



# 情412:CAD

(Computer Aided Design)

## イントロダクション

---

琉球大学工学部情報工学科  
ファイヤー和田 知久

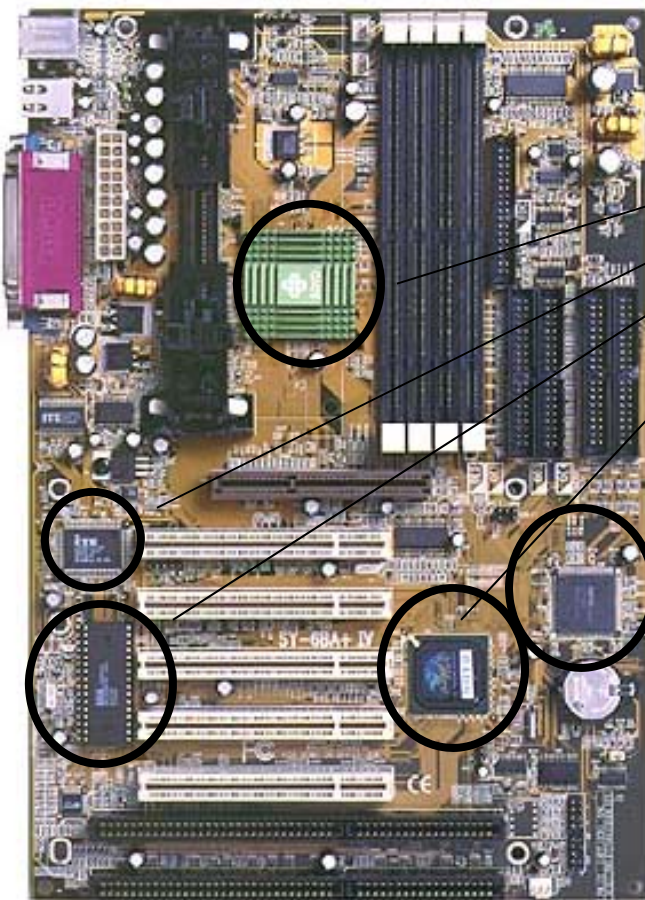
wada@ie.u-ryukyu.ac.jp

<http://www.ie.u-ryukyu.ac.jp/~wada/>

# デジタルを支える半導体技術



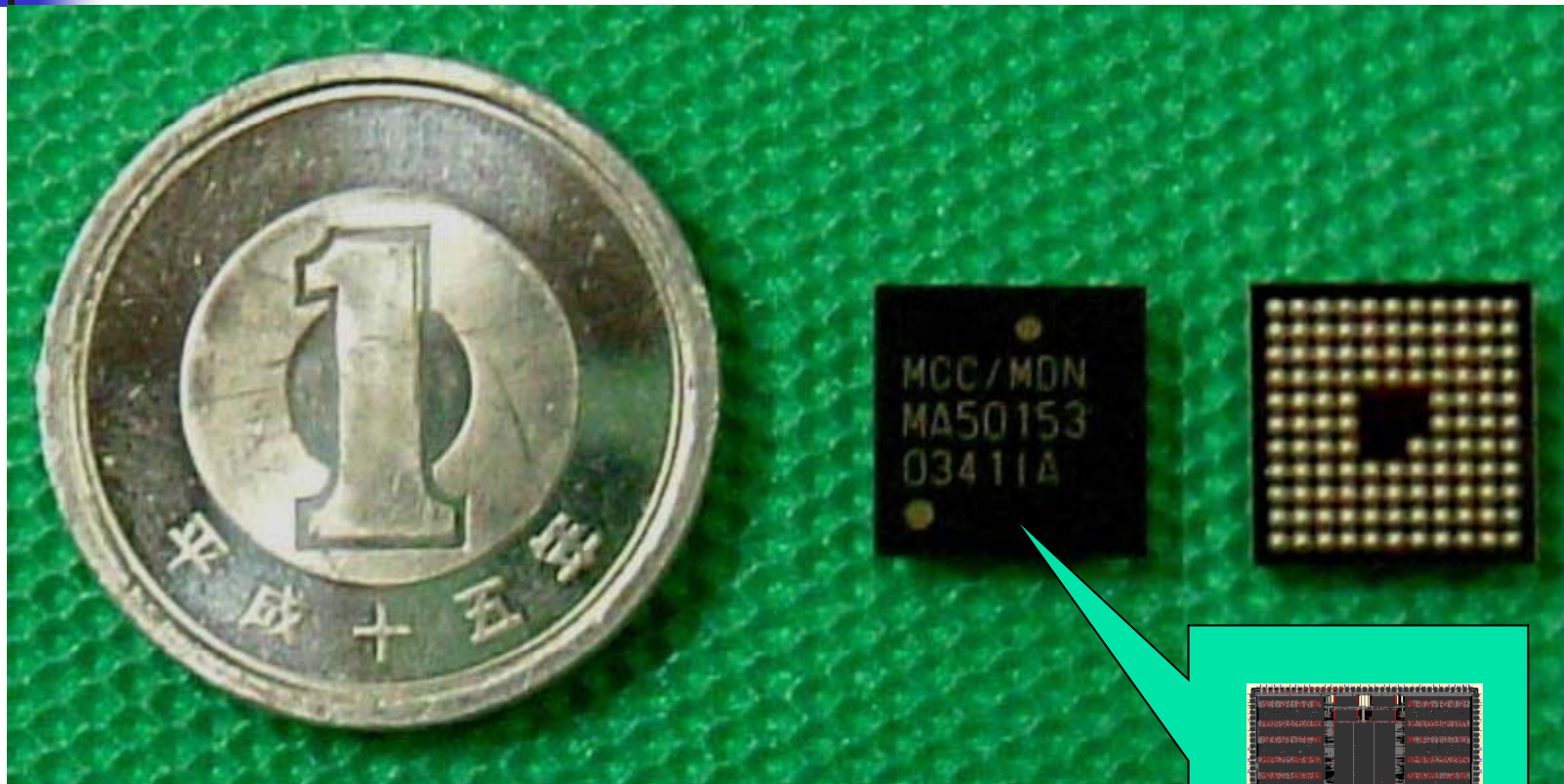
# パソコンを開けてみると...



- VLSIとソケットばかり

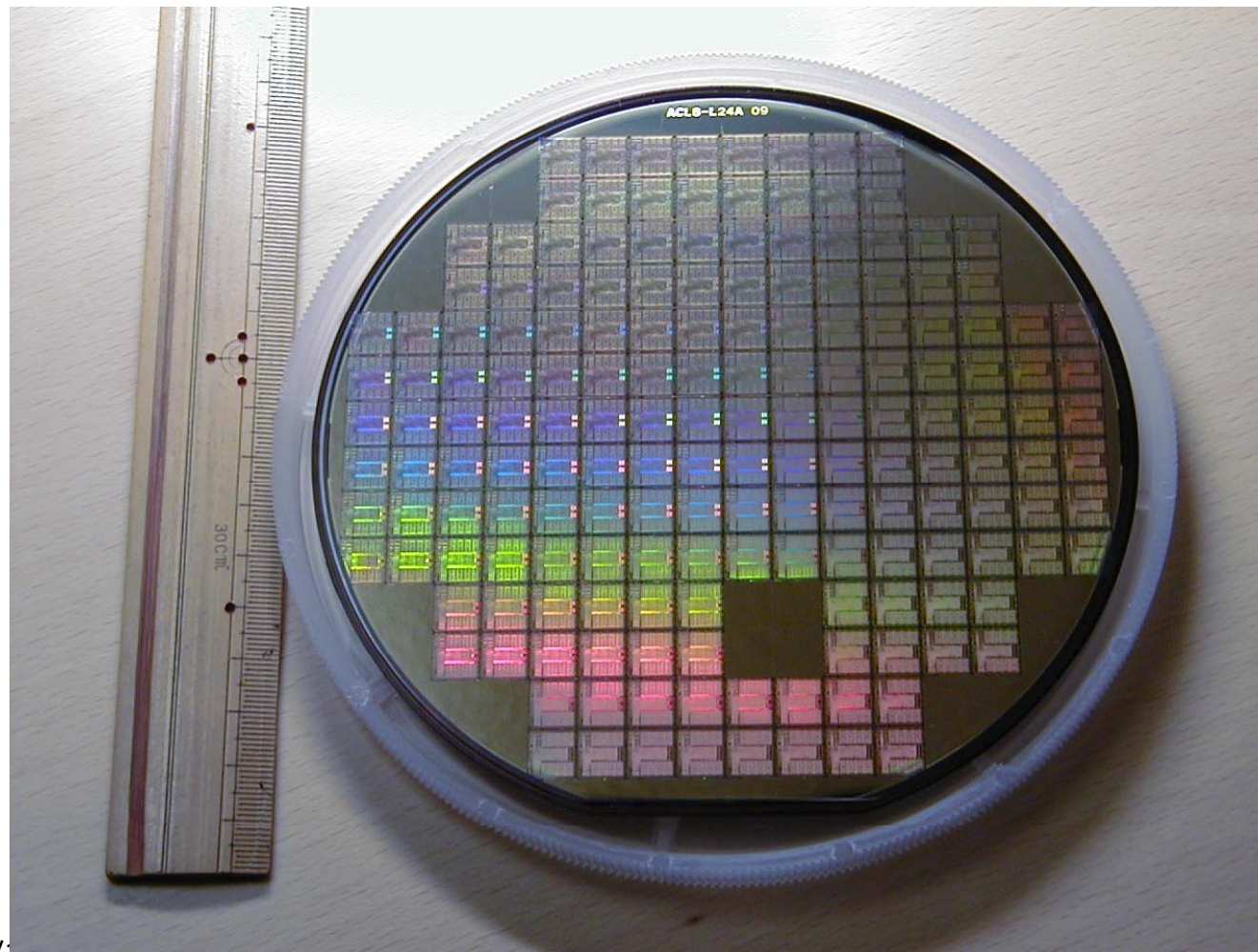
- 最近の電子機器は必ずVLSIで構成される  
(携帯電話、プレイステーション等)

# ファイヤー和田研 & マグナ社の 携帯電話用デジタルTV受信LSI

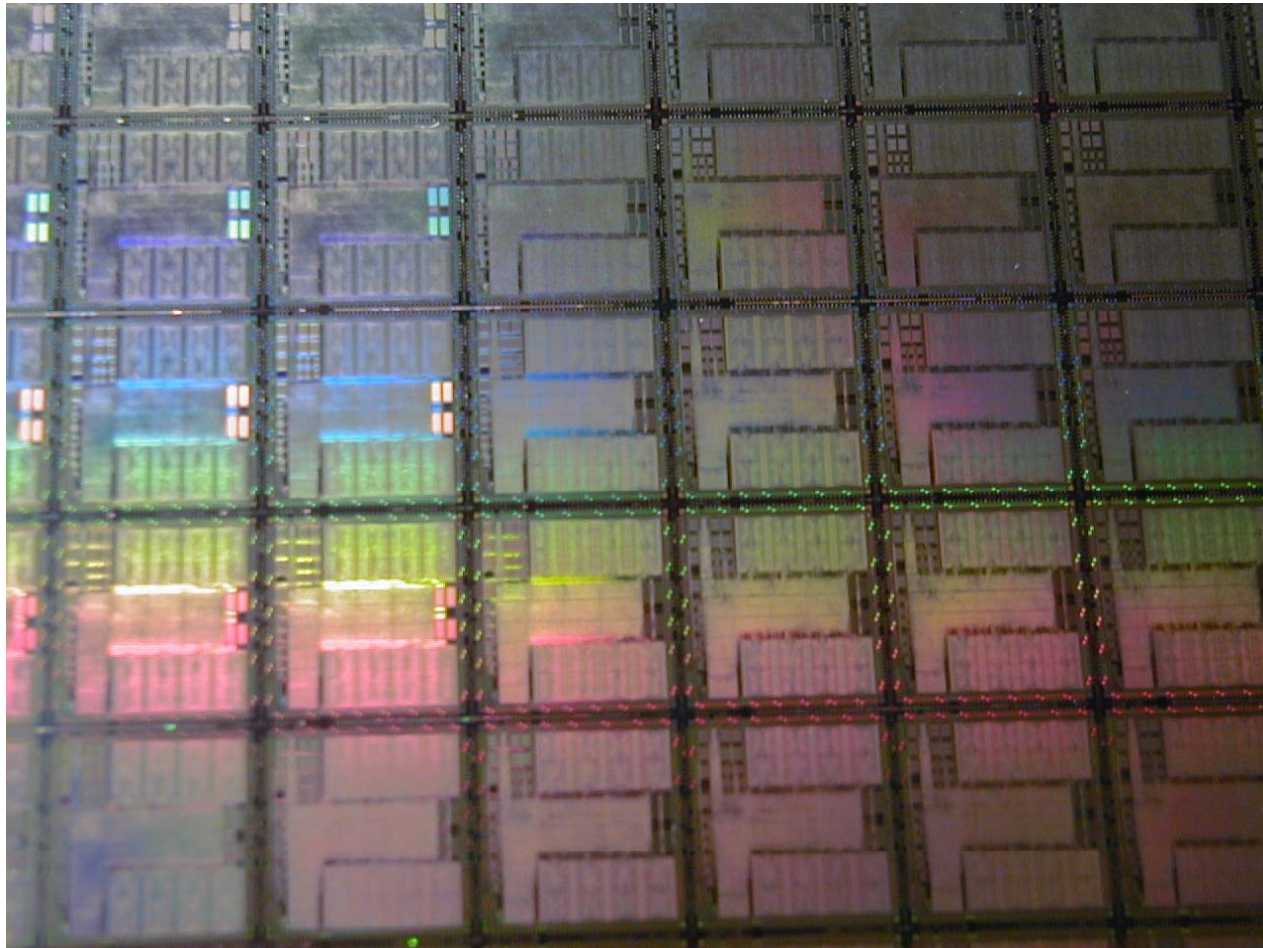


このチップには数百万トランジスタが集積されている

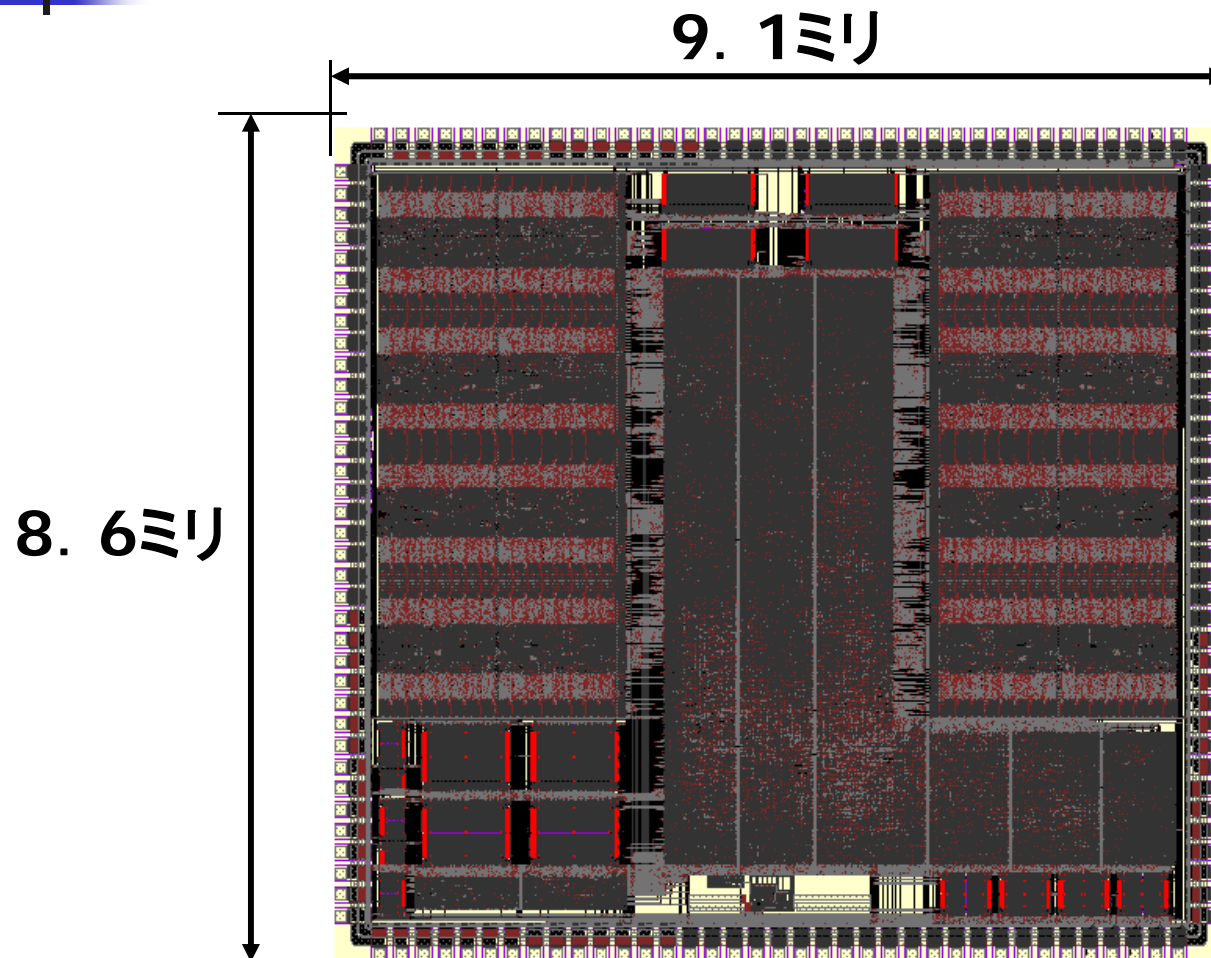
# シリコンウエハ



多数のチップを同時にウエハ上に作成する。



# 1つのチップ



だいたい  
小指の爪  
の大きさ。  
200万  
トランジス  
タ  
を集積

# チップの断面の電子顕微鏡写真



1ミクロンは1ミリの千分の1

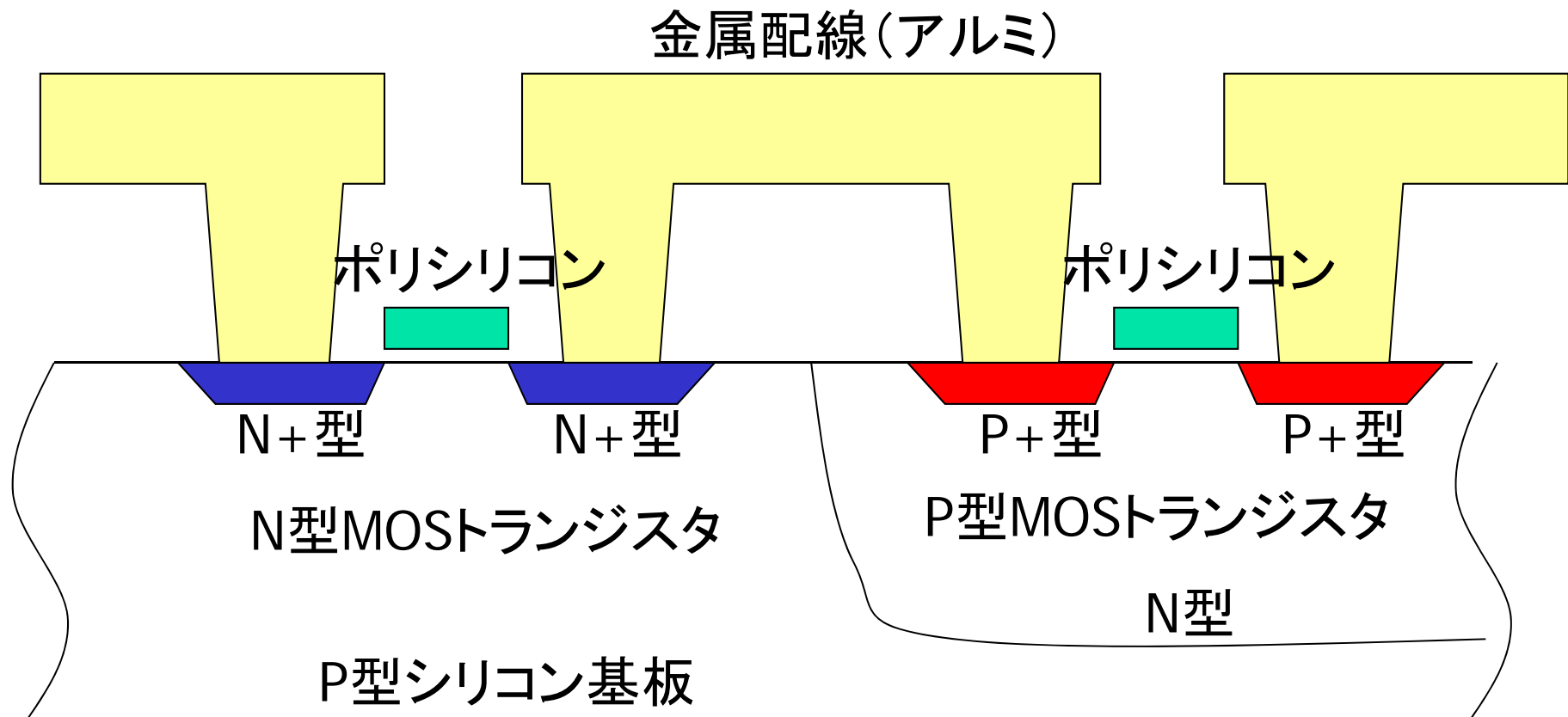
0.5μm



0.5ミクロン

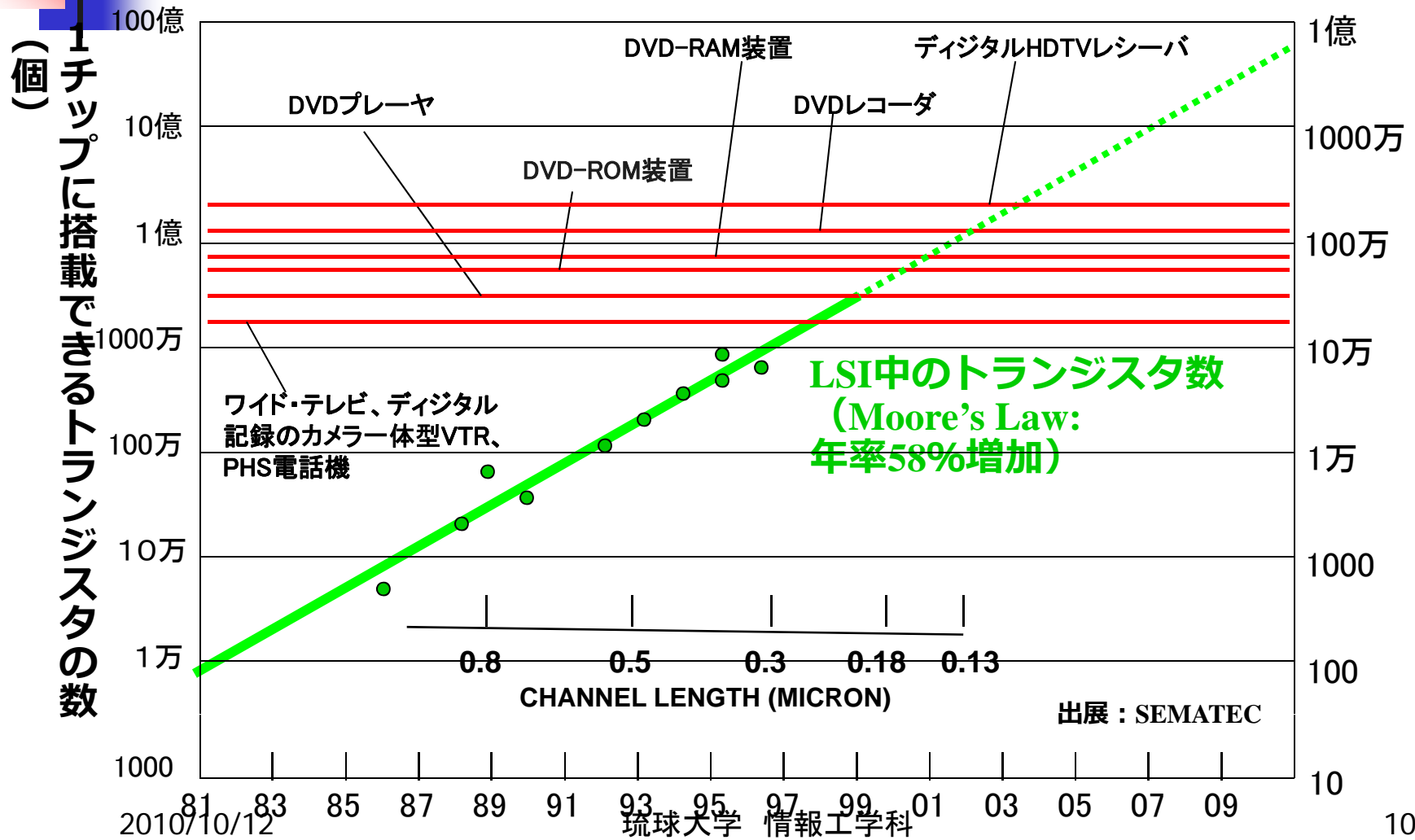


# VLSIの構造



# Mooreの法則に乗る集積素子数の推移

このような大規模な回路を言語を用いて設計する





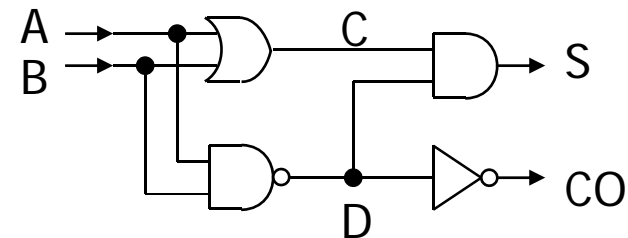
# VHDLの歴史

---

- 1970年代に米国国防省にてVery High Speed ICプロジェクト発足
- 1981年にVHSICプロジェクトの一環として機能記述言語VHDLが提案
- 1986年にIEEEにてVHDLバージョン7.2を基本言語として採択し、標準化スタート
- 1987年にLanguage Reference Manualをリリースして、IEEE-1076Bとして正式に標準化

# VHDLの簡単な例題

```
library IEEE;
use IEEE.std_logic_1164.all;
entity HALF_ADDER is
    port ( A, B : in std_logic;
          S, CO : out std_logic);
end HALF_ADDER;
architecture DATAFLOW of HALF_ADDER is
    signal C, D : std_logic;
begin
    C <= A or B;
    D <= A nand B;
    CO <= not D;
    S <= C and D;
end DATAFLOW;
```

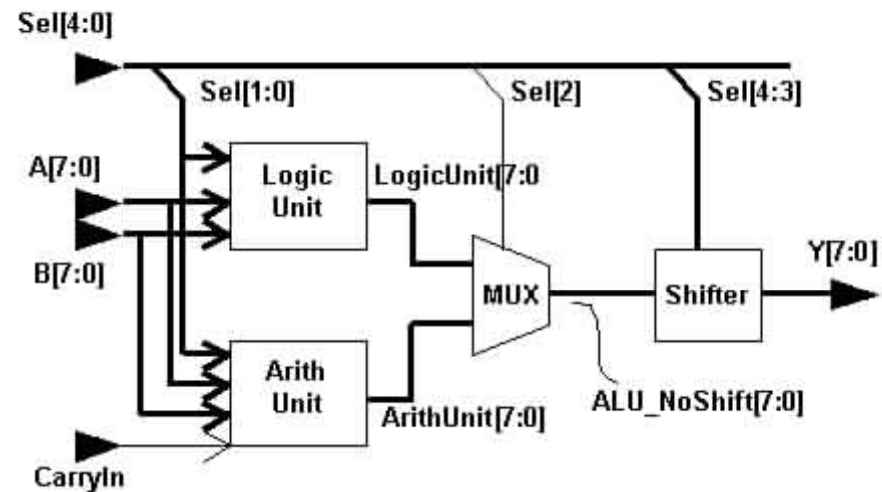


# VHDLを簡単に見てゆくと

<pre>library IEEE; use IEEE.std_logic_1164.all;</pre>	ライブラリ宣言、通常STD_LOGICを使用するので、必ず書く
<pre>entity HALF_ADDER is</pre>	エンティティ宣言
<pre>port ( A, B : IN std_logic;       S, CO : OUT std_logic );</pre>	ポート宣言(入出力信号と信号のデータタイプを定義)
<pre>end HALF_ADDER;</pre>	エンティティ終了。";"忘れるな!
<pre>architecture DATA_FLOW of HALF_ADDER is</pre>	アーキテクチャ宣言
<pre>signal C, D : std_logic;</pre>	内部信号を定義
<pre>begin   C &lt;= A or B ;   D &lt;= A nand B ;   CO &lt;= not D;   S &lt;= C and D; end DATAFLOW;</pre>	同時処理文! コンピュータプログラムとは違う、この場合は4つの同時動作するゲートに対応

# ALUのVHDL記述の紹介

- ALUとは算術(加算や減算)や論理演算を行う回路である
- 今回2つの8ビットの数AとBの入力に対してYを出力する、14種類の演算を行うALUの記述を紹介する



# ALUの動作

S4	S3	S2	S1	S0	Cin	動作	説明	実行ブロック
0	0	0	0	0	0	$Y \leftarrow A$	Aを転送	Arithmetic Unit
0	0	0	0	0	1	$Y \leftarrow A + 1$	Aをインクリメント	Arithmetic Unit
0	0	0	0	1	0	$Y \leftarrow A + B$	加算	Arithmetic Unit
0	0	0	0	1	1	$Y \leftarrow A + B + 1$	キャリー付加算	Arithmetic Unit
0	0	0	1	0	0	$Y \leftarrow A + \bar{B}$	AとBの1の補数をたす	Arithmetic Unit
0	0	0	1	0	1	$Y \leftarrow A + \bar{B} + 1$	減算	Arithmetic Unit
0	0	0	1	1	0	$Y \leftarrow A - 1$	デクリメント	Arithmetic Unit
0	0	0	1	1	1	$Y \leftarrow A$	Aを転送	Arithmetic Unit
0	0	1	0	0	0	$Y \leftarrow A \text{ and } B$	論理積	Logic Unit
0	0	1	0	1	0	$Y \leftarrow A \text{ or } B$	論理和	Logic Unit
0	0	1	1	0	0	$Y \leftarrow A \text{ xor } B$	排他的論理和	Logic Unit
0	0	1	1	1	0	$Y \leftarrow \bar{A}$	1の補数	Logic Unit
0	0	0	0	0	0	$Y \leftarrow A$	Aを転送	Shifter Unit
0	1	0	0	0	0	$Y \leftarrow \text{shl } A$	Aを左シフト	Shifter Unit
1	0	0	0	0	0	$Y \leftarrow \text{shr } A$	Aを右シフト	Shifter Unit
1	1	0	0	0	0	$Y \leftarrow 0$	0を転送	Shifter Unit



# alu.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
```

```
entity ALU is
  port(Sel : in unsigned(4 downto 0);
        CarryIn : in std_logic;
        A, B : in unsigned(7 downto 0);
        Y : out unsigned(7 downto 0));
end entity ALU;
```

```
architecture COND_DATA_FLOW of ALU is
begin
```

```
  ALU_AND_SHIFT:
  process (Sel, A, B, CarryIn)
    variable Sel0_1_CarryIn : unsigned(2 downto 0);
    variable LogicUnit, ArithUnit,
              ALU_NoShift : unsigned(7 downto 0);
```

```
begin
```

```
  -----
  -- Logic Unit
  -----
```

```
  LOGIC_UNIT: case Sel(1 downto 0) is
    when "00" => LogicUnit := A and B;
    when "01" => LogicUnit := A or B;
    when "10" => LogicUnit := A xor B;
    when "11" => LogicUnit := not A;
    when others => LogicUnit := (others => 'X');
  end case LOGIC_UNIT;
```

```
  -----
  -- Arithmetic Unit
  -----
```

```
  Sel0_1_CarryIN := Sel(1 downto 0) & CarryIN;
  ARITH_UNIT: case Sel0_1_CarryIn is
    when "000" => ArithUnit := A;
    when "001" => ArithUnit := A+1;
    when "010" => ArithUnit := A+B;
    when "011" => ArithUnit := A+B+1;
    when "100" => ArithUnit := A + not B;
    when "101" => ArithUnit := A-B;
    when "110" => ArithUnit := A-1;
    when "111" => ArithUnit := A;
    when others => ArithUnit := (others => 'X');
  end case ARITH_UNIT;
```

```
  -----
  -- Multiplex
  -----
```

```
  LA_MUX: if (Sel(2) = '1') then
    ALU_NoShift := LogicUnit;
  else
    ALU_NoShift := ArithUnit;
  end if LA_MUX;
```

```
  -----
  -- Shift operation
  -----
```

```
  SHIFT: case Sel(4 downto 3) is
    when "00" => Y <= ALU_NoShift;
    when "01" => Y <= Shift_left(ALU_NoShift, 1);
    when "10" => Y <= Shift_right(ALU_NoShift, 1);
    when "11" => Y <= (others => '0');
    when others => Y <= (others => 'X');
  end case SHIFT;
```

```
end process ALU_AND_SHIFT;
end architecture COND_DATA_FLOW;
```