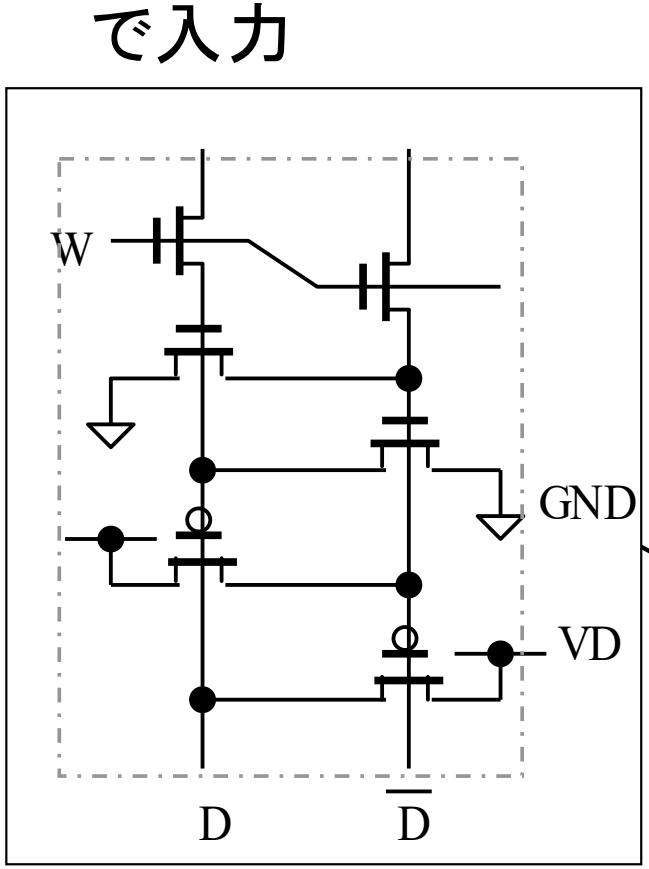
7.2 Implementation of design

各種実装方式の回路構成

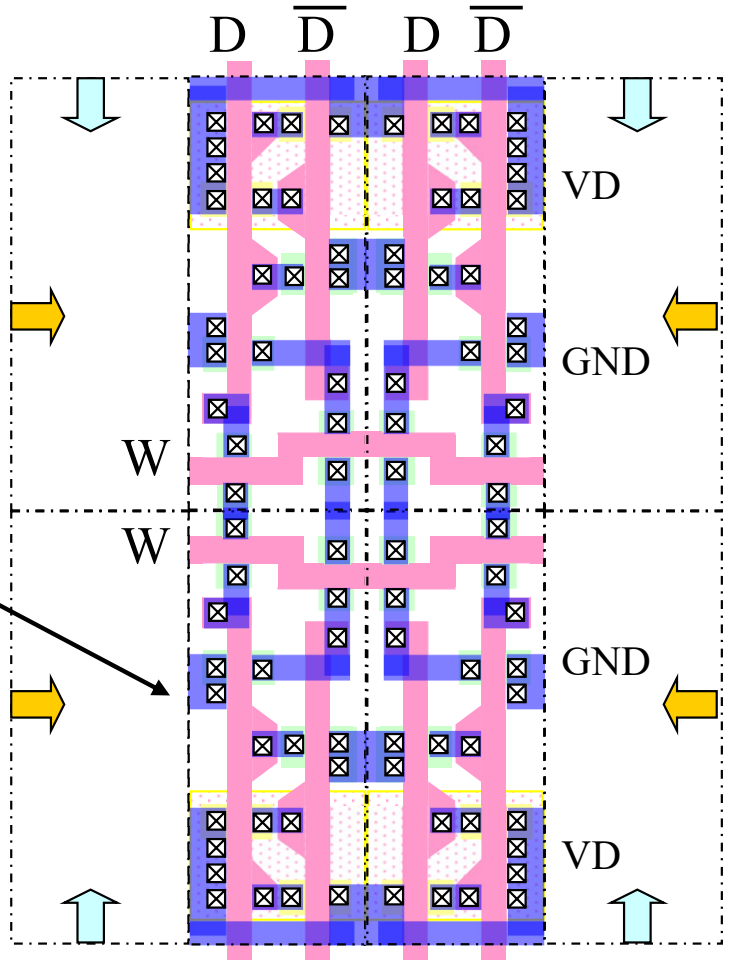
７．２．１　フルカスタム

要素機能のアートワーク

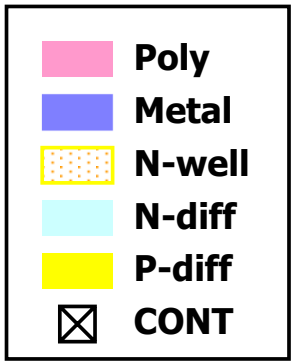
- レイアウト用CADツール（レイアウトエディタ）を用い、手で入力



6T CMOS SRAM

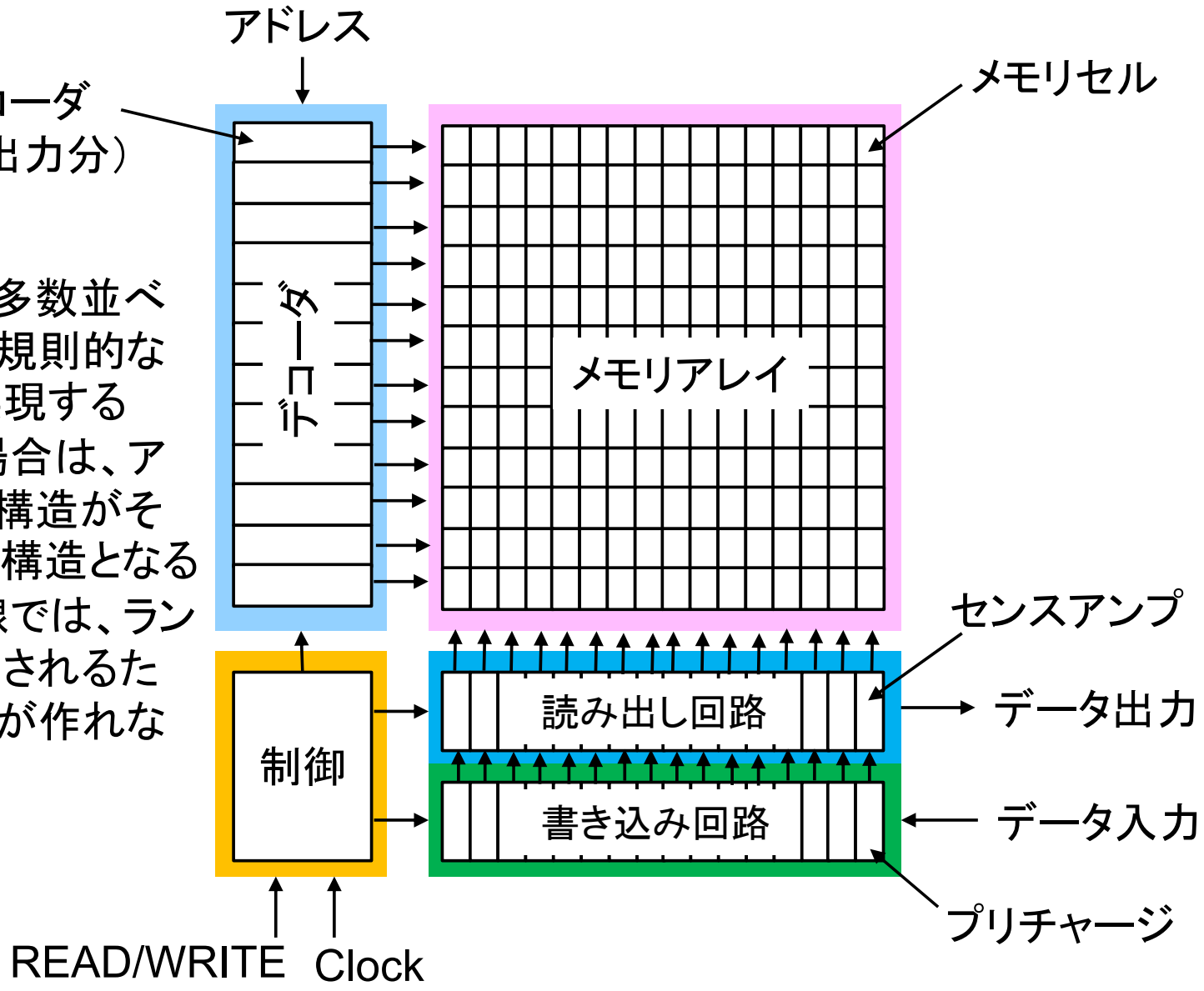


SRAMセルのレイアウト例（メモリも自動配置配線不可）



各種機能の結合

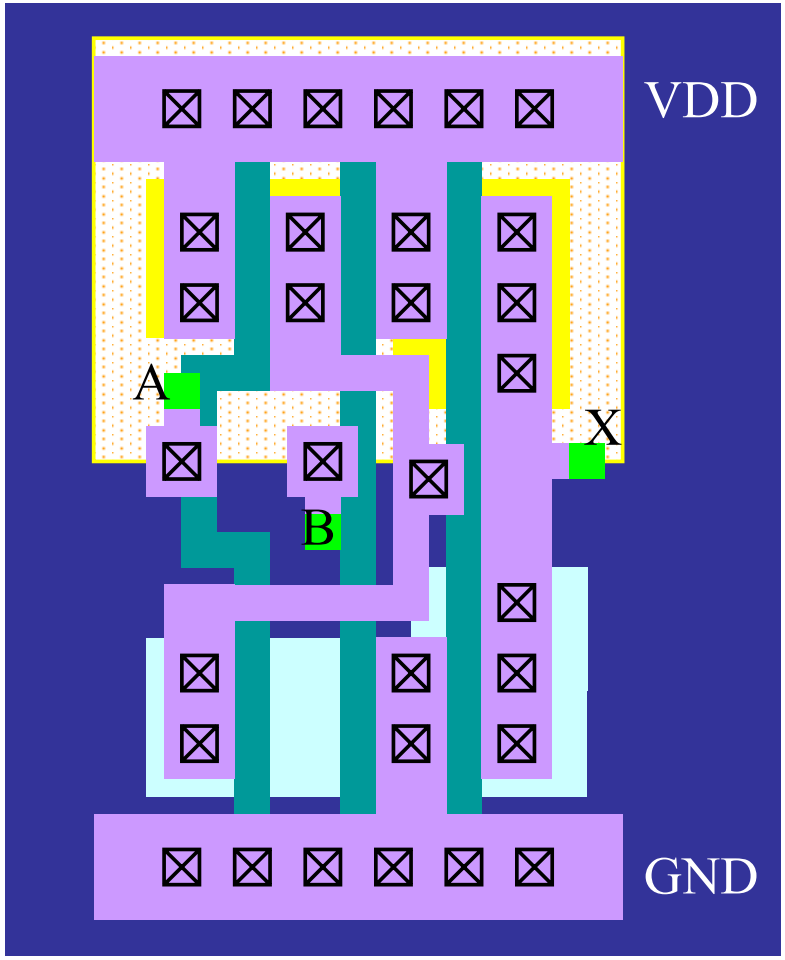
- 小さい回路を多数並べることにより、規則的な回路構造を実現する
- 演算回路の場合は、アルゴリズムの構造がそのまま回路の構造となる
- 自動配置配線では、ランダムに最適化されるため、規則構造が作れない



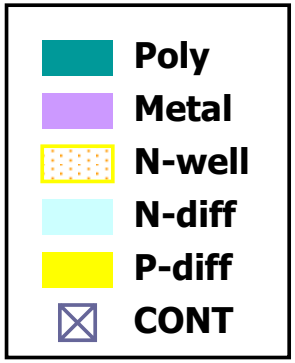
7．2．2　セルベースIC
(Cell based IC)

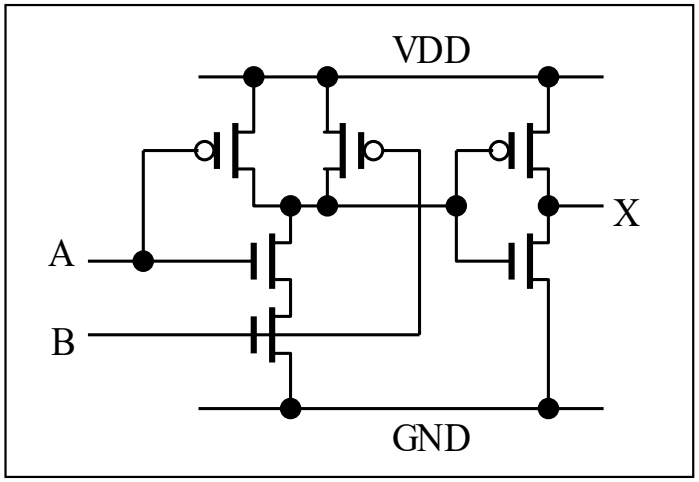
スタンダードセルの構造

- スタンダードセル用の2入力ANDセルのレイアウト例
- 各種のセルを同じ高さになるようにレイアウトする



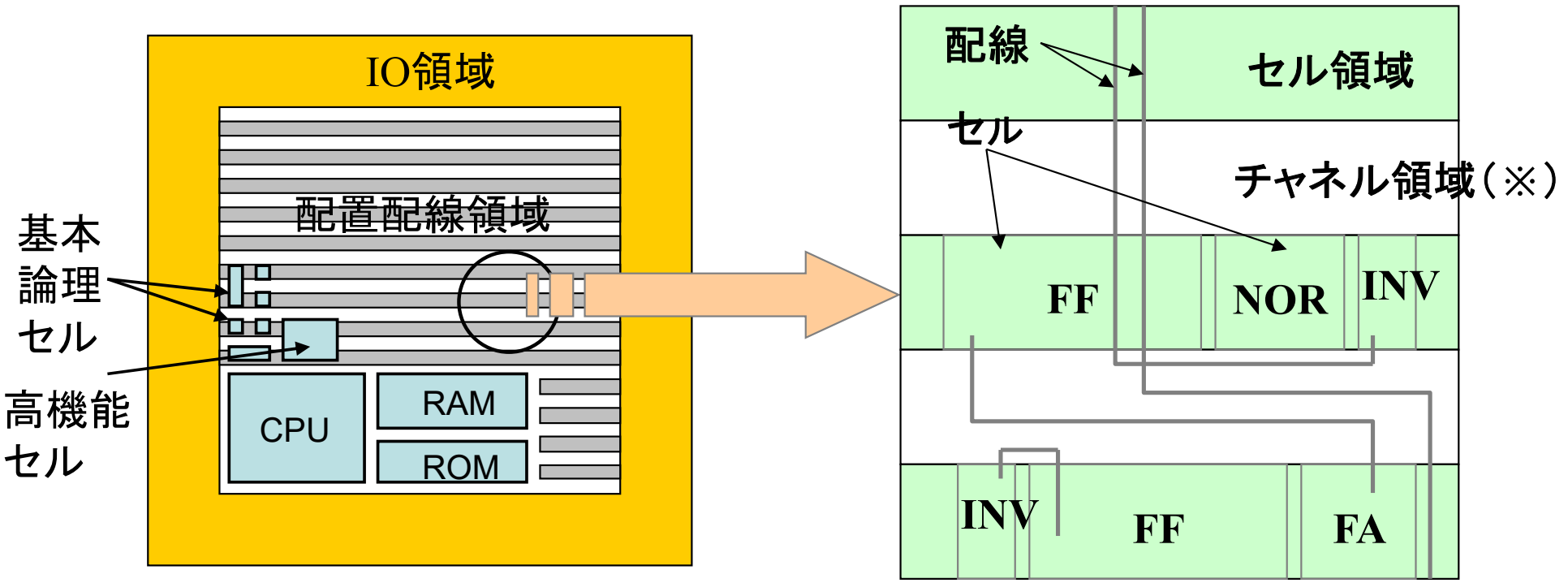
入出力端子





セルベースICの構造

- 配置配線領域に論理ゲートのセルを自動的に最適配置し、さらに自動的にそれらに配線を行う

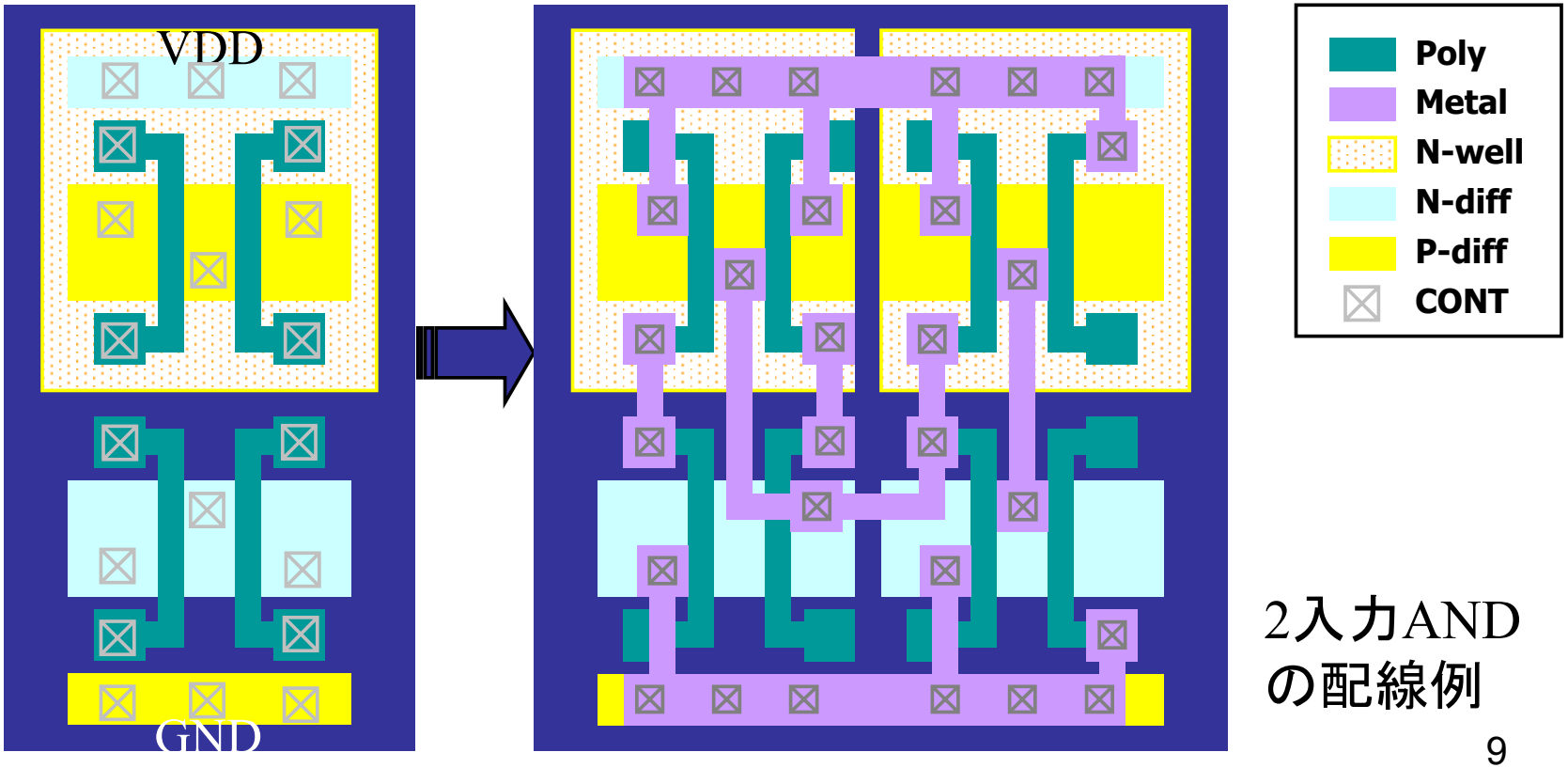


※ 配線層数が多い場合はチャネル領域は不要

7．2．3　マスタースライス

Gate Array

- ゲートアレイセルと呼ばれる数トランジスタ単位のパターン（下地セル）が敷詰められたウエーハ（マスタースライス）上に、配線の加工だけを行う

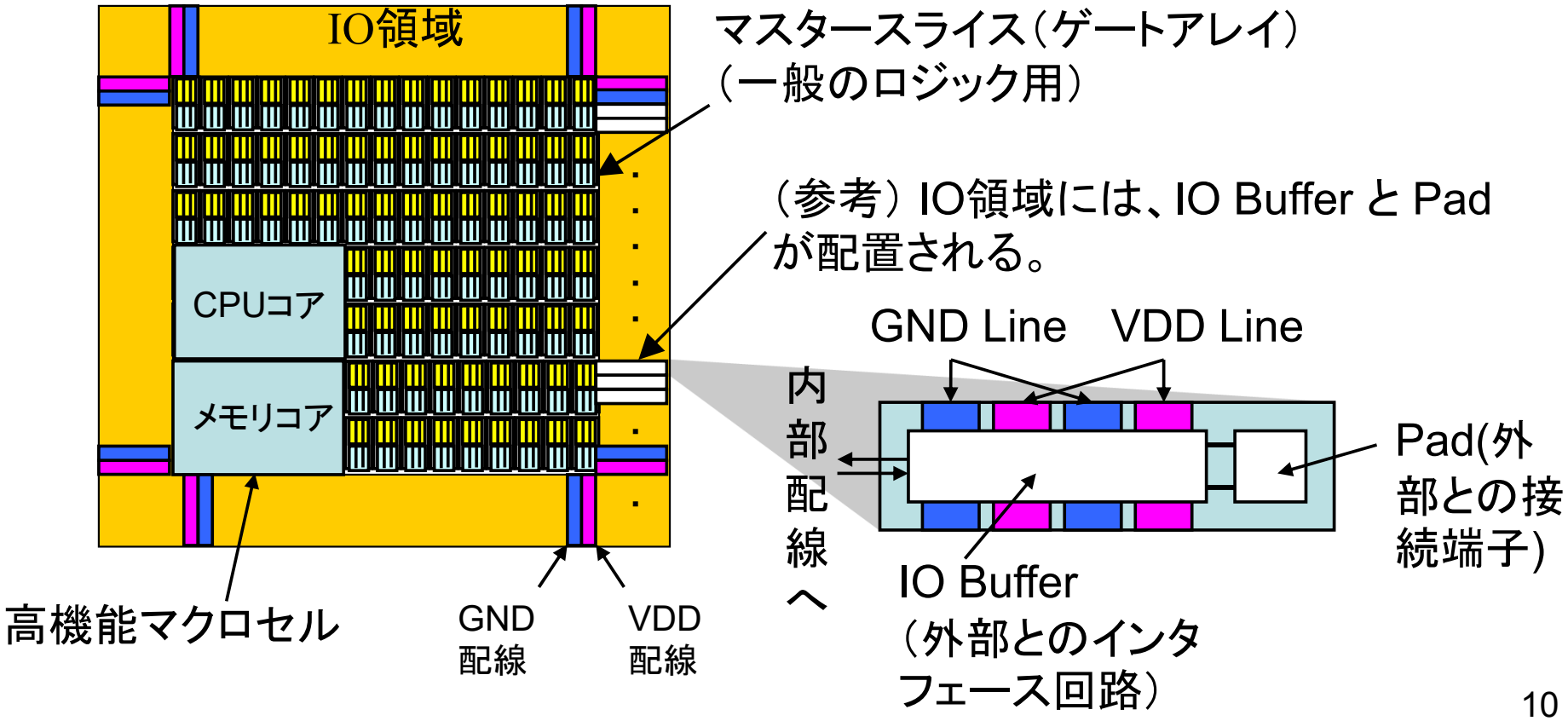


下地セル

2入力AND
の配線例

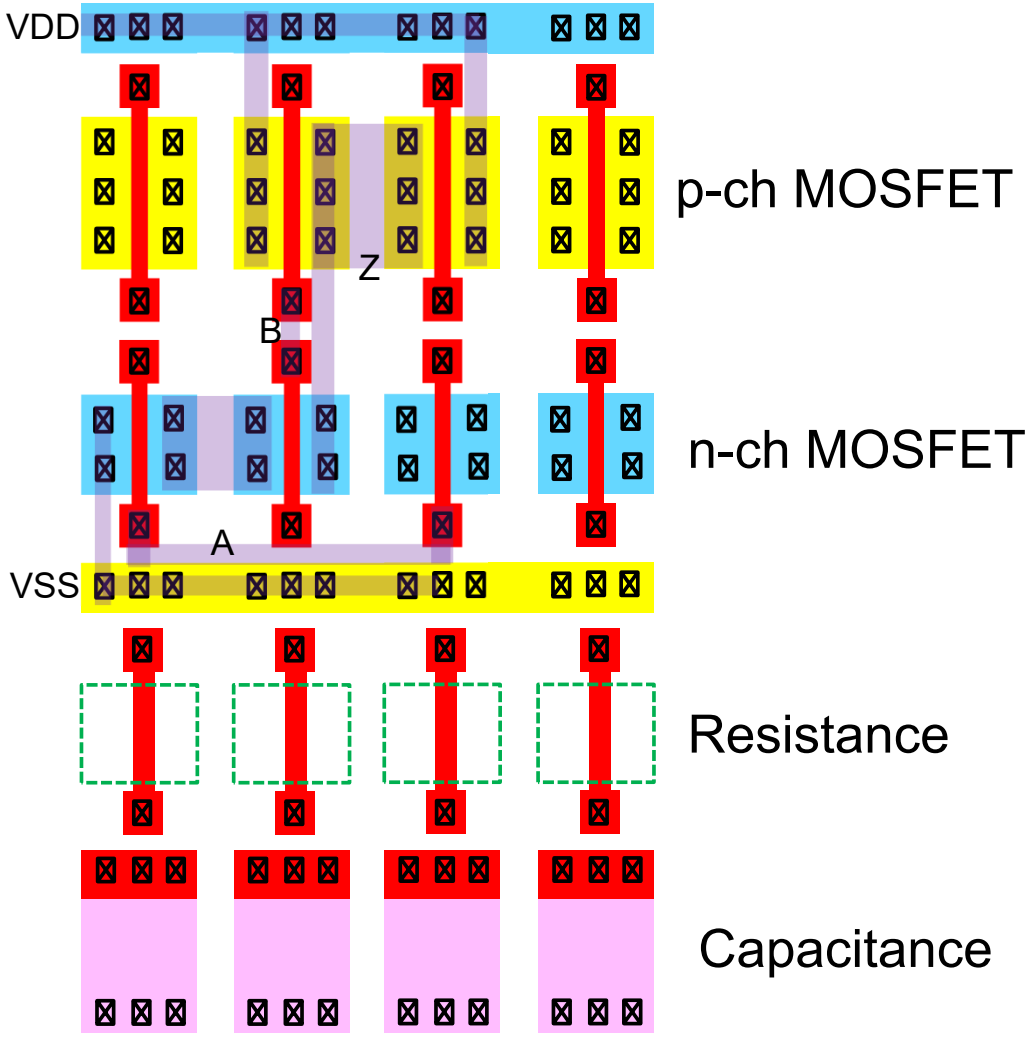
Embedded Array

- ゲートアレイ＋作り付けの高機能マクロセル
- 短TAT（turn around time）と高集積化を両立



アナログマスタースライス

- アナログ回路の設計に必要なMOSFET, C, Rのマスタースライス上で、配線加工だけを行う
- 面積利用効率が低くなるが、ディジタル回路の自動配置配線も可能

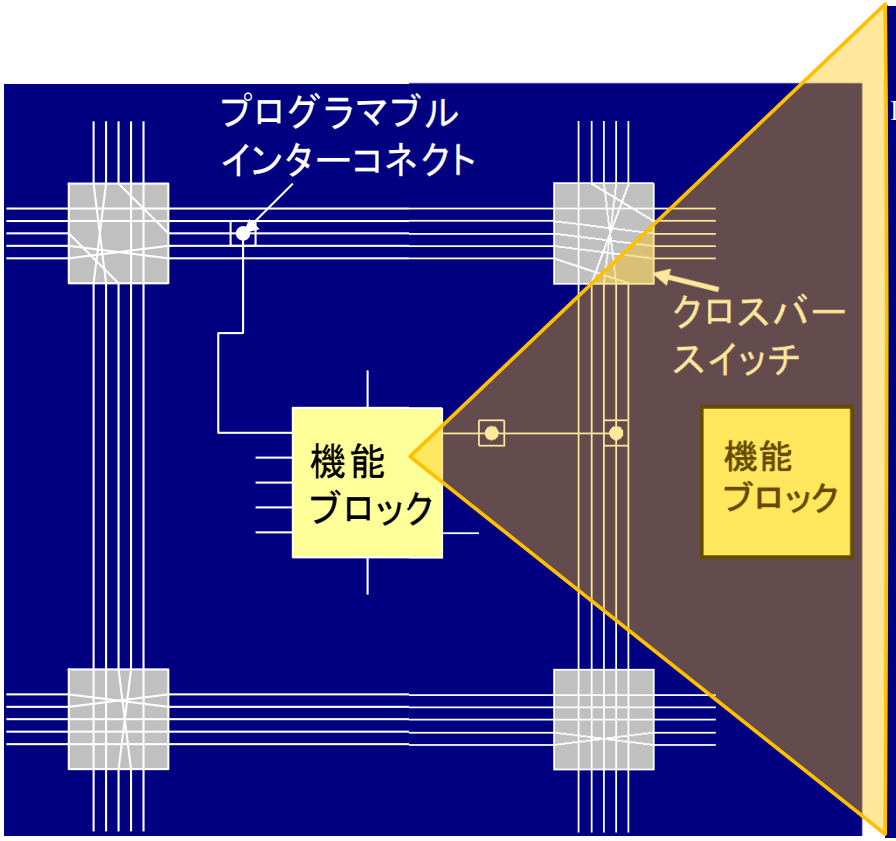


2入力NANDの配線例

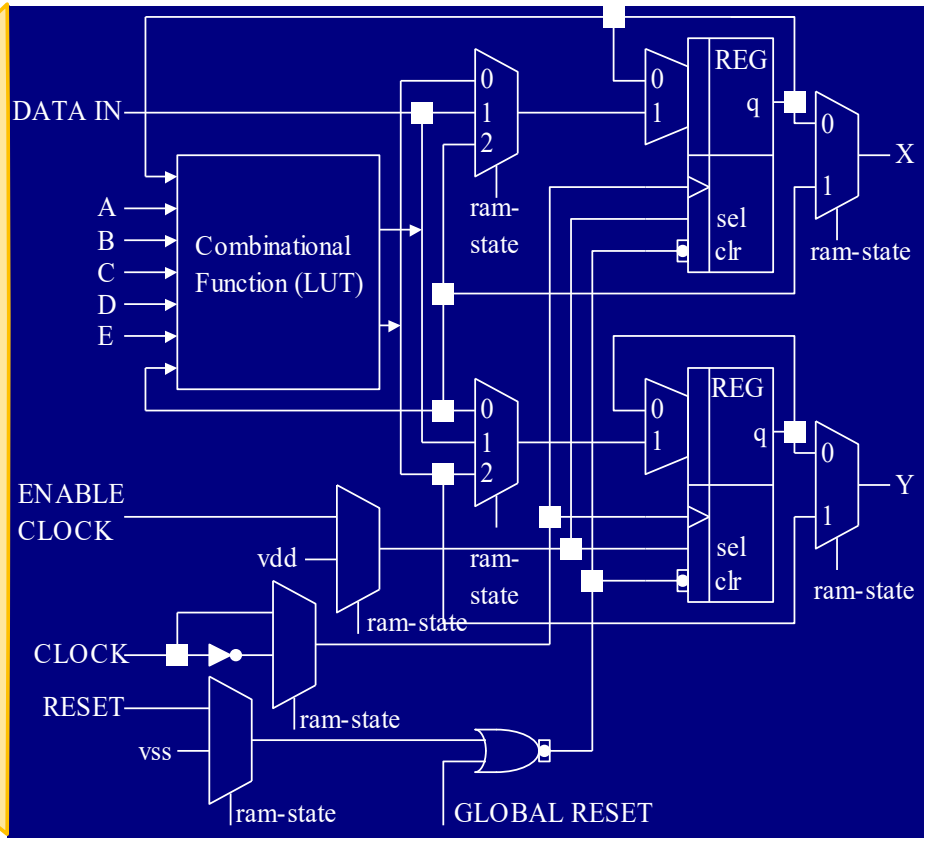
7．2．4　PLD(Programmable Logic Device)

FPGA

- 論理機能(演算器やレジスタが実現できる)ブロックがアレイに並べられており、そのブロックの機能と配線のデータをRAMに書き込むと必要な機能が実現される



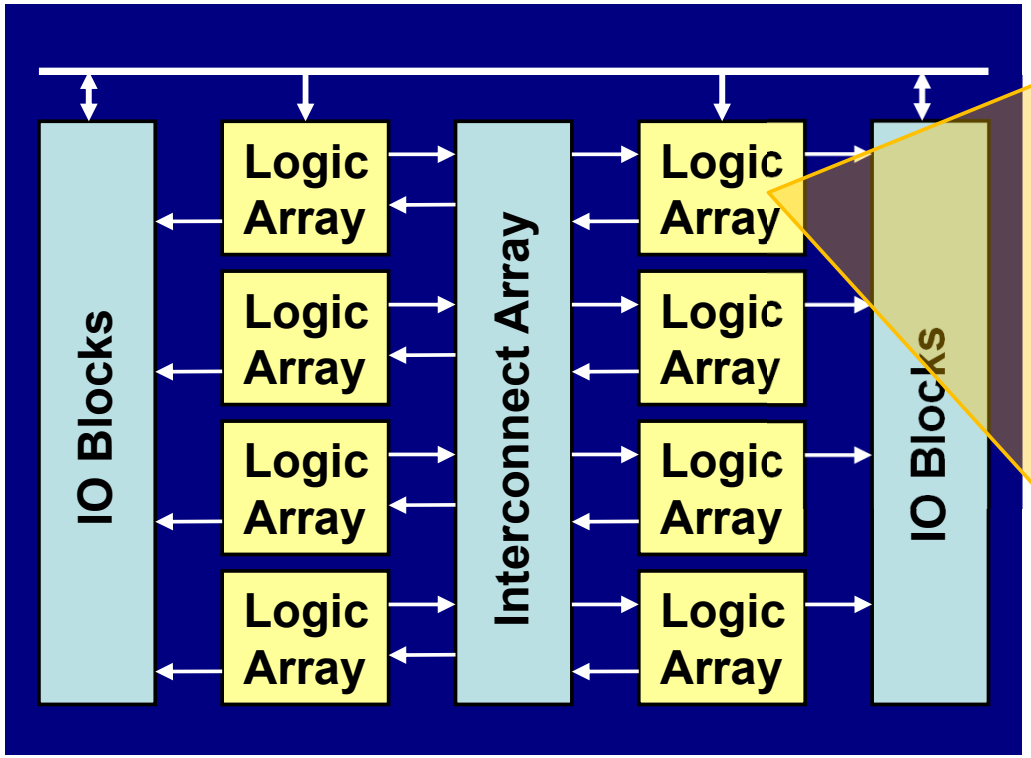
クロスバーとローカル配線



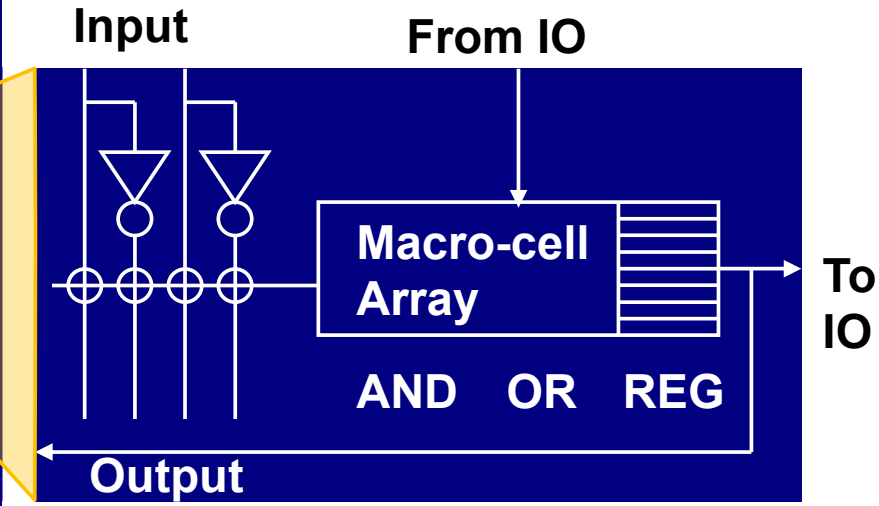
機能ブロックの構造(Xilinx社の例)

CPLD

- 多数のLogic ArrayをInterconnect Array（配線とスイッチ）で結合
  - Logic Array: 複数のPLA (Programmable Logic Array = プログラム可能な組合せ回路)と出力レジスタ
    - PLAは、加法標準形（積項の和）で表現された論理式をANDとORのアレイと配線用スイッチで表現する回路



CPLD(Altera社の例)



Logic Array

傾向としては:
FPGAは大規模回路向き
CPLDは小規模・高速・低価格向き

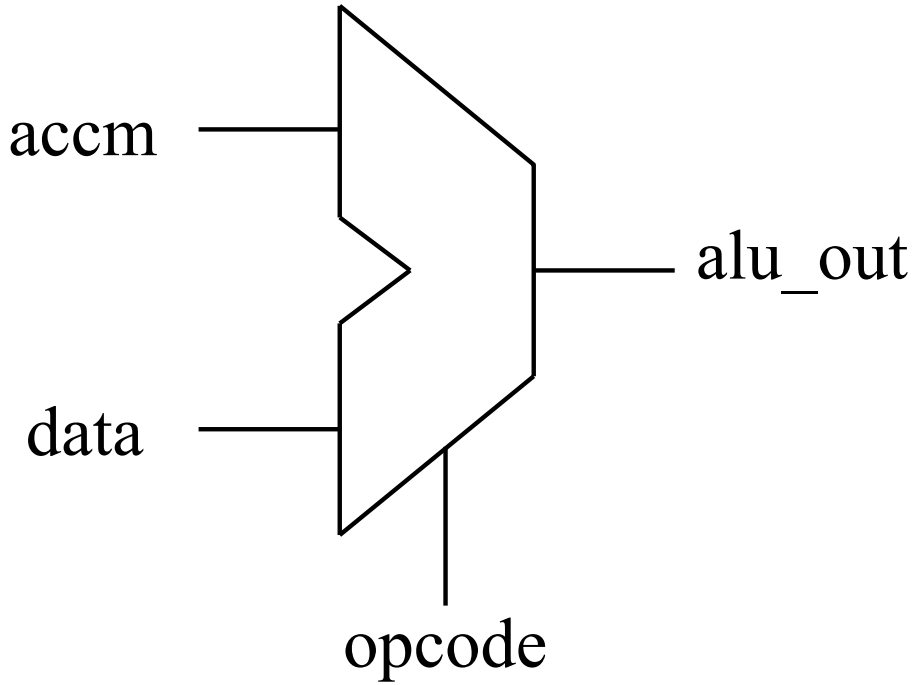
実装方式の選択

	フルカスタム	セルベース	マスタースライス	FPGA, CPLD	MCU
開発費	×	△	△	○	◎
開発期間	×	△	○	◎	◎
単価	○	○	○	×~△※1	○
設計自由度と性能	◎	○	○	△	×
機能集積度	○	△	×	×	○

※1　FPGA, CPLDなどのプログラマブル・デバイスの高集積化、高性能化、低価格化が進行しており、時代とともに得失は変化する。

７．２．５　セルベースIC設計フロー

ALU（算術論理演算ユニット）の設計例



HDL(Hardware Description Language)記述

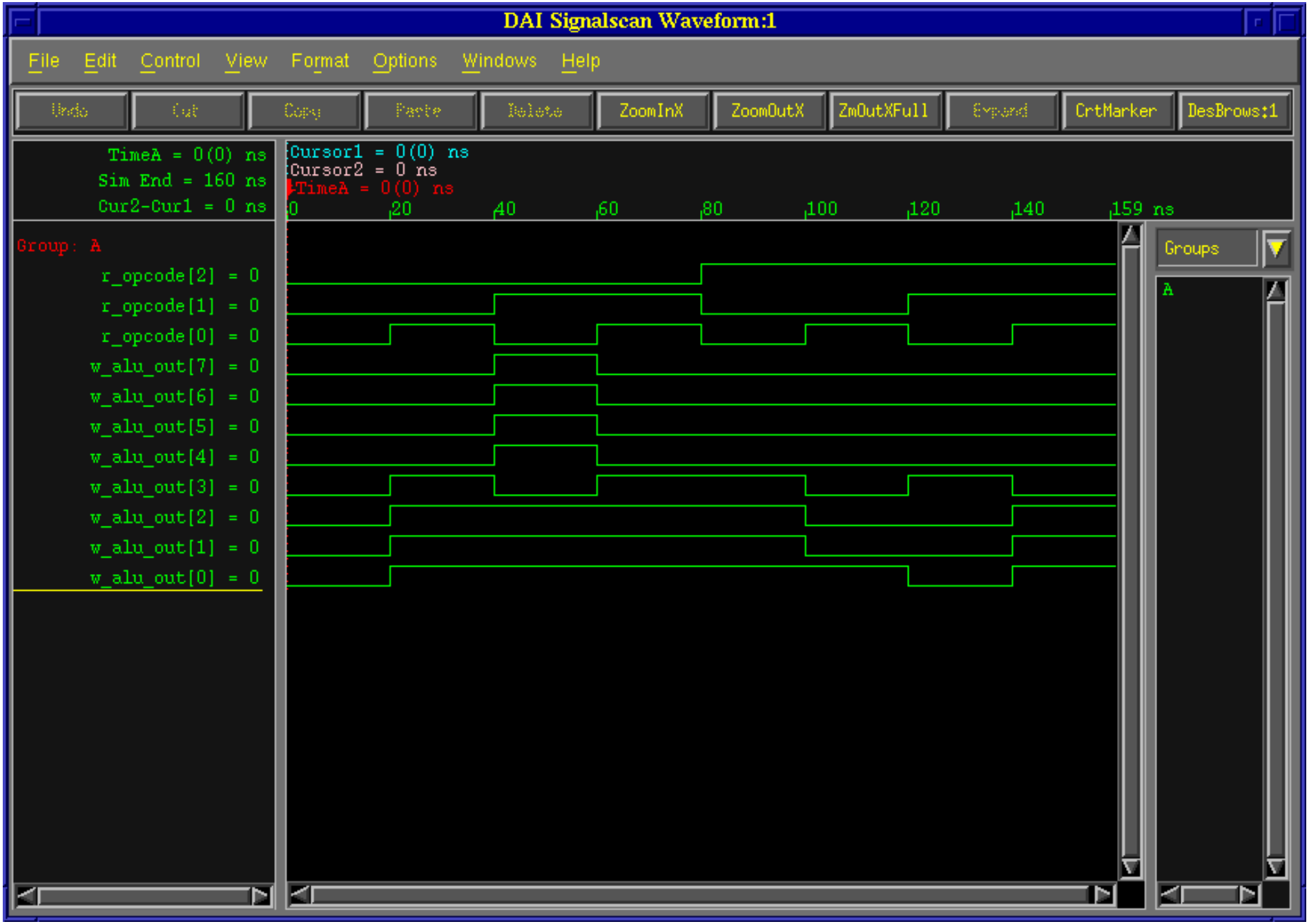
- RTLと呼ばれる論理合成可能な記述スタイル

kterm

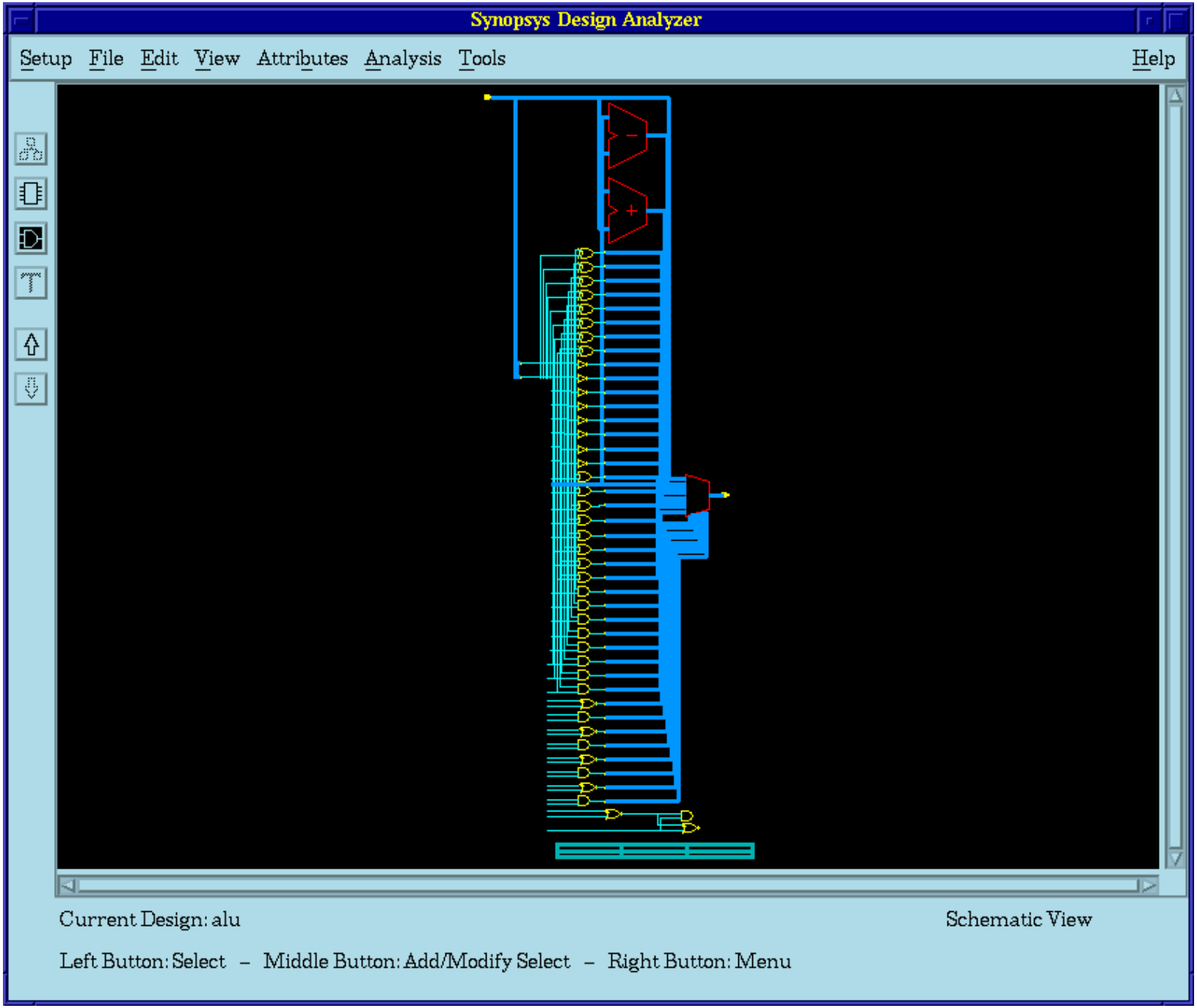
```
vlsisv04{kitagawa}25: cat alu.v
`timescale 1ns/1ns

`define AND 3'b000
`define OR  3'b001
`define NOT 3'b010
`define XOR 3'b011
`define ADD 3'b100
`define SUB 3'b101
`define ACC 3'b110
`define DAT 3'b111

module alu (alu_out, accm, data, opcode);

    input  [7:0] accm, data;
    input  [2:0] opcode;
    output [7:0] alu_out;
    reg    [7:0] alu_out;

    always @(accm or data or opcode) begin

        case (opcode)
            `AND : alu_out <= accm & data;
            `OR  : alu_out <= accm | data;
            `NOT : alu_out <= ~accm;
            `XOR : alu_out <= accm ^ data;
            `ADD : alu_out <= accm + data;
            `SUB : alu_out <= accm - data;
            `ACC : alu_out <= accm;
            `DAT : alu_out <= data;
           default : alu_out <= 8'bxxxxxxxx;
        endcase

    end
endmodule
vlsisv04{kitagawa}26:
```

Verilog HDLによる8bit ALUの記述例

機能シミュレーション

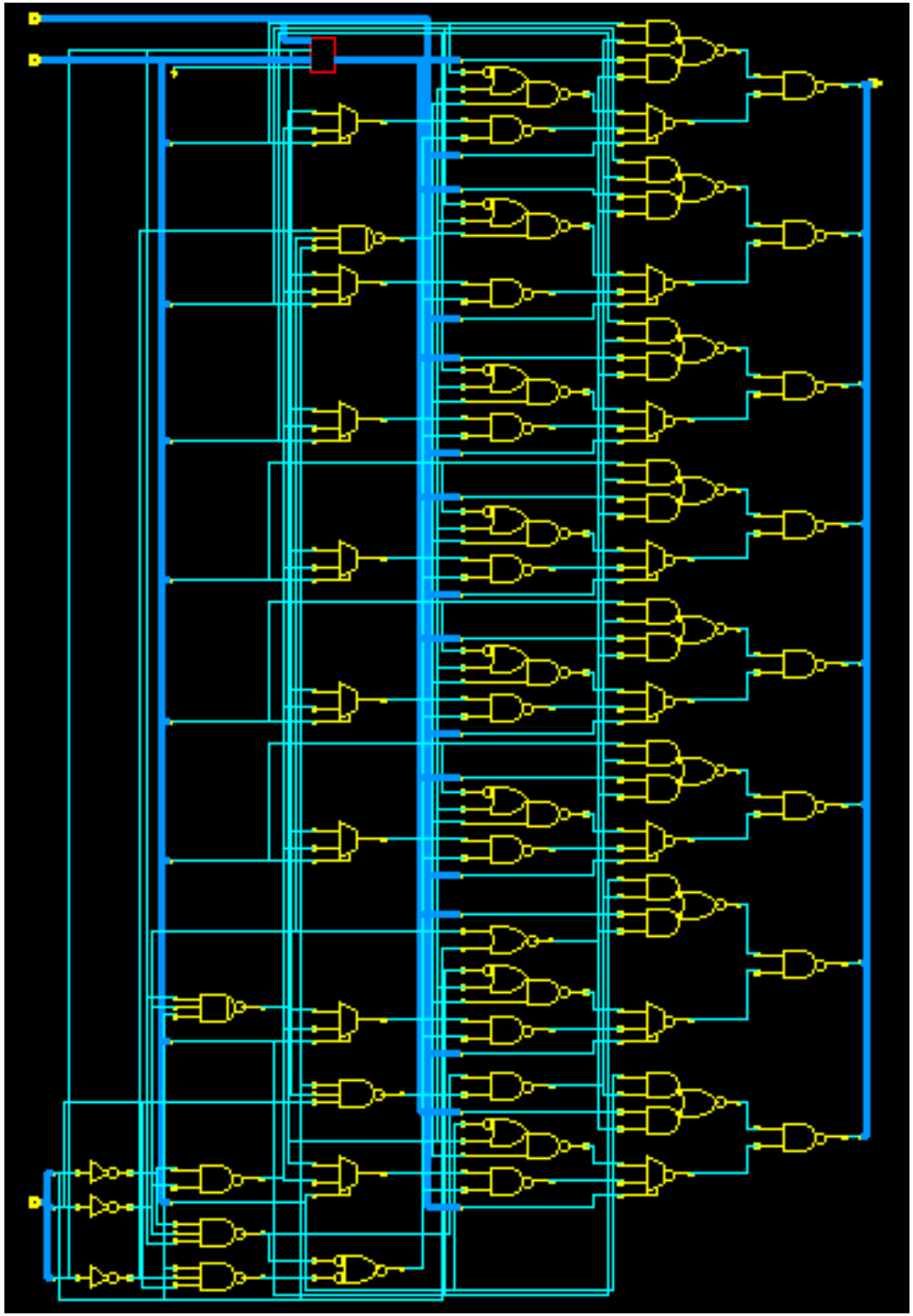

DAI Signalscan Waveform:1
File Edit Control View Format Options Windows Help
Undo Cut Copy Paste Delete ZoomInX ZoomOutX ZmOutXFull Expand CrtMarker DesBrows:1
TimeA = 0(0) ns
Sim End = 160 ns
Cur2-Cur1 = 0 ns
Cursor1 = 0(0) ns
Cursor2 = 0 ns
TimeA = 0(0) ns
0 20 40 60 80 100 120 140 159 ns
Group: A
r_opcode[2] = 0
r_opcode[1] = 0
r_opcode[0] = 0
w_alu_out[7] = 0
w_alu_out[6] = 0
w_alu_out[5] = 0
w_alu_out[4] = 0
w_alu_out[3] = 0
w_alu_out[2] = 0
w_alu_out[1] = 0
w_alu_out[0] = 0
Groups
A

論理合成（HDLそのまま回路図化）



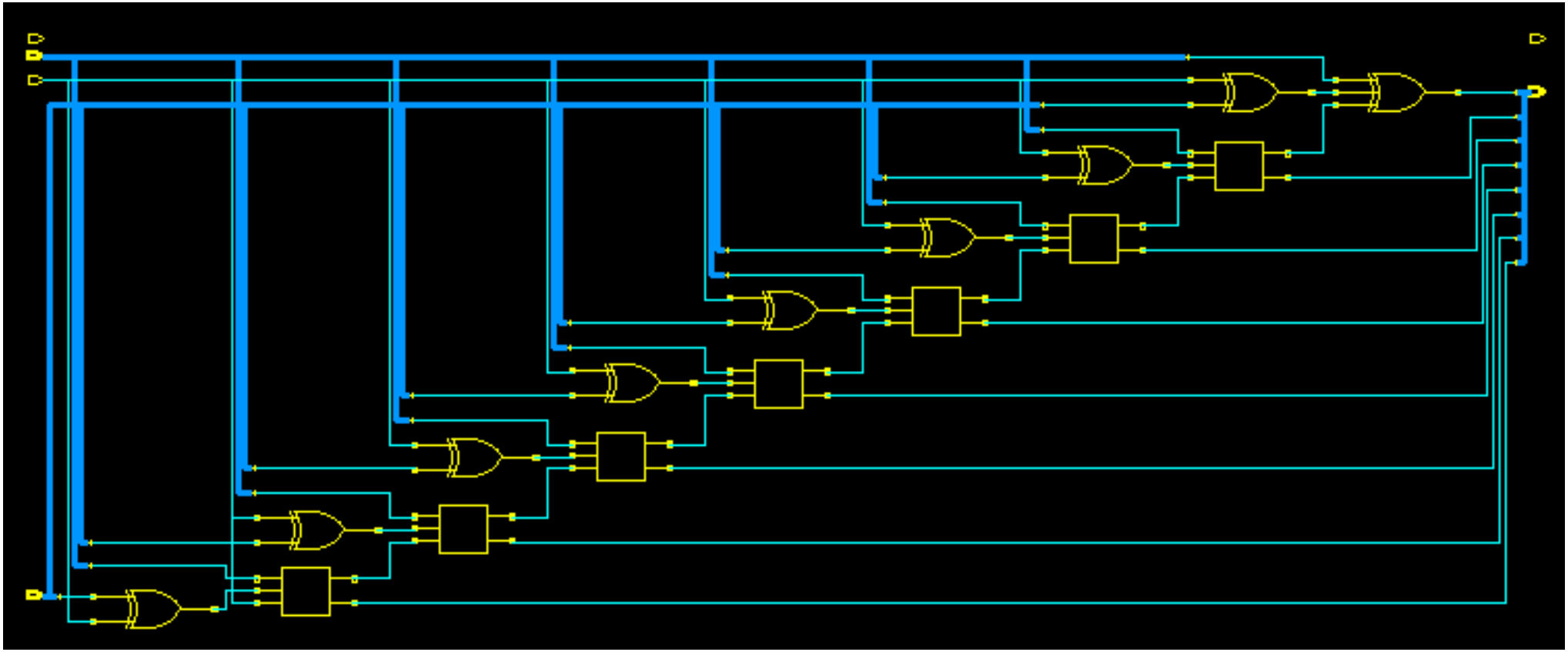
最適化

- 論理演算およびセレクタ部

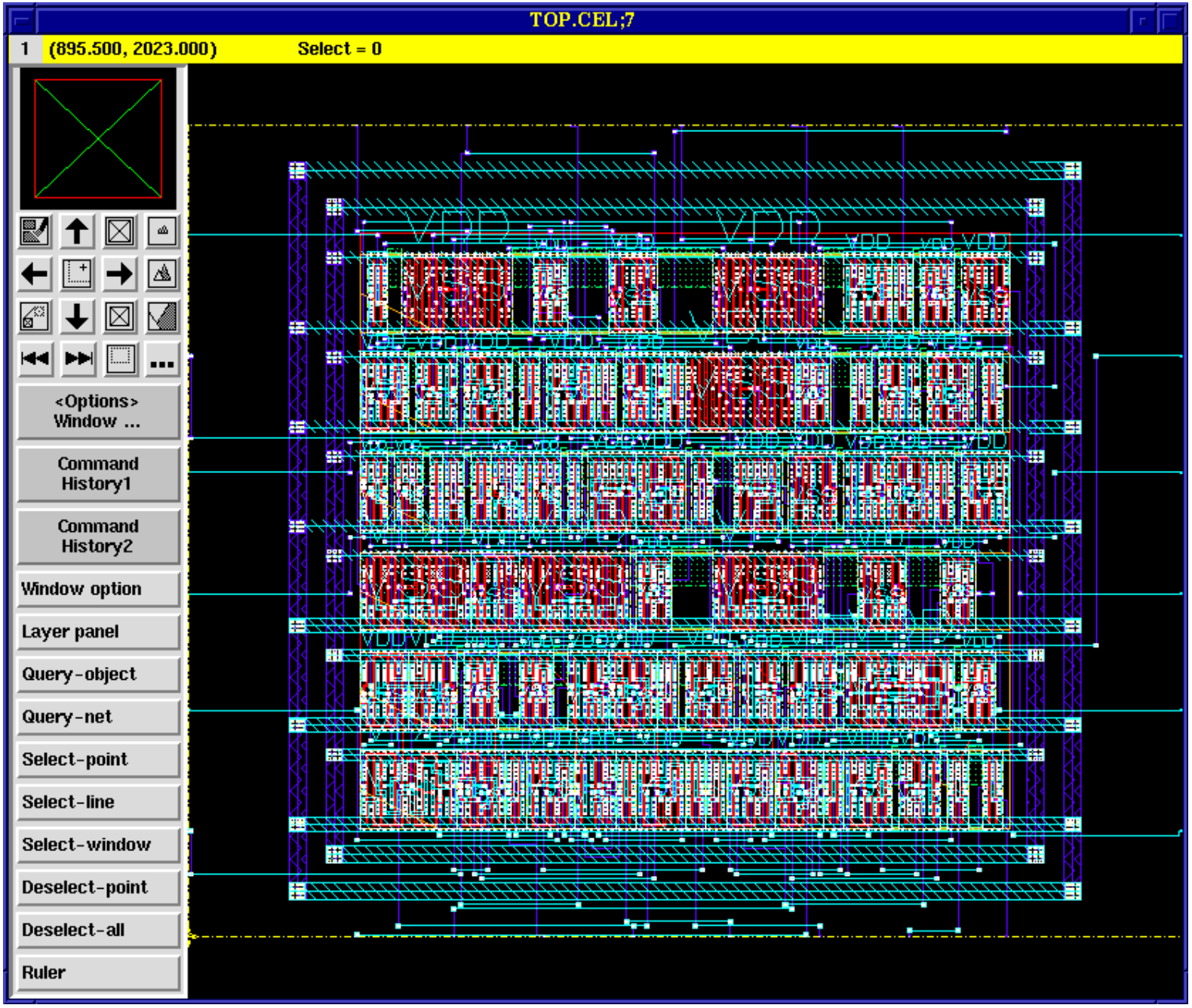


加算器のシェア

- 加算と減算を1個の加算器で実現

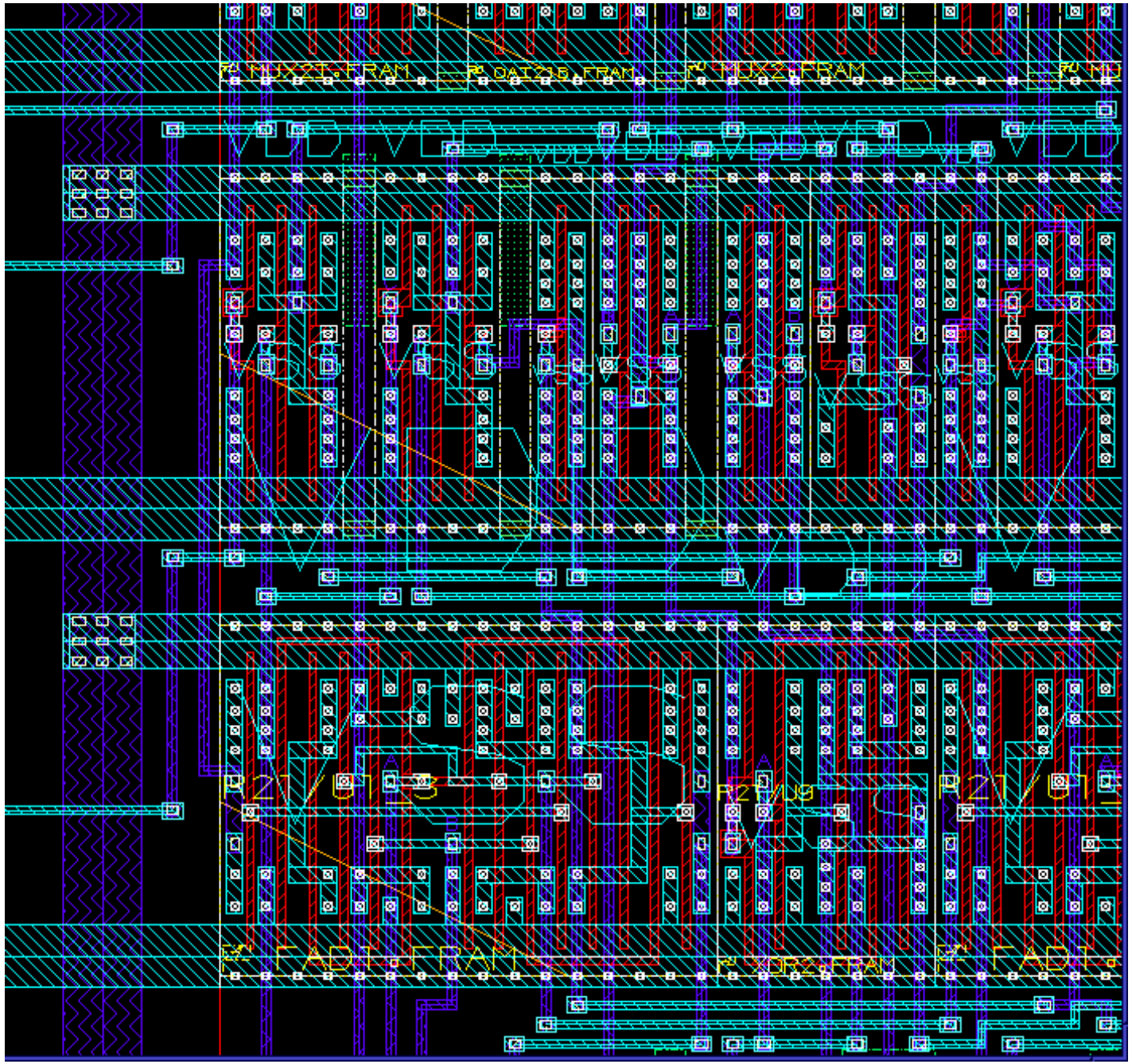


自動配置配線

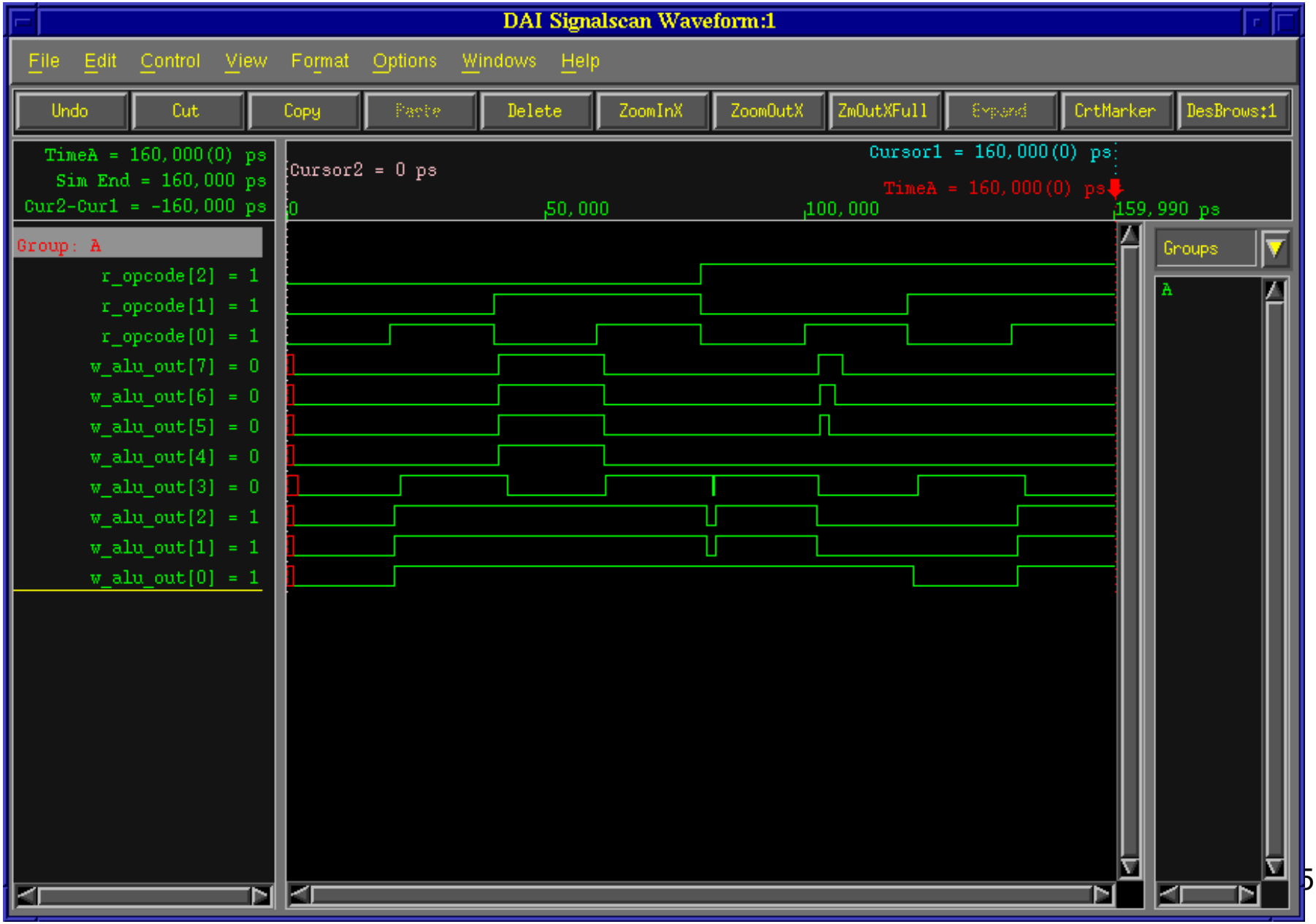

TOP.CEL;7
1 (895.500, 2023.000) Select = 0
<Options> Window ...
Command History1
Command History2
Window option
Layer panel
Query-object
Query-net
Select-point
Select-line
Select-window
Deselect-point
Deselect-all
Ruler

論理セルの構造

- 赤色：Poly
- 水色：M1
- 紫色：M2



ポストレイアウト・シミュレーション



７．２．６　オープンチップ開発

ポストムーアの時代

- スケーリング則によりより高速、低消費電力、低価格を求めて微細化が進行したが、スケーリングには限界がある（終焉が近いと言われている）
- データセントリック社会の実現には、微細化以外の高性能化と低コスト化指針が必要
  - 効率の高い専用ハードウェアによるデータ通信の高効率化と低消費電力化
  - 成熟した製造技術と既存設備の活用による少量生産～大量生産への対応と低コスト化
  - ミクストシグナル（アナログ-ディジタル-RF）、量子-古典（量子ビット-ディジタル）混合システムの構築

オープンチップ開発で求められていること

- 開発ツールのオープンソース化または低価格化
- IP（設計資産）のオープンソース化
- NDA（秘密保持契約）からの解放
- 無償または低価格の試作サービス
- 短TAT(Turn Around Time)の製造技術

オープンチップ関連の動向

- ミニマルEDA
  - https://www.minimalfab.com/update/170
- MakeLSI
  - https://www.facebook.com/makelsi/
- Googleオープンチップ開発ポータル
  - https://developers.google.com/silicon
- Googleのオープンチップ開発の解説（日本語訳）
  - https://developers-jp.googleblog.com/2022/06/build-open-silicon.html