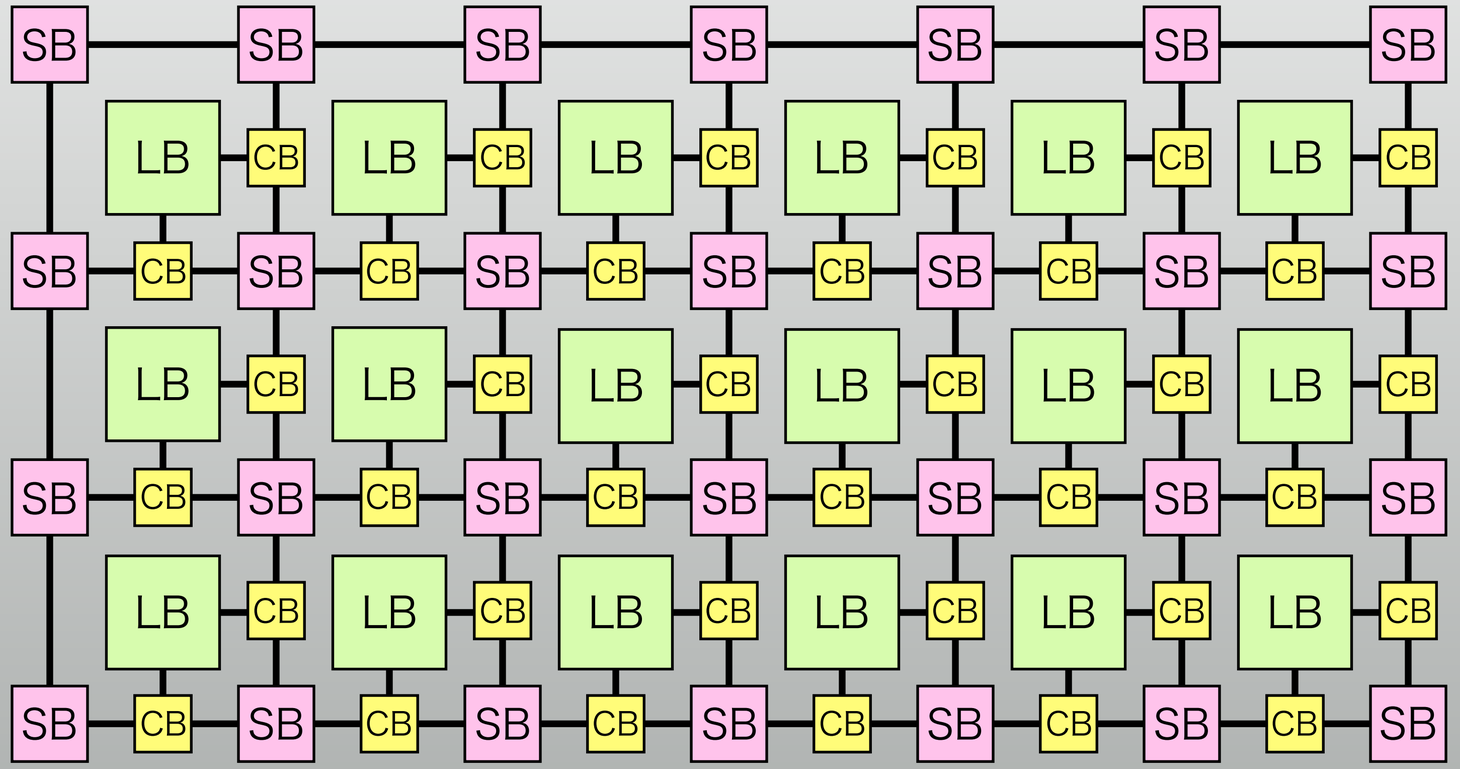
再構成型アーキテクチャ特論 （12）

osana@eee.u-ryukyu.ac.jp

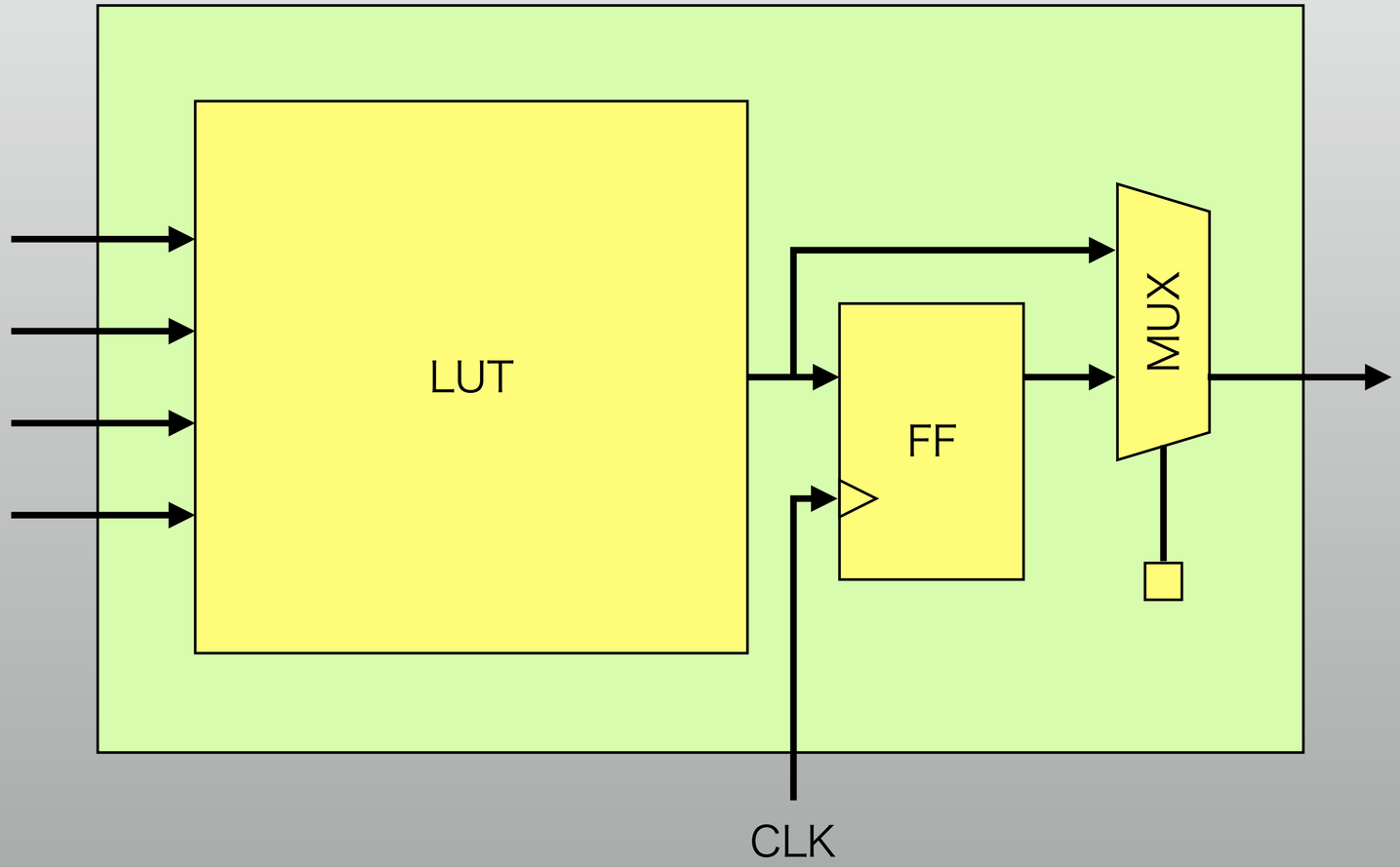
基本の構成要素

- 論理ブロック (LB: Logic Block)
- 接続ブロック (CB: Connection Block)
- スイッチブロック (SB: Switch Block)
- 配線


SB
LB
CB

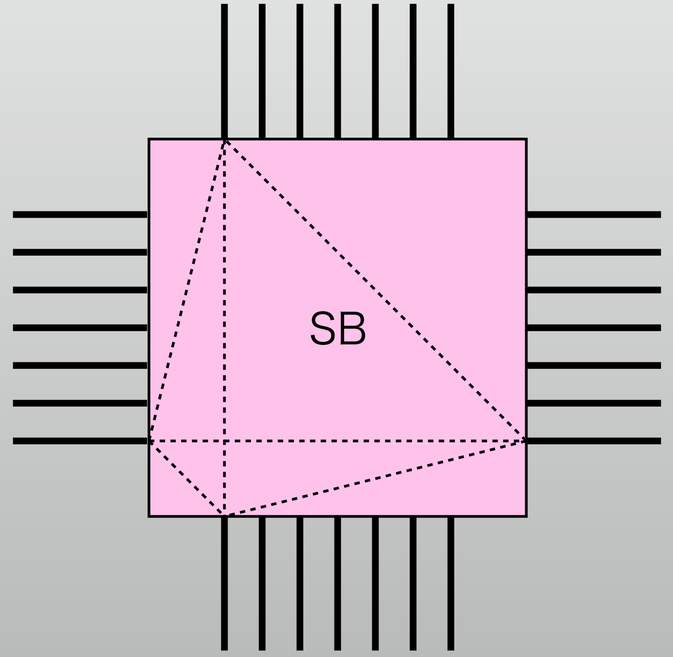
論理ブロック

* ごく基本的な構成はLUT + FF + MUX



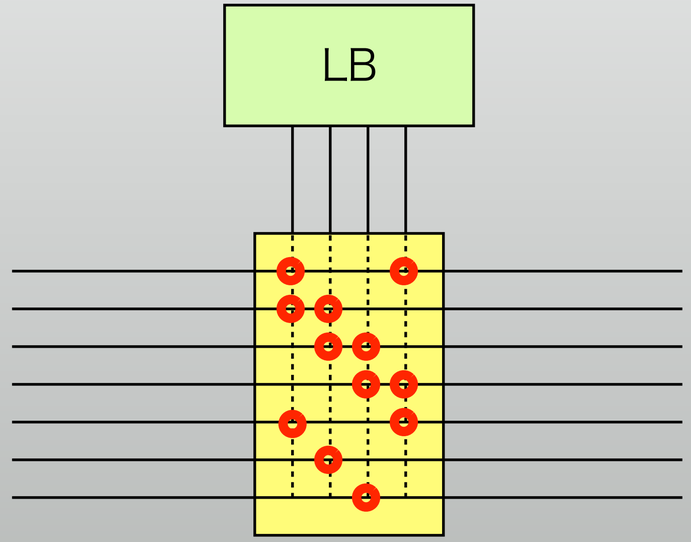
配線とSB

- 縦横の配線は複数のトラックで構成
  - 完全結合は大変：ベンダ次第



LBと配線とCB

- ロジックブロックからの配線を
  トラックに接続
  - これも完全結合ではない

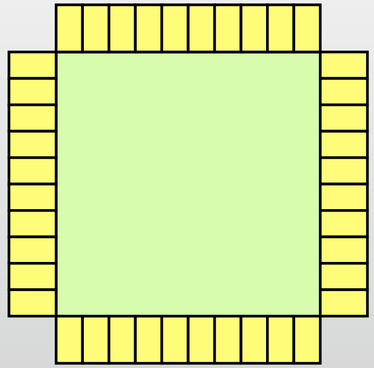


その他構成要素

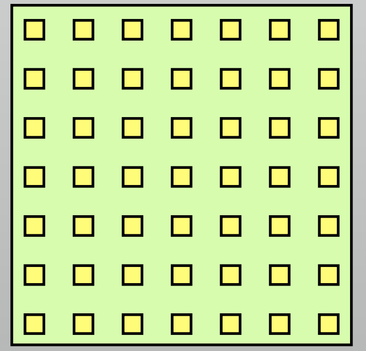
* 入出力ブロック
* クロック配線

入出力

- ワイヤボンディング
  - I/O は周辺部だけに存在
- フリップチップ
  - I/O を分散できて配線短縮
  - 周辺部にまとめる場合も



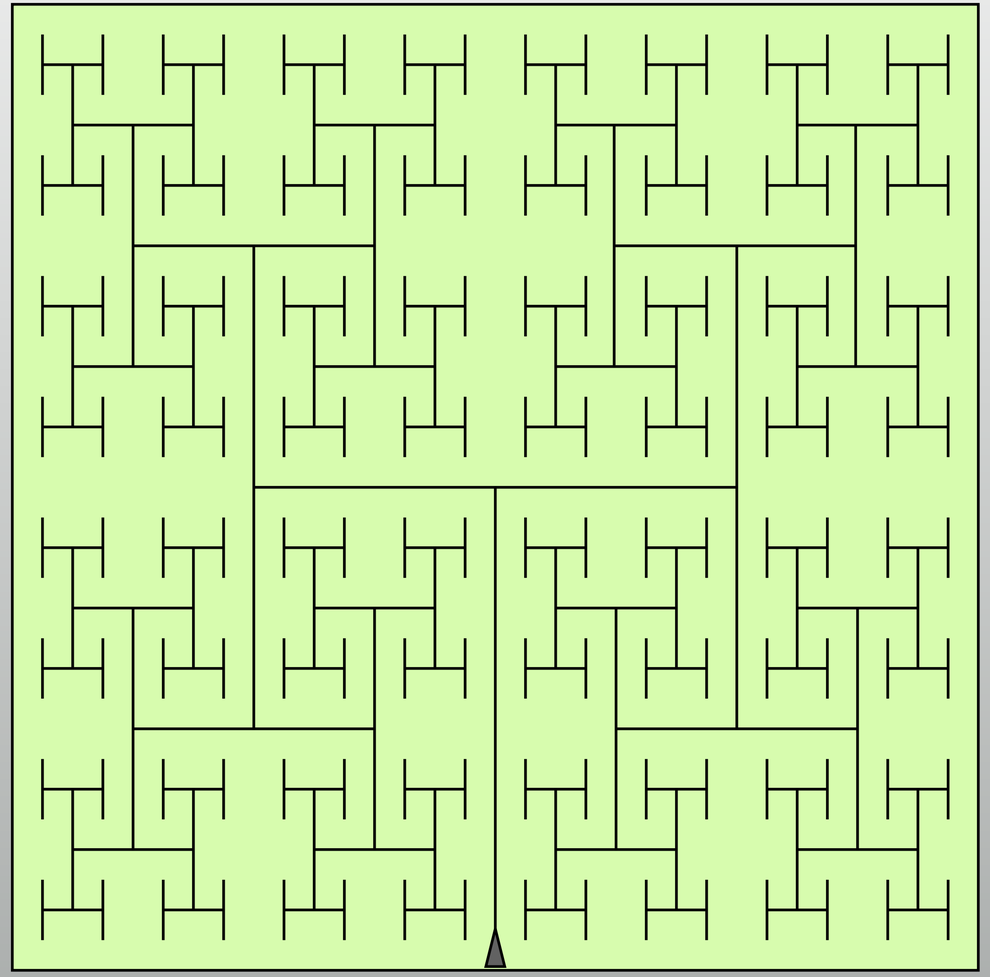
従来型



フリップチップ

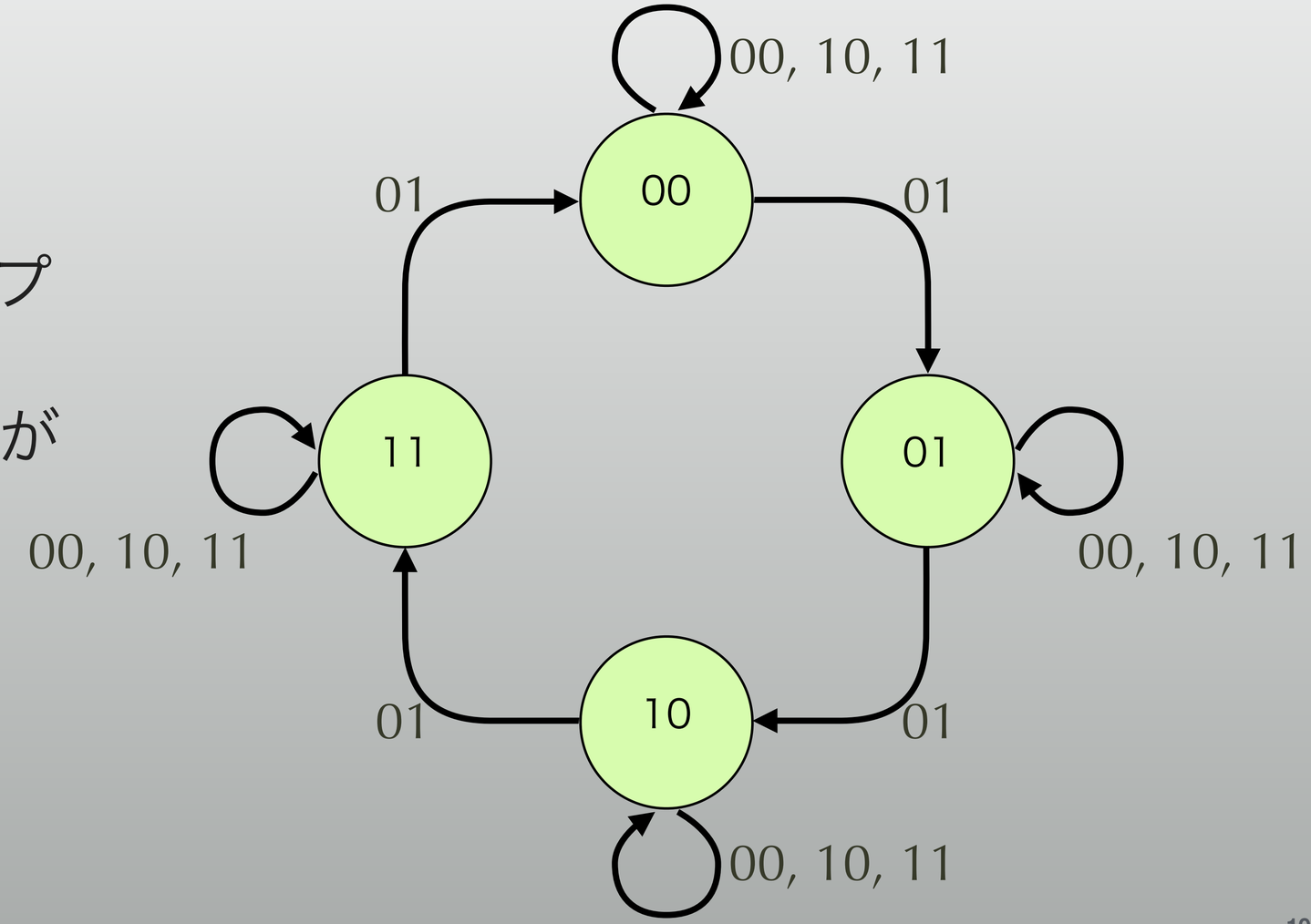
クロックツリー

- すべてのFFへクロックを等長配線
  - 専用の配線ルートを使う
  - ロジック用の配線とは別

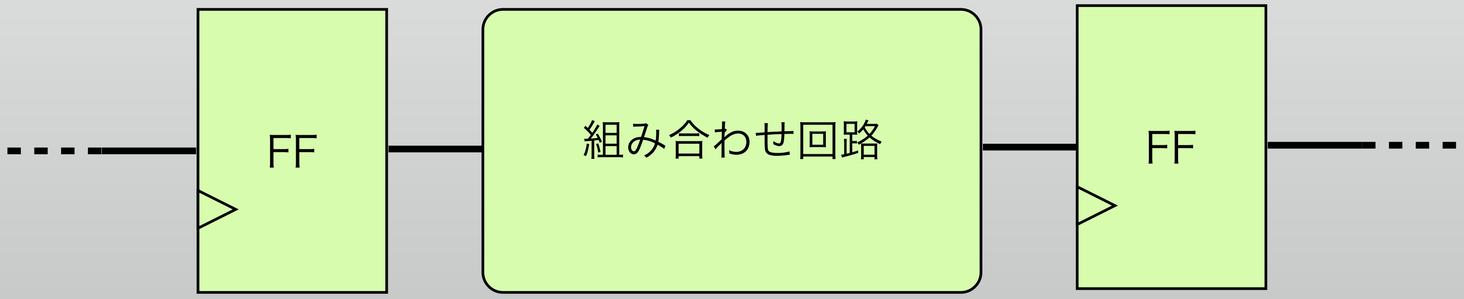


論理回路

- 組合せ回路とフリップフロップ
  - 状態遷移図とかそういうのがいろいろ出てくるけれど...



結局のところこれ



性能を決めるもの

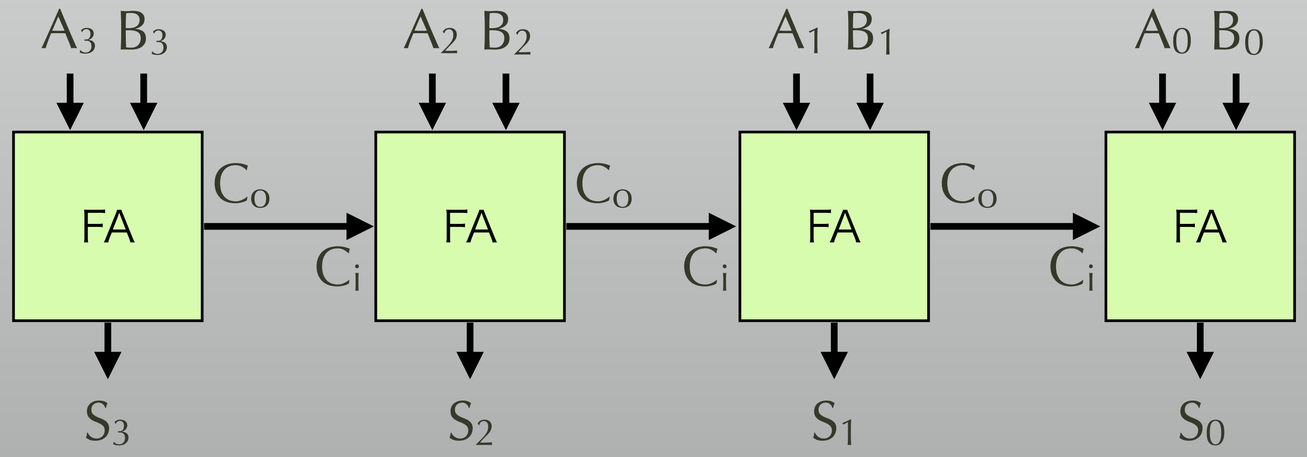
- レジスタ(FF)の段数
  - 何クロックサイクルで目的の状態に達するか
- 組み合わせ回路の遅延
  - Worst case delayがクロックの周期を決定
  - ロジック遅延＋配線

よく使われるビルディングブロック

- こういうのを効率よく作れることが重要
  - シフトレジスタ
  - 加算器
  - メモリ

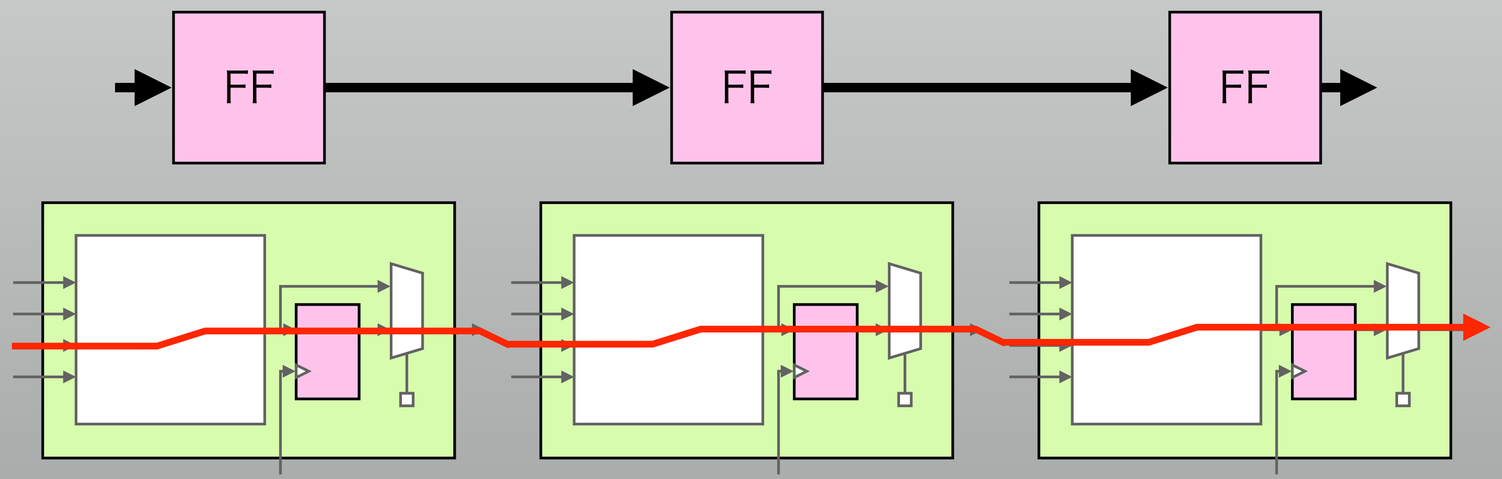
加算器

- 全加算器を数珠つなぎにする
  - 全加算器はLUTで作れる
  - 問題はキャリーチェーンの遅延が大きいこと



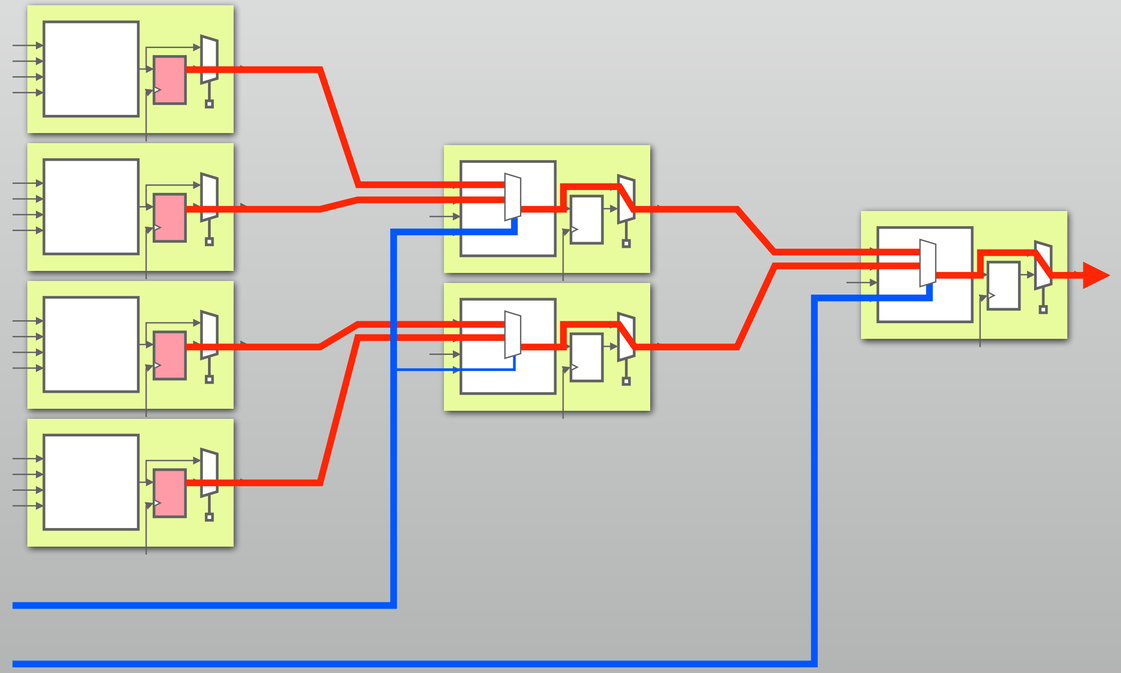
シフトレジスタ

- LBを順につなげばできる
  - 遅延はあまり問題ない
  - でもLUTが使われないので面積効率が悪い



メモリ

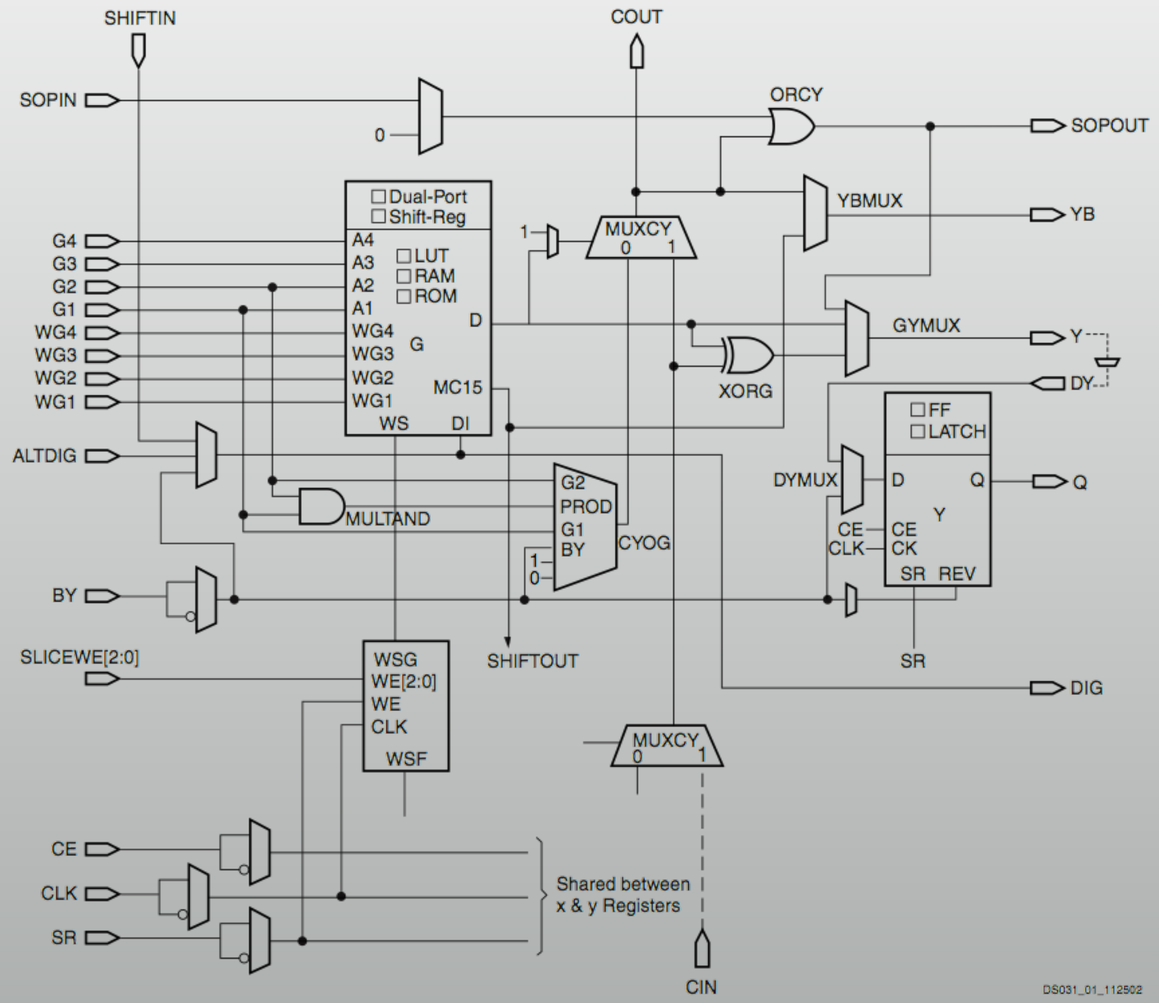
* LUTはデコーダ/マルチプレクサにしか使われない...



もったいない？

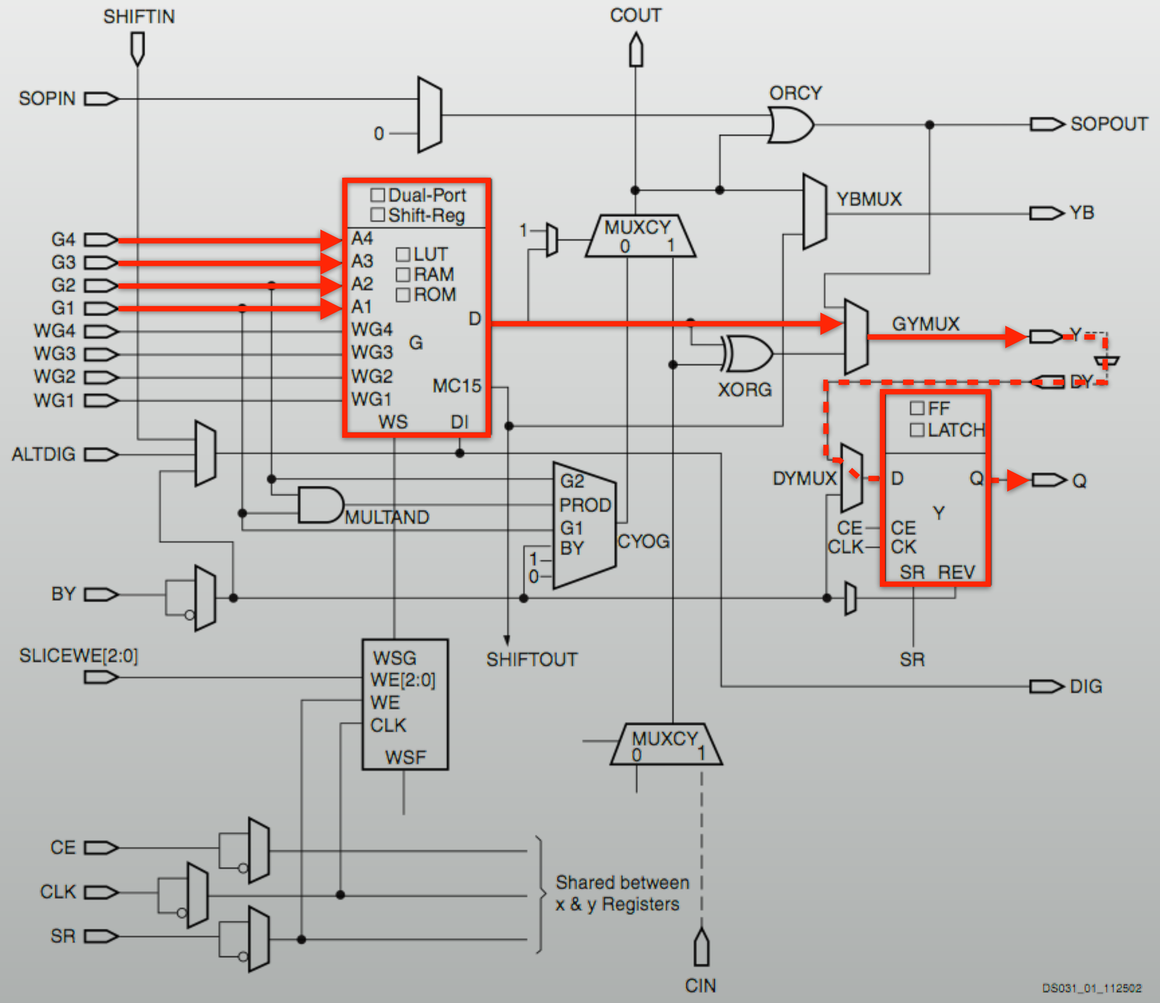
- LUT はそもそもメモリ
  - 4入力なら16ビットある
  - 当然アドレスデコーダもある
- 頑張ればシフトレジスタにも使えそう...

Virtex-II のLB



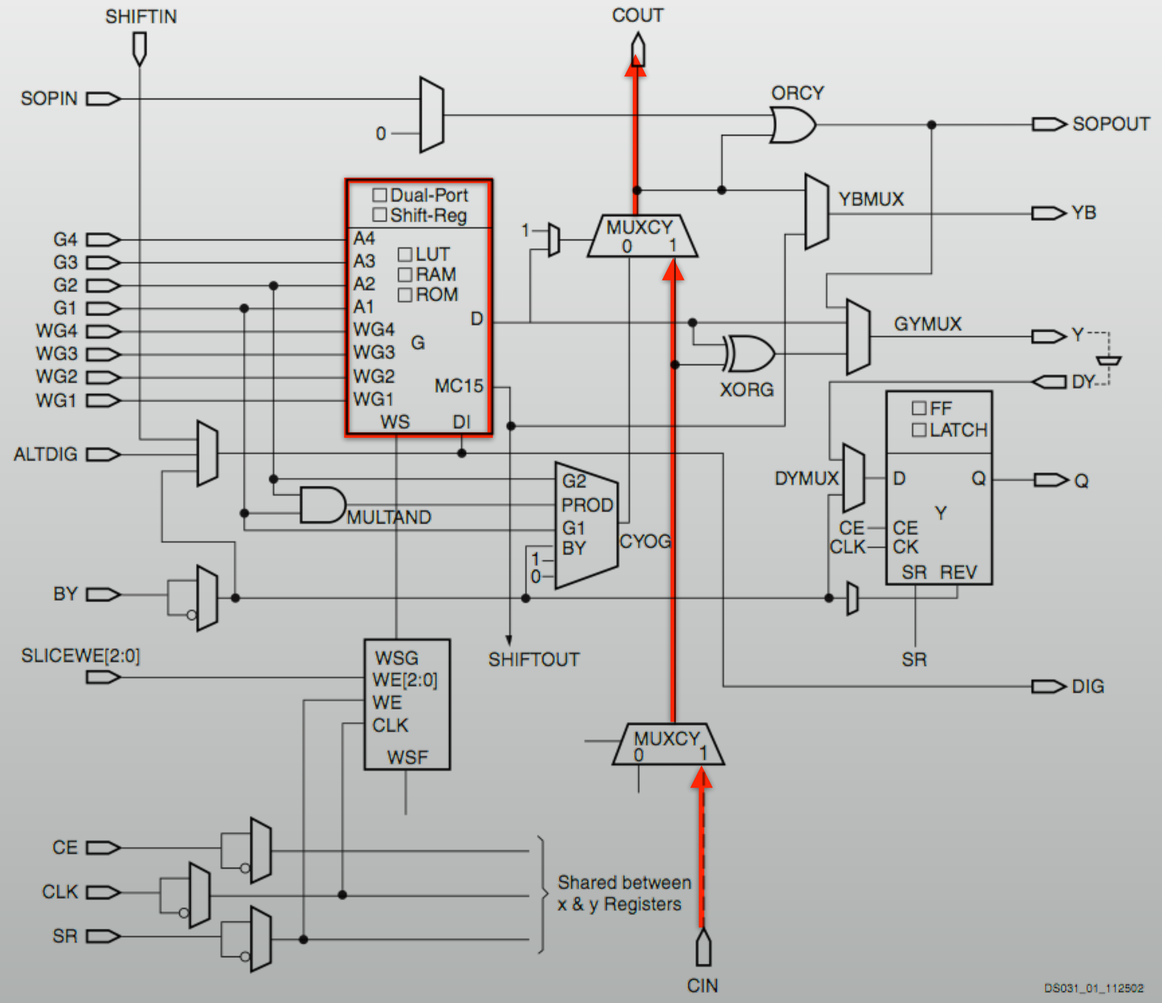
Xilinx Virtex-II FPGA Datasheet (DS031) より

基本構造



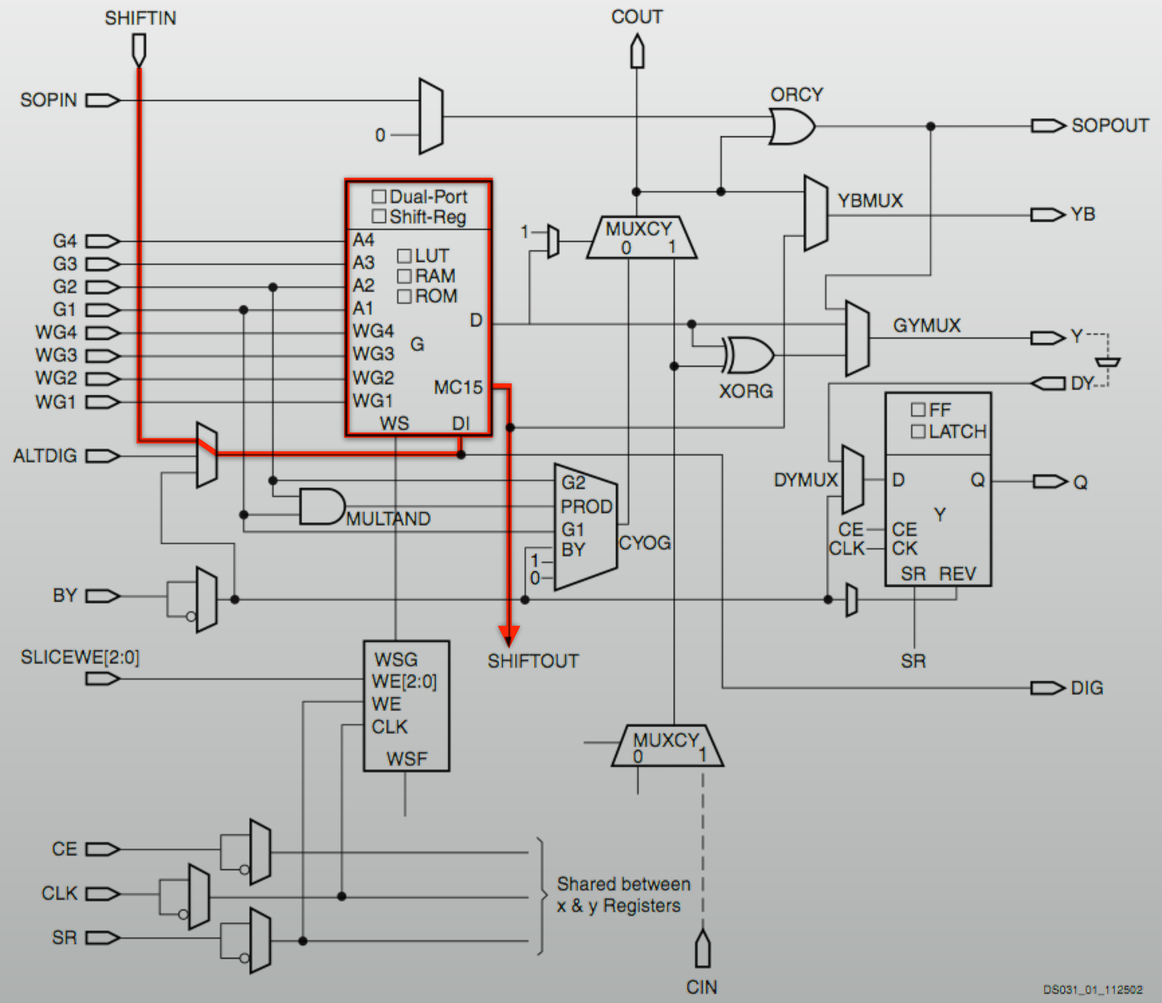
Xilinx Virtex-II FPGA Datasheet (DS031) より

キャリーチェーン



Xilinx Virtex-II FPGA Datasheet (DS031) より

シフトレジスタ



Xilinx Virtex-II FPGA Datasheet (DS031) より

LB+CB+SB=FPGA？

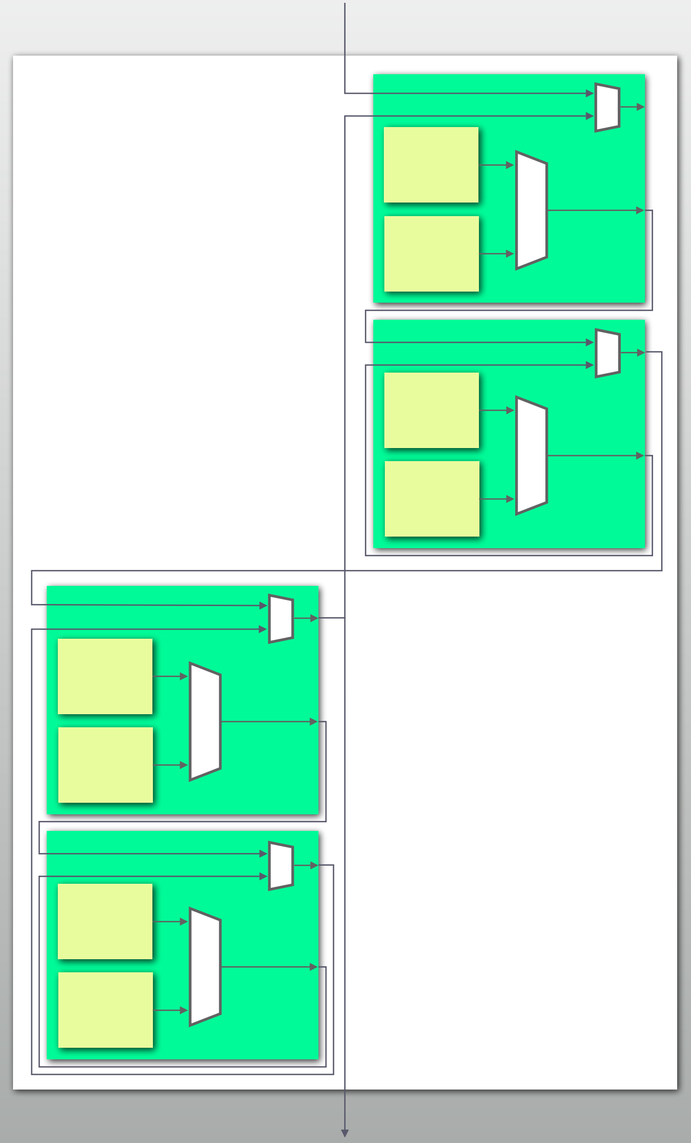
- 実際にはこれだと性能がでない
  - LUT+FF+MUX を通る度に配線トラックへ？
  - それだと配線を通る回数が多く、遅い
  - 実際の Configurable Logic Block の構造はもうちょっと複雑

柔軟性 と 性能

- いちいち配線を通ると性能が落ちる
  - LUT+FF+MUXをいくつかグループにする
  - そのグループのなかではローカルな配線が使えるようにする
  - 各LUT+FF+MUXの入出力は個別に外へ接続
- これがクラスタリング

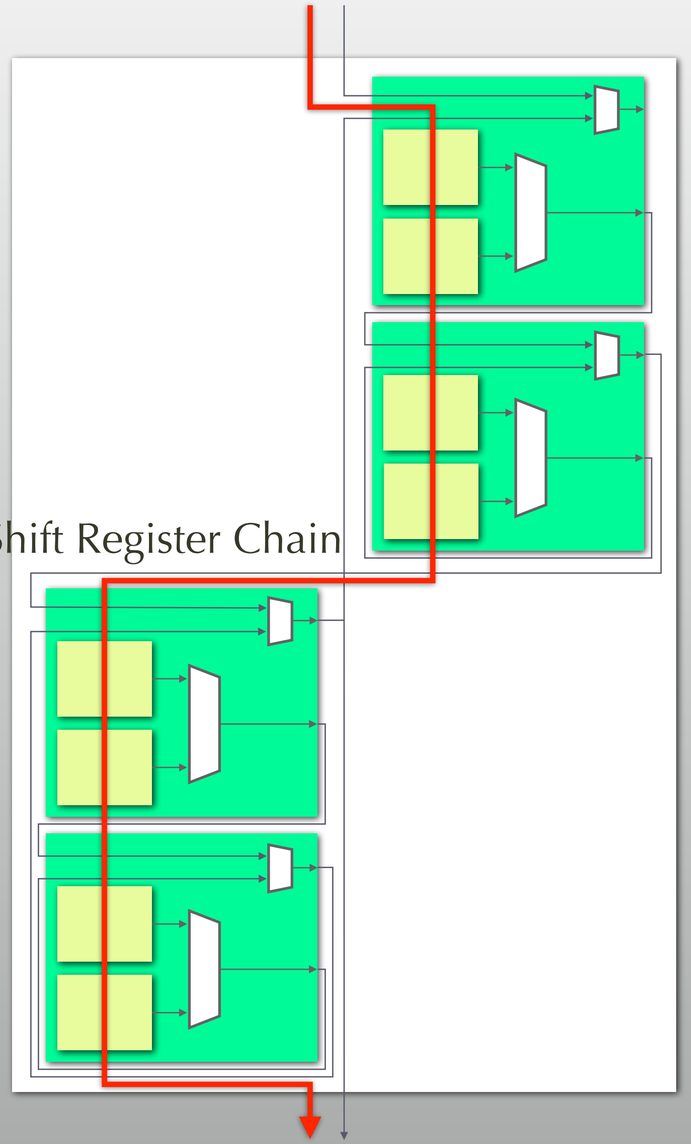
クラスタリング

* ローカルな配線は
  比較的簡単な構成
* キャリー・シフトレジスタ用の
  専用配線なども



クラスタリング

* ローカルな配線は
  比較的簡単な構成
* キャリー・
  シフトレジスタ用の
  専用配線なども



配線アーキテクチャ

- 長さ1だけのwire segmentでは性能が上がらない
  - どれだけ多くのLBを低遅延で接続できるか
  - 長さの違うwire segmentを使うのがポイント

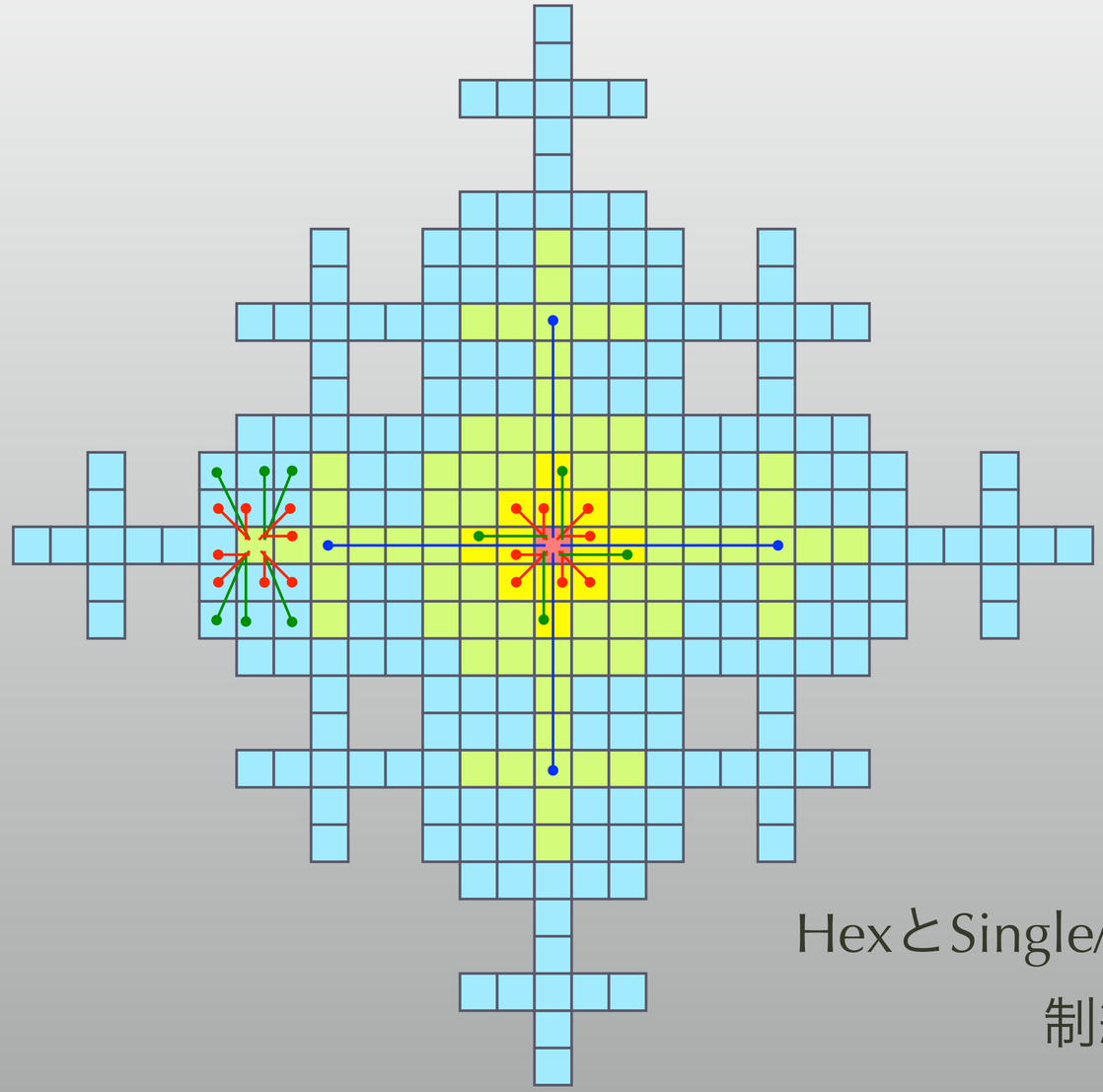
Virtex-4 & 5

* 正確なことは公式な資料だけではわかりづらい
* 以下は公開資料からの推測で、かつ説明用
  * いろいろ省略してあるので注意
* 自信のない数字も出てきますので「なんとなくこういうもの」程度に

Xilinx Virtex-4

- Double (2), Hex (6), Long (端から端まで) の3種類
  - DoubleとHexの併用で近隣のLBを低遅延で接続
  - 配線構造しては古典的な手法の延長

Virtex-4 Interconnect

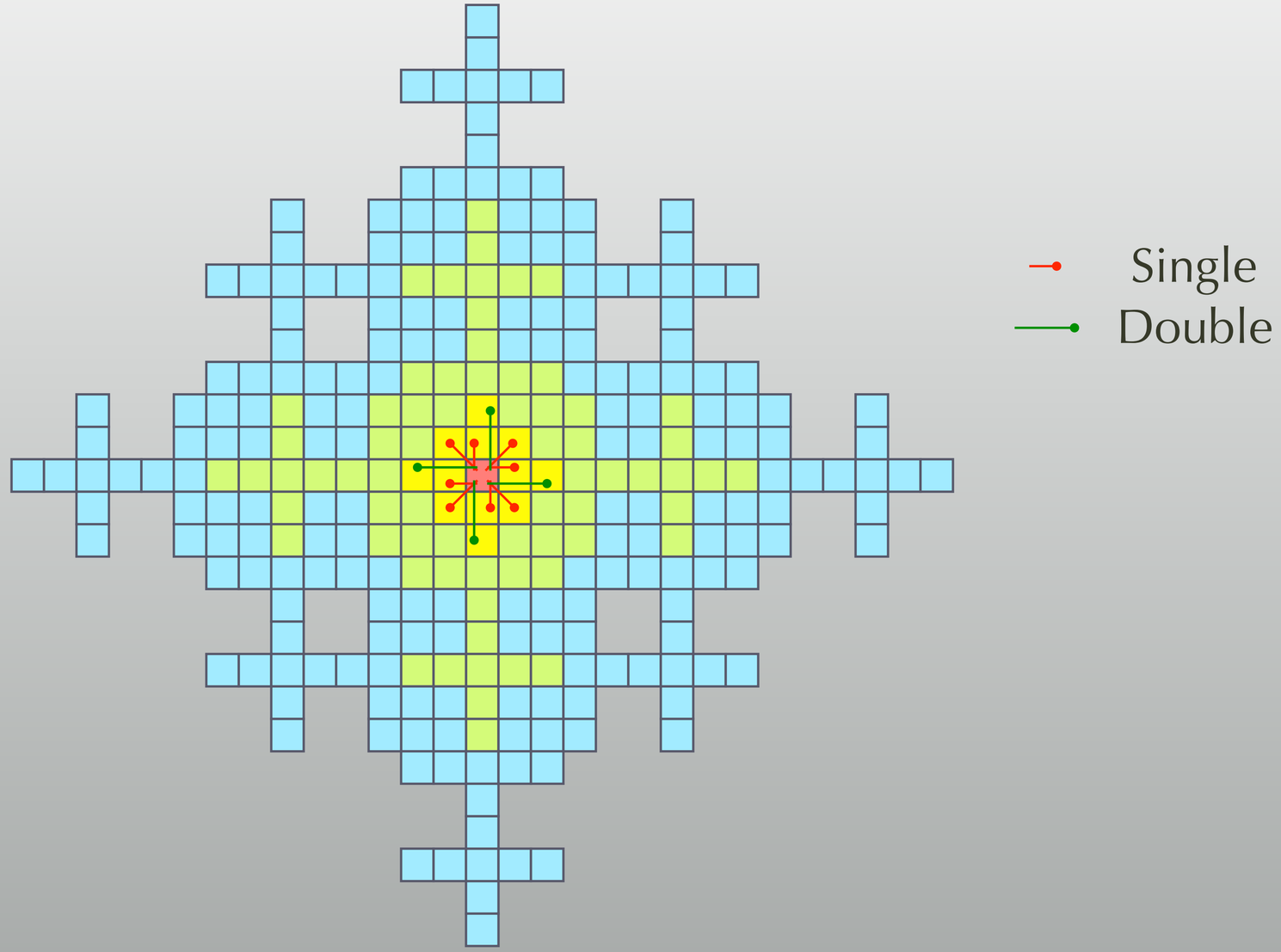


Single
Double
Hex

HexとSingle/Doubleの接続に
制約あり

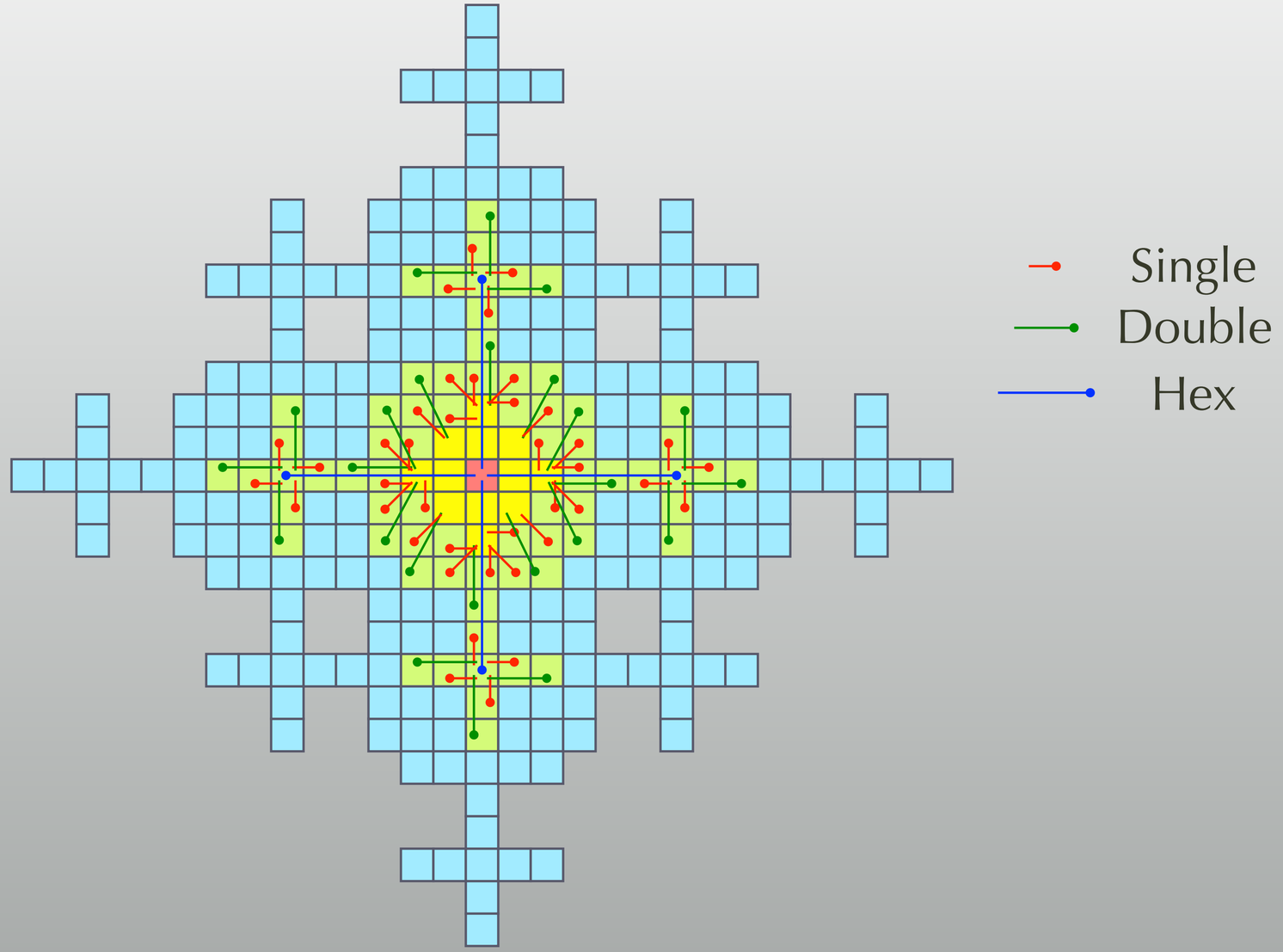
Virtex-4 Interconnect

1 Hop



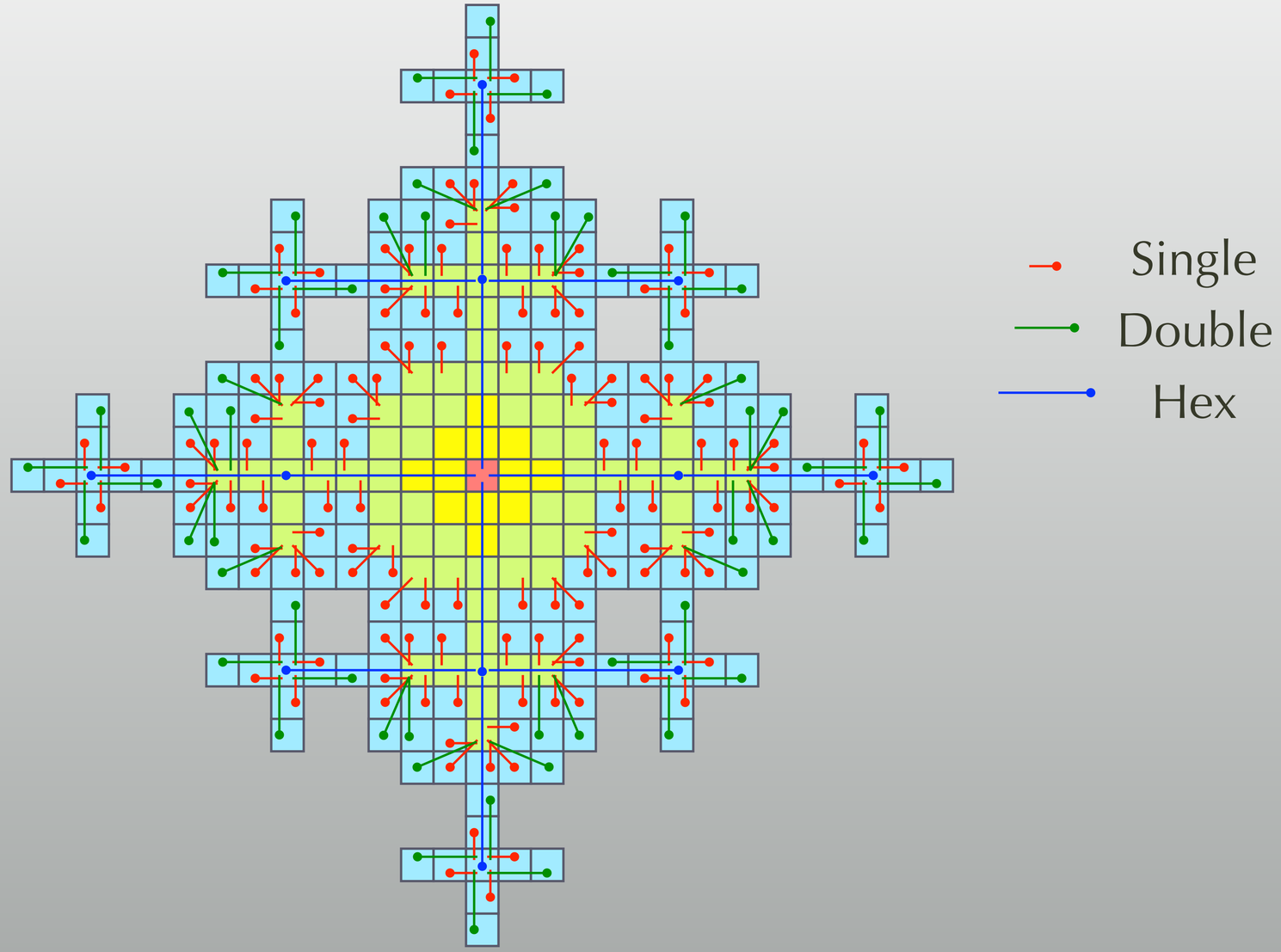
Virtex-4 Interconnect

2 Hops



Virtex-4 Interconnect

3 Hops

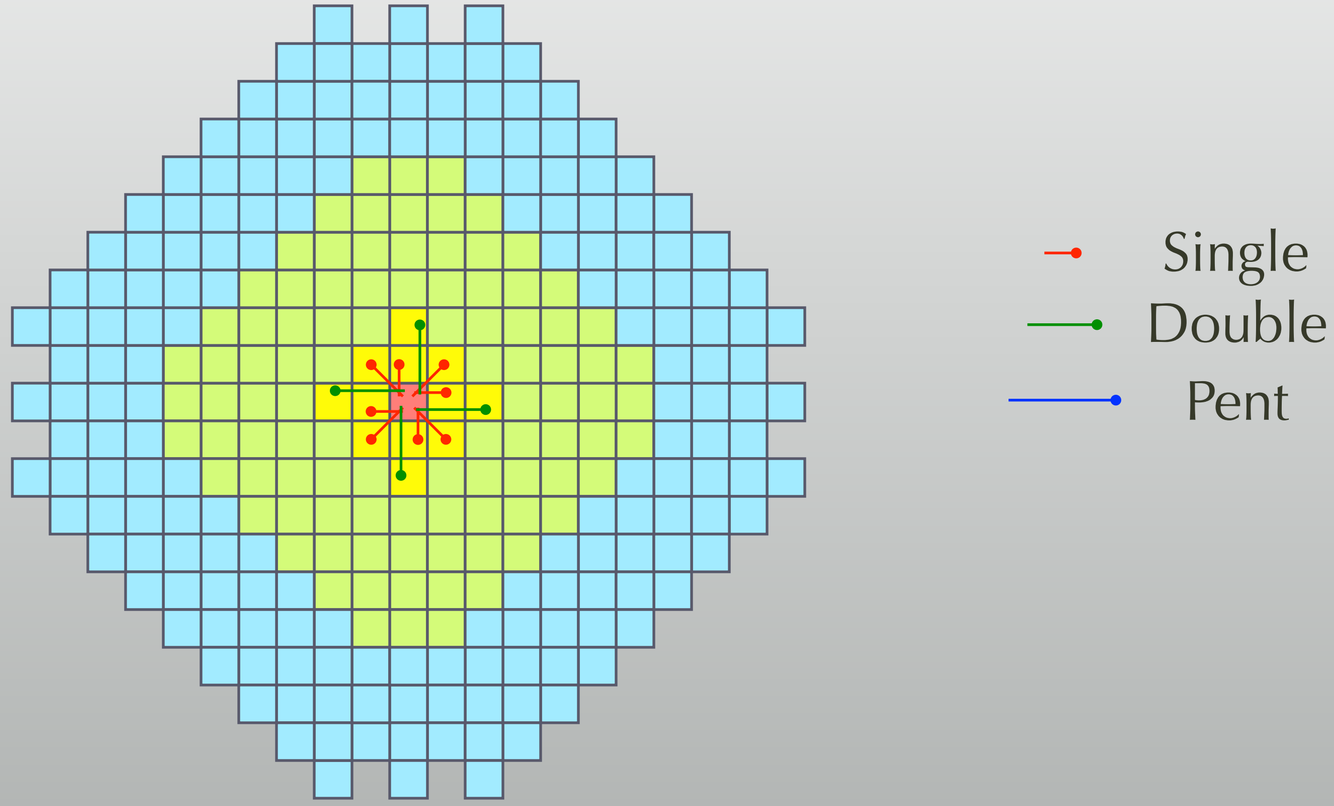


Xilinx Virtex-5

- 6-LUT を採用
  - 単純に考えて 50% 多くの配線が必要
  - L 字型の配線 (Pent) を導入、Hex はなくなった
  - より多くの近隣のLBを接続

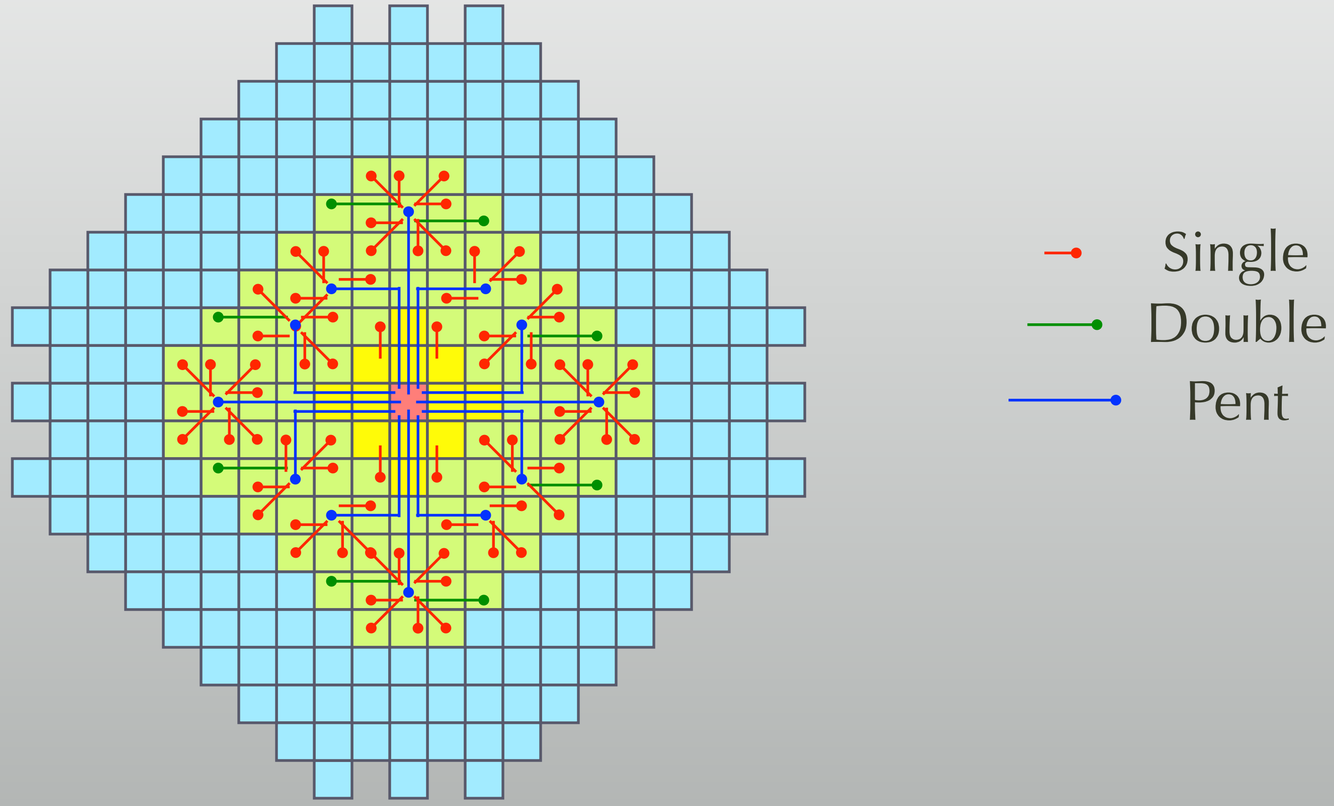
Virtex-5 Interconnect

1 Hop



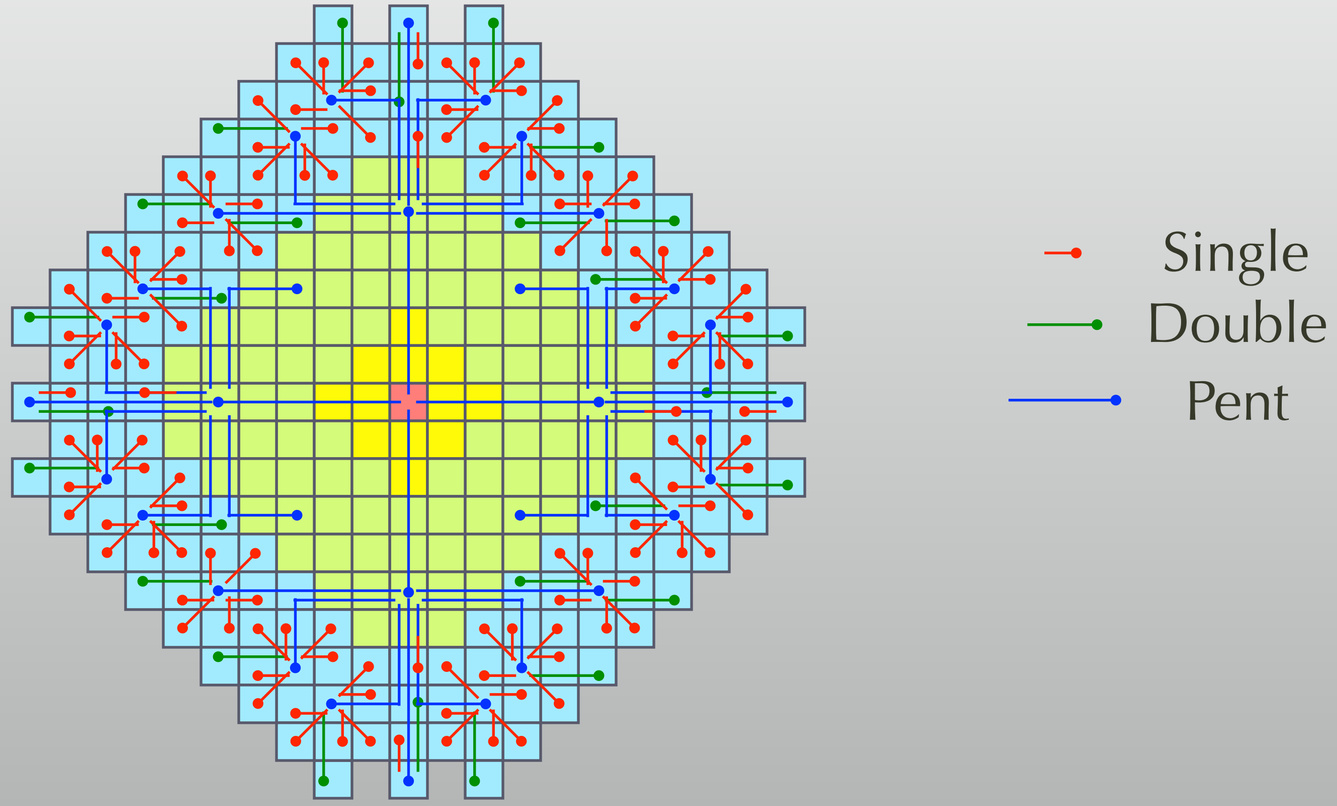
Virtex-5 Interconnect

2 Hops



Virtex-5 Interconnect

3 Hops



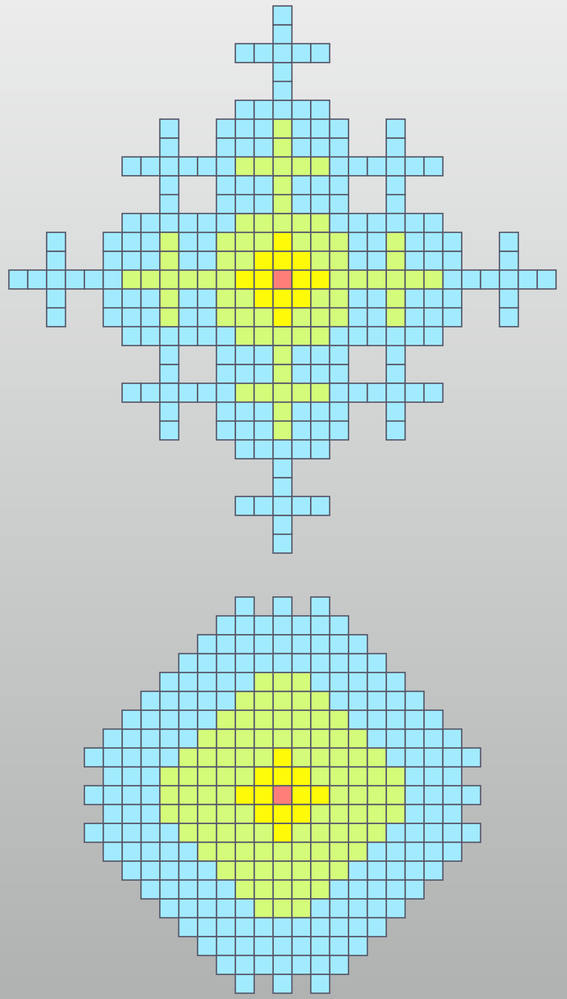
比較

Reachable CLBs

	Virtex-4	Virtex-5
1 Hop	12	12
2 Hops	68	96
3 Hops	200	180
Total	280	288

Wire segments

	Virtex-4	Virtex-5
Double	40	42
Hex	120	-
Pent	-	120
Long	24	18
Total	184	180



配線は増やせる？

- 配線技術の進歩によって配線層が増加、配線の自由度が向上
  - Virtex-II 150nm 6層メタル (2001)
  - Virtex-II Pro 130nm 7層メタル (2002)
  - Virtex-4 90nm 10層メタル (2004)
  - Virtex-5 65nm 11層メタル (2008)
  - Virtex-6 40nm 12層メタル (2009)
  - Virtex-7 28nm (2011)
  - Virtex Ultrascale 20nm (2014) / Ultascale+ 16nm FinFET

入出力

- 世の中の流れは多電圧化、だけではない
- 昔からある入出力規格:
  - TTL (5V), LVTTL (3.3V)
  - CMOS (5V), LVCMOS (3.3V)
- これらはロジックの配線をそのまま引き出せばOKだった

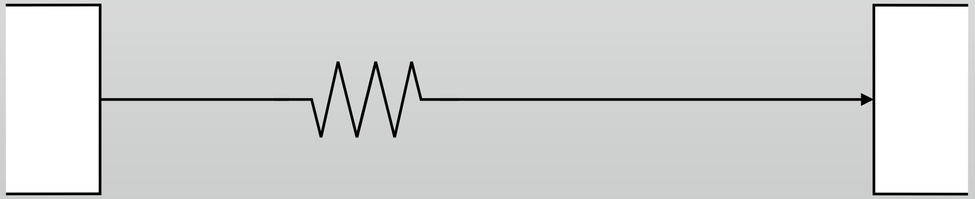
入出力

- 5V/3.3V の TTL/CMOS 信号そのままが基本
  - GNDとの電位差で H/L を伝える
  - 回路の構成が簡単
  - ノイズ、グラウンドバウンスに弱い
  - 66MHz くらいを超えると反射などの影響が出る

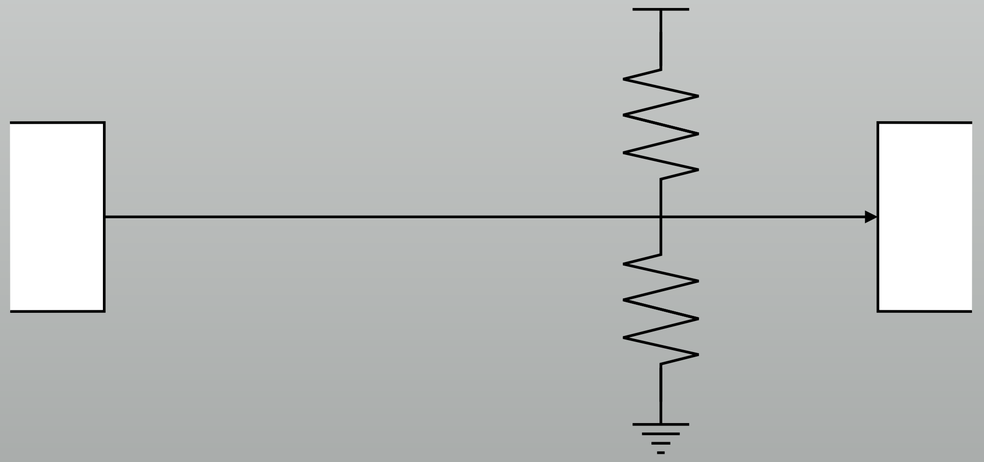
シングルエンド伝送

* 高速になるとインピーダンス整合が重要になる

ダンピング抵抗
（直列終端）



テブナン終端抵抗

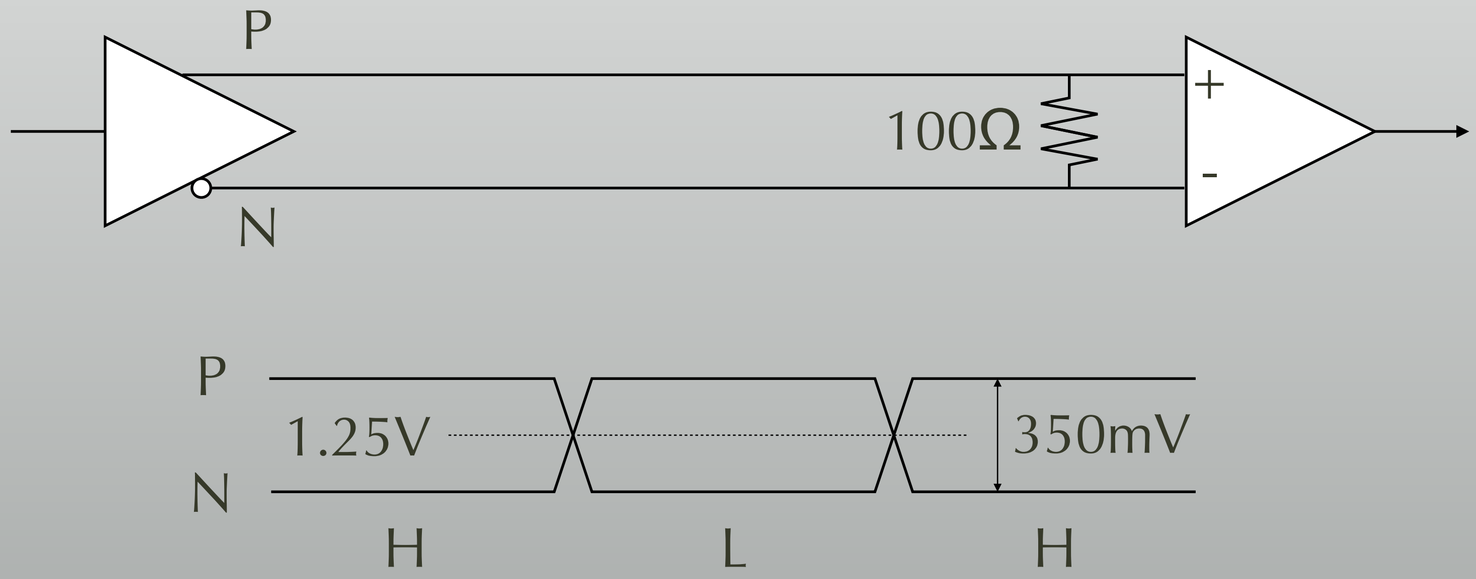


入出力（長距離）

- 差動信号の登場
  - LVDS, LVPECL, CML... など
  - シリアルバスなどへ応用されている
    - 液晶パネルの接続、USB, IEEE1394, ...
    - 本数が少ない超高速伝送向け

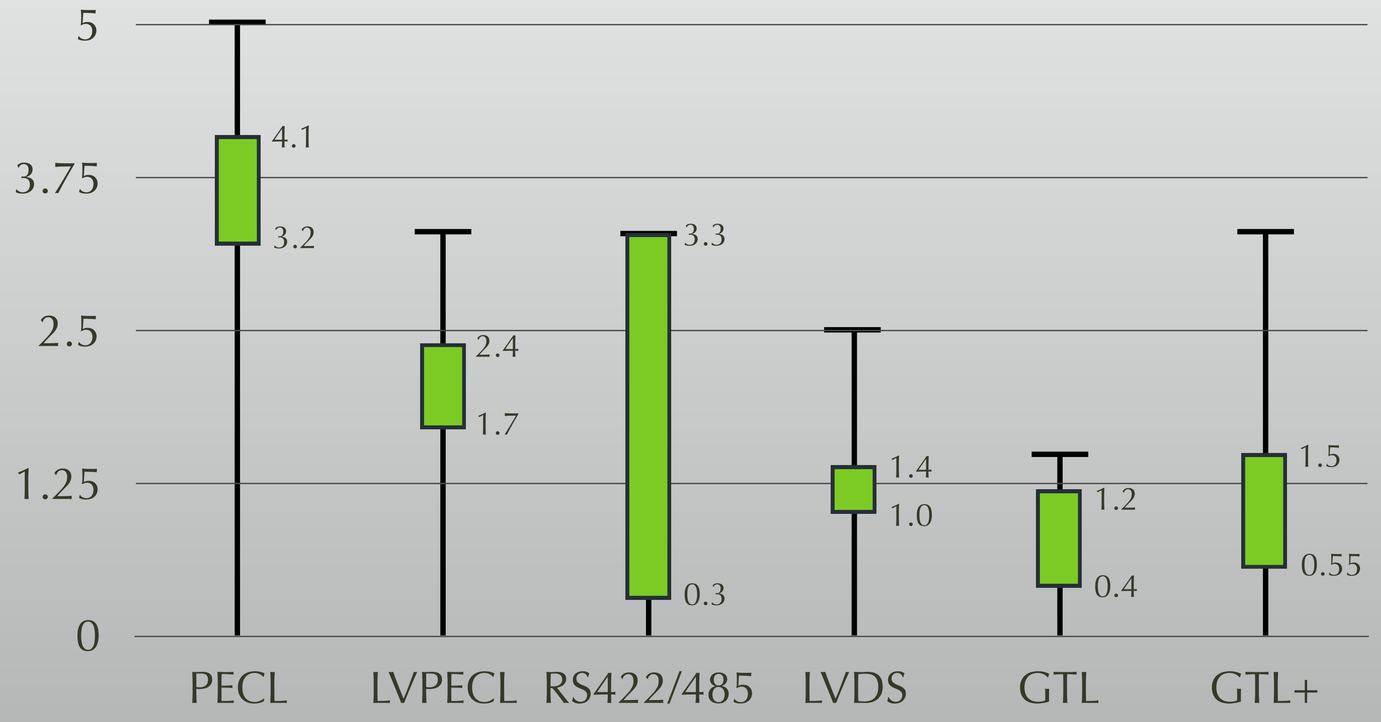
LVDS

- 3.5mA の電流モード　（コモンモード1.25V）
  - 100Ωで終端なので350mV振幅



差動信号規格

* 電源レールの電圧と振幅範囲

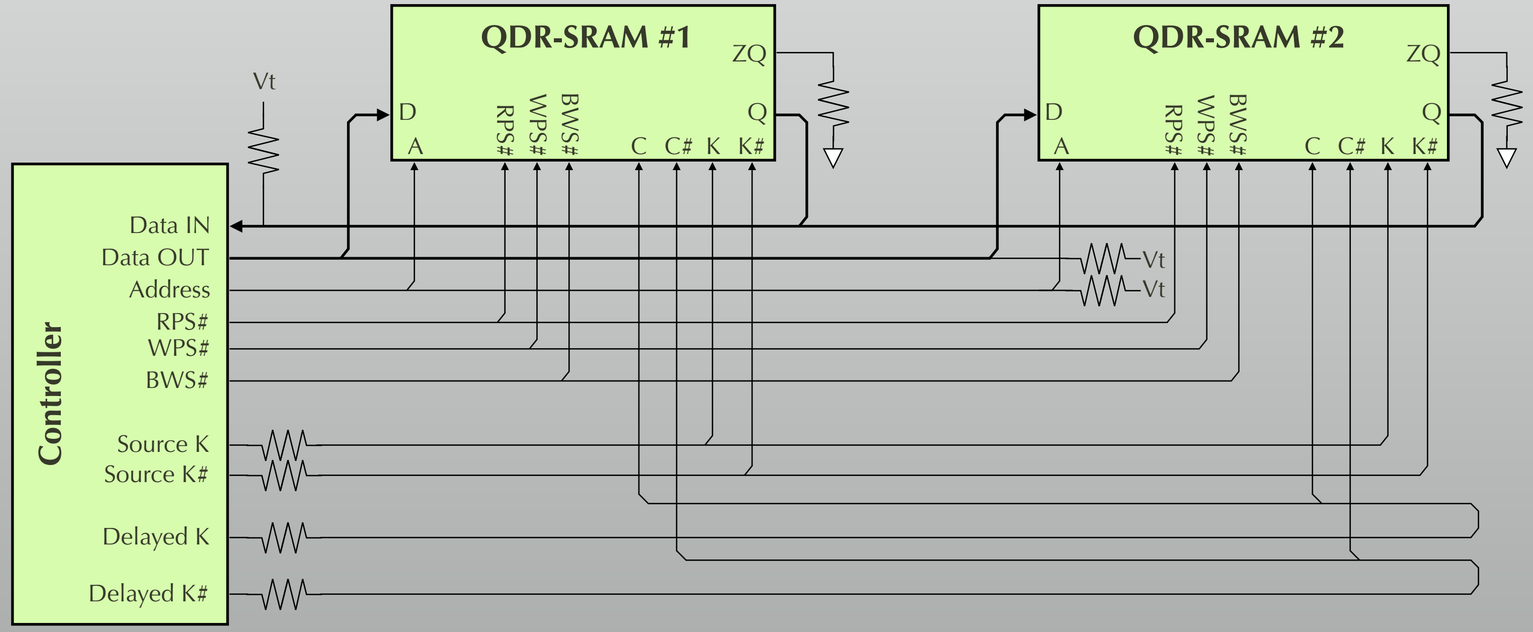


入出力（高速）

- 差動信号では配線が2本必要
  - パッケージのピン数や基板の面積で不利
  - シングルエンドでも高速動作する伝送規格
    - アクティブ終端の導入
  - HSTL、SSTL など、メモリなどに使用

HSTL

- 同時スイッチングの影響を減らすアクティブ終端
- クロックには直列終端



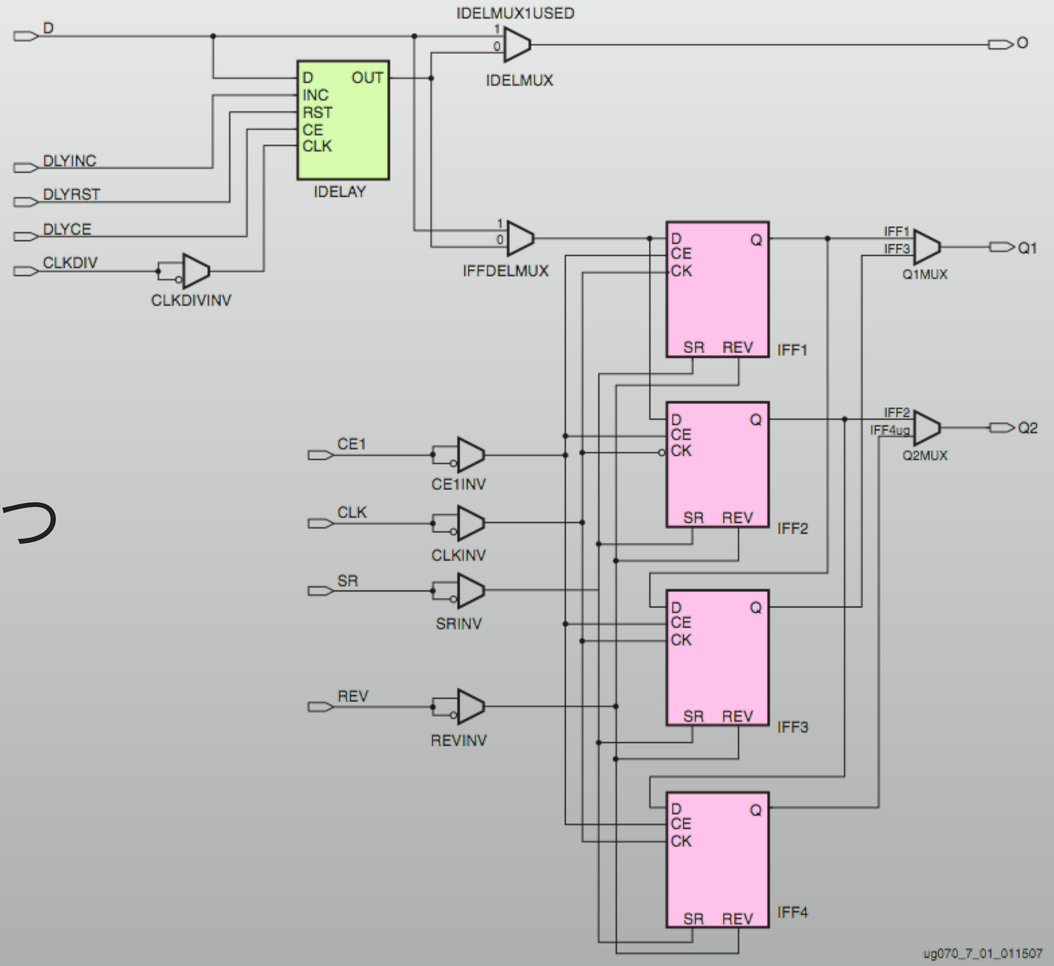
$Vt=V_{ddq}/2$

主要なI/O規格

- LVTTL (3.3V)
- LVCMOS (3.3/2.5/1.5/1.2V)
- HSTL (1.8/1.5/1.2V)
- SSTL (2.5V/1.8V/1.5V)
- LVDS
- LVPECL
- GTL, GTL+
- (PCI, PCI-X)

入出力: オンチップのことも忘れない

- Virtex-4 の ILOGIC ブロック
  - ただの入力バッファではない
  - 遅延素子 (IDELAY) とレジスタ4つ
- 出力側もだいたい同じ構造



Xilinx UG070 Virtex-4 User Guide

遅延素子はなぜ必要?

- 外部のデバイスからの遅延は均一ではない
  - 電流の速度は有限、配線長やインピーダンスにはバラツキがでる
  - パラレルな信号線を扱うためには遅延時間の調整が必要
- 入力 (最近では出力も) ピンごとに遅延時間を細かく変えられる
  - システムの初期化時にテストパターンを流してキャリブレーション

入力レジスタの役割

- チップ外配線の遅延とチップ内配線の遅延
  - チップ外は基板設計者がマジメならある程度コントロールされている
  - チップ内は配置配線ツールがチェックしてくれる
- ただし内外の両方になると検証がめんどう
  - 入出力レジスタはこの問題を (大幅に) 緩和してくれる

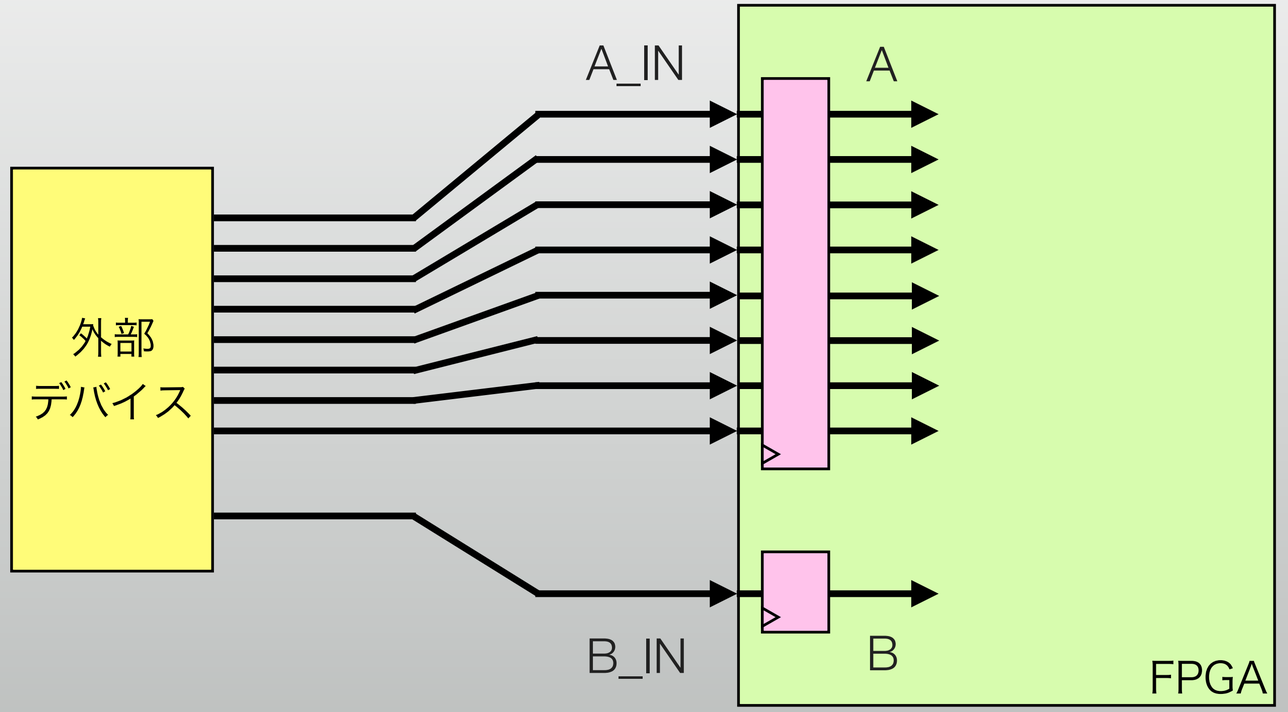
入力バッファにレジスタで解決

```
module some_fast_one
   ( input CLK, RST,
     input [7:0] A_IN,
     input       B_IN,
     . . . );

   reg [7:0] A,
   reg       B;

   always @ (posedge CLK) begin
      A <= A_IN;
      B <= B_IN;
   end

   . . .
endmodule
```



配線の遅延が目標動作周波数に
間に合うように基板設計

FFでクロックのエッジに
信号を揃えることで
タイミングマージンを確保

DDRとかそれ以上とか

- SDR (Single Data Rate):
  - データはクロックの立ち上がりエッジに同期
  - データの周波数はクロックの半分
- DDR (Double Data Rate):
  - 立ち上がり/立ち下がりでデータが変化、クロックと同じ周波数

内部の回路は…

- always @ **(CLK)** と書けば DDR で動く
  - 動くけど周波数が高い場合が多いので設計が難しくなる (事実上ムリ)
  - 2倍のビット幅の SDR の信号として扱えれば FPGA の設計が楽に
    - 入出力ブロックにあるレジスタを使う

入力DDRの例

- レジスタを2つか3つ利用
  - 片方は立ち下がりエッジで動作

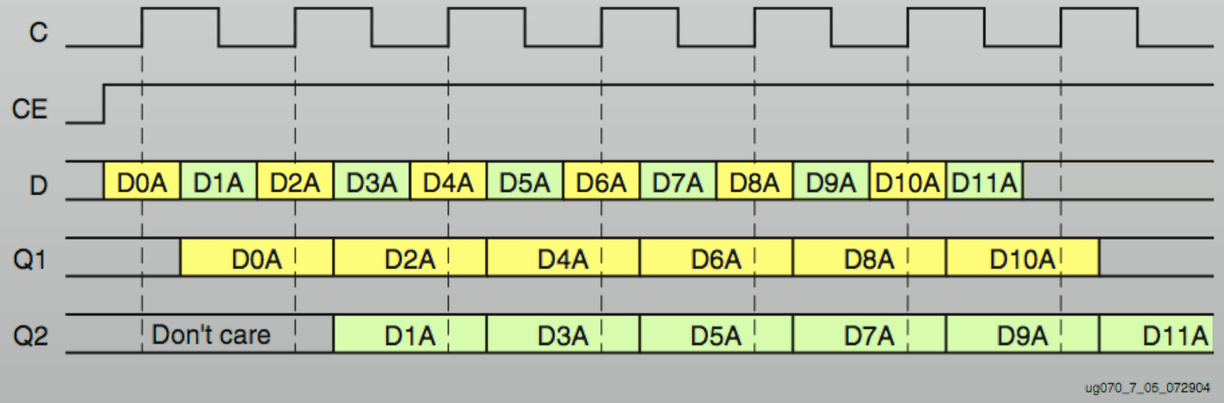


Figure 7-5: **Input DDR Timing in SAME_EDGE Mode**

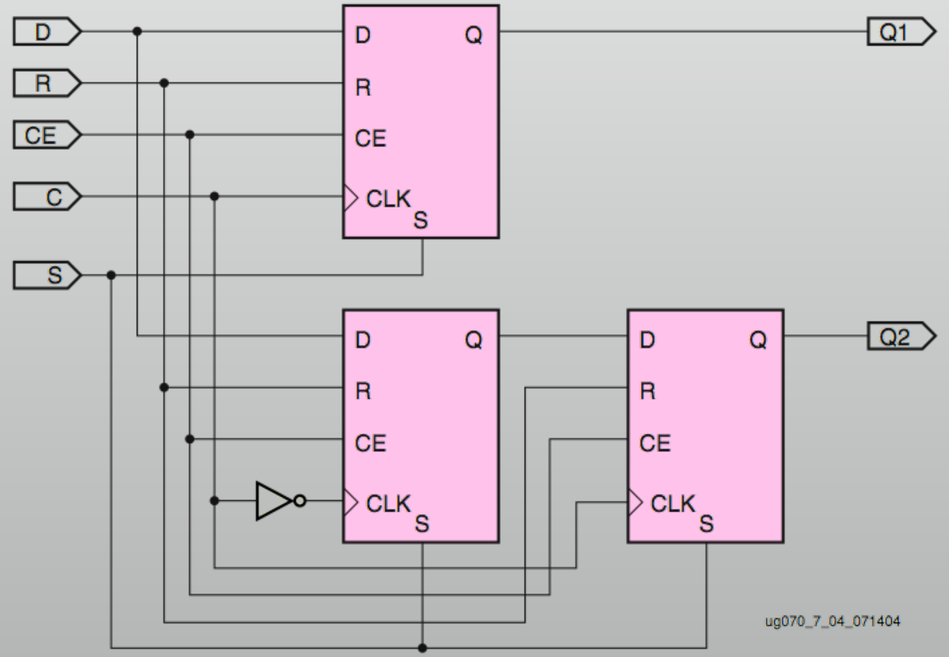
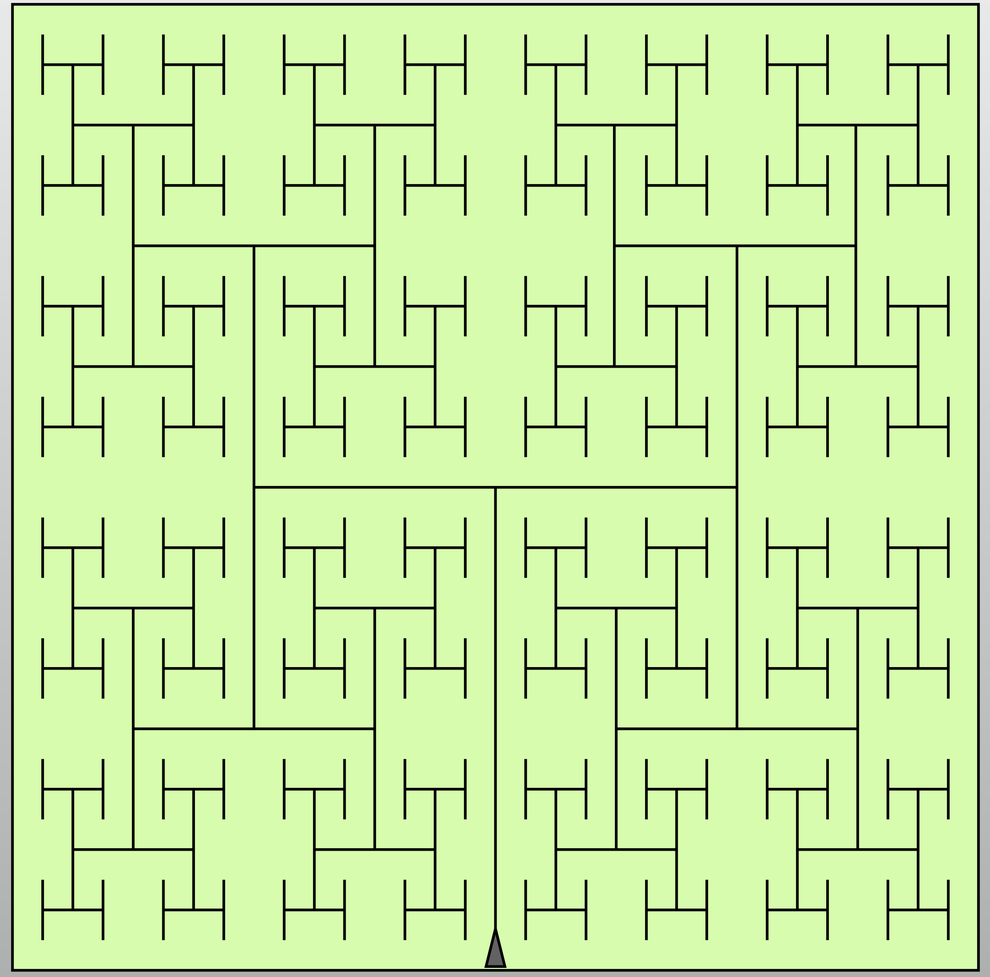


Figure 7-4: **Input DDR in SAME_EDGE Mode**

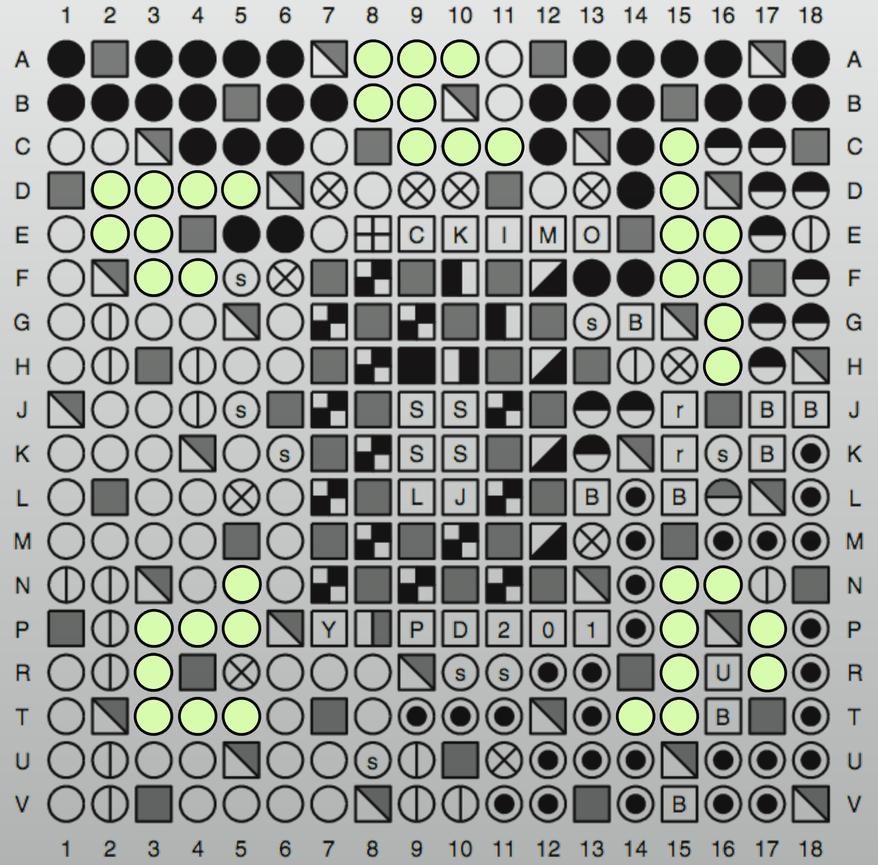
クロック

- クロックツリーを駆動するのは誰？
  - クロックバッファ
- クロックバッファを駆動するのは？
  - クロック入力可能ピン (Clock Capable Pin)
  - CMT などのクロックマネージャ



Clock Capable Pin

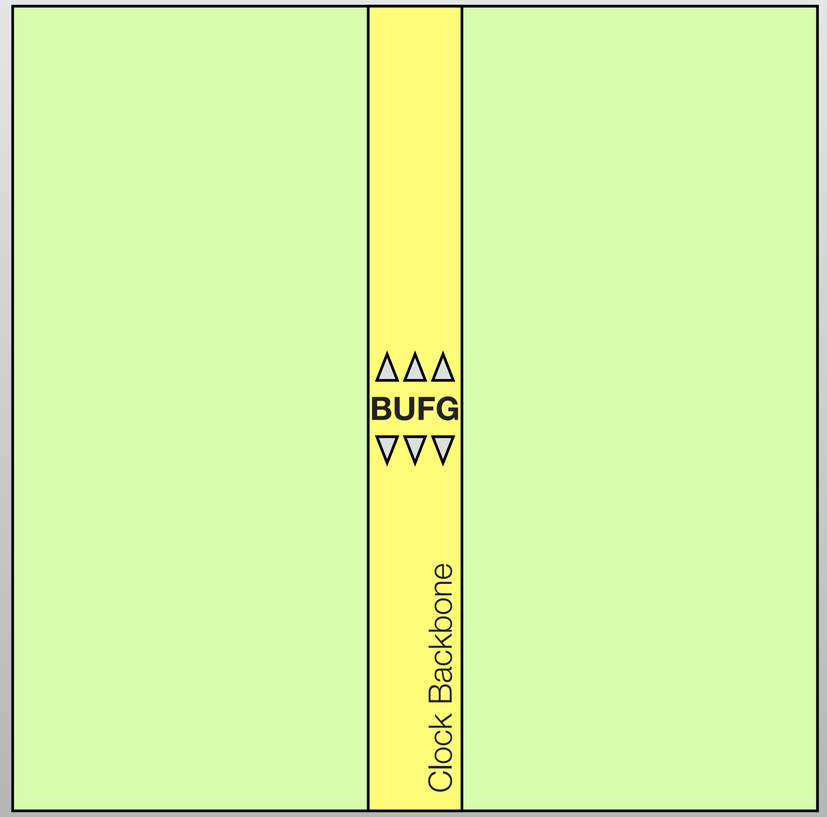
- わりと限られている
  - 差動でなければ使えるのは半分
  - 全部が同じように使えるわけではない
  - 詳しくは次で



Xilinx UG475: 7 Series FPGAs Packaging and Pinout

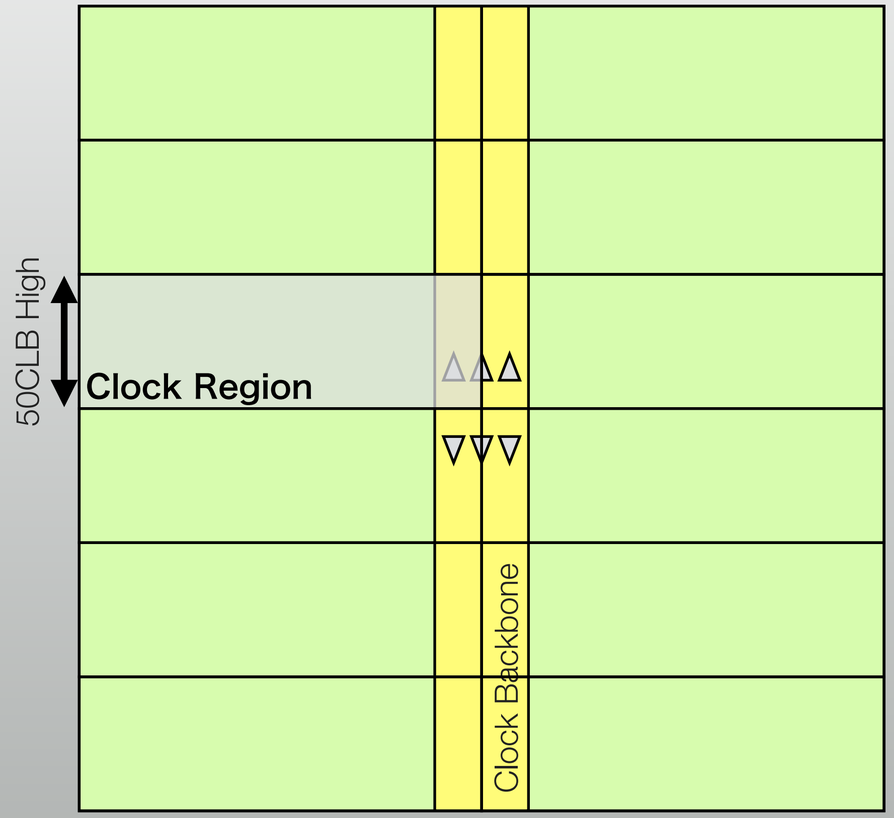
Global Clock

- Global Clock Buffer
  - 基本的にデバイスの中央付近にある
  - 数は決まっている



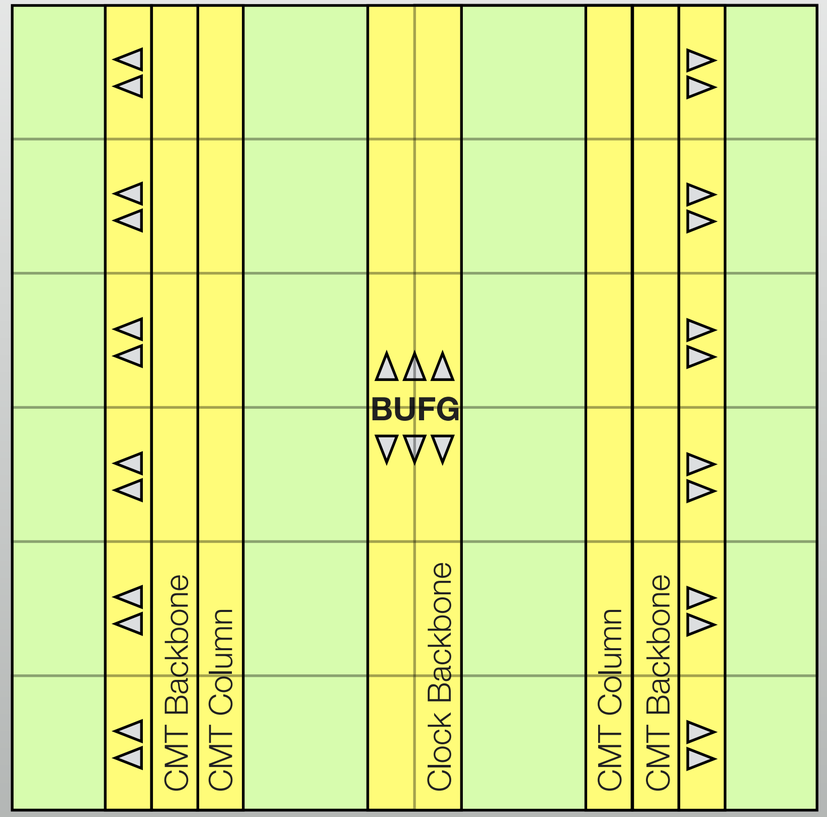
Regional Clock

- デバイス全体で使わないクロックもある
  - たとえばメモリを動かすためのクロック
- 一定の範囲でクロックを供給できればOK
  - Regional Clock を使いましょう
  - Global Clock より多くの系統が使える



Regional Clock

- 各 Clock Region には
  - 複数の Clock Capable I/O
  - 複数の Regional Clock Buffer を配置
- CMT は Clock Region ごとにあったり
  デバイスによっては、そんなになかったり



まとめ

- FPGAは基本的に均一な構造: LB, CB, SB
  - LB でプログラマブルな論理、CB+SB でプログラマブルな配線
- 特別な回路要素も必ず必要になる: 入出力ブロック・クロックツリー

LB に関する工夫

- LUT + FF + MUX が基本
- 「よくある回路」を効率よくつくれることが大事
  - LUT をシフトレジスタやメモリとして利用する仕掛け
  - キャリーチェーン専用の配線

クラスタ化

- クラスタ化することで CB-SB を経由せず LB どうしを接続可能に
  - 1つのLBに入りきらない場合について、ある程度高速化が期待できる
  - 1つのLUTに入らない大きめのメモリやシフトレジスタも構成

インターコネクト

- 隣接ブロックとの接続だけでは性能が出ない
  - 長距離・多数のブロック間を効率よく結ぶ必要がある
  - なので、トポロジはいろいろ工夫されている

入出力 (オフチップ)

- さまざまなシングルエンド・差動の I/O 規格に対応
  - いろいろな周辺デバイスと接続ができる
  - シングルエンド (パッシブ終端・アクティブ終端)
  - 差動

入出力 (オンチップ)

- 入出力ブロックには複数のレジスタを配置
  - オンチップとオフチップの遅延をきれいに切り分ける役目
  - 複数あるので DDR 入出力などにも利用できる

クロック

- 専用の入力ピン・クロックバッファ・クロックツリー配線
  - グローバルクロックは FPGA 全体で使える
    - 普通に論理合成されるのはグローバルクロック
  - リージョナルクロックを明示的に書けば特定のモジュールだけで使うようなローカルなクロック配線を使うことができる