FPGAコンピューティング
最近の情勢

2018.11

慶應義塾大学
天野英晴

自己紹介

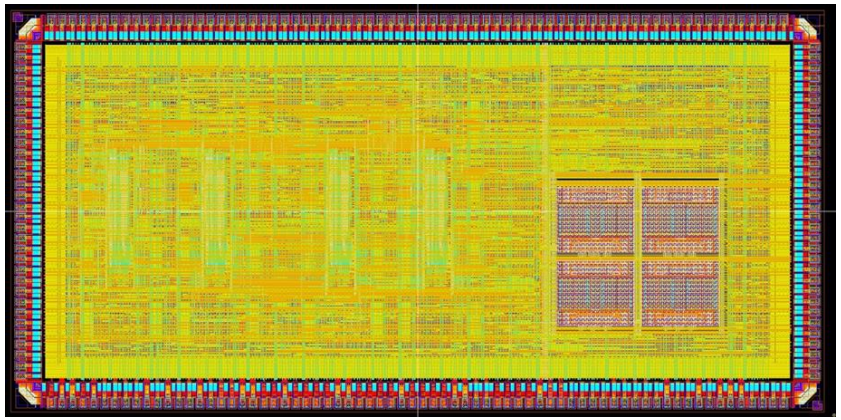
- 主に並列計算機、リコンフィギャラブルシステムを研究してきた
  - 交信用メモリ
  - 結合網
  - リコンフィギャラブルシステム
    - FPGAを利用したシステム
    - 粗粒度リコンフィギャラブルシステム
- ハードウェアを作ってきた
  - コアをLSIチップで実装
  - システム全体を作る
- ずっと慶應に居た

http://www.am.ics.keio.ac.jp/members/hunga

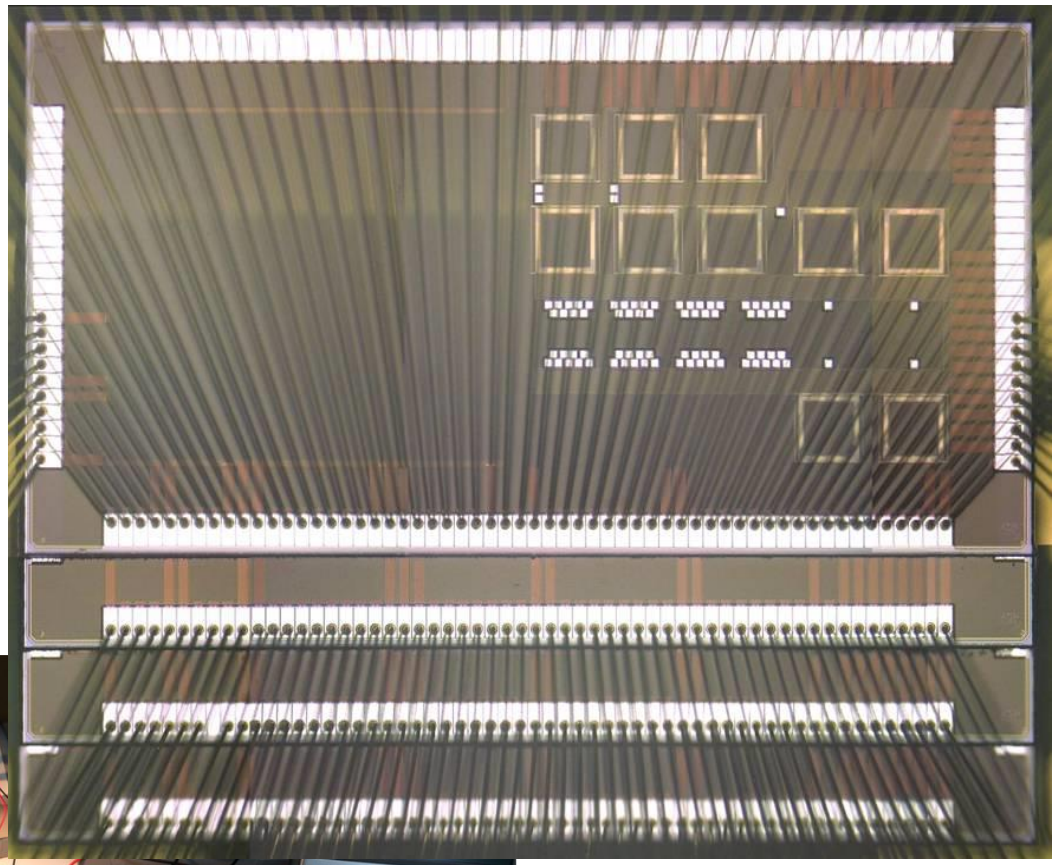
2016年フェロー就任講演　「しくじりフェロー」　をご覧ください。

WASMII： 新しいアーキテクチャをチップ化するグループ

- **超低電力なチップを作る。チップ同士を重ねてネットワークを構成する**
- **新しいアーキテクチャ CGRA（Coarse Grained Reconfigurable Array)**
- **ニューロチップ**

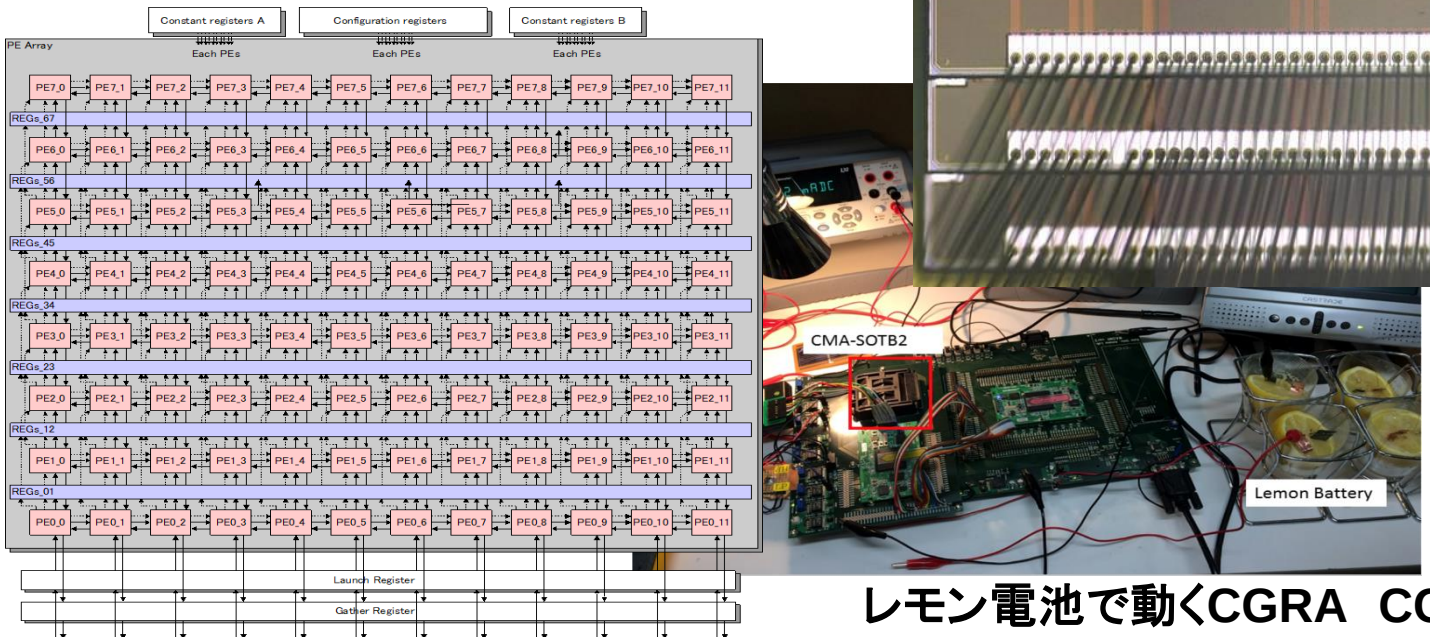


Neuro Chip： SNACC



3次元チップ間

無線接続



レモン電池で動くCGRA CC-SOTB2

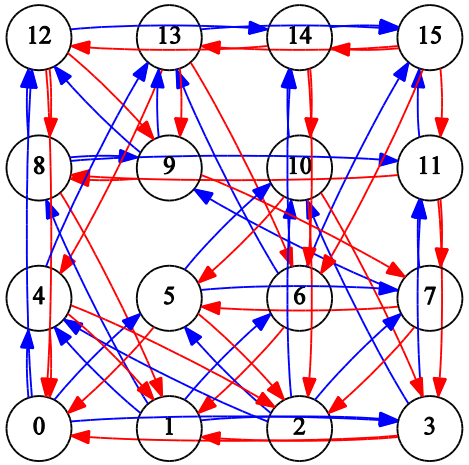
Blackbus： Interconnection Networkを研究するグループ

大規模並列コンピュータのノード同士をどのように接続するか、ルーチングするか？

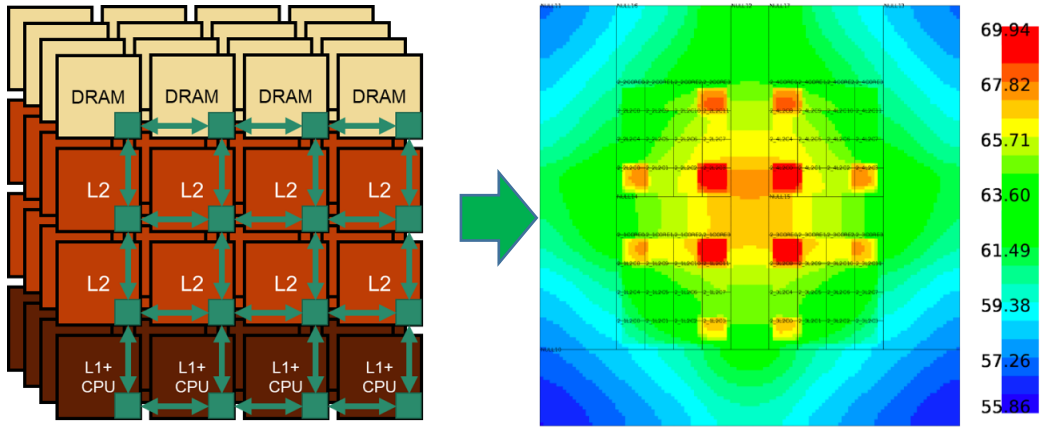
NoC（Network on Chip)をどのように作るか？

シミュレーションで評価を取るのに費やす時間が多い

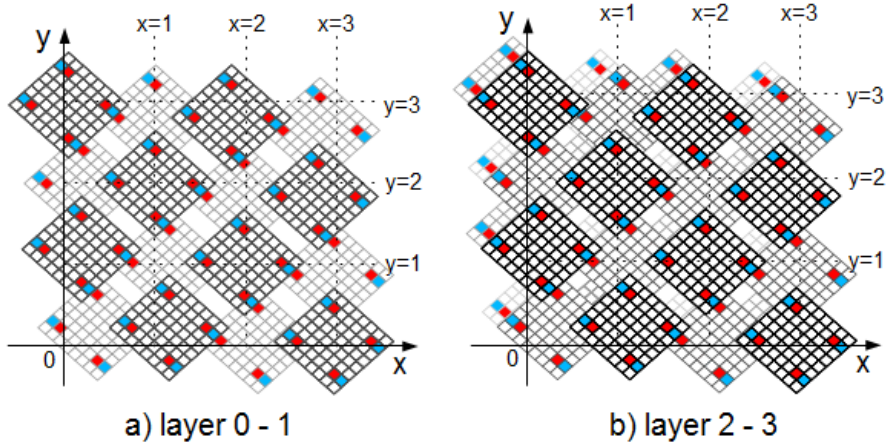
有名論文誌に論文がたくさん通る

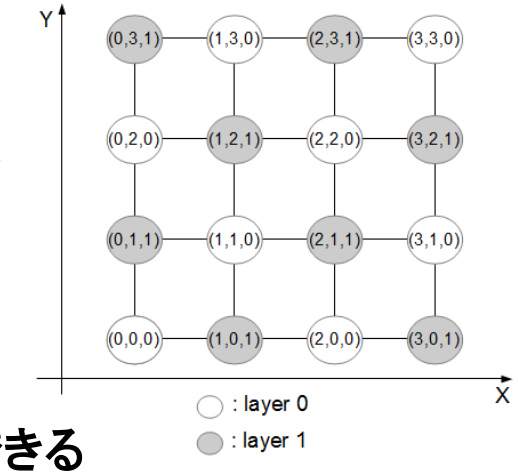


ランダムネットワークによる低レイテンシィ化



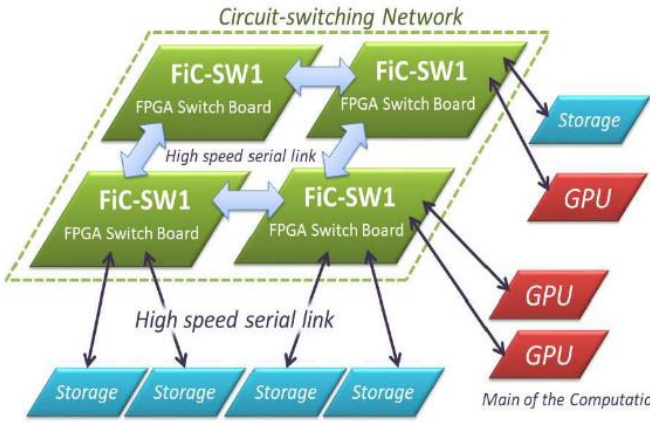
積層チップの熱解析ツールの開発

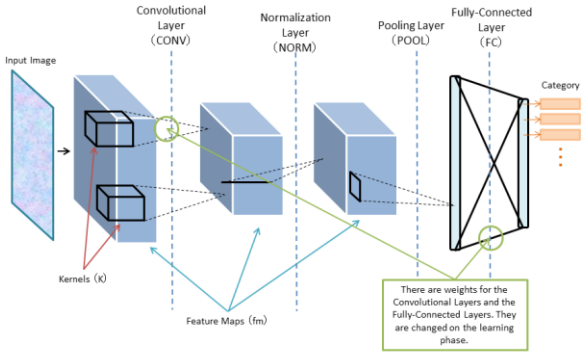




チップを重ねるとネットワークができる

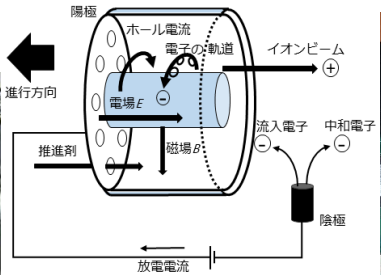
ASAP： FPGA、GPUによるアクセラレータの研究
エッジ側、クラウド側の両方、HLS（高位合成）を使う人が多い





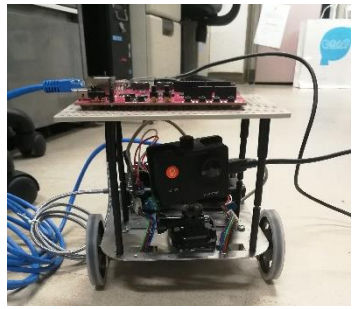
大規模マルチFPGAシステムFiC（Flow in Cloud)

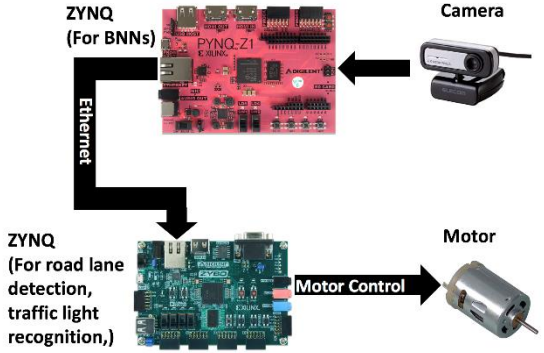
GPUをExpEtherで接続したGPUBOX



Intel FPGAを使った

エンジンシミュレーション





自動走行車

本日の流れ

- FPGAの基本
- FPGAの最近の動向
  1. 基本構造の変化
  2. FPGAの分化
  3. IPとSoC型FPGA
  4. アクセラレータとしてのFPGA
  5. FPGA in Cloud
- Virtual Large FPGA by FiC-SW

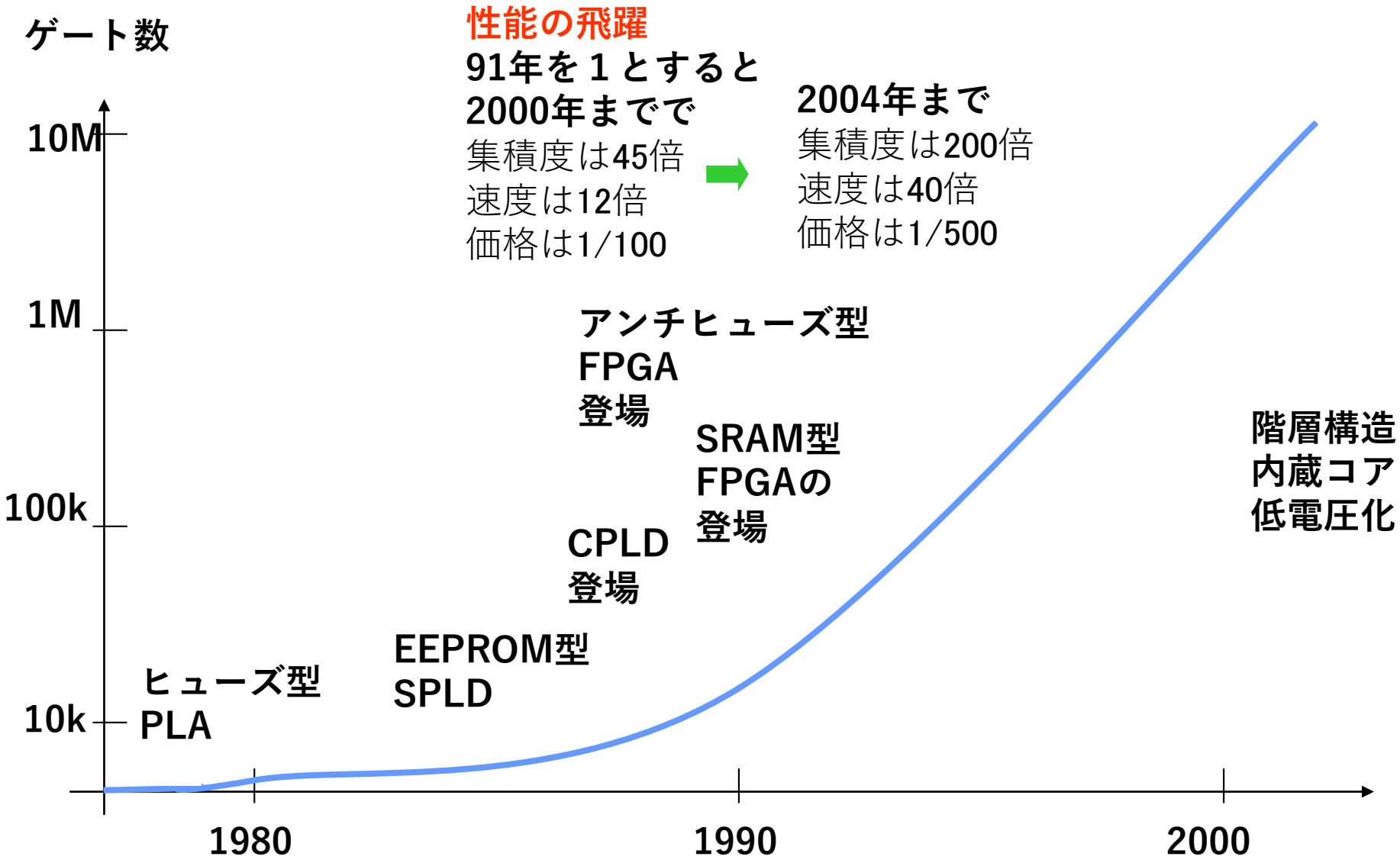
1. FPGAの基本

FPGAはPLD (Programmable Logic Device)の一種

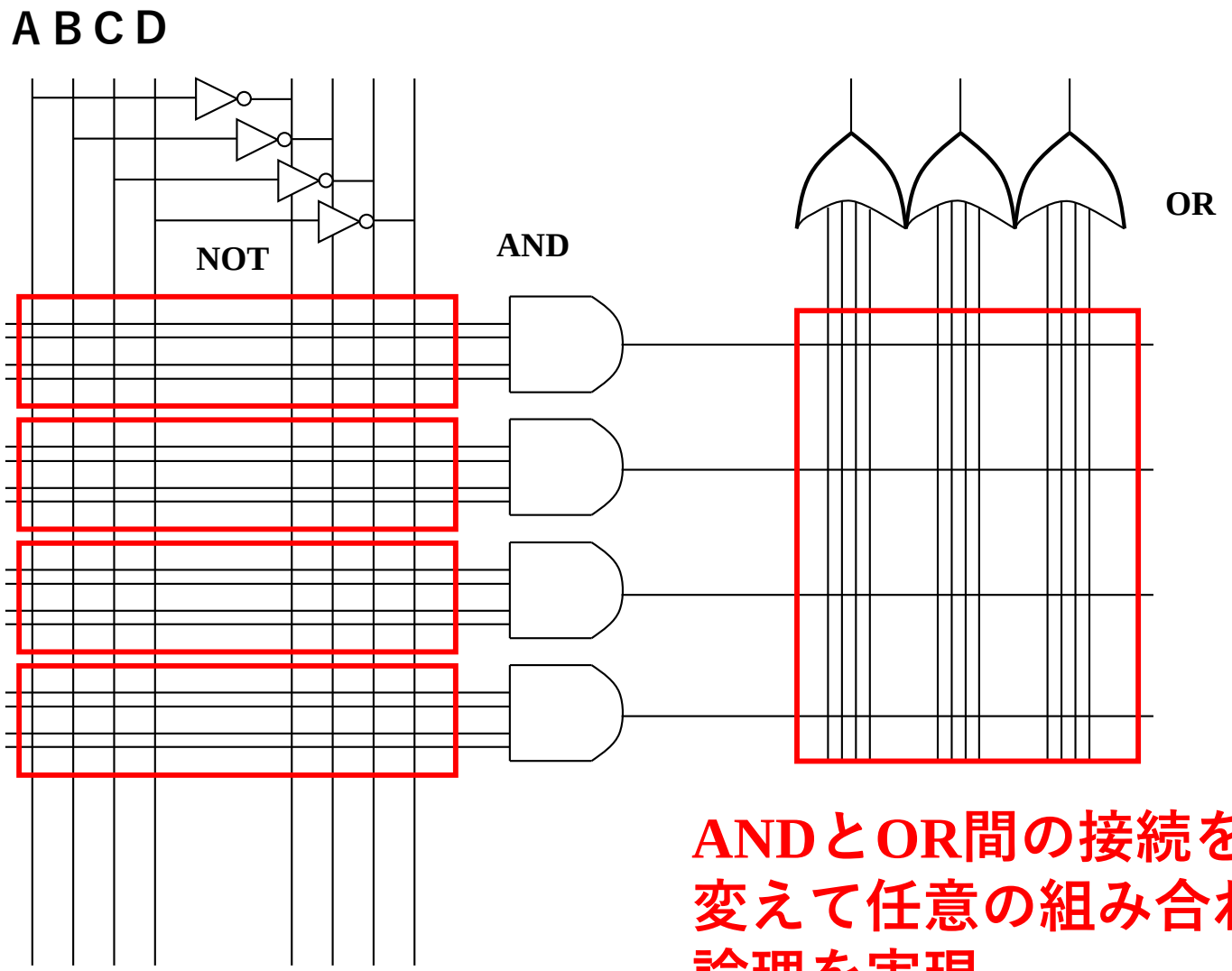
- ユーザが論理機能を決めることのできるIC
  ⇔ メモリ、CPU、ASIC （Application Specific IC）
- SPLD（Simple PLD） /
  PLA（Programmable Logic Array ）
  - 小規模なAND-OR構造のもの
- CPLD （Complex PLD）
  - 主としてAND-OR構造を拡張して大規模化したもの
- FPGA （Field Programmable Gate Array）
  - LUT構造を用いた大規模なもの

用語は混乱していて、使い分けは統一されていないので注意！

PLDの成長

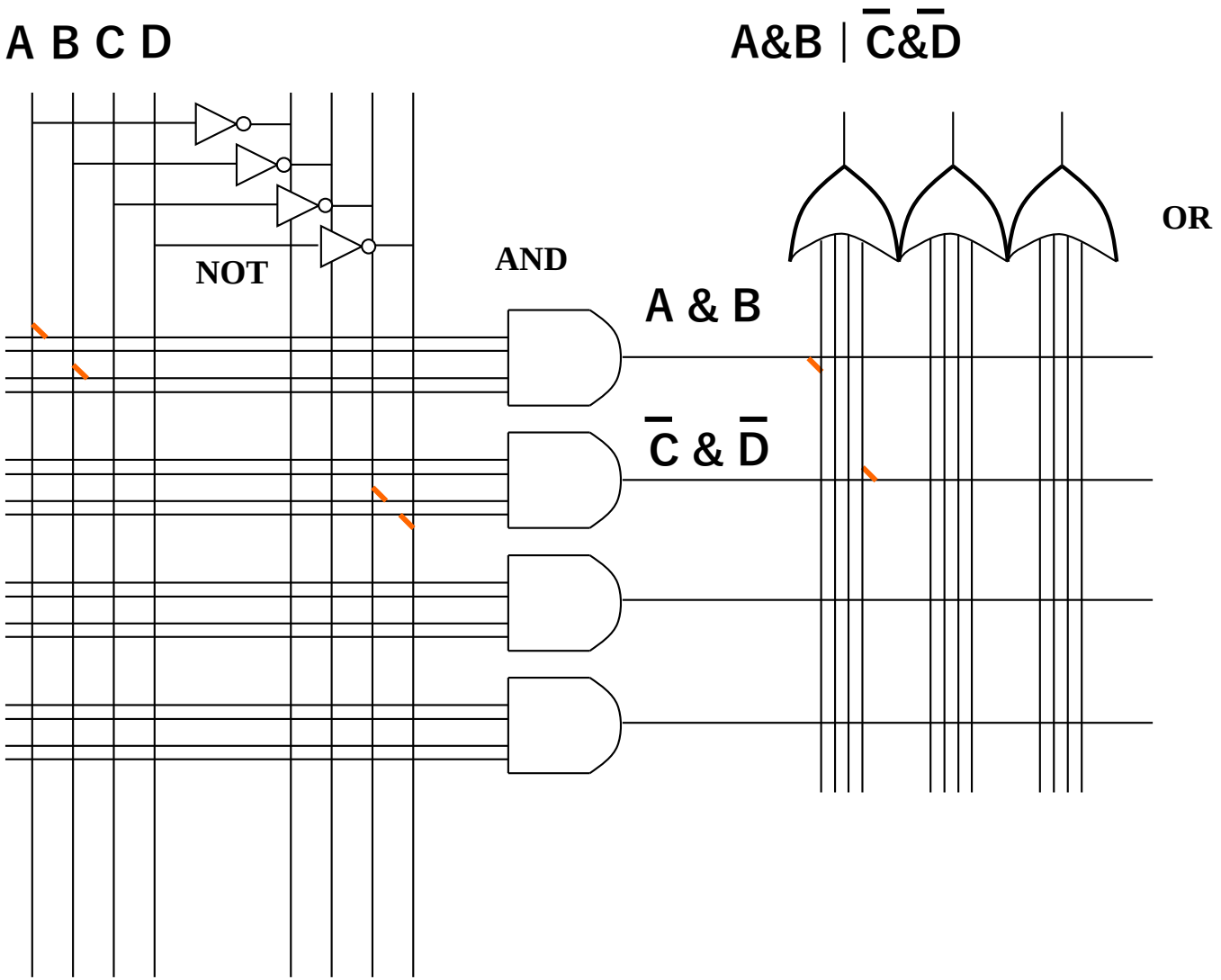

ゲート数
性能の飛躍
91年を１とすると
2000年までで
集積度は45倍
速度は12倍
価格は1/100
2004年まで
集積度は200倍
速度は40倍
価格は1/500
10M
1M
100k
10k
アンチヒューズ型
FPGA
登場
SRAM型
FPGAの
登場
CPLD
登場
EEPROM型
SPLD
ヒューズ型
PLA
階層構造
内蔵コア
低電圧化
1980
1990
2000

SPLD（Simple　PLD:プロダクトターム構造/AND-OR構造）

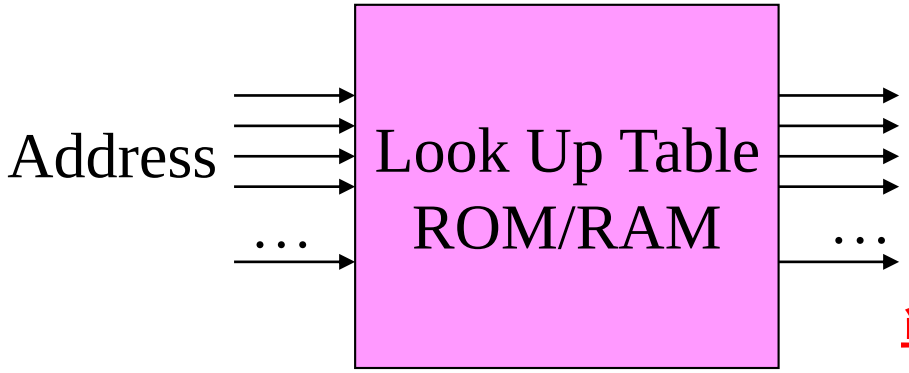


ANDとOR間の接続を
変えて任意の組み合わせ
論理を実現

プロダクトターム構造の作り方

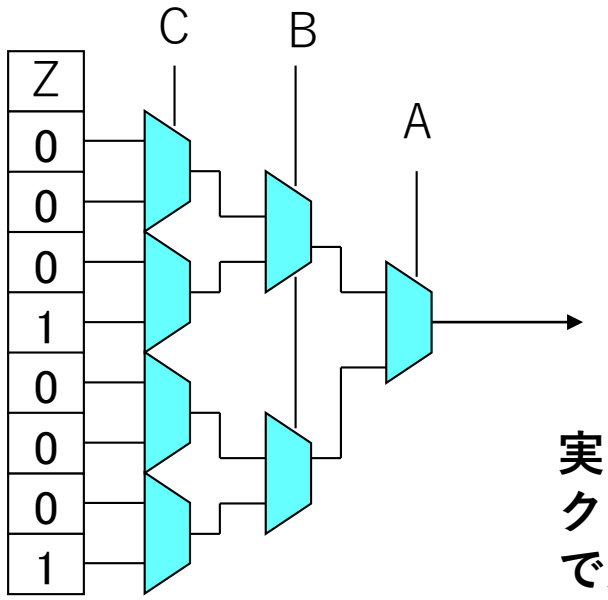


LUT：Look Up Table方式による論理の実現



単純なROMまたはRAMによっても
任意の組み合わせ回路が実現できる

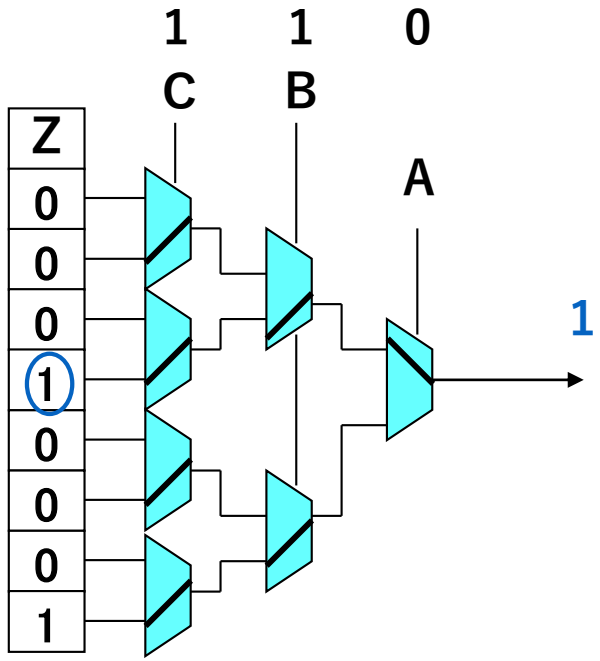
ABC	Z
000	0
001	0
010	0
011	1
100	0
101	0
110	0
111	1



実際はメモリとマルチプレクサ
で実現する

LUT：Look Up Tableによる論理の実現例

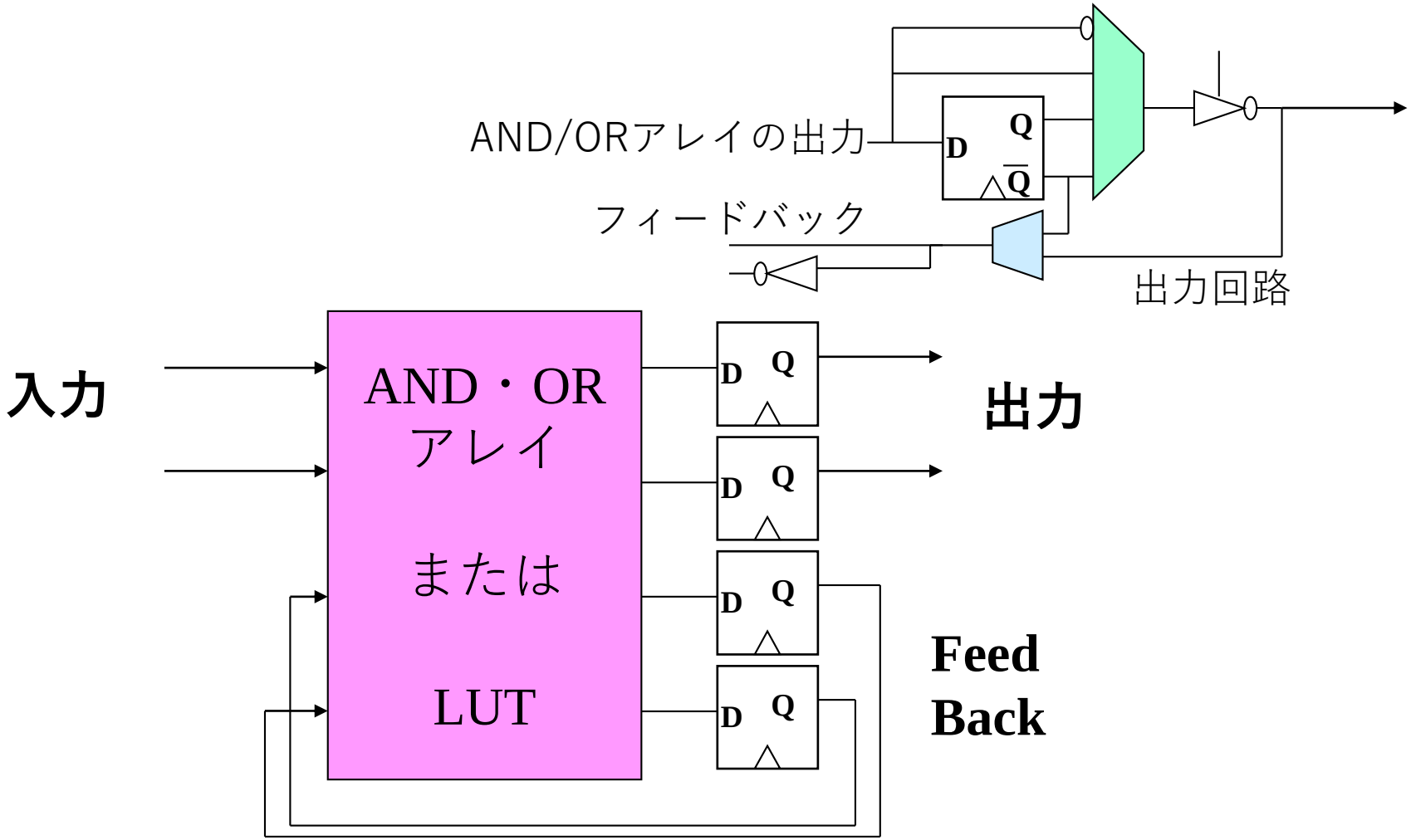
ABC	Z
000	0
001	0
010	0
011	1
100	0
101	0
110	0
111	1



プロダクトターム方式　vs.　LUT

- プロダクトターム方式（AND-OR構造）
  - 多入力多出力回路が効率良く実現できる
  - 場合によっては入力項数が不足する
  - EEPROM、フラッシュROMでの実現に適している
- LUT
  - 任意の論理が実現できる
  - 出力が少なく小規模な論理に有利
  - フラッシュ、アンチヒューズ、SRAM型に適している

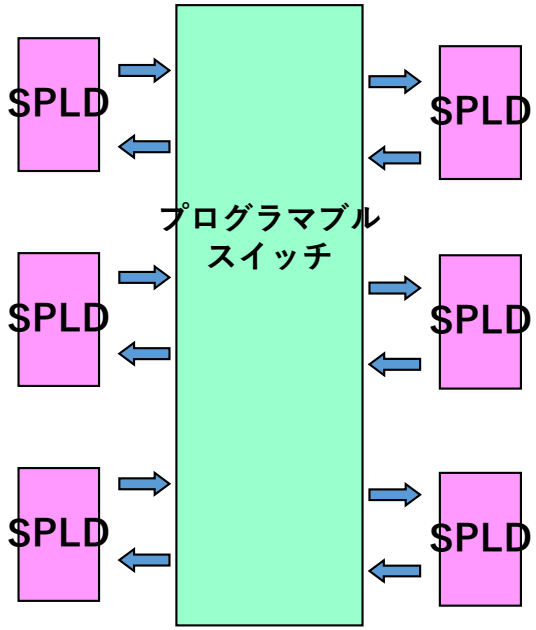
順序回路の実現



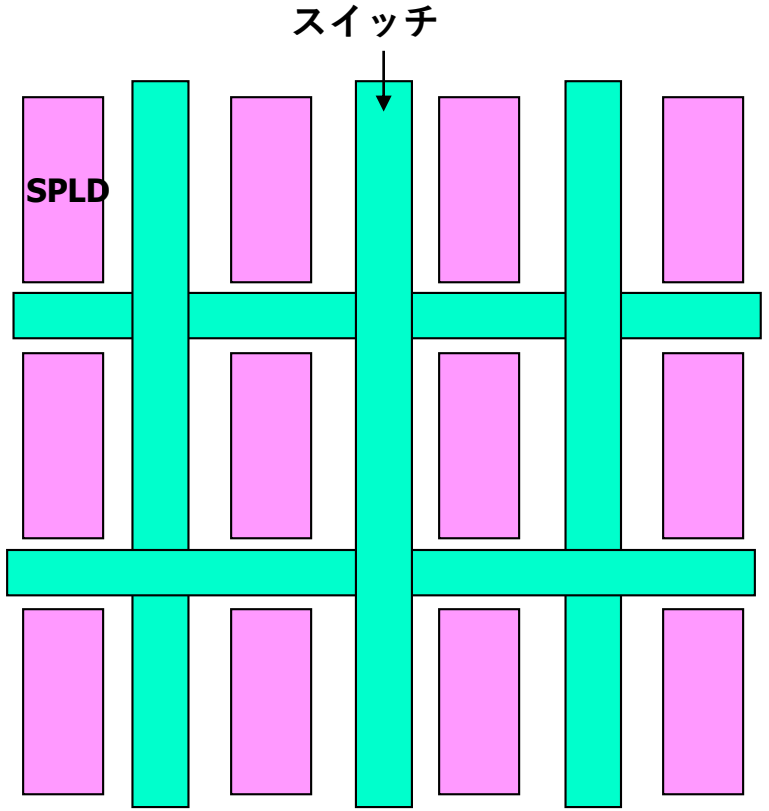
出力にF.F.を付けて、フィードバックラインを装備すれば任意の順序回路が実現できる

CPLD　(Complex　PLD)

AND/OR　ロジックブロック複数をスイッチで接続プログラマブル



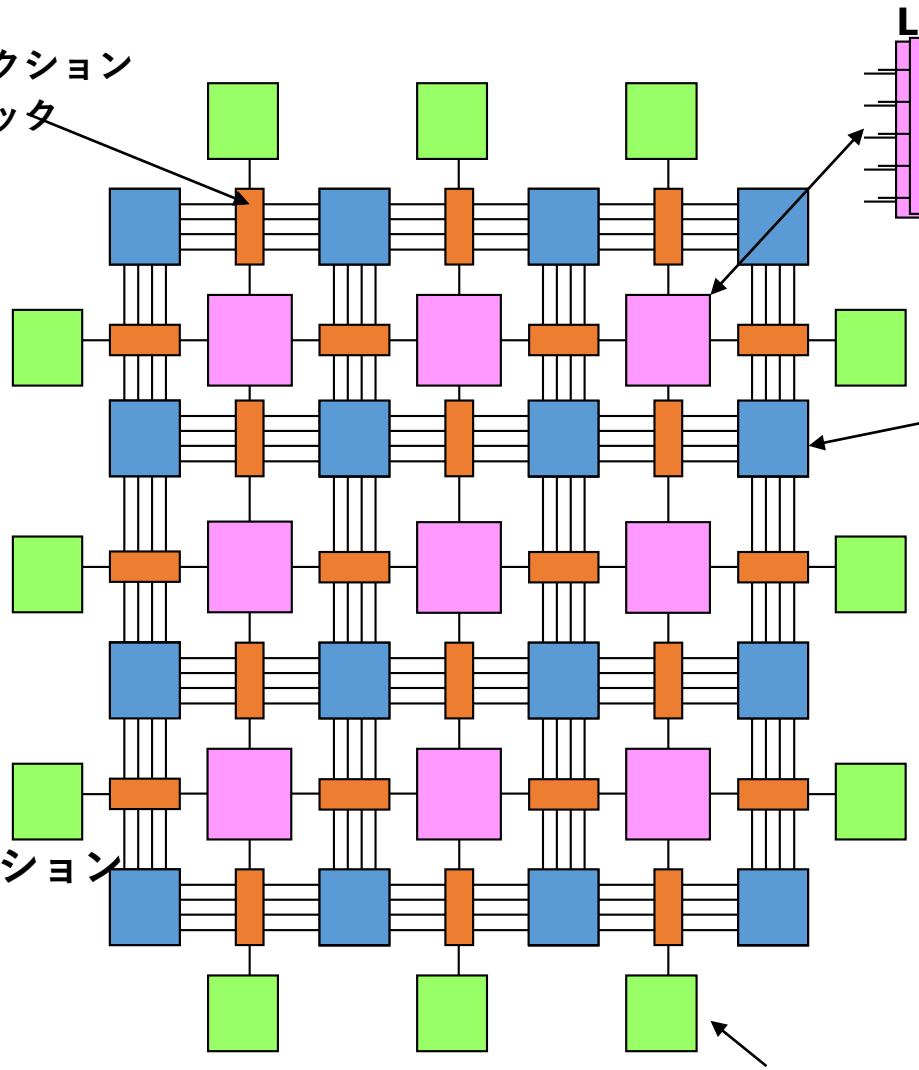
Altera社
MAXシリーズなど

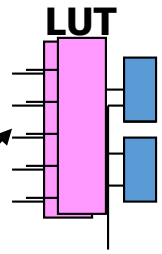


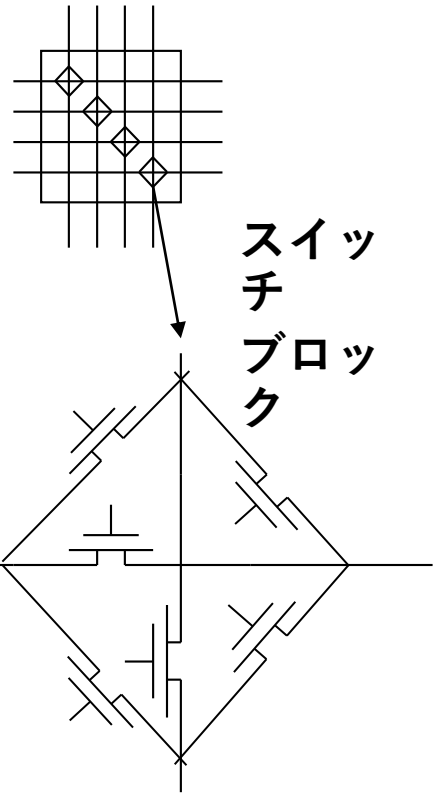
2次元構造で大規模化

FPGA(Field Programmable Gate Array)

アイランド
スタイル

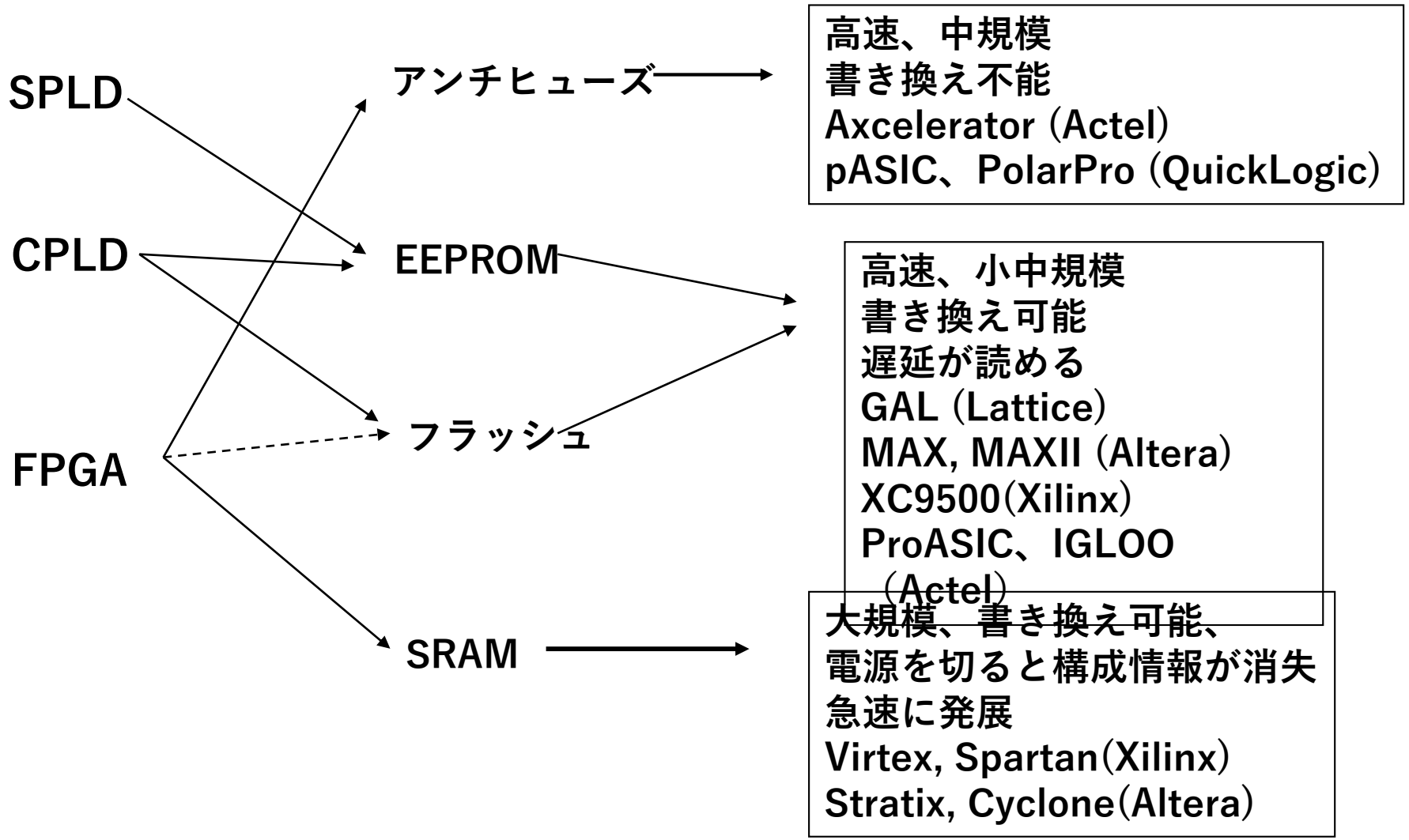




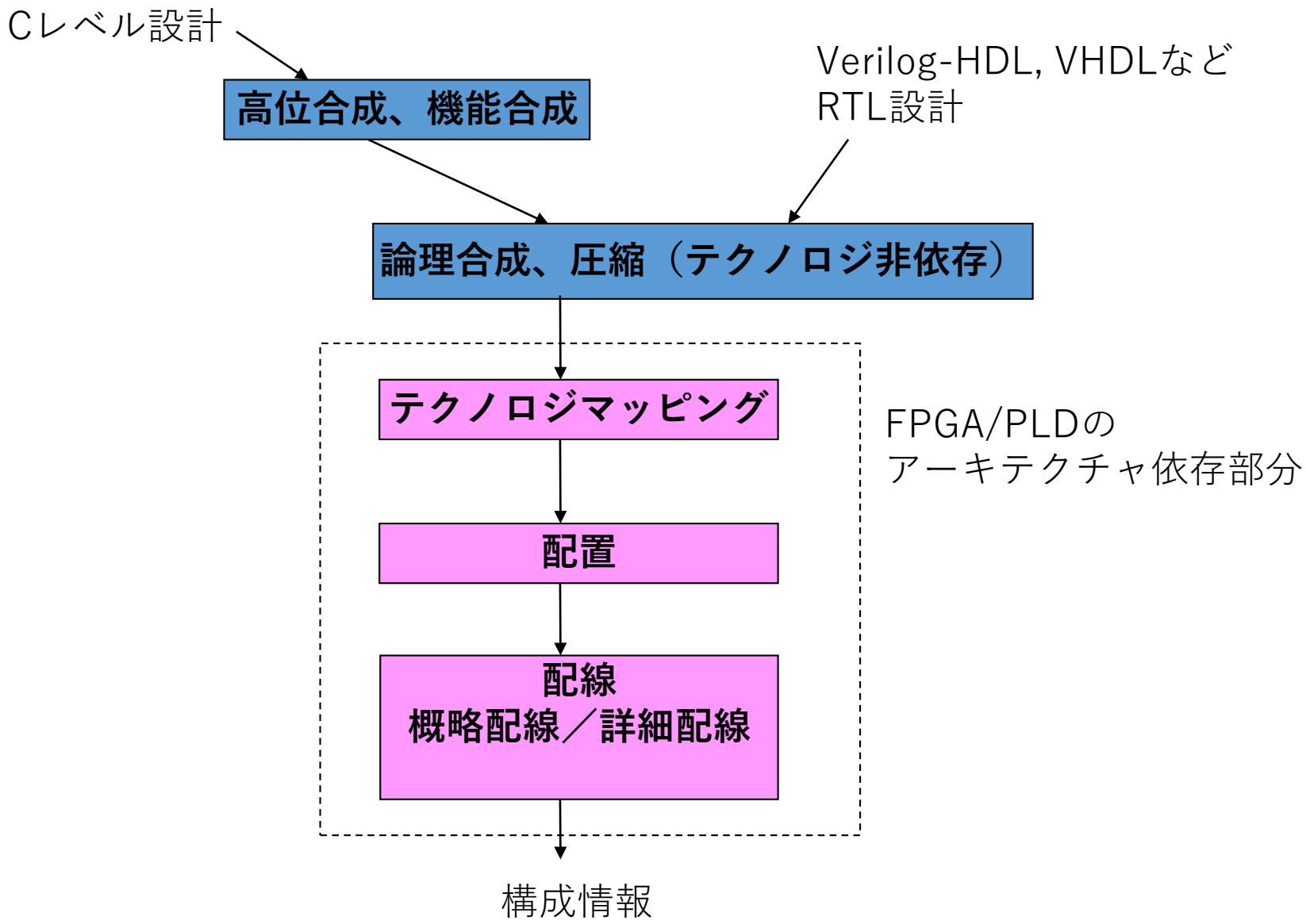


LUT, スイッチ
ブロックの設定は
SRAM内の
構成情報
（コンフィギュレーション
データ）
で変更可能

構成方式と柔軟性実現技術



FPGAの設計


Cレベル設計
高位合成、機能合成
Verilog-HDL, VHDLなど
RTL設計
論理合成、圧縮（テクノロジ非依存）
テクノロジマッピング
配置
配線
概略配線／詳細配線
FPGA/PLDの
アーキテクチャ依存部分
構成情報

FPGAの設計ツール

- ベンダが統合型のツールを提供
  - シミュレーション、配置、配線、構成情報の転送、デバッグツールが一式入っている
  - お試し版Web packでもかなりのことができる
- 設計入力
  - Verilog HDL、VHDLなどハードウェア記述言語が多い
  - 最近、C言語設計が発達
    - Impulse Cなどサードパーティの供給
    - Xilinx Vibado, SDAccel
    - Intel(Altera)　OpenCL

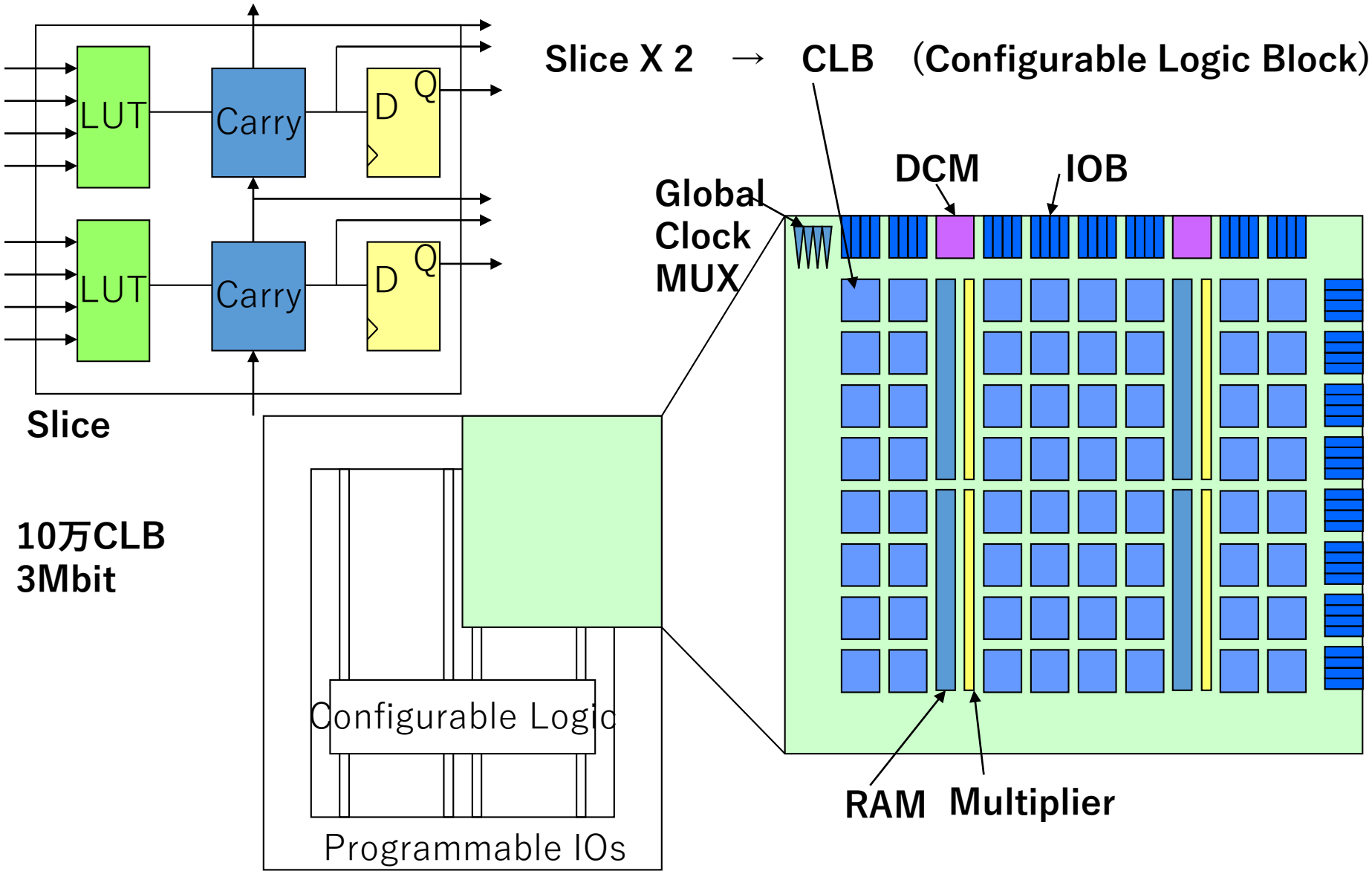
２．FPGAの最近のトレンド

1. 階層的構造による大規模化、高速化：
   - Xilinx社Virtexシリーズ、Altera社Stratixシリーズ
2. ハイエンドとローコストの分化、ミドルレンジの登場

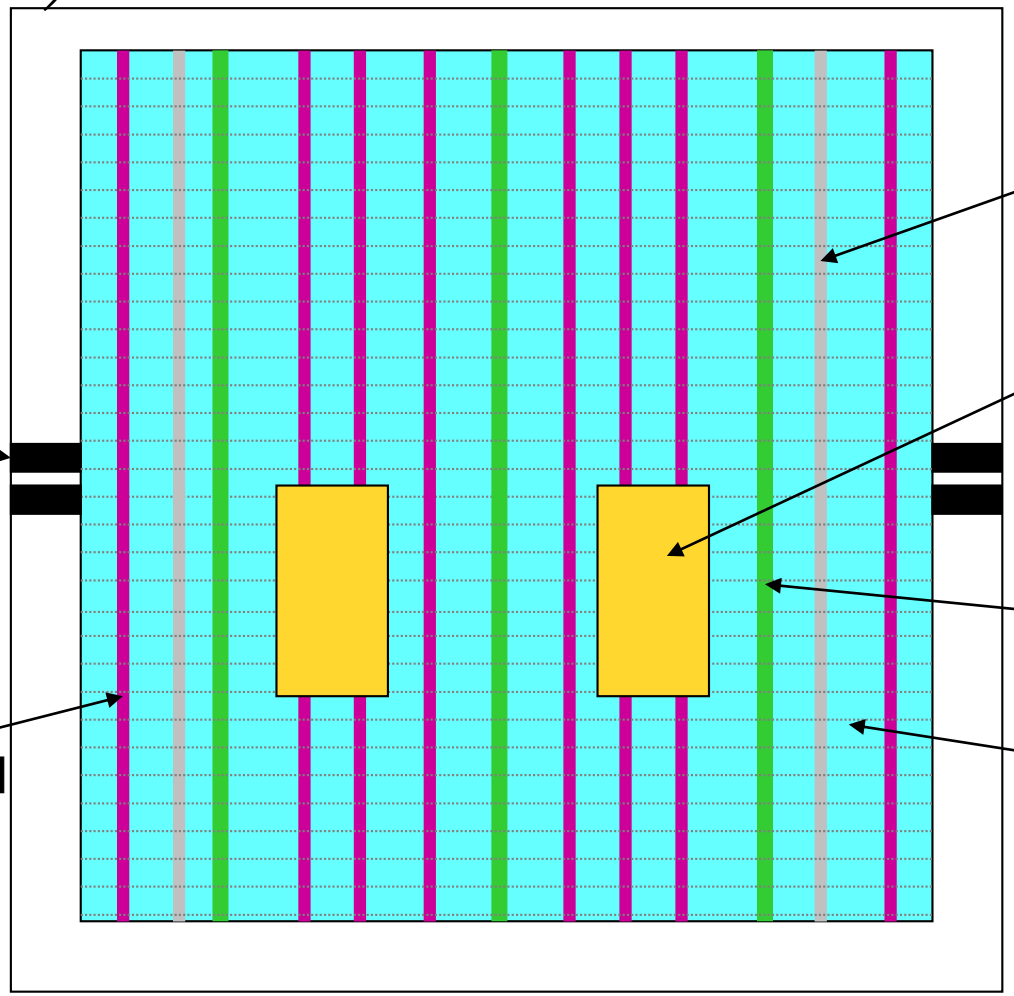
- System on Programmable Device
  - DLL、DSP、メモリ、乗算器、高速リンクをハードIPとして混載
  - SoC(System-on-a Chip)タイプのFPGA
    - CPU（ARM)を混載
- アクセラレータとしてのFPGA
- 部分再構成機能の充実

2-1. 基本構造の変化

Xilinx　Virtex シリーズの基本構成



Intel(Altera)　Stratix II



DSP　Blocks

Mega RAM
Blocks

M4K RAM
Blocks

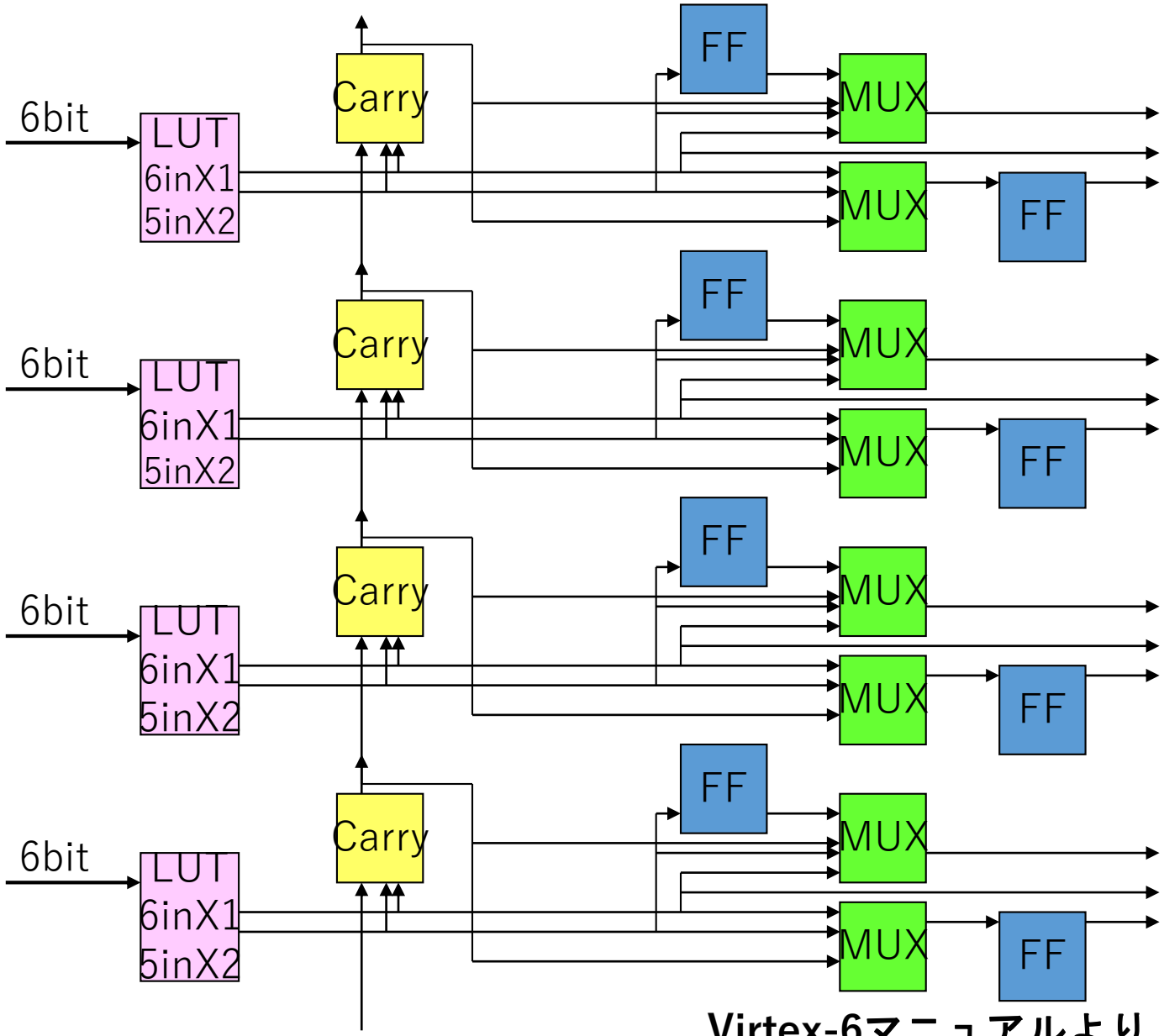
PLL

M512 RAM
Blocks

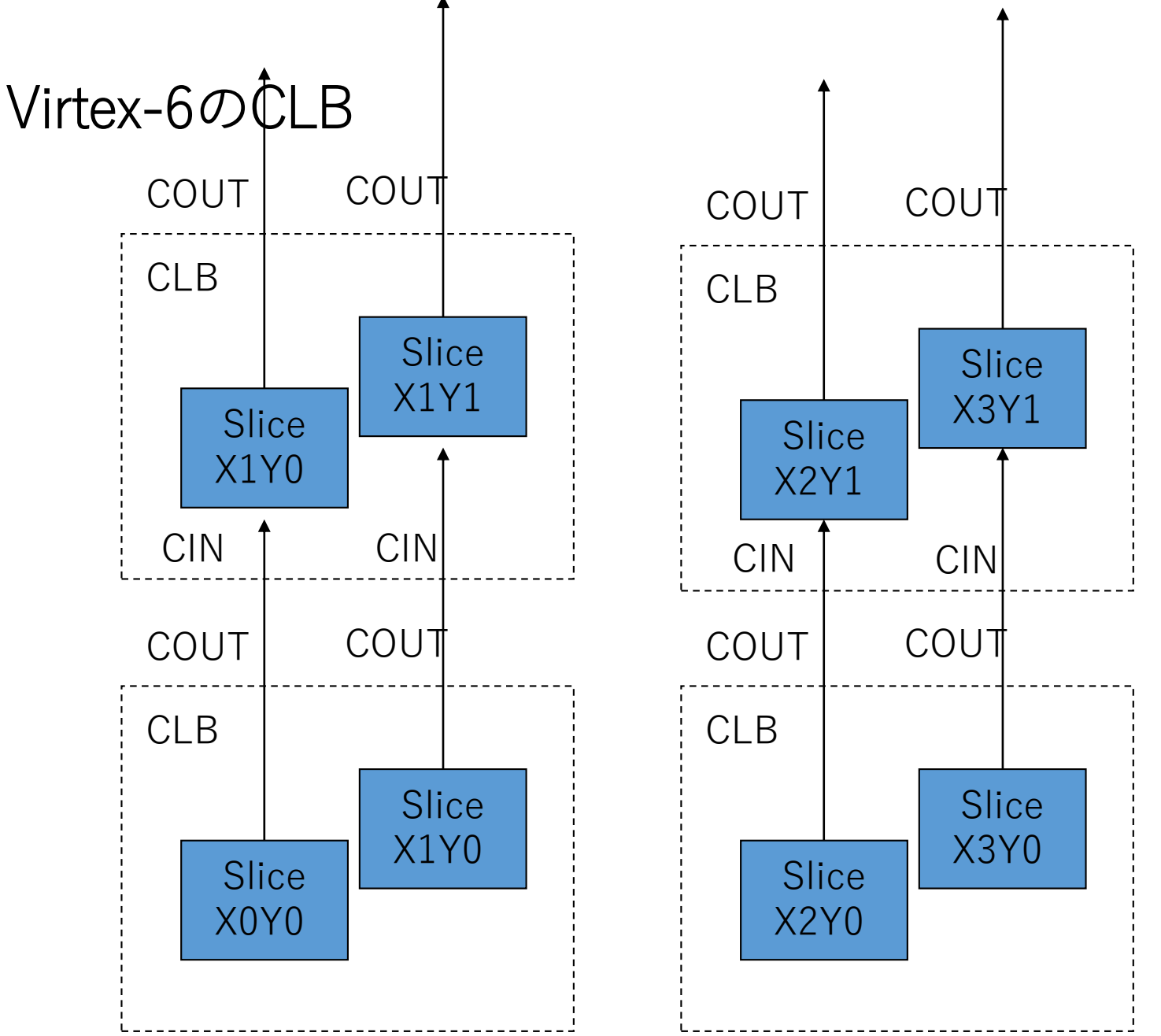
LAB：Logic Array Block
4入力のLUTとF.F.から成る
LE　10個から構成される

ローカルコネクトと
グローバルコネクトにより
高速なデータ転送を実現

Virtex-6のSlice構造

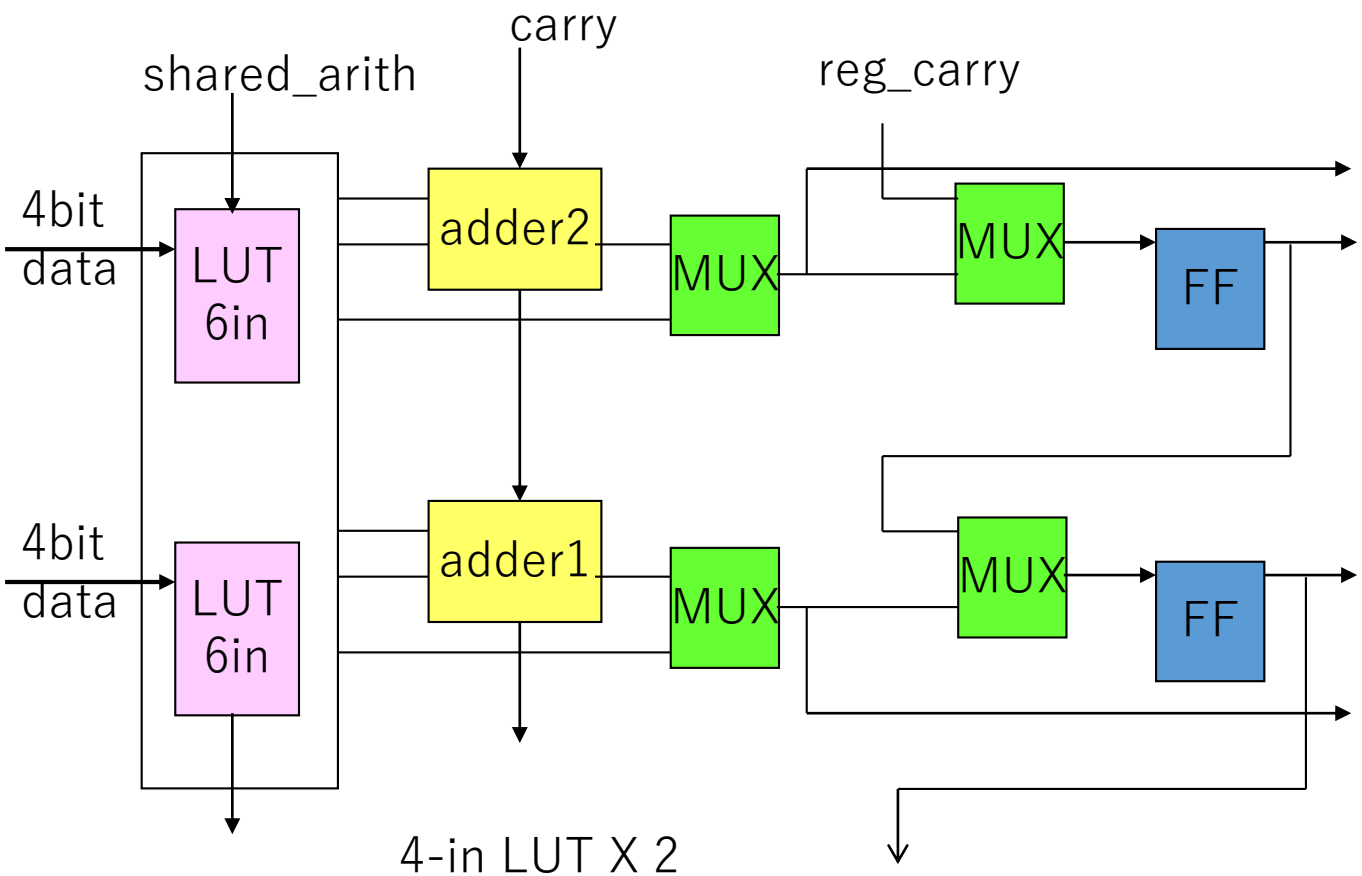


Virtex-6マニュアルより



Virtex-6マニュアルより

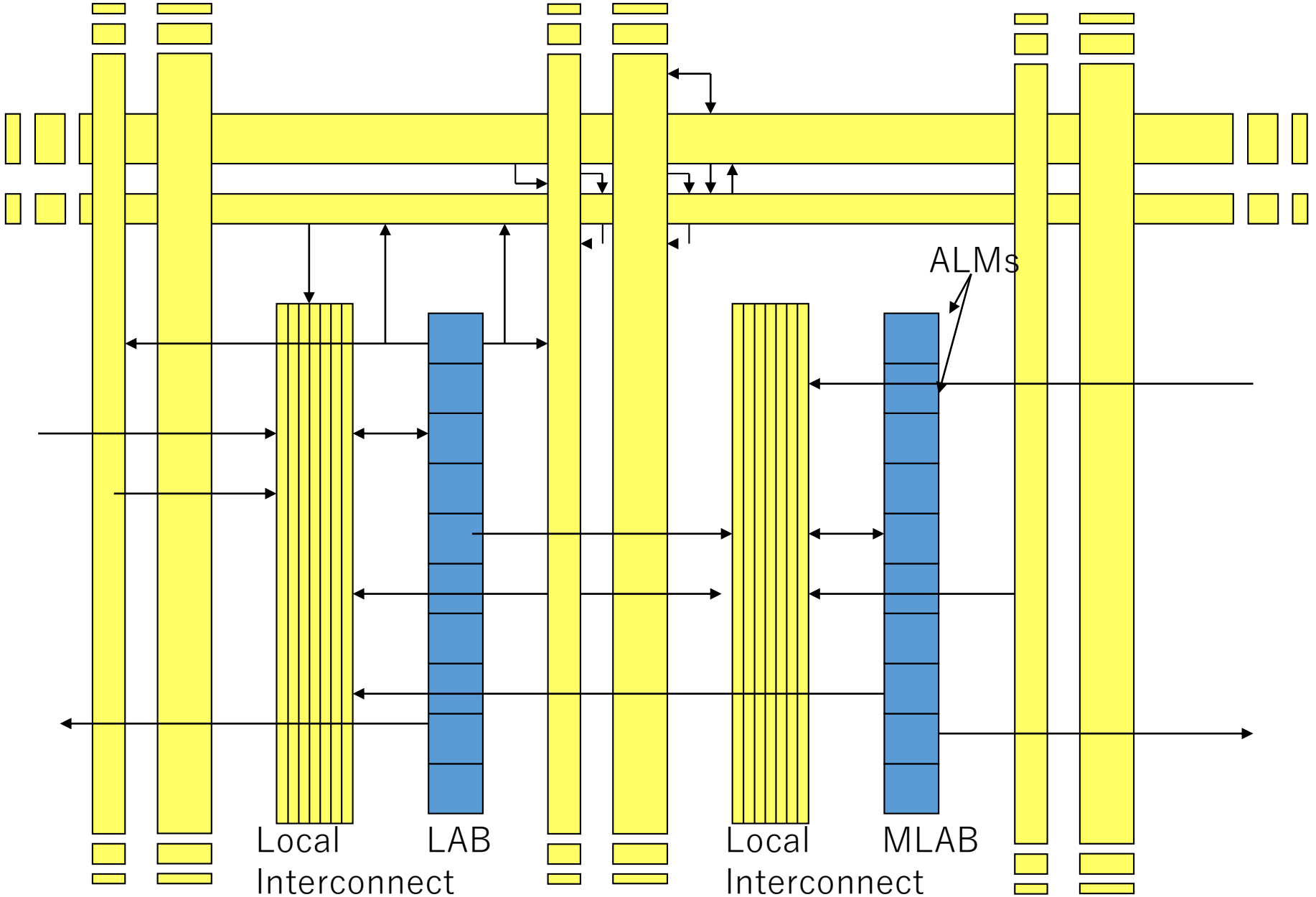
Stratix－IVのALM



4-in LUT X 2
5-in LUT + 3-in LUT
5-in LUT + 4-in LUT 1-input shared
5-in LUT + 5-in LUT 2-input shared
6-in LUT
6-in LUT + 6-in LUT 4-input shared

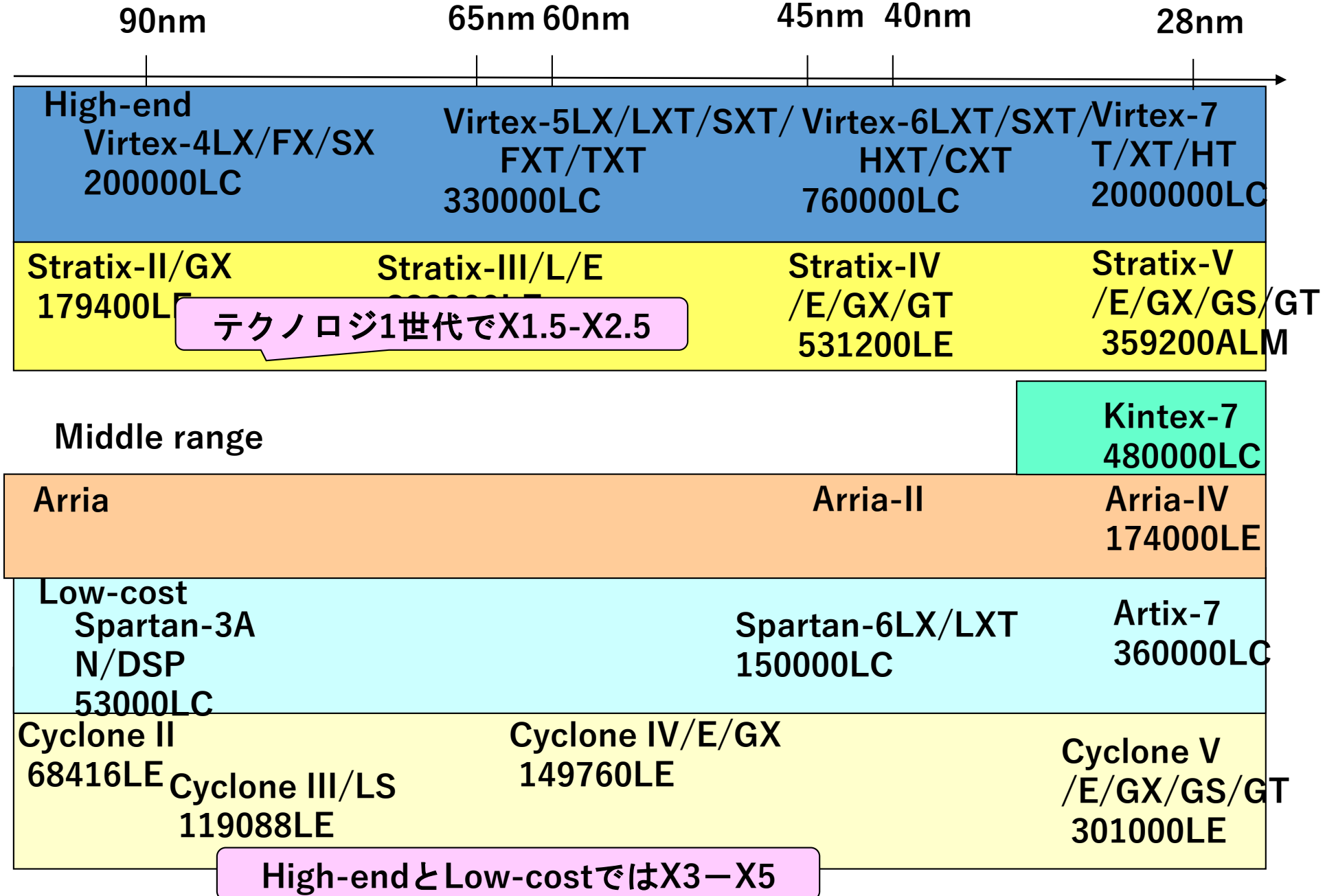
Stratix-IVマニュアルより

StratixーIVのLAB構造

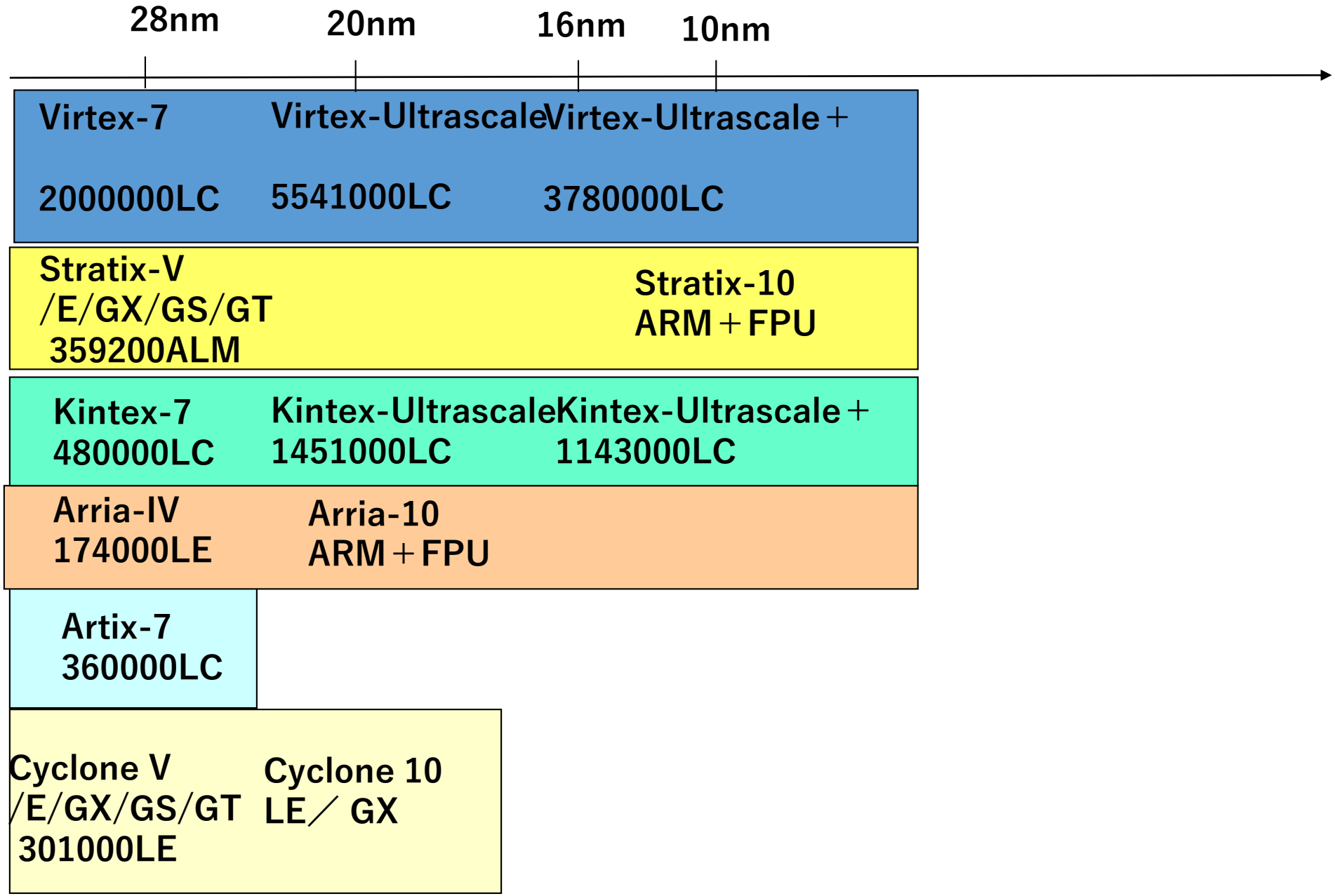


Stratix-IVマニュアルより

2-2. FPGAの分化→ハイエンド、ローコスト、ミドルレンジ



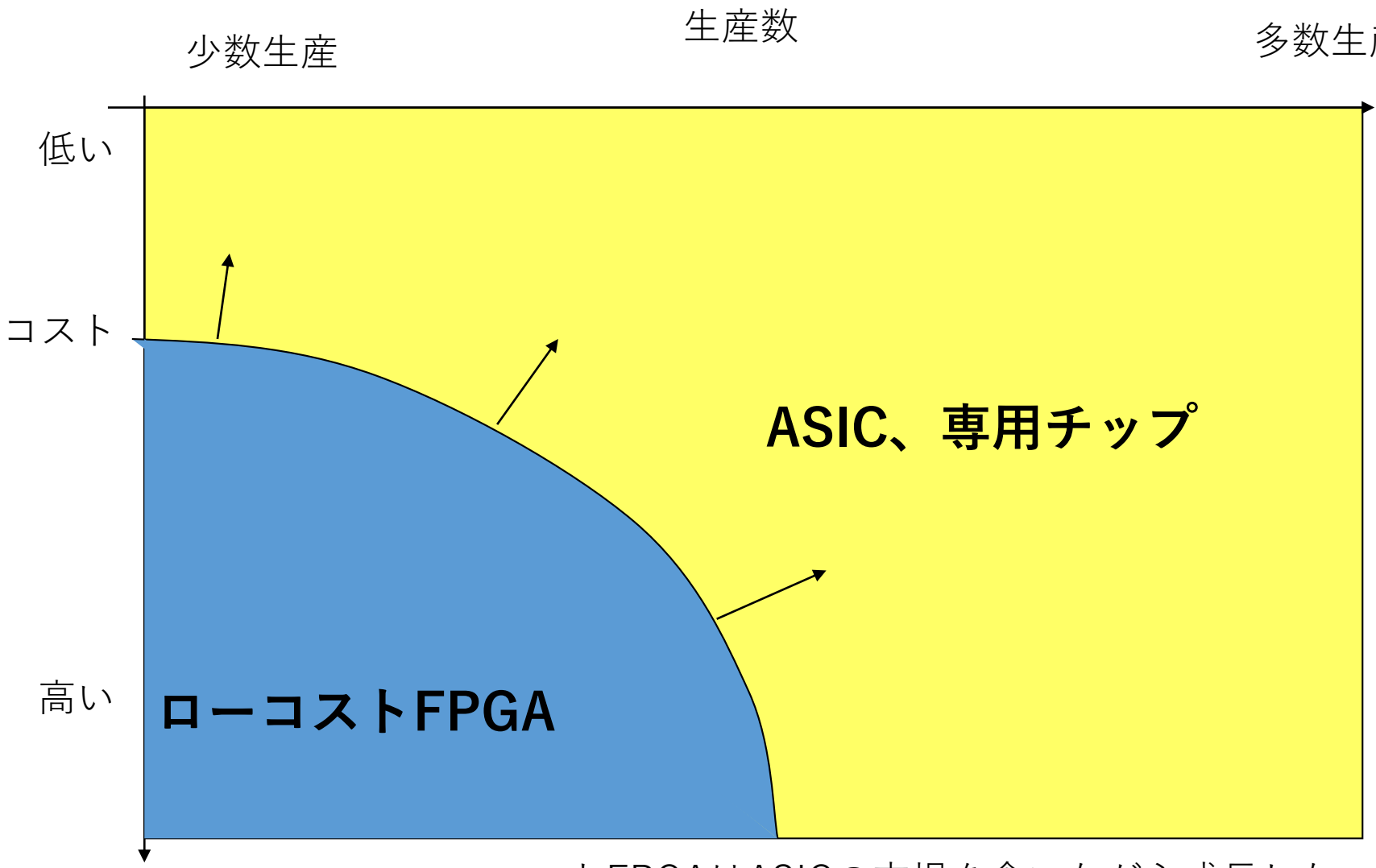
FPGAの発展 つづき


28nm
20nm
16nm
10nm
Virtex-7
2000000LC
Virtex-Ultrascale
5541000LC
Virtex-Ultrascale＋
3780000LC
Stratix-V
/E/GX/GS/GT
359200ALM
Stratix-10
ARM＋FPU
Kintex-7
480000LC
Kintex-Ultrascale
1451000LC
Kintex-Ultrascale＋
1143000LC
Arria-IV
174000LE
Arria-10
ARM＋FPU
Artix-7
360000LC
Cyclone V
/E/GX/GS/GT
301000LE
Cyclone 10
LE／ GX

FPGA vs. ASIC

- ASIC(Application Specific IC) 特殊目的用IC
  - 製品に特化したICチップを特別に開発するもの
  - スマートフォン用、カメラ用、地デジ用、etc.
  - SoC(System on a Chip)、システムLSIと呼ばれ日本の半導体の主力だった
  - 最近のプロセスは開発コストが高騰し、非常に多くの個数を開発しない限り元が取れないことが多い
  - ビジネスとして難しい
  - 本来、FPGAよりも高速、低消費電力、低コスト
- FPGAの低価格品は、ASICの市場を食いつぶして成長
- FPGAは最新プロセスを使うため、小規模なチップでは性能的にはASICとあまり変わらない
- 消費電力、コスト（大量生産できれば）の点でASICが有利

ローコストFPGAの進撃



ローコストFPGAはASICの市場を食いながら成長した

最新プロセス取り入れモデルの限界

- 以前に比べて新シリーズ投入が遅れている
  - FPGAといえども内部構造が複雑化しすぎている
    - Stratix10の遅れの一つの原因
  - ASICはもっと遅れているのでアドバンテージはキープしているが、追いつかれつつある
    - Artix7は28nmプロセスが3万円程度で利用可能
    - 小規模なCPLDの需要は存在
- ASICは低コストFPGAの挑戦を退けた？
  - Artix, Cycloneは最新プロセスを使わない（使えない）
- 高性能FPGAは高性能CPU、GPUと同じ土俵で勝負せざるを得なくなった。

2-3.　IP（Intellectual Property)とSoC型 FPGA

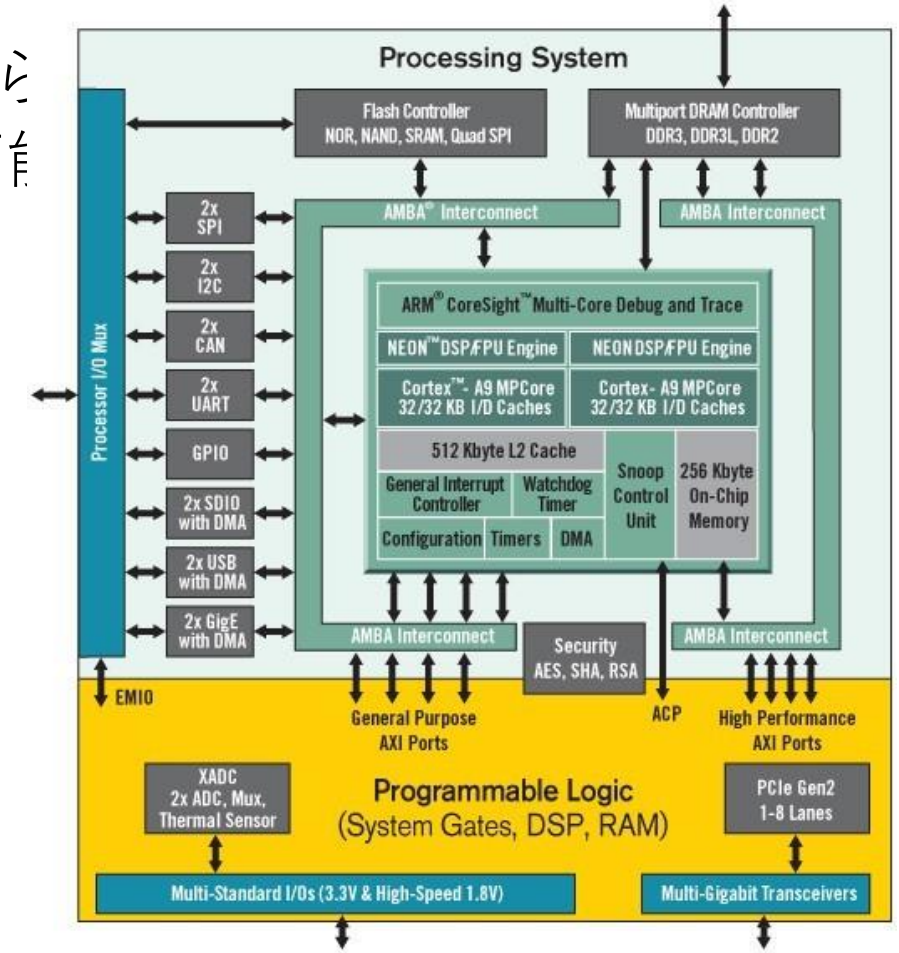
- 標準装備IP
  - 内蔵RAM
  - Multiplier, DSPモジュール（乗算器＋ALU)
  - Clock Manager
  - 高速シリアルI/O（ハイエンド、ローエンドの一部）
- 最近のIP
  - PCI Express（Virtex-6, Stratix－IV)
  - EthernetのMACコントロール（Virtex-6)
  - DRAM用メモリコントロールブロック（Spartan-6)
  - ハードコアCPU　→　ARMの普及(Arria10、Zynq)
  - 浮動小数演算器　Area 10、Stratix 10

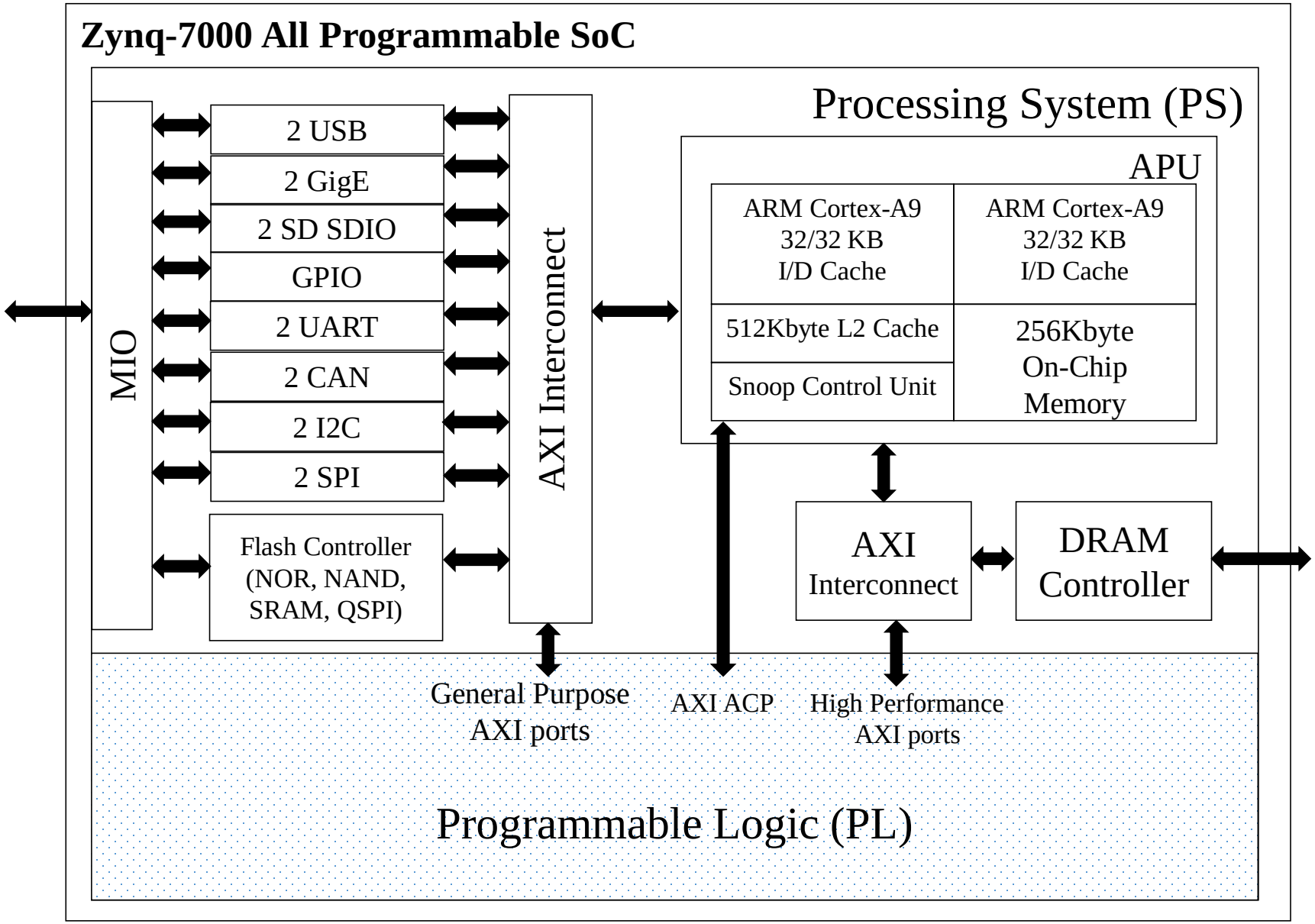
ハードコアとソフトコア

- ハードコア
  - レイアウトレベルで組み込んである
  - 標準的な命令セットを持つ
  - 利用可能なFPGAはかなり高価となる
  - ARM　（Xilinx　Xynq,　Intel(Altera）　Aria-10、Stratix10）
- ソフトコア
  - FPGA上のロジックブロックで構成する
  - 一定のサイズがあればどの製品でも利用可能
  - MicroBlaze　(Xilinx)
    - 32ビット長、32レジスタを持つ
    - 約500スライス、85MHz動作(Spartan-3)
  - PicoBlaze　(Xilinx)
    - 18ビット長、16レジスタを持つ
    - 96スライス、44MHz動作(Spartan-3)
  - Nios (Altera)
    - 16ビットのNios16と32ビットのNios32
    - 状況に応じて構成をチューンすることが可能
  - Cortex-M1(Actel)
    - ARM互換32bitプロセッサ

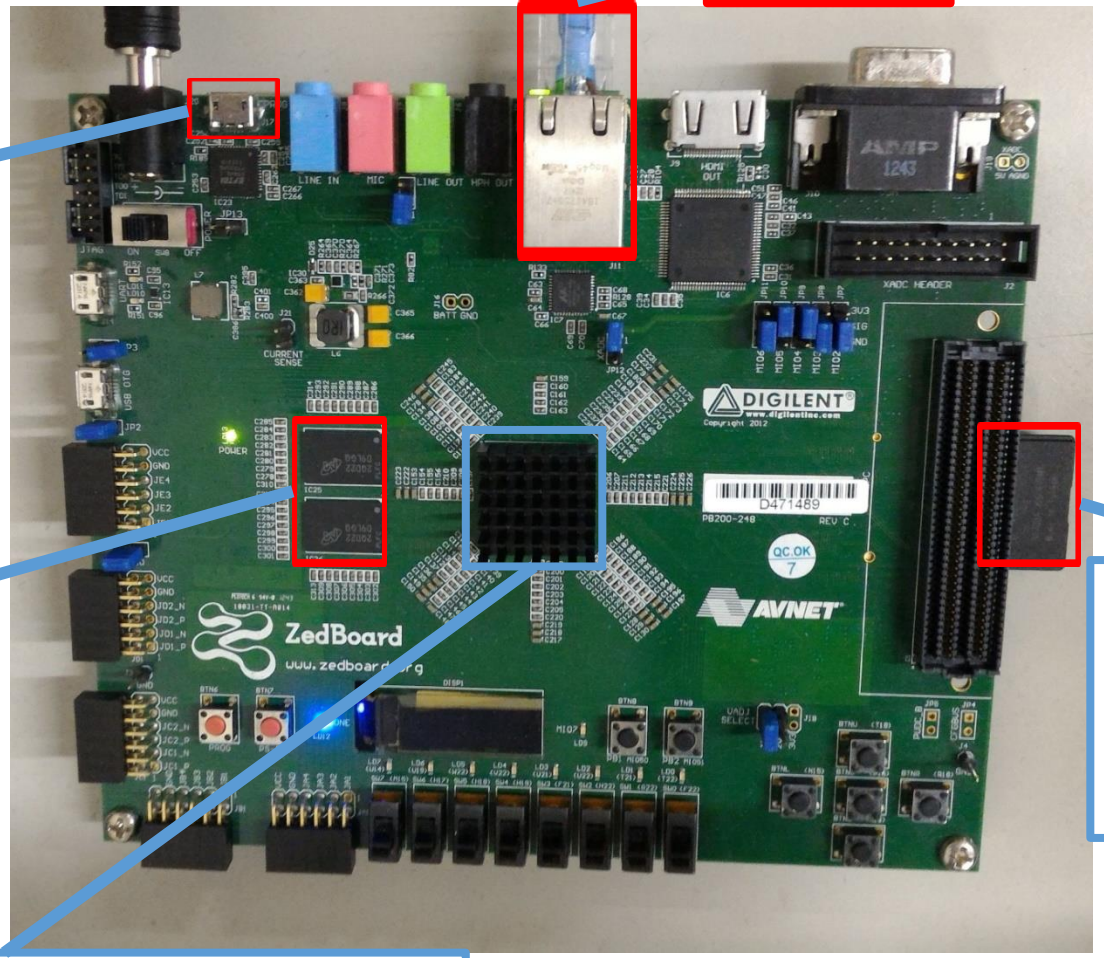
Zynq All programmable SoC

- ARM Cotex-A9プロセッサ（PS part)と、28nm Artix-7/Kintex-7 ベースのFPGA（PL part)の組み合わせ
- Vivado開発環境によりソフトウェア、ハードウェアの統合開発が可能
  - ソフトウェアで動かしてから
    HLSで簡単にオフロードが可能
- 安価はテストボードが普及
  - ZYBO Zynqボード
  - Zed　Board
  - MicroZed　Board




Zynq-7000 All Programmable SoC
Processing System (PS)
APU
ARM Cortex-A9
32/32 KB
I/D Cache
ARM Cortex-A9
32/32 KB
I/D Cache
512Kbyte L2 Cache
Snoop Control Unit
256Kbyte
On-Chip
Memory
MIO
2 USB
2 GigE
2 SD SDIO
GPIO
2 UART
2 CAN
2 I2C
2 SPI
Flash Controller
(NOR, NAND,
SRAM, QSPI)
AXI Interconnect
AXI
Interconnect
DRAM
Controller
General Purpose
AXI ports
AXI ACP
High Performance
AXI ports
Programmable Logic (PL)

ZedBoard



Etherne
t

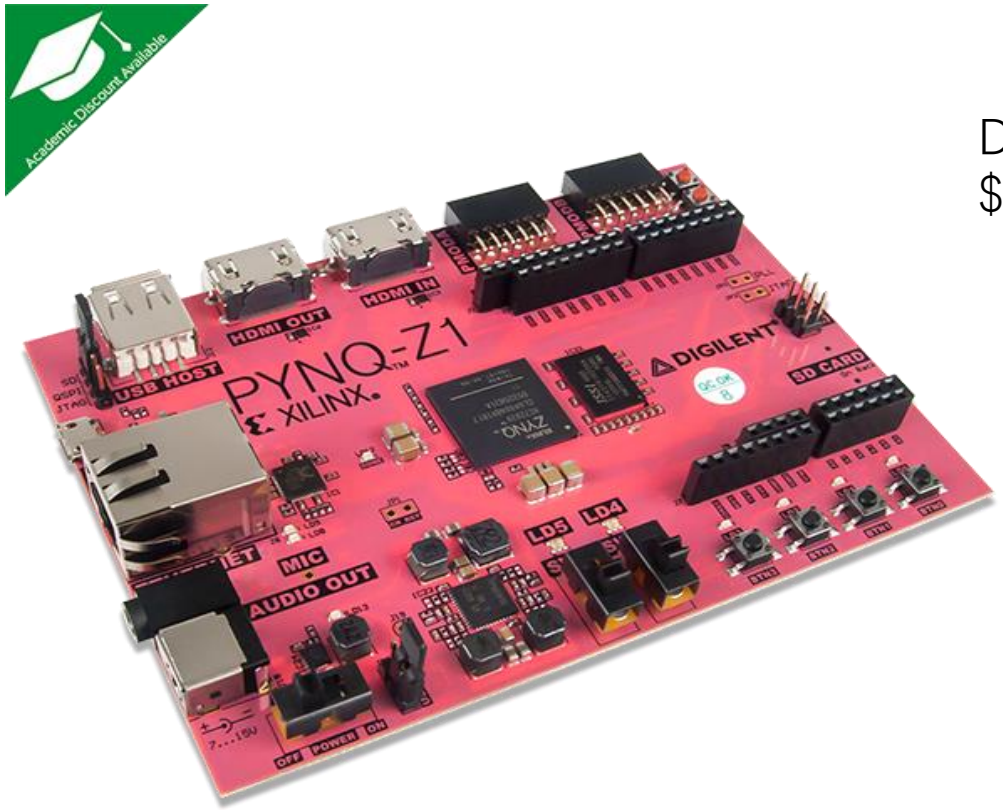
JTAG

DDR3 512MB

SD Card
installed
Linaro
Ubuntu

Zynq-7000 All Programmable
SoC
XC7Z020-CLG484-1

PYNQ：Python+Zynq



DegilentのWebより
$229

PYNQとは？

- ハードウェアはZynq
- ARM上でLinuxが動作し、Pythonが使える
- ハードウェア部はPythonから使えるライブラリの形で供給される
  - Pythonで高位合成ができるのではない
- Pythonを使って、PL部でBNNによるDeep Learningのアクセラレーションができる
- FPGAによるDeep Learningの導入に優れた環境

４.アクセラレータとしてのFPGA

- Stratix 10
  - 14nm Intelプロセス利用
  - HyperFlexの採用によりGHｚ台の動作周波数
    - 配線構造上にレジスタを置く
  - 最大10TFLOPSの浮動小数DSPモジュール
  - ARM Cortex A53 Quad Core
- Arria 10
  - 20nm TSMCプロセス
  - 最大1.5TFLOPS
  - SoCタイプはDual Core ARM Cortex

Open-CLの利用

- GPU同様のアクセラレータとして扱う
  - ホストから入力データを転送→処理を起動→結果を収集
  - ホストの選択
    - PCIe経由でIntelのCPUを使う
    - SoCタイプでは内蔵ARMを使う
- BSP(Board Support Package)が必要
  - ボード依存性を吸収
  - 自作ボードにチップを使ってOpenCLを使うのが極めて難しい→ボード単位で利用するしかない
- HDLモジュールとの接続が難しい

Arria10 SoCボード

SSDより簡単にLinuxがブートする
Ethernetでネットワークに接続
内蔵ARMをホストにOpenCLでの設計ができる

ではGPUより速いのか？

	Stratix 10	Tesla P100
最大TFLOPS	9.2？	9.3
最大電力（W）	33－45	250
価格	177万（開発キット）	81万（Amazon）

- まだStratix 10とGPUとの比較は国際会議などでは出てきていない
- Arria10の場合は、アプリケーションとチューニングのテクニックによるがGPUには絶対性能では勝てない場合が多い。しかし電力性能では勝つ
- コストは現時点では不明（出だしなので高価すぎる）だが、Stratix Vを考える（シリーズによるが1チップ当たり130万円くらいする）と、GPUに比べて倍以上するのでは？

アクセラレータとしてのFPGAは成功するか？

- OpenCLの導入によりソフトウェア開発者、スパコン屋が利用可能になった
  - しかしFPGAの良い所が半減したかもしれない
- 科学技術計算では、GPUに性能面では勝てそうもない
  - Stratix 10ならば良い勝負ができるかもしれない
  - しかし値段が相当違う、、、
  - そのうちスパコン屋に飽きられるのでは？
- AI分野は？
  - OpenCLを使っているとGPUに比べてメリットが少ない
- Intelには勝算があるのだろうか？？？

OpemCLの無料化
intel HLSの登場

ソリューションの提供

- Stratix10クラスの性能を持ち、ターゲットアプリケーションが決まっている（例えばCNNの学習）場合
  - HDLを使って専門家が根性でチューニングする
  - 性能、電力でGPUに勝てる可能性がある
- カスタマがはっきりしていれば勝機がある

→Maxelerのビジネスモデル

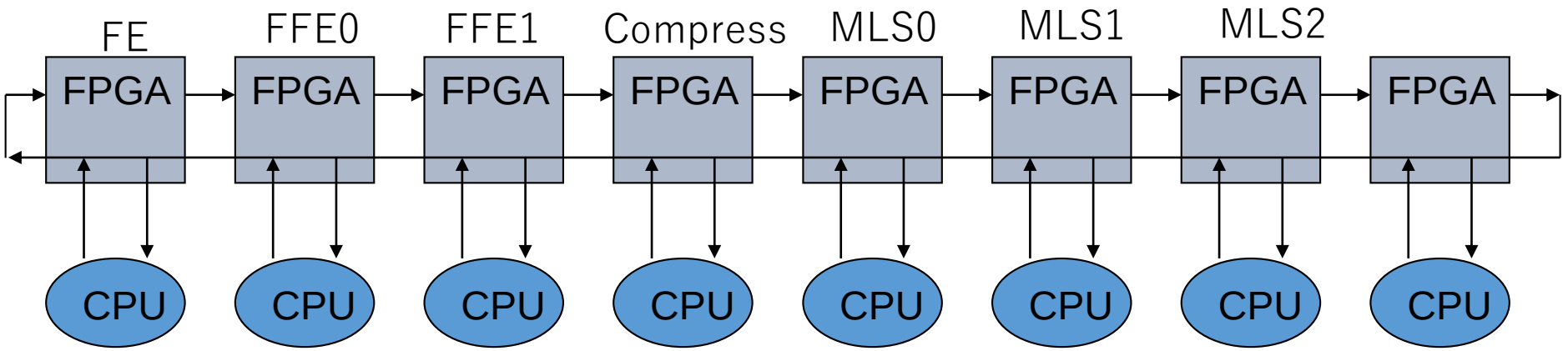
- Cloudで提供すれば機会が広がる

5.FPGA in Cloud

- Catapult project　by Microsoft [ISCA14他]
  - FPGAを用いた検索エンジンの導入
- FPGAの仮想化 [FPGA17]
- FPGA Supervessel Cloud by IBM
  - FPGAによるサービスの提供[ICFPT2016]
  - Xilinx SDACcelによるアプリケーション開発
    - OpenCLに似た開発環境
- Amazon EC2 F1インスタンス
  - Cloud上でのFPGAアプリケーション開発環境
  - Xilinx Ultrascale+を利用
  - 衝撃的な内容だが良くわからない

Microsoft's Catapult

Rank computation for Web search on Bing.
Task Level Macro-Pipelining (MISD)
FE: Feature Extraction
FFE: Free Form Expression: Synthesis of feature values
MLS: Machine Learning Scoring



FPGA: Altera's Stratix V

2-Dimensional Mesh is formed (8x6) for 1 cluster.

元々CloudにはFPGAが使われている

- ネットワークインタフェース、ネットワーク用スイッチはFPGAの主戦場
- ネットワークの機能を拡張するのはFPGAの利用法としては正統的
- 開発はベンダに限定、難しい
- NetFPGAは研究機関にも開放している

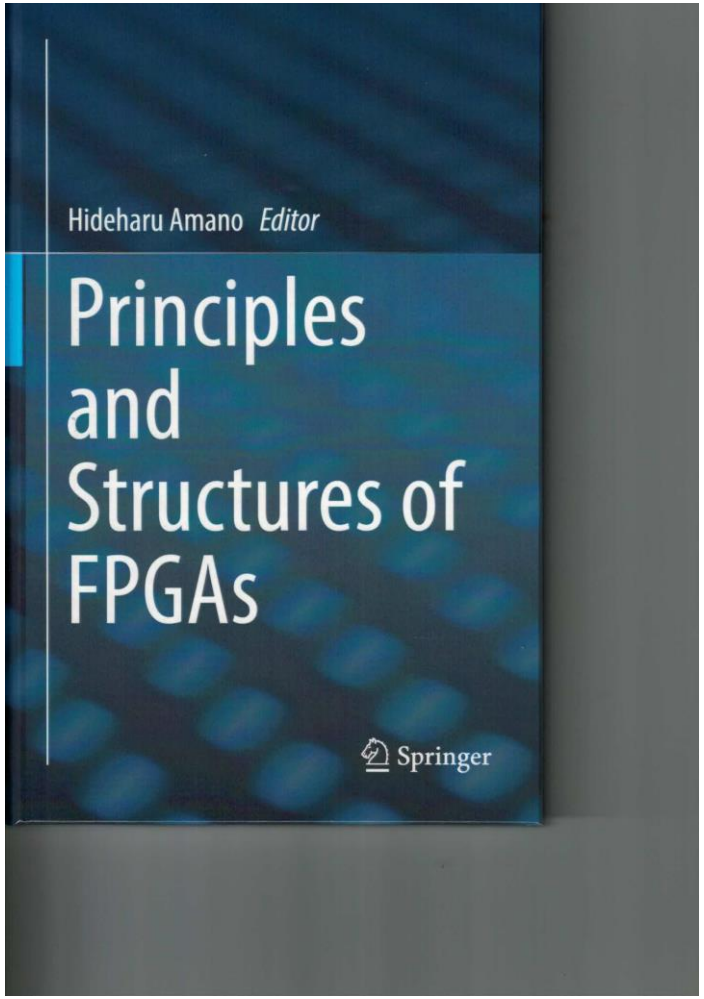
FPGA in Cloudの将来

- Cloud Computingのサービスの一部をFPGAが行うのは自然の流れ
  - Catapultなどの検索エンジン
  - 画像、文章などの認識処理
- ネットワーク制御の拡張は将来性がある
- CloudでFPGA開発を行う試みはどうか？
  - 環境を囲い込めるのでメリットはある
  - EC2　F1インスタンスは革命的
    - うまく行けばFPGA設計者にとってすごいことになる
    - しかし現実が付いてこないようだが、、、

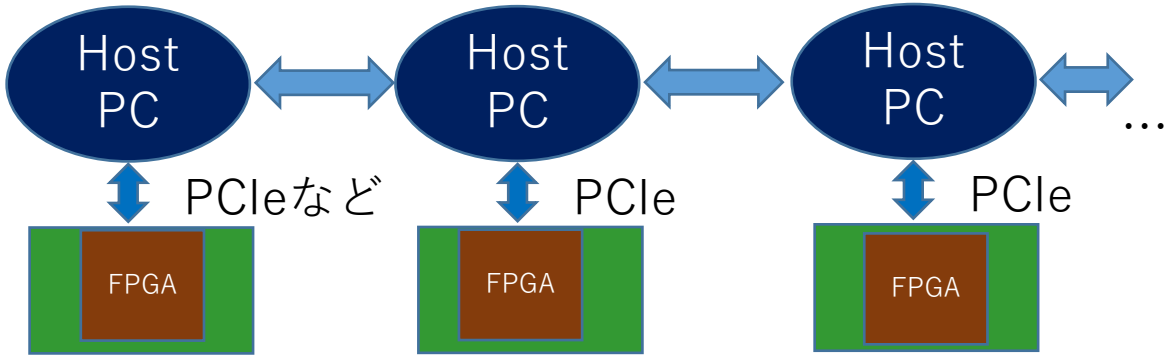
ここまでは一般論

宣伝

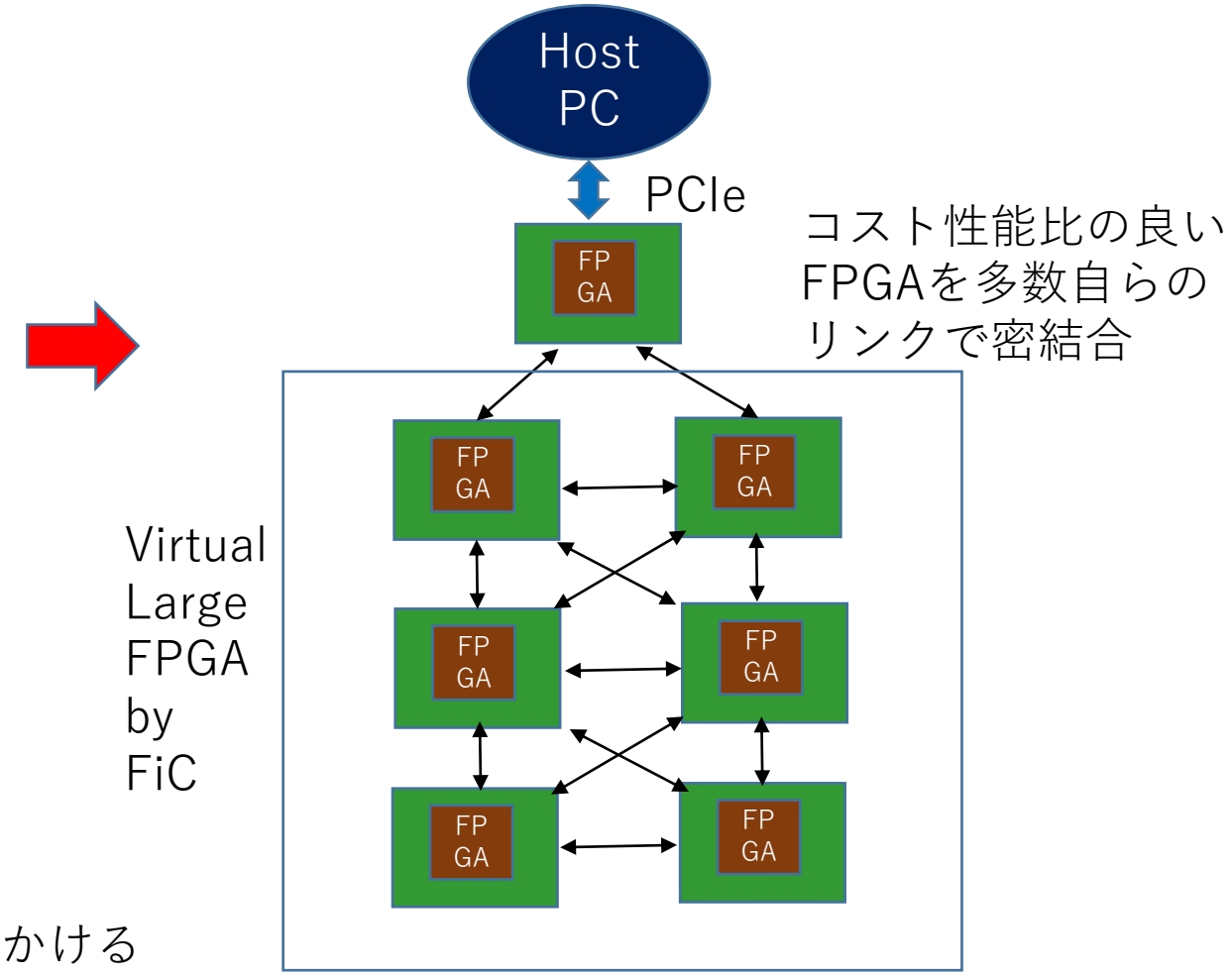

FPGAの
原理と構成
天野英晴 編
尼崎太樹・飯田全広・泉知論・長名保範・佐野健太郎・
柴田裕一郎・末吉敏則・中原啓貴・張山昌論・丸山勉・
密山幸男・本村真人・山口佳樹・渡邊 実 共著
1章—FPGAを理解するための基本事項
2章—FPGAの概要
3章—FPGAの構成
4章—設計フローとツール
5章—設計技術
6章—ハードウェアアルゴリズム
7章—PLD/FPGAの応用事例
8章—新しいデバイス, アーキテクチャ
Ohmsha


Hideharu Amano Editor
Principles
and
Structures of
FPGAs
Springer

我々のプロジェクト：
Virtual Large FPGA (NEDO)



通常のFPGA in Cloudは個々の
ノードに強力なFPGAを接続
FPGAの通信機能が活かされない

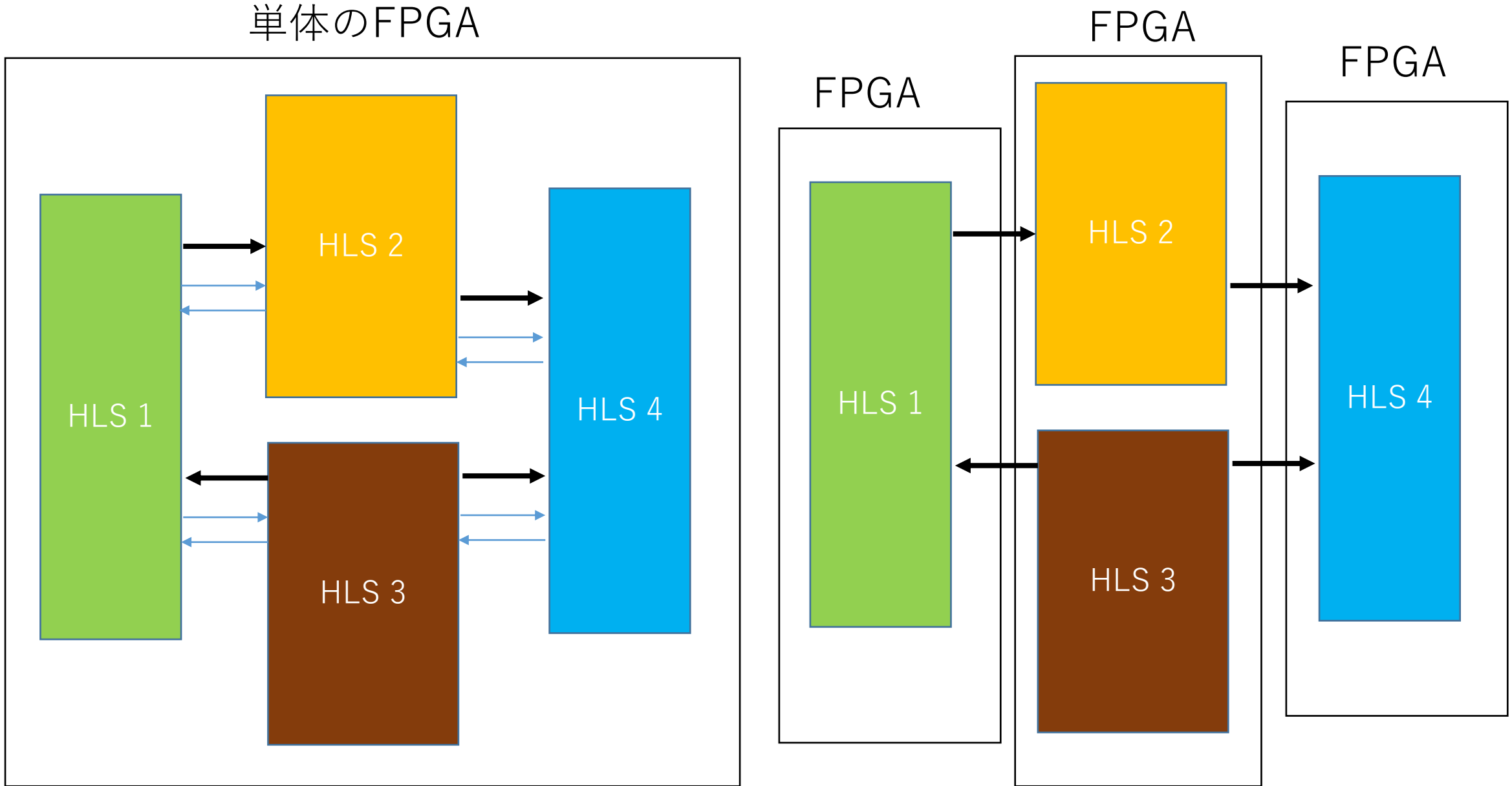


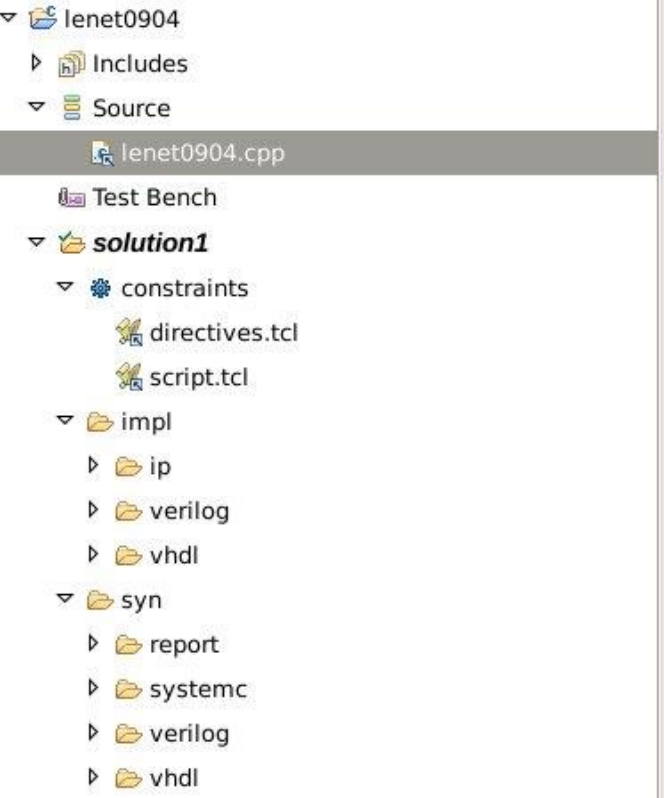
類似のシステムにCatapultがあるが、
ホストPCはなくFPGAの集合体を1つのFPGAに見せかける
Catapultより個々のFPGAがコスト性能比重視
ネットワークは強力
HLSのIPの相互接続を単一の巨大FPGAを想定して実現

Multi-FPGA in cloud

- FPGAはクラウドに置いた場合GPUと比べて何が有利か？
  - 演算性能は圧倒的に負ける
  - データ転送性能は有利、そもそもFPGAはスイッチ用素子といっていい
    - NVLINKとかあるが安価なGPUでは使えない
    - GPUとネットワークとの遅延、ホストによる制御が必要
  - 電力性能は優れている
- コスト性能比の良いFPGAを自身のネットワーク機能で密結合して、全体として超巨大な演算用のFPGAに見せかけられないか？
  - 現在、大規模なプログラミングはHLSによるIPの組み合わせになっている
  - IP間のハンドシェークがMulti-FPGAでは難しい

ハンドシェークを取らなければ、マルチFPGA化は容易





Project: lenet0904
Solution: solution1
Product kintexu
Target device: xcku095-ffvb2104-1-c

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
21730880	22175328	21730881	22175329	none

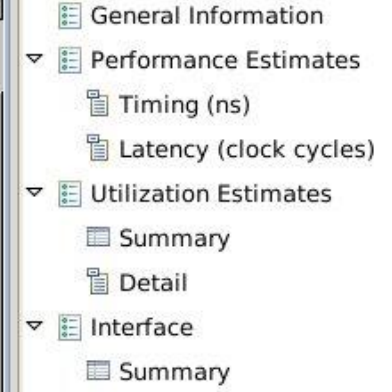
Detail

Instance

		Latency		Interval		
Instance	Module	min	max	min	max	Type
grp_fc2_fu_88	fc2	22793	22793	22793	22793	none
grp_fc1_fu_104	fc1	1841501	1841501	1841501	1841501	none
grp_conv2_fu_116	conv2	16777745	16777745	16777745	16777745	none
grp_load_wb_fu_128	load_wb	444448	444448	444448	444448	none
grp_conv1_fu_150	conv1	3030961	3030961	3030961	3030961	none
grp_pool1_fu_162	pool1	40841	40841	40841	40841	none
grp_pool2_fu_170	pool2	11701	11701	11701	11701	none
grp_load_input_fu_178	load_input	3194	3194	3194	3194	none
grp_flatten_fu_188	flatten	2101	2101	2101	2101	none
grp_store_output_fu_196	store_output	24	24	24	24	none

Loop

Utilization Estimates



Console | Errors | Warnings

Vivado HLS Console

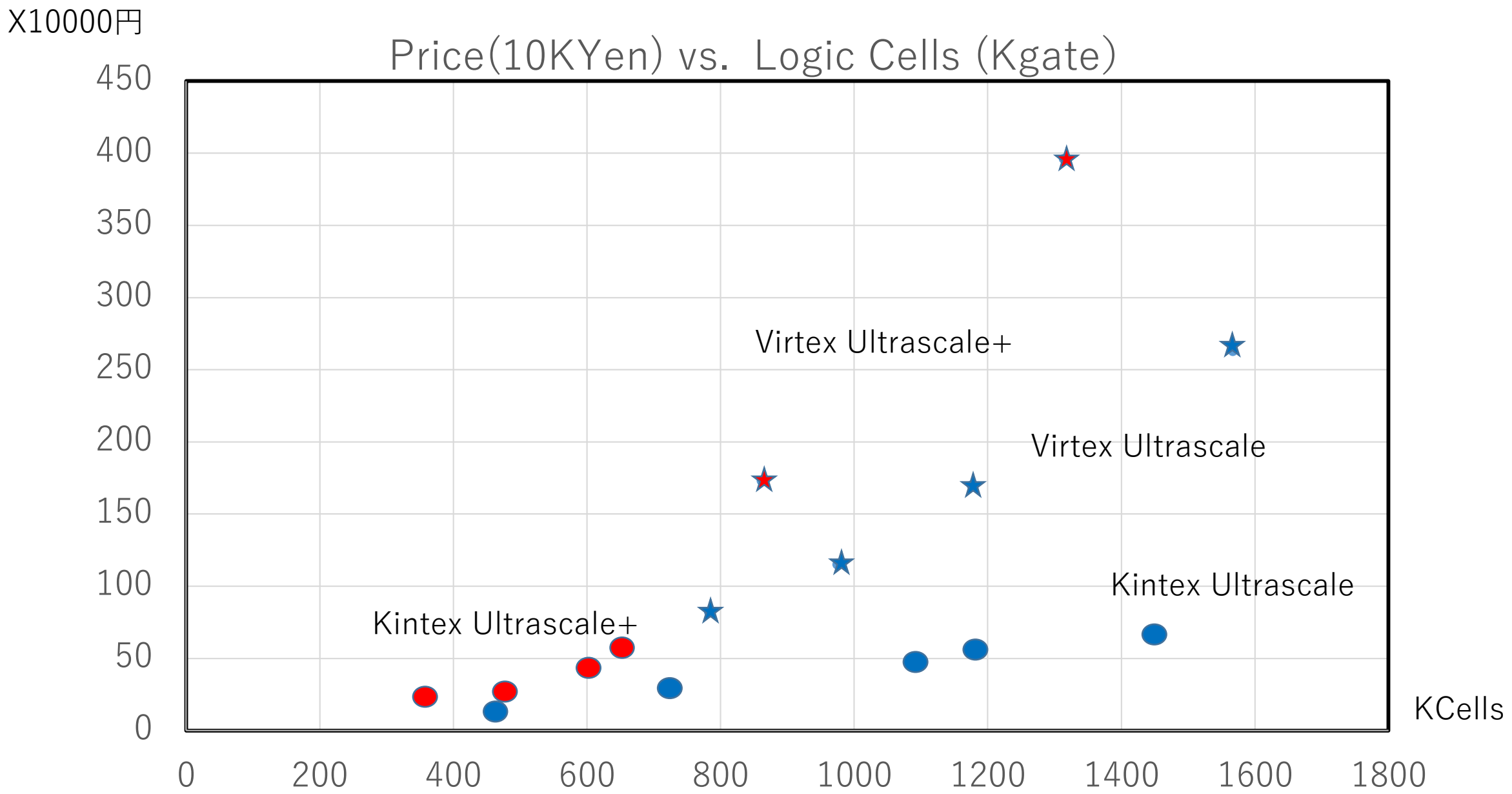
Virtual Large FPGAの概念

- HLSは合成時に1回の反復の実行サイクル数、転送データ量を見積もってくれる。
- FPGAの高速リンクを多数（32）利用してボード同士を密結合
  - 各9.9Gbit/sec
  - 4本1組のFireflyケーブル＝5900円
  - 基板1枚当たり23600円
  - 3-5ホップの遅延で数百ボードを接続可能
- STDM（Static Time Division Multiplexing)の導入
  - 転送時間を予測可能にして、ハンドシェークを省略する
  - スケジューラとマッパ（NII）によりSlot数を削減
  - 全体をスケジュールして、FIFO長、オーバーヘッドを最小化
- Partial Reconfigurationによりスイッチ部とHLS部を完全分離
- 様々なサイズの仮想FPGAを切り出して運用
  - Flow OS（産総研）の利用
- 単一のFPGAボードと同程度のコスト効率、電力効率でスケーラブルに高い性能を実現

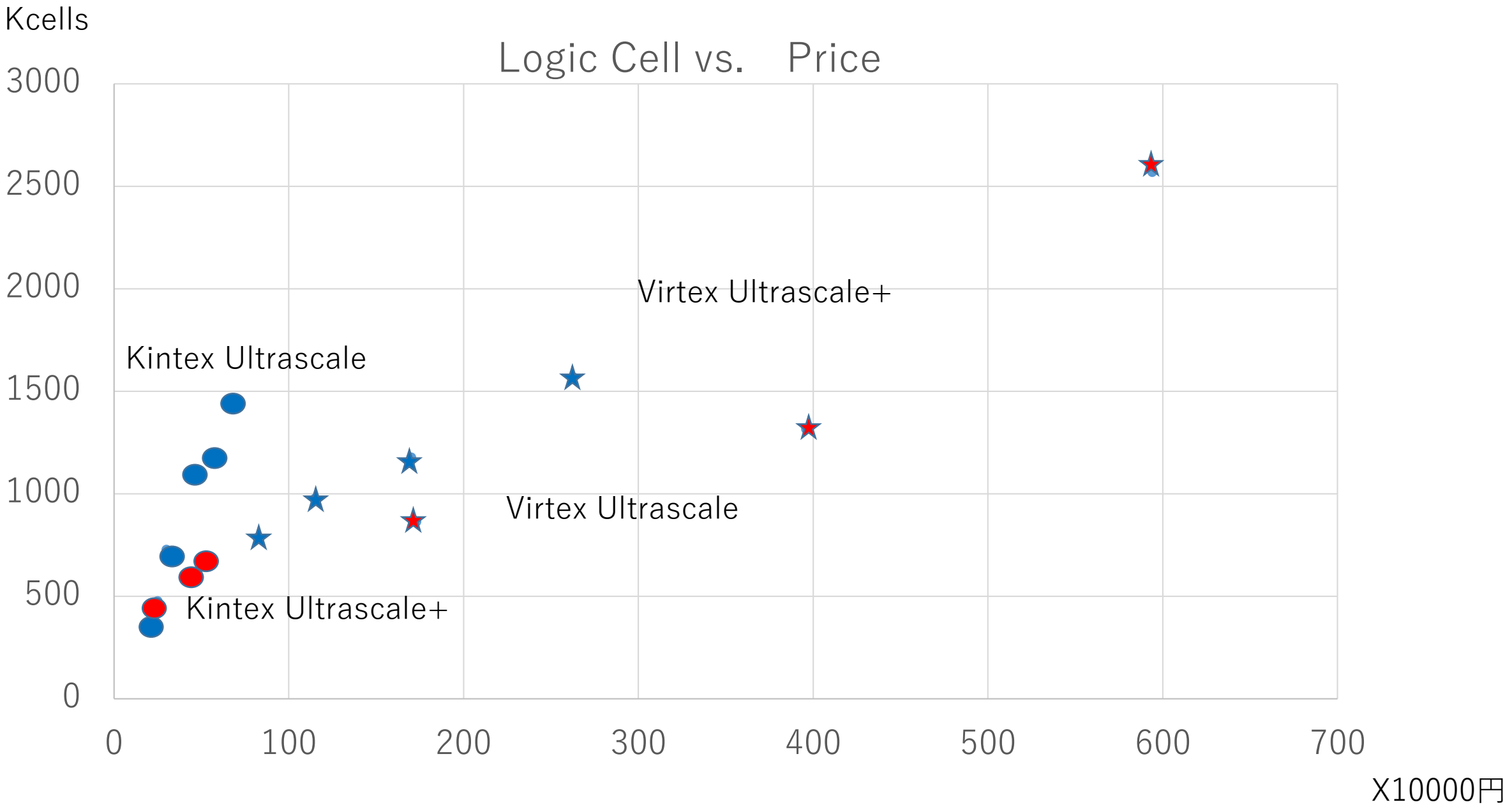
FPGAの性能価格比

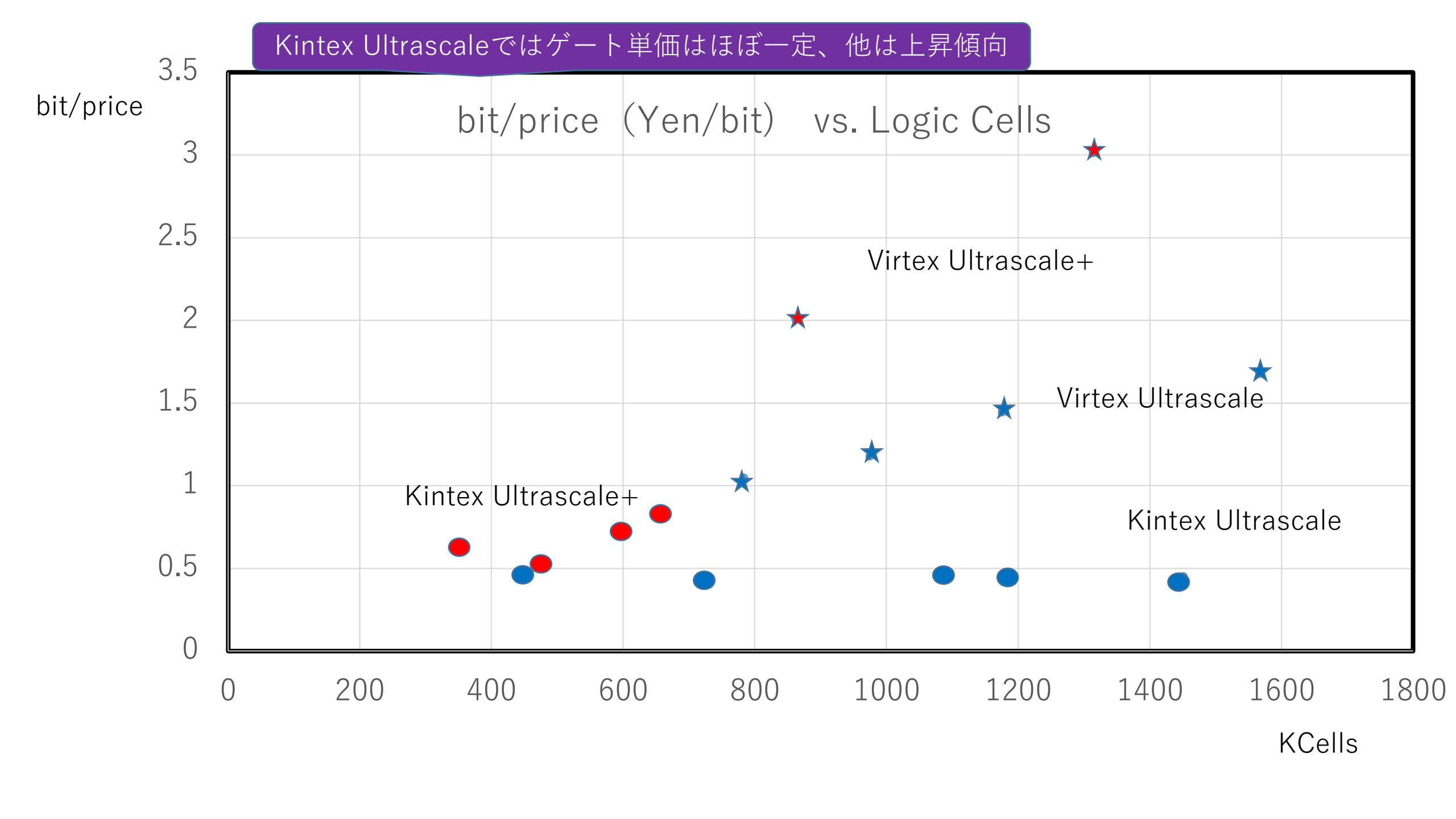
- そもそも性能価格比が高いFPGAを多数高速リンクで接続することで、強力なFPGAに勝てるか？
- Kintex Ultrascale KU085の使用時、
  - 2ボードでVirtex Ultrascale+の最強のFPGAと同等、チップコストは1/5
  - 5ボードでVirtex Ultrascaleの最強のFPGAと同等、チップコストは1/3
- マルチボード化のオーバーヘッドは主として高速シリアルケーブル
- そもそもFPGAのロジックセルと価格の関係はどうなのか？
  - 価格はDegikeyによる
  - 同じ種類でも結構幅があるので目安と考えて欲しい

搭載ゲート数と価格は直線あるいはややオーバーハング



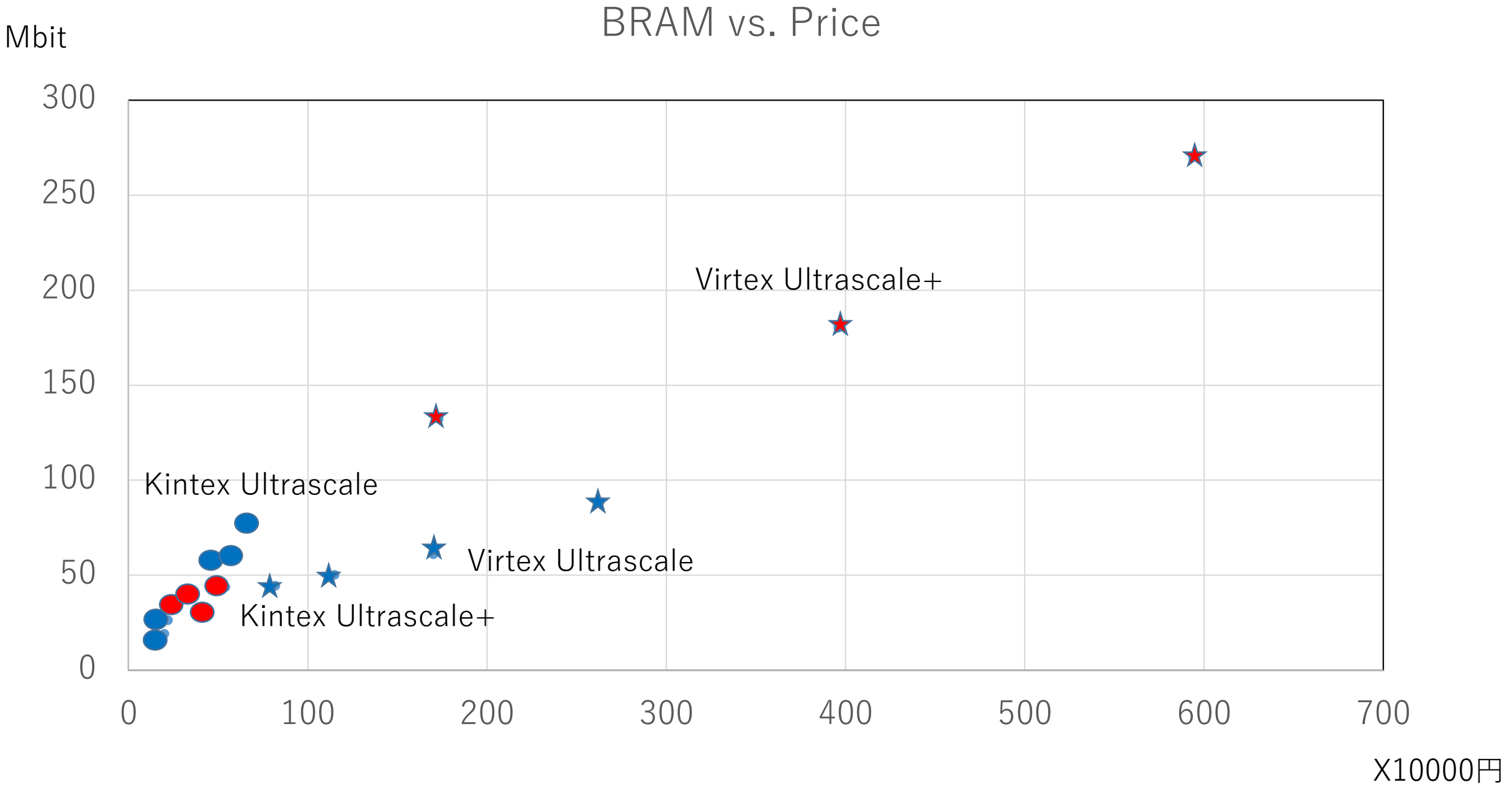
お金を縦軸に取ったもの、左上がコスト比が高い

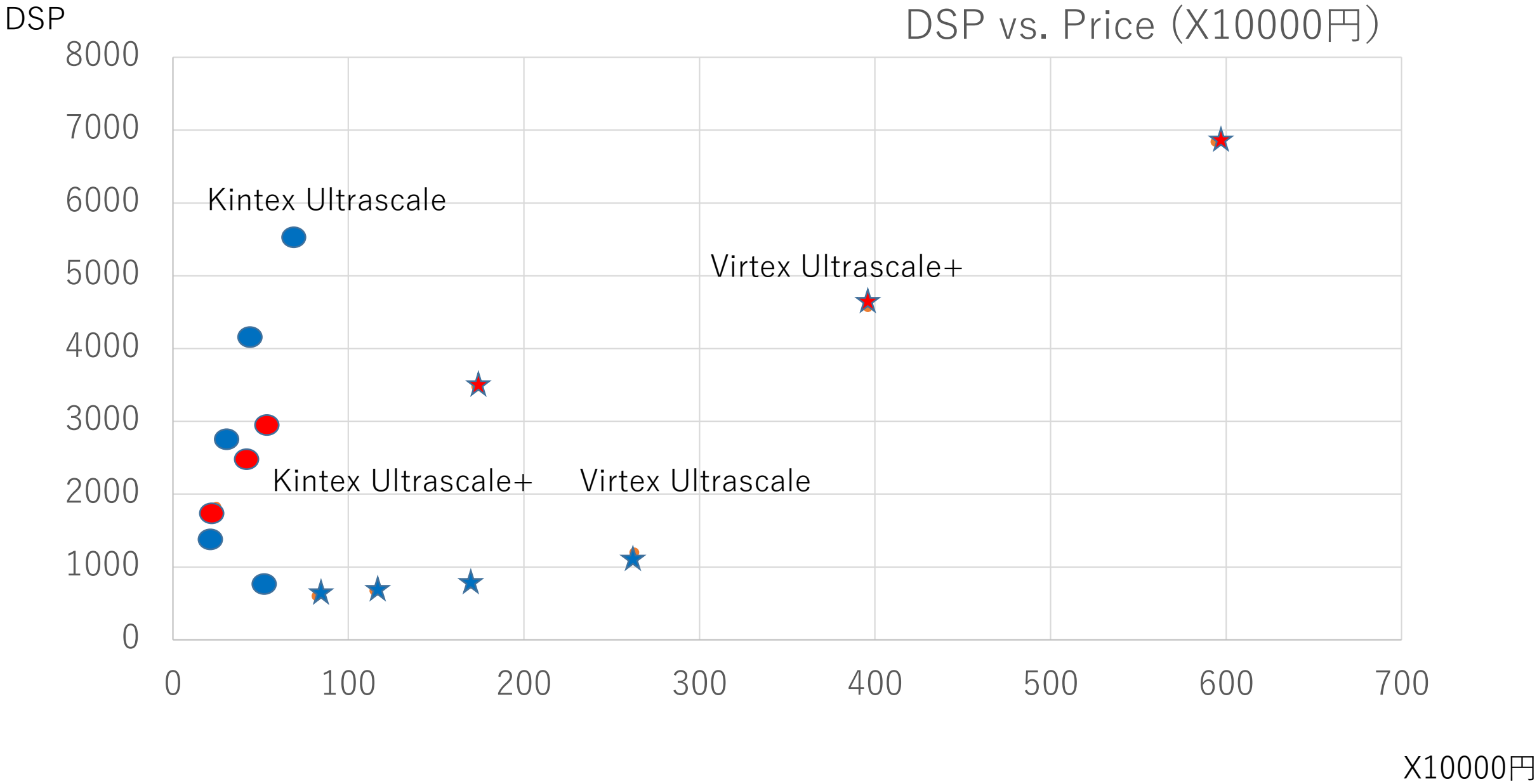



Kintex Ultrascaleではゲート単価はほぼ一定、他は上昇傾向
bit/price
bit/price（Yen/bit） vs. Logic Cells
3.5
3
2.5
2
1.5
1
0.5
0
0
200
400
600
800
1000
1200
1400
1600
1800
KCells
Virtex Ultrascale+
Virtex Ultrascale
Kintex Ultrascale+
Kintex Ultrascale

ロジックセル当たりの価格

- Kintex Ultrascaleでは、Logic Cell単価はサイズに依らずほぼ一定
  - 0.4円/Logic Cell程度
- 他のシリーズは、サイズに対してやや増大する傾向がある
  - Virtex Ultrascale:　1～1.6円/Logic Cell
  - Kintex Ultrascale+:　0.6～0.8円/Logic Cell
  - Virtex Ultrascale+:　2～3円/Logic Cell
- Kintex Ultrascaleの高速シリアルリンクを使ってチップを多数接続すれば、低コストで全体として巨大なロジックセル数が得られるはず
  - KU060以上のチップは32本以上リンクを接続可能


BRAM vs. Price
Mbit
300
250
200
150
100
50
0
0
100
200
300
400
500
600
700
X10000円
Virtex Ultrascale+
Kintex Ultrascale
Virtex Ultrascale
Kintex Ultrascale+


DSP
DSP vs. Price (X10000円)
8000
7000
6000
5000
4000
3000
2000
1000
0
Kintex Ultrascale
Virtex Ultrascale+
Kintex Ultrascale+
Virtex Ultrascale
0
100
200
300
400
500
600
700
X10000円

メモリとDSPモジュール数とサイズの関係

- メモリ
  - BRAMのbit数はVirtex UltrascaleよりもKintex Ultrascaleが有利
  - Ultrascale+ではUltra RAMが使える（しかし総容量はさほど大きくない）
  - Kintex Ultrascaleではbit単価がサイズによって若干上がる
    - 0.7円～0.9円/bit

- DSP
  - Kintex Ultrascaleが圧倒的に有利（なぜだろう？）
  - Kintex Ultrasclaleでは、DSP単価はほぼ一定（例外がKCU095）
    - 110円/１DSP

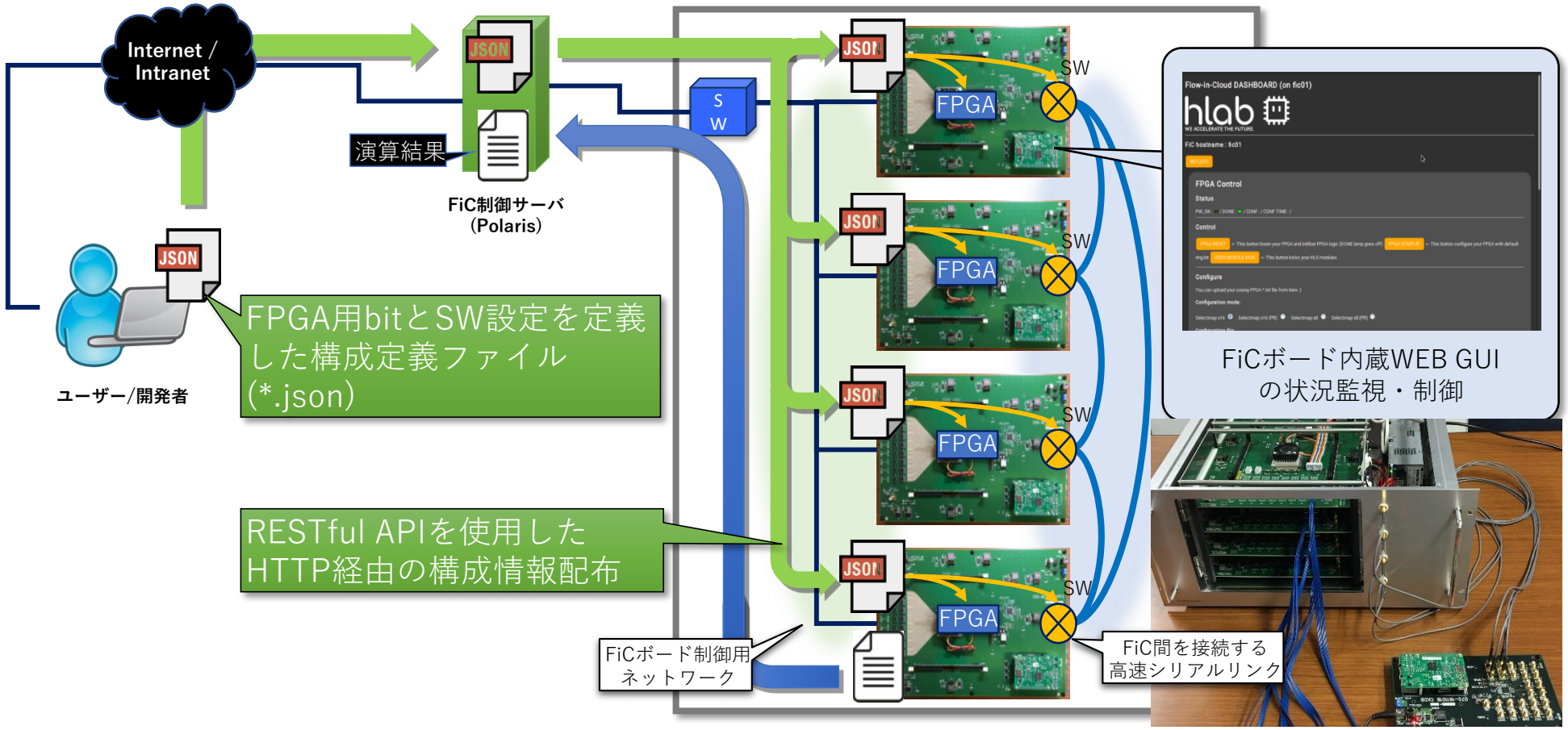
結論：Virtual Large FPGAは可能

- KU085レベルのKintex Ultrascaleが最も性能価格比が高い
  - 1088KLogic cell、56.9Mbit BRAM、4100DSPが45万円程度
  - 高速リンクで密結合、STDMで遅延を予想、スケジュールによりHLSの演算と転送遅延を可能な限りオーバーラップ
  - ケーブルは直結、コストは+23600円
- 単一のFPGAと同じ程度のコスト・電力効率で、スケーラブルに高い性能を実装可能

Virtual Large FPGAシステム

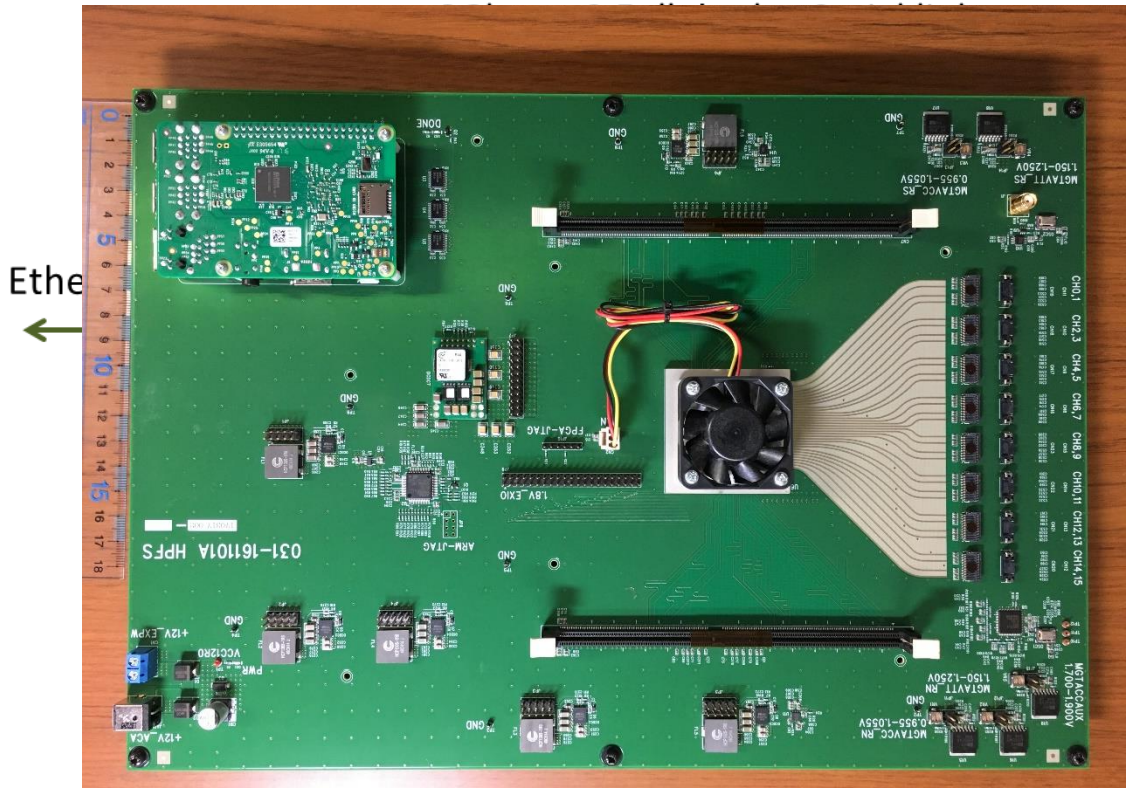
32枚のFiC SW ボードから構成される

9Gbps X 32のリンクで密結合
HLSでアプリケーションを記述



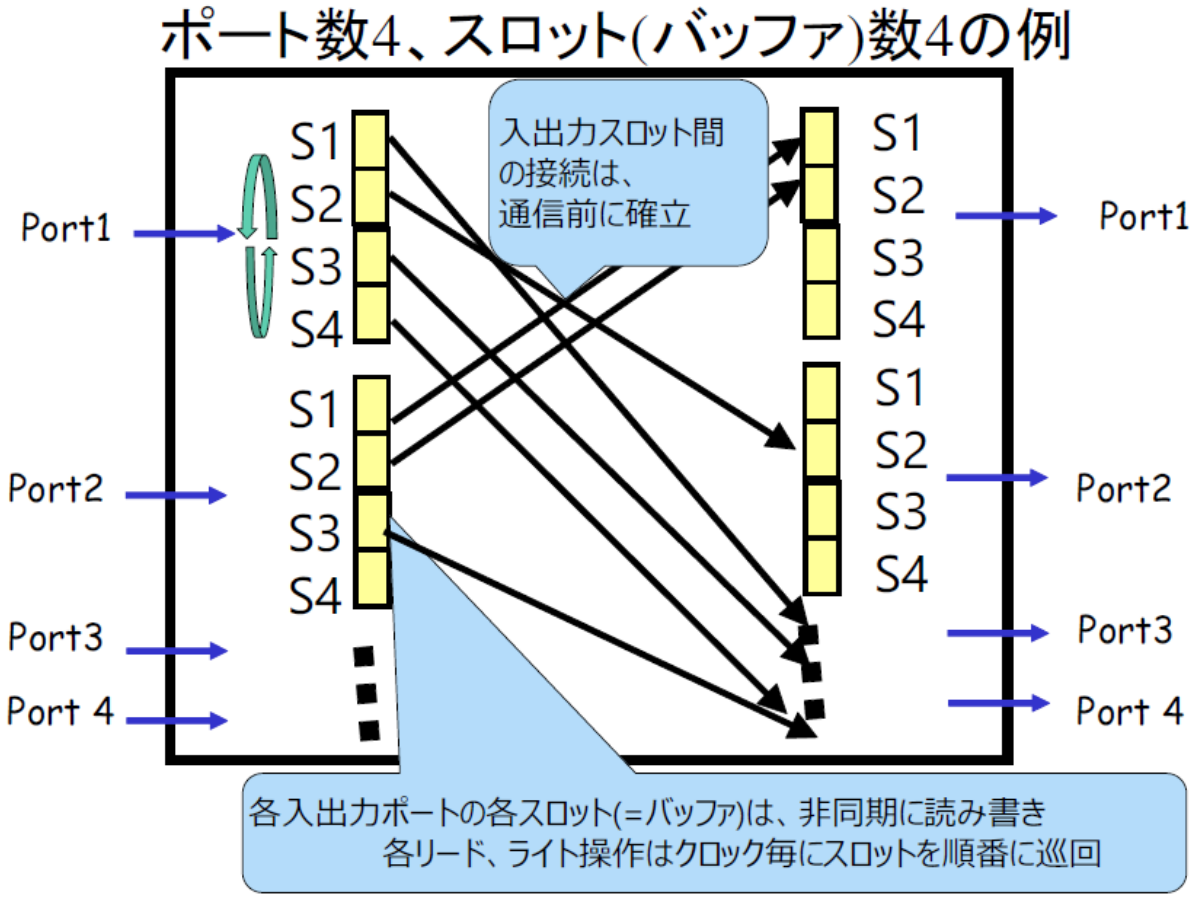
ベースとなるFiC-SW1ボードの構成

- FPGA : Xilinx Kintex Ultrascale XCKU095
- RAM：DDR-4 SDRAM16GB ×4 組
- 9Gbps全二重のシリアルリンク×32組
- （制御用）RaspberryPi 3 ボード+Ethernet … FPGA構成など



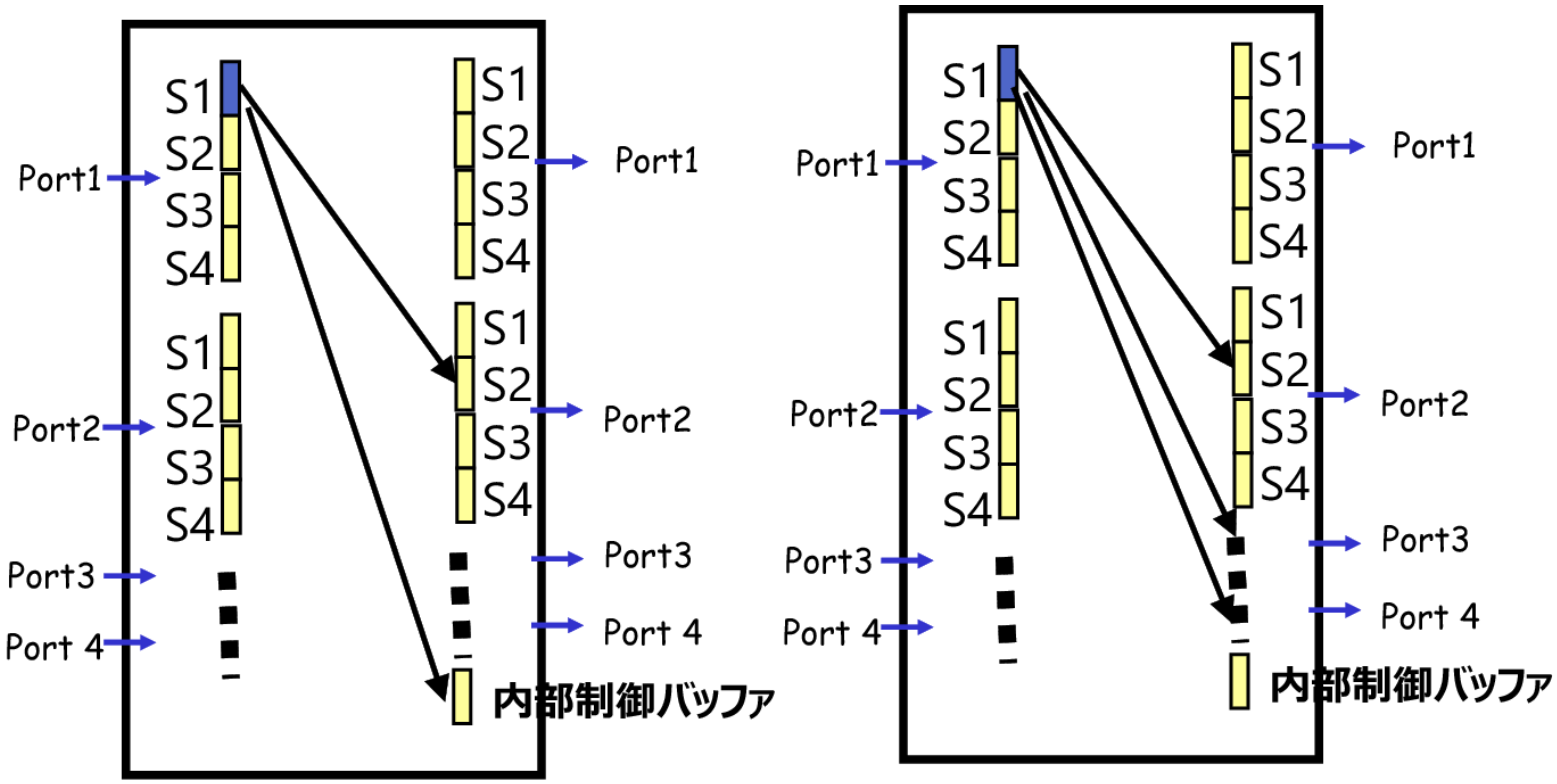
FiC-SWのサーキットスイッチング

- スロットへの書き込み位置により経路が一意に定まるように、入出力スロットの接続を事前に設定
- クロック毎にスロットを巡回し読み書きを行う



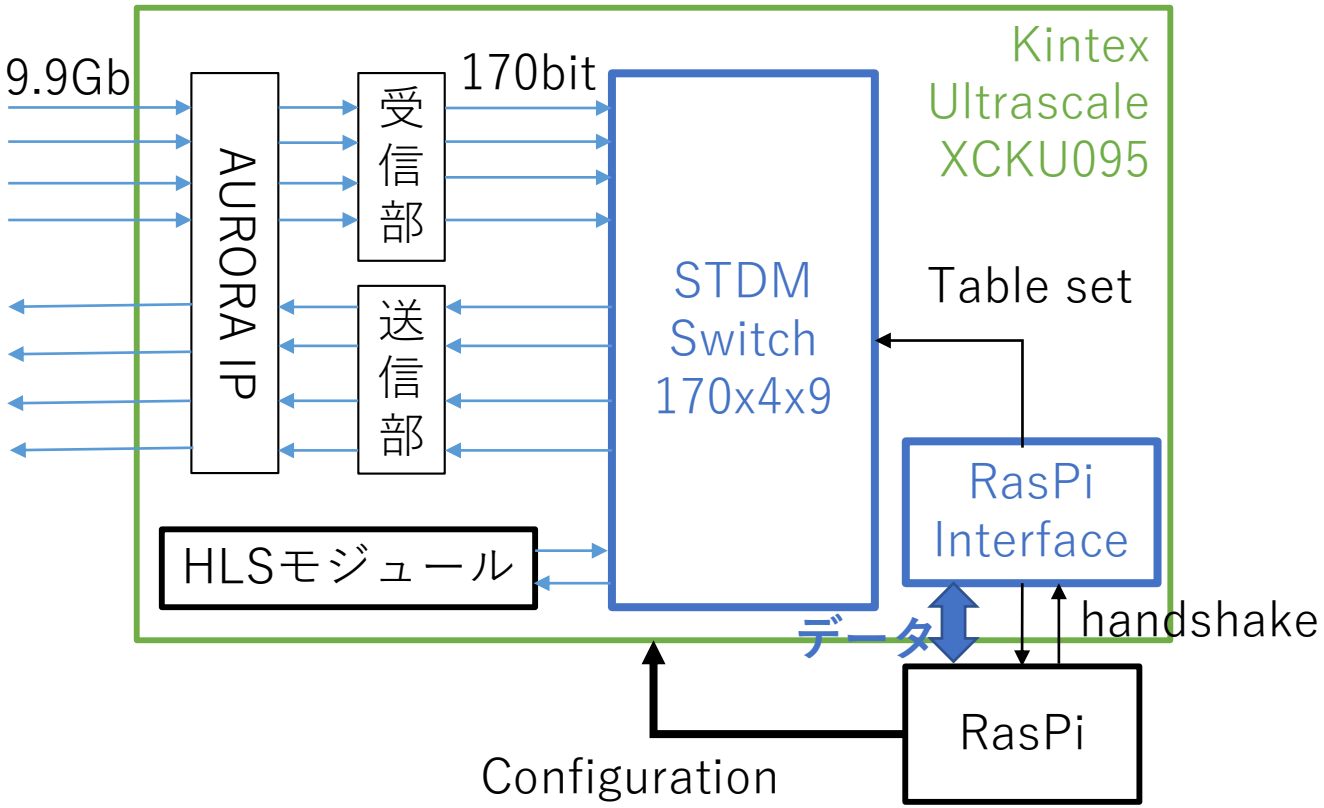
FiC-SWのサーキットスイッチング

- マルチキャストの様子
  - 内部向けのバッファに特定の値を書き込むことで経路を再構成



(a) 内部制御バッファへの書き込み(1:2)　(b) 他スイッチへのマルチキャストの設定例

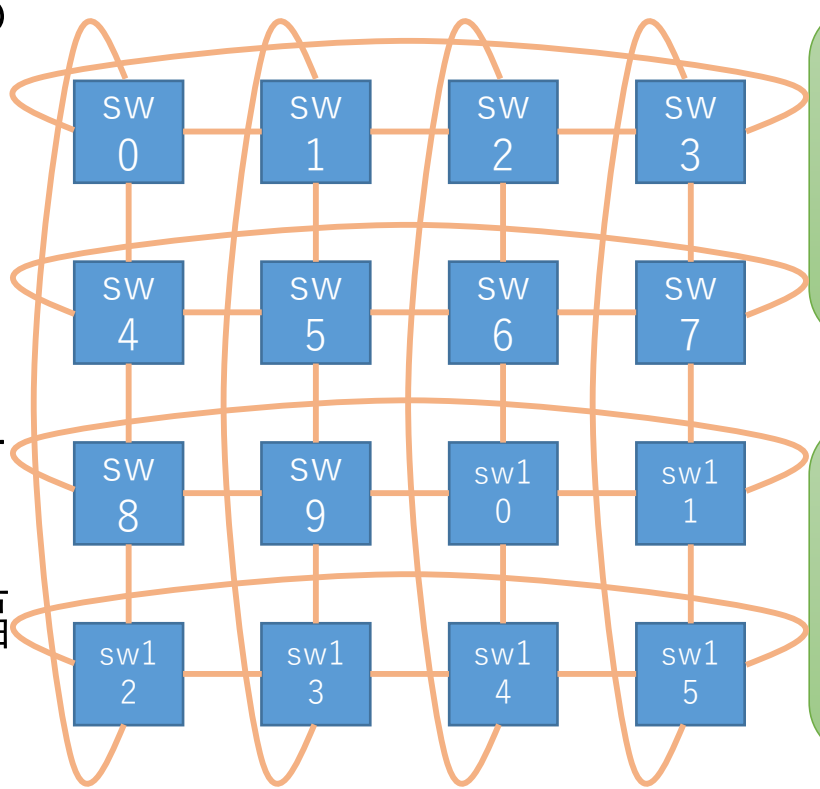
FPGA内の構成



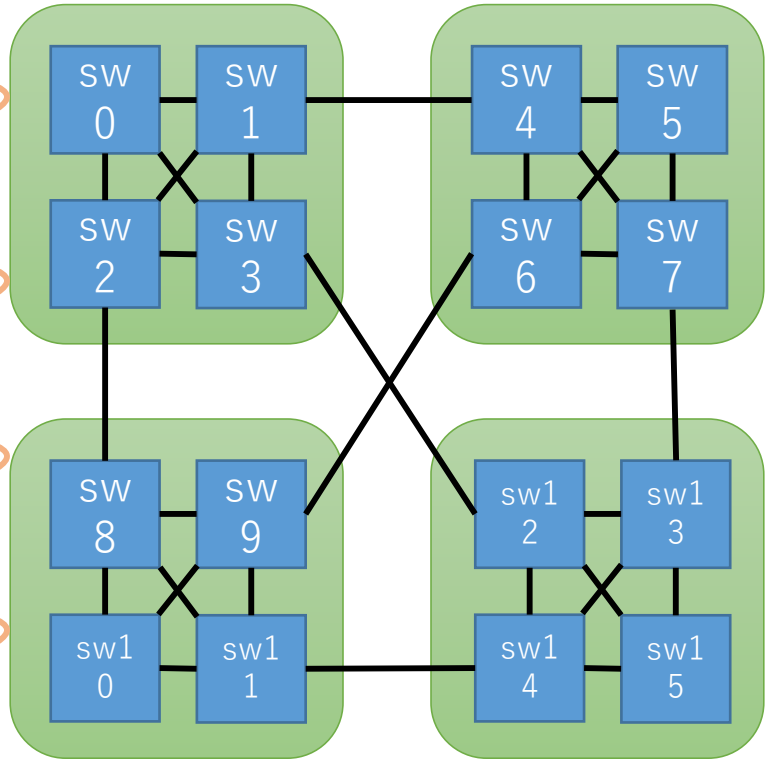
- 最大9つの入出力でスイッチング
  - パラメタライズされたスイッチが必要
- スイッチはポート数+1
  - +1にアクセラレータを接続
  - Acclerator in Swtich(AiS)で演算をオフロード
- AiSにConvolutional Neural Network(CNN)の推論フェーズが実装
- AiS部の部分的再構成が可能
- スイッチのテーブルをRasPiから変更可

SWと接続トポロジ

- ルーティングテーブルを決めるのは　トポロジ
  - 物理的/論理的トポロジー
- STDMスイッチではスロット数が増えるほど性能が低下
- 物理的トポロジーとしてトーラスとDragonfly
- 予測可能な通信遅延と帯域幅
  - 演算と通信をオーバーラップ



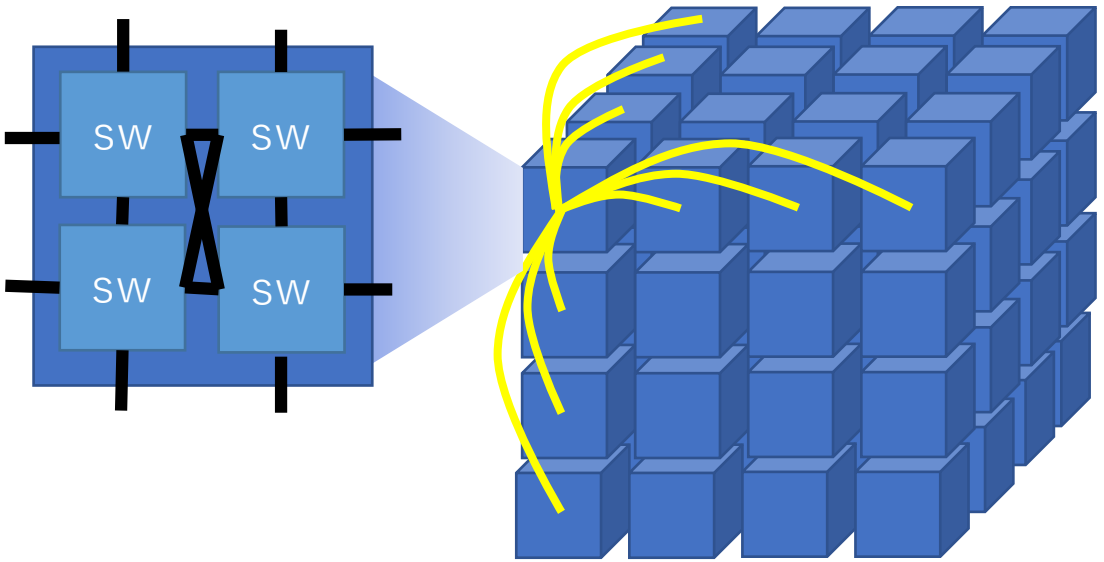
4x4 torus



4x4 Dragonfly

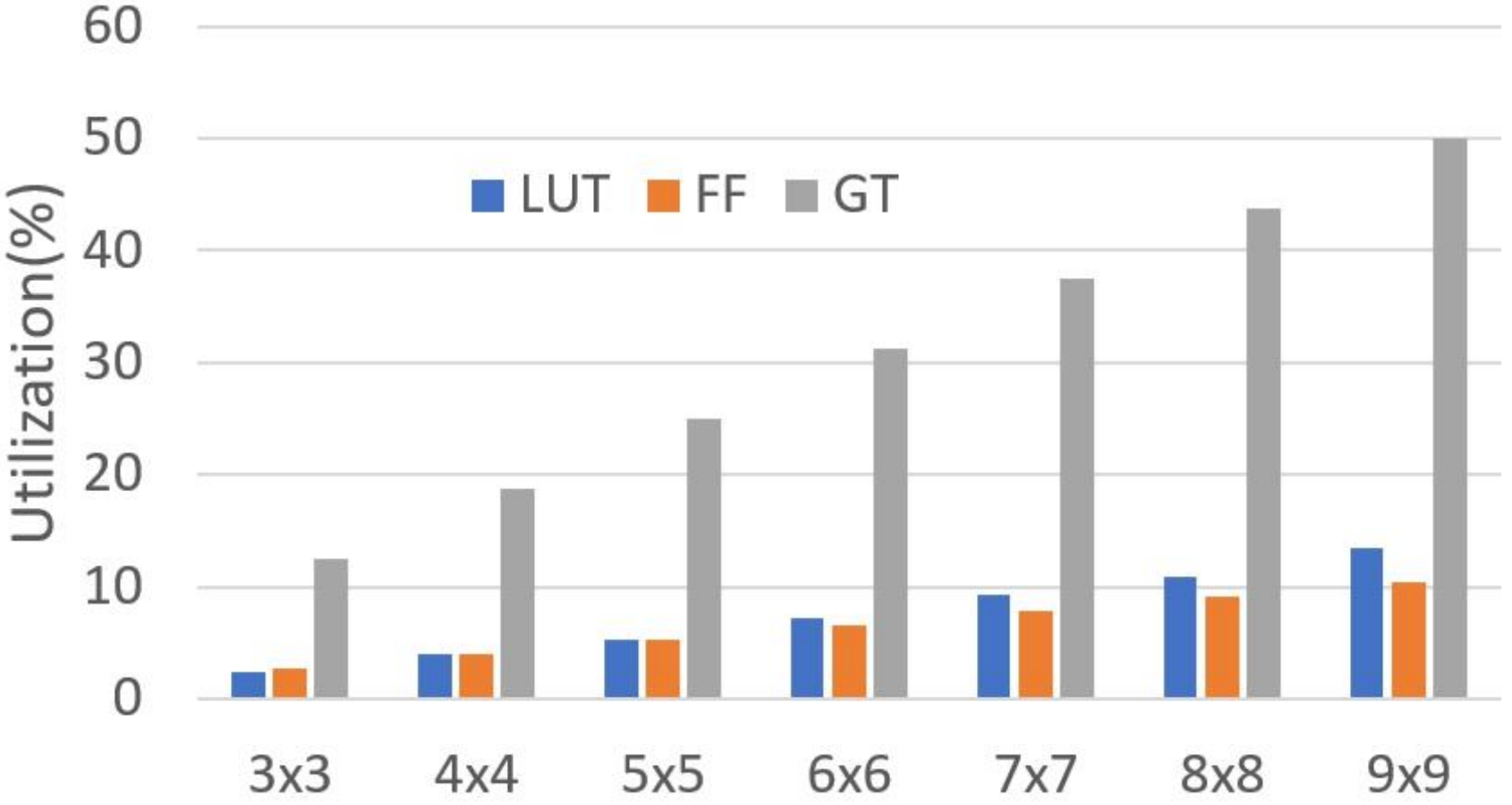
最大スロット数の結果

Slot数	4x4 torus	4x4 Dragonfly	8x8 torus	8x8 Dragonfly	16x16 torus	4x4x4x4 Flattened butterfly
Bit complement	1	1	3	2	7	4
Bit shuffle	1	1	3	2	7	4



- 物理×論理的トポロジーによる結果
- 256ノードではFlattened butterfly
  - 4ノードの全結合を3次元に
- 高度なトポロジーでは最大スロット数は小さい

STDMスイッチのリソース使用率



GT: 高速リンク利用率以外は12%前後

HLS部に十分リソースが残る

ニューラルネットワーク識別層の実装結果

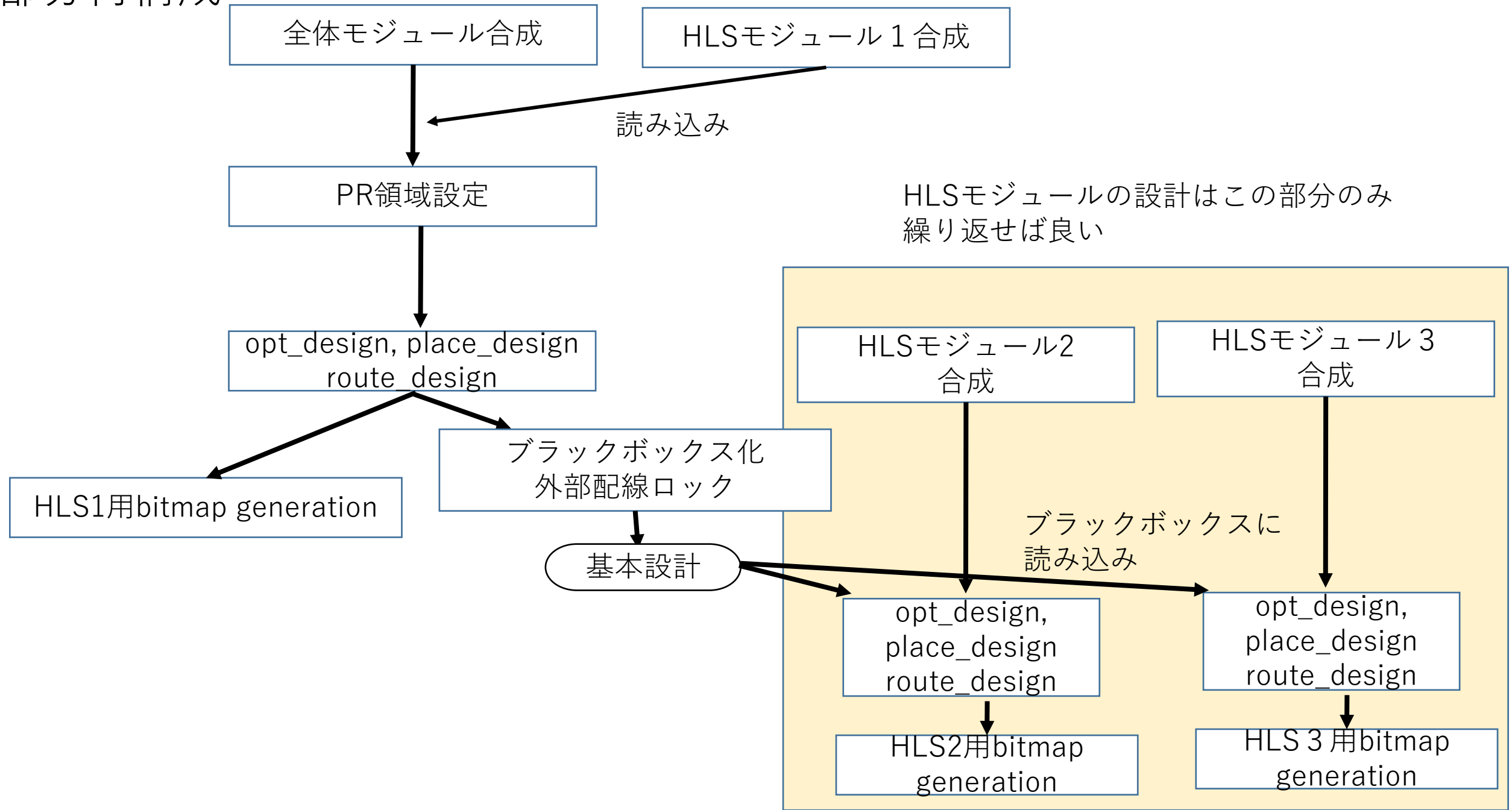
	周波数（MH z ）	電力（W）	image/sec	GOPS	GOPS/W
1 board	100	17.89	23551	120.58	6.74
4 boards	100	71.56	94226	482.34	6.74

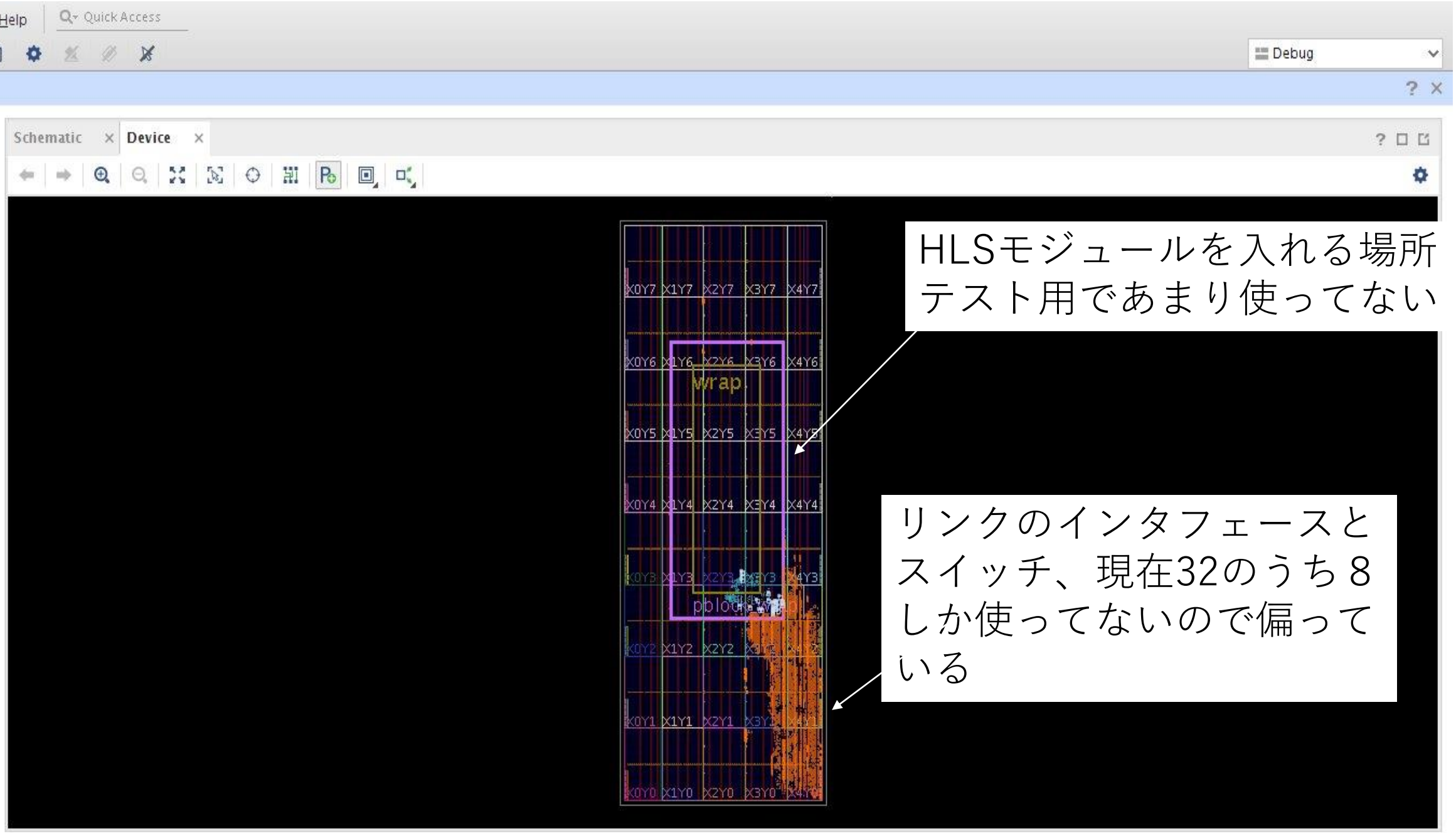
	周波数（MHz）	電力（W）	GOPS	GOPS/W
FiC 4boards	100	71.56	482.34	6.74
Stratex-V [1]	120	25.8	136.5	5.29
FCN　[2]	200	25.0	172.0	6.88

単一のFPGA実装と同程度の電力効率で高い性能を達成

[1] N.Suda, V.Chandra, G.Dasika, et.al “Throughput-optizemed open-based FPGA Accelerator for large-scale combolutional neural networks,” FPGA2016.
[2] C.Zhang, Z.Fang, P.Zhao, P.Pan, J.Cong, “Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks,” ICCCAD2017.

部分再構成


全体モジュール合成
HLSモジュール１合成
読み込み
PR領域設定
HLSモジュールの設計はこの部分のみ
繰り返せば良い
opt_design, place_design
route_design
HLSモジュール2
合成
HLSモジュール３
合成
ブラックボックス化
外部配線ロック
HLS1用bitmap generation
ブラックボックスに
読み込み
基本設計
opt_design,
place_design
route_design
opt_design,
place_design
route_design
HLS2用bitmap
generation
HLS３用bitmap
generation


Help
Quick Access
Debug
Schematic
Device
wrap
HLSモジュールを入れる場所
テスト用であまり使ってない
リンクのインタフェースと
スイッチ、現在32のうち８
しか使ってないので偏って
いる

ここでデモビデオを見せる

おわりに

- 今季中に32ボードのシステムを稼働させる予定
- スケジューラ、設計環境はまだまだこれから
- 4本束用のネットワークを検討中
- 様々なアプリケーションを試している