D-Fabric	Panasonic	Deliver	4	Homo
Xpp	PACT	Deliver	24	Homo
S5/S6 engine	Stretch	Deliver	4/8	Hetero
CS2112	[illegible]	[illegible]	[illegible]	Homo
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
ADRES	[illegible]	[illegible]	[illegible]	Homo
FE-GA	Hitachi	Multi-C	16	Hetero
For Car-tuners	SANYO	Multi-C(4)	24	Homo
FlexSword(SAKE)	Toshiba	Multi-C(4/16)	16	Homo
Cluster	Fujitsu	Multi-C	16	Hetero

絶頂期！日本企業のほとんどが動的リコンフィギャラブルプロセッサを検討

動的再構成プロセッサのプロジェクト
Shonan Meeting [2012]

Product	Vendor	Context	Data	PE
D-Fabric	Panasonic	Deliver	4	Homo
Xpp	PACT	Deliver	24	Homo
S5/S6 engine	Stretch	Deliver	4/8	Hetero
CS2112	[illegible]	[illegible]	16/32	Homo
DAPD[illegible]	[illegible]	[illegible]	[illegible]	Hetero
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Kilocore	[illegible]	[illegible]	[illegible]	Homo
ADRES/SRP	IMEC→Used in [illegible] smart phone	Multi-C（32）	16	Homo
FE-GA	Hitachi	Multi-C	16	Hetero
For Car-tuners	SANYO	Multi-C(4)	24	Homo
FlexSword(SAKE)	Toshiba	Multi-C(4/16)	16	Homo
Cluster	Fujitsu	Multi-C	16	Hetero

半導体不況
SoCビジネスの崩壊
日本半導体産業の壊滅

教訓

企業に頼れば企業の業績次第に

しかし、これについては
しくじったと思ってない

超低消費電力領域で巻き返し
が可能　CMA-SOTB

チップ開発が再び可能に！

- Embedded Arrayの絶滅と共にHDLテープアウトは極端な金持ち以外は不可能となった→チップのレイアウト設計をやらなければ、、、
- しかし、アーキテクチャ研究者がレイアウトまでやってチップを作ると努力に応じた業績に結びつかない
    - 2000年初頭のAstroの環境は酷かった
    - 状況に

コンピュータアーキテクチャ屋がチップを作れる環境が再来

- 2008年頃
    - IC CompilerとSOC Encounterのレベルが上がる
    - e-shuttle 65nmの登場

しかし、スピードでは勝てない

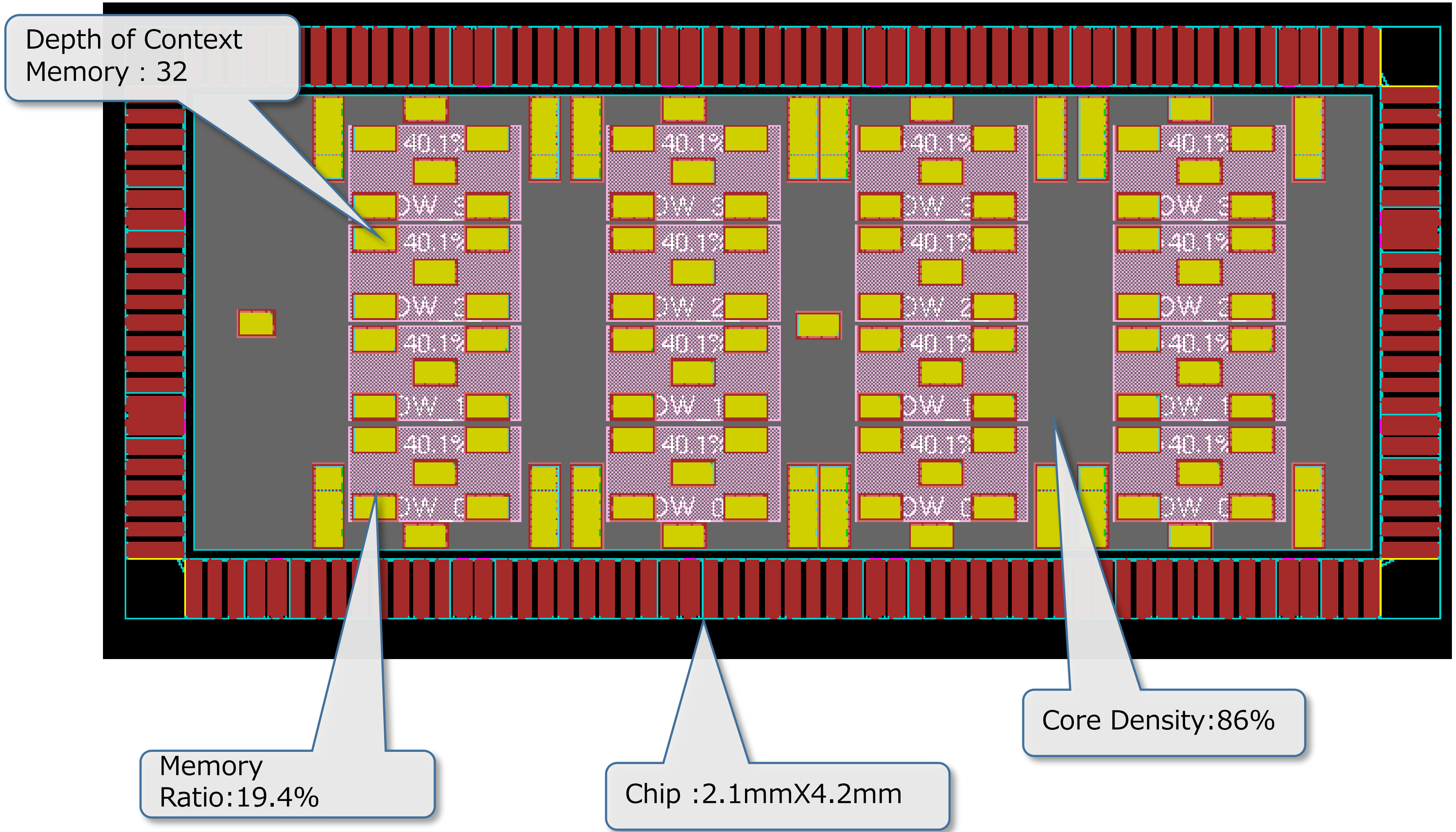
- 高性能システムはFPGAボードでいいじゃないか！
  - ReCSiPボード、PEACH
- 消費電力評価→FPGA、GPUには圧勝できる
- 超低電力チップで世界一を目指す
  - 比較が難しいが、ある程度は行ける
- 純粋ディジタルでない技術との組み合わせ
  - [illegible]

もはやちょっとやそっとの
しくじりでは動じない

そもそも回路よ[illegible]ないチップ設計をやらざるを得ない

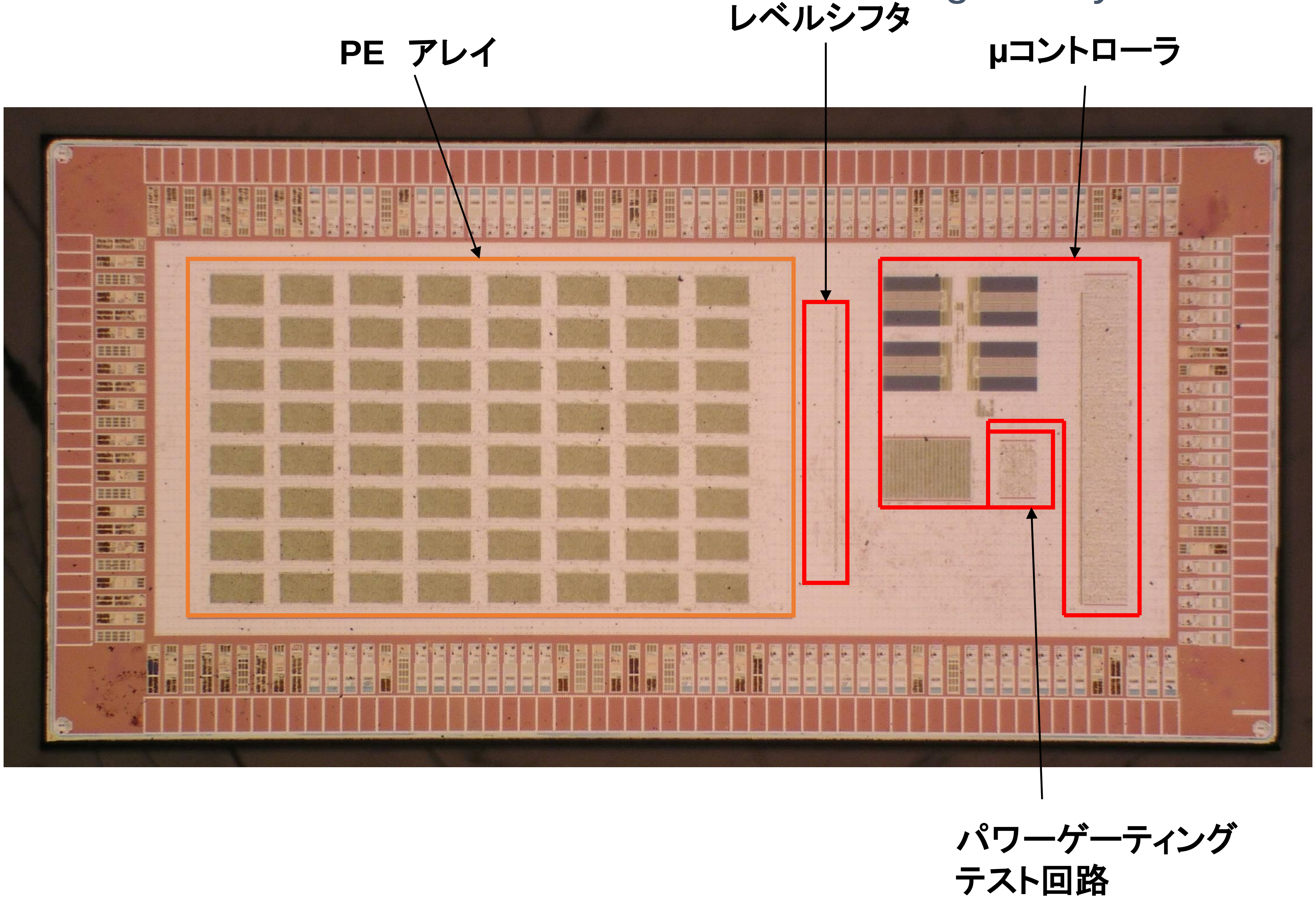
結果として失敗率が飛躍的に上がった！

低消費電力動的リコンフィギャラブルプロセッサ MuCCRA－3

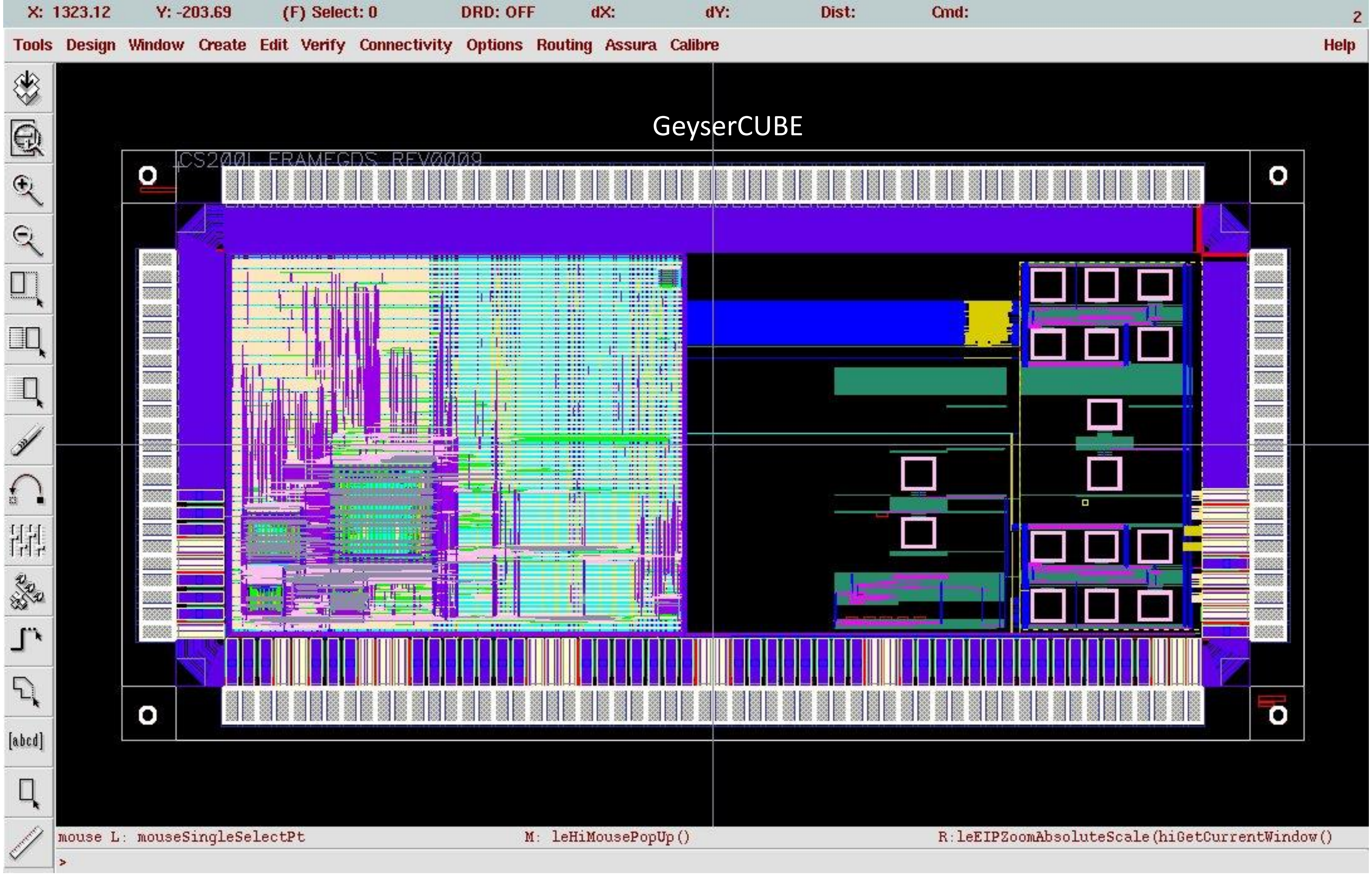


Fujitsu’s 65nm 16-metal layer
16bit PEs are used

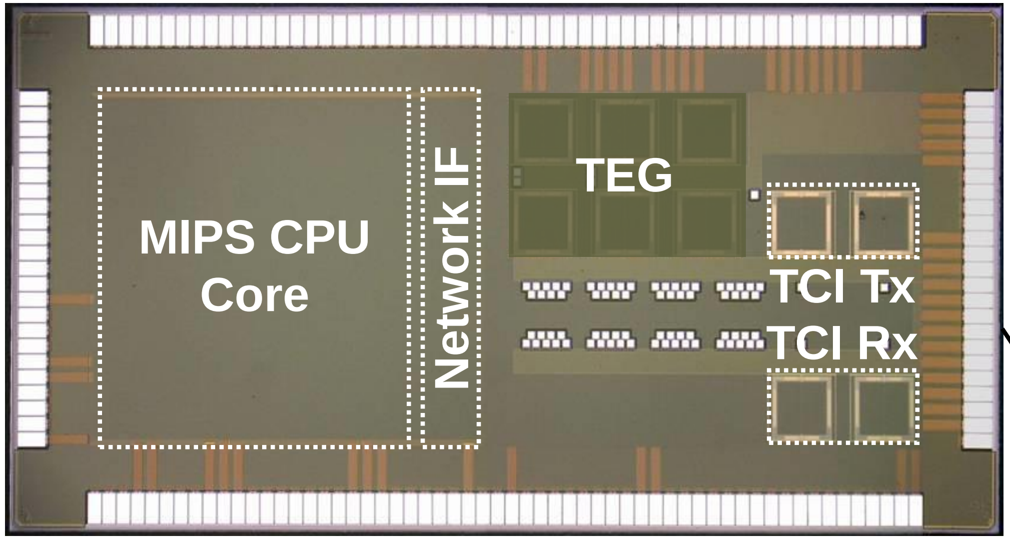
世界一:2.7GOPS ／11mW　Cool Mega Array



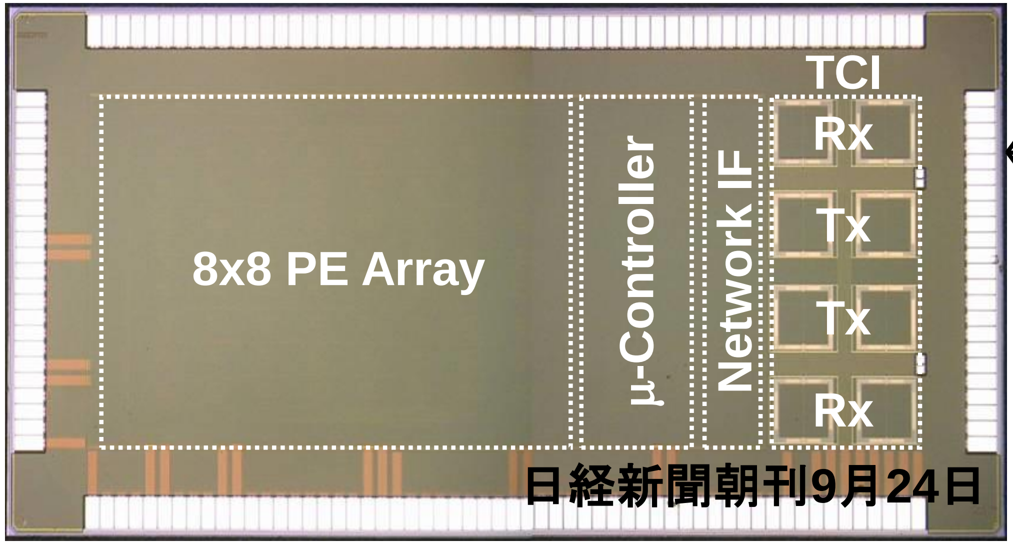
GeyserCUBE


X: 1323.12
Y: -203.69
(F) Select: 0
DRD: OFF
dX:
dY:
Dist:
Cmd:
2
Tools Design Window Create Edit Verify Connectivity Options Routing Assura Calibre
Help
CS200L FRAMEGDS REV0009
[abcd]
mouse L: mouseSingleSelectPt
M: leHiMousePopUp()
R:leEIPZoomAbsoluteScale(hiGetCurrentWindow()

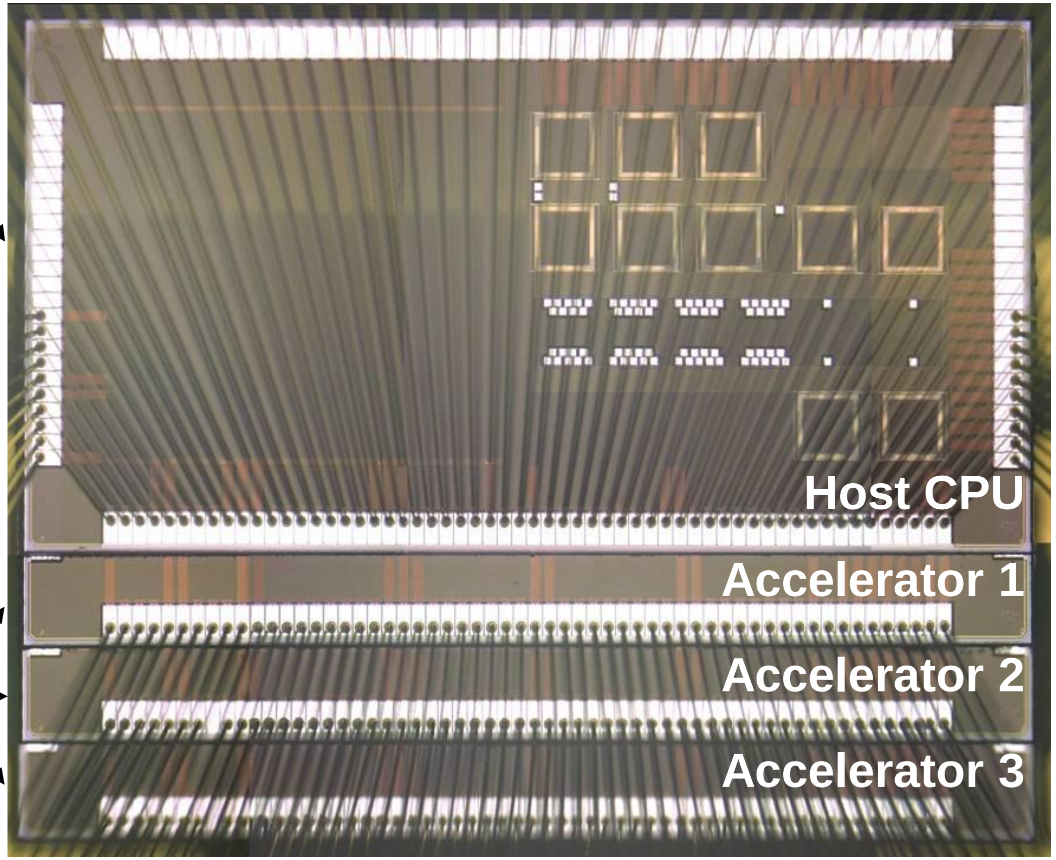
ワイヤレス積層マルチコアシステム
Cube-1



Host CPU Chip



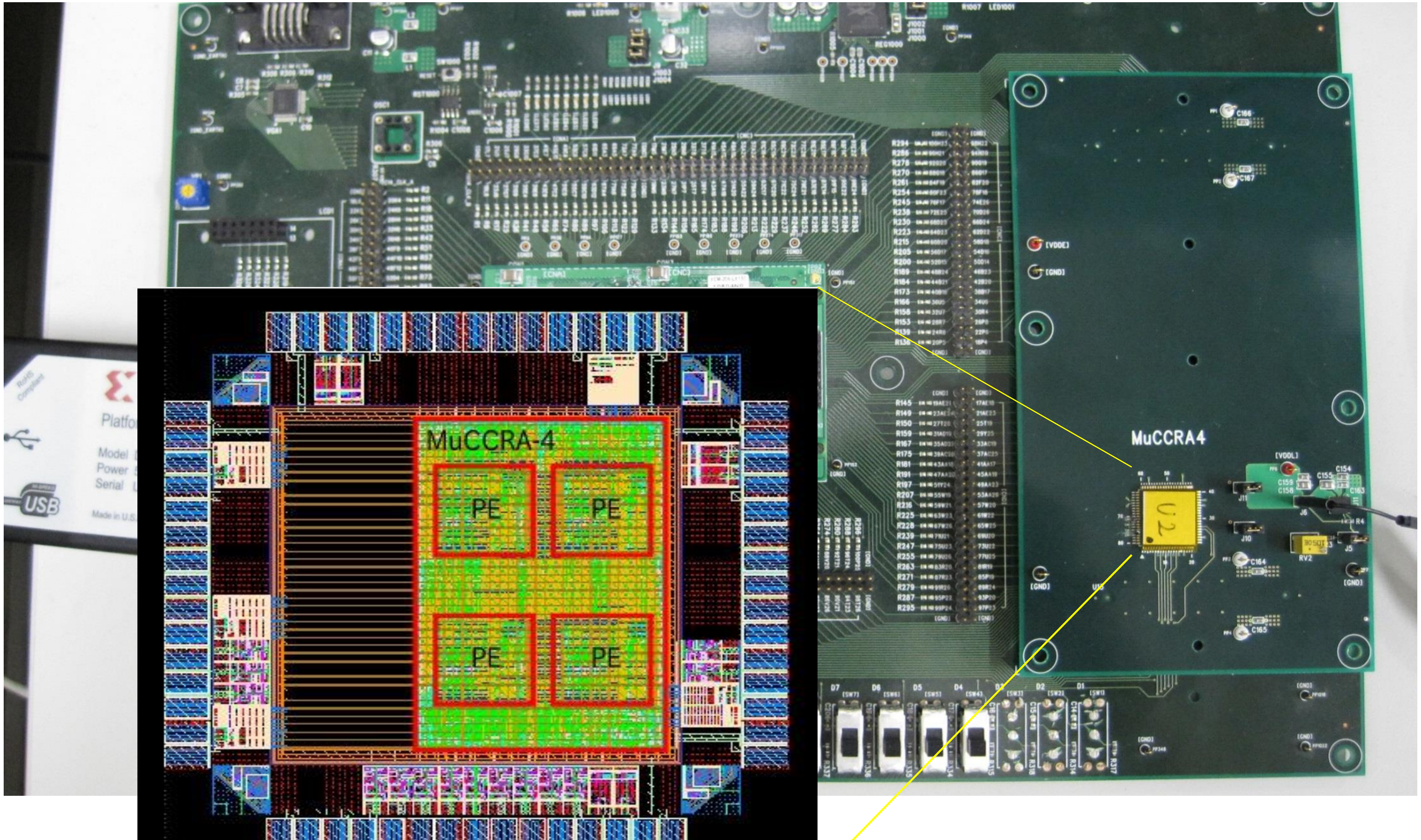
Accelerator Chip



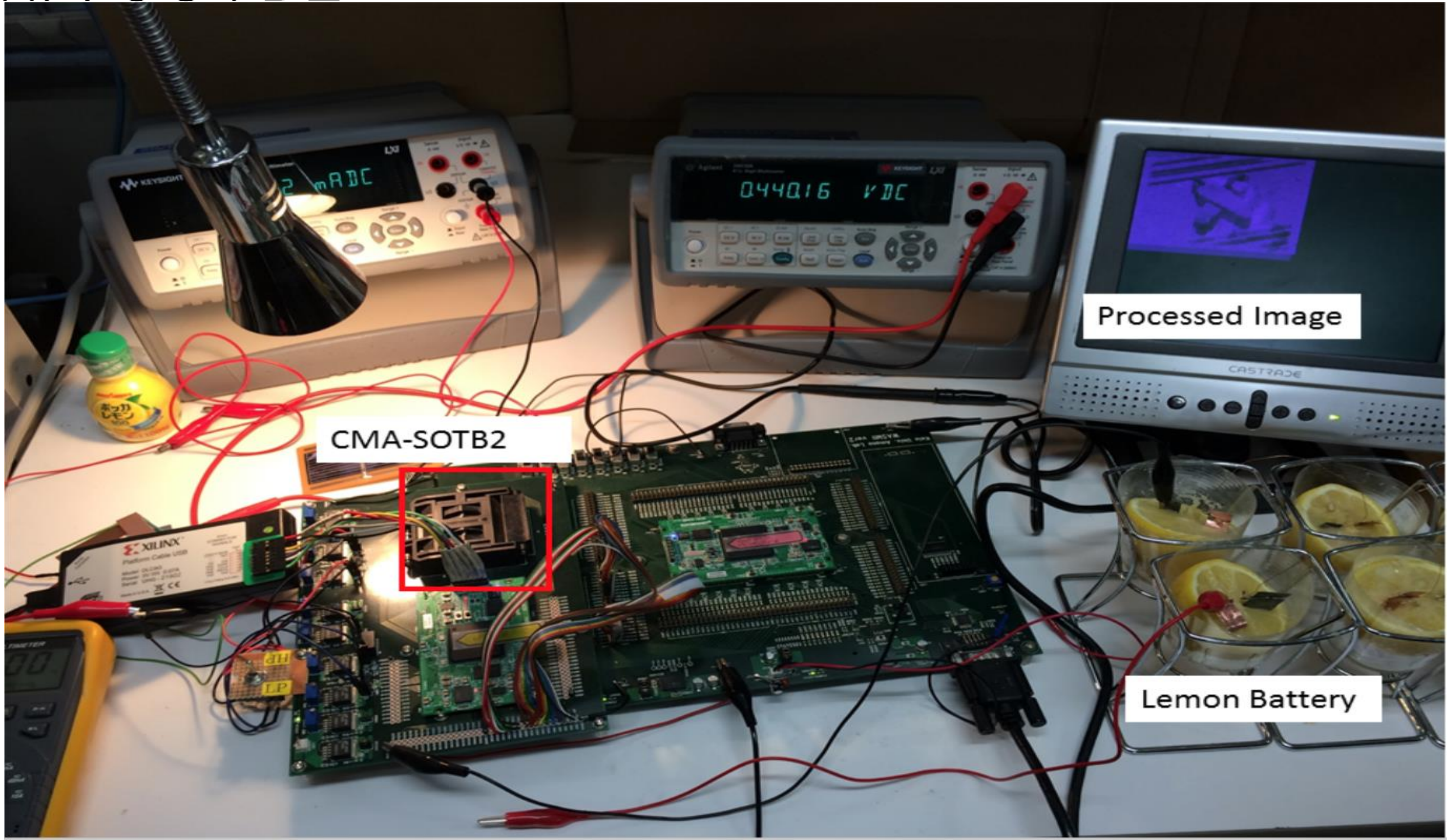
Host CPU + Accelerator x3 Chip Stack
Fabricated in 65nm CMOS

Microphotograph of stacked test chips.

最新プロセス28nmを使った動的リコンフィギャラブルプロセッサ MuCCRA-4　稼動！



レモン電池で動作する
CMA-SOTB2



教訓のまとめ

年を食ったが、設計環境が進歩したお蔭で設計し続けていられる

1. 自分の実装能力を限界としない
2. 専用に閉じこもらず、なるべく汎用を目指そう
3. 企業と連携しよう、他人に頼ろう(一緒にやろう)
4. 高性能はFPGAで、電力消費はチップで

もはやちょっとやそっとの
しくじりでは動じない

では、若い人を同じ戦略をお奨めできるか？

- もちろんできる
  - SOTBプロセスが利用可能に
  - 誘導結合TCIもIPに
- 設計、実装環境はどんどん改善されている
- コンピュータの応用分野はどんどん広がっていく
  - IoT、ビッグデータ、ウェアラブルコンピューティング、自動運転、ディープラーニング、etc.
- 1990年代 [illegible] クチャ分野は確実

いっしょに挑戦し、
しくじりましょう