

システム設計演習(前期分) 第2回

秋田純一

<http://j.mp/akita-class>

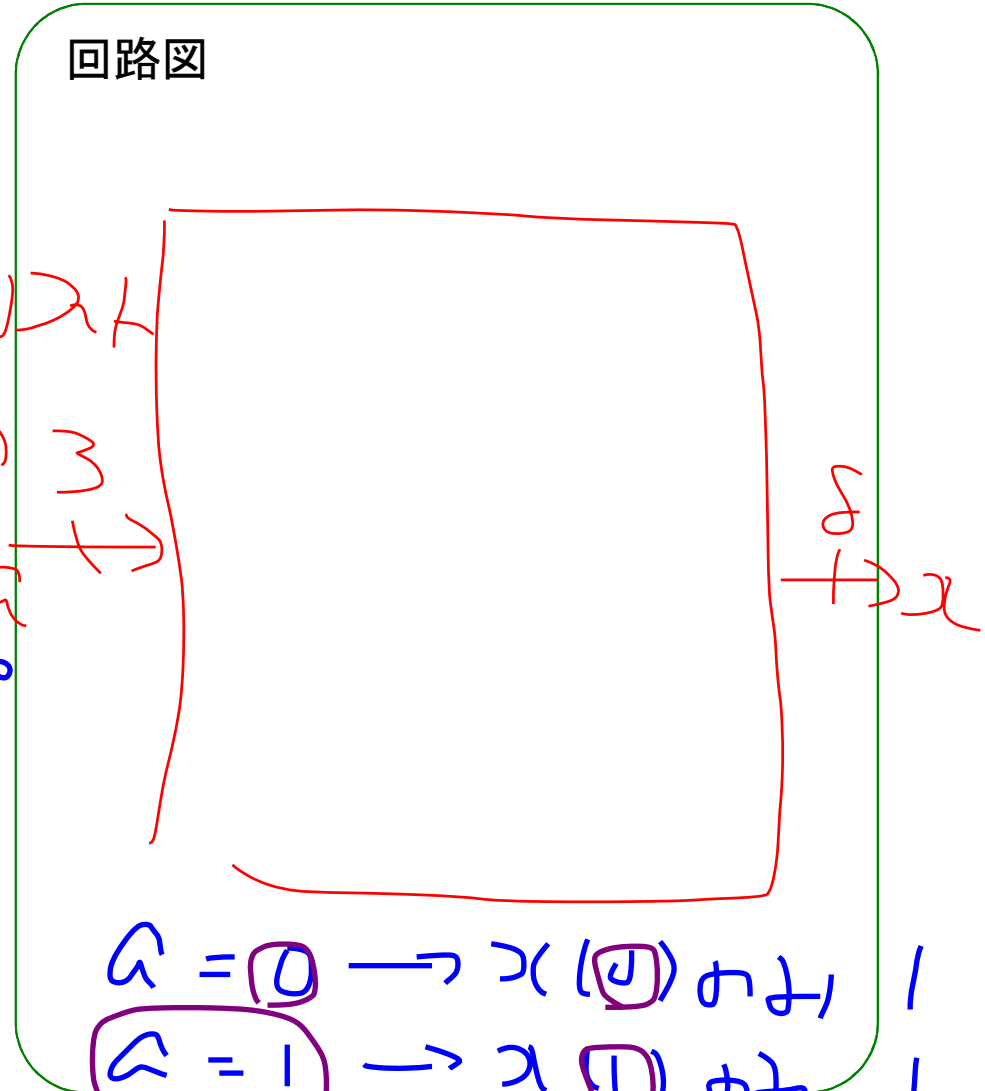
akita@ifdl.jp (@akita11)

HDL記述の例: デコーダ (p.62)

```
entity dec is
  port (
    a: in std_logic_vector(2 downto 0);
    x: out std_logic_vector(7 downto 0)
  );
end dec;
```

```
architecture arch of dec is
begin
  process (a) begin
    case a is
      when "000" => x <= "00000001";
      when "001" => x <= "00000010";
      when "010" => x <= "00000100";
      when "011" => x <= "00001000";
      when "100" => x <= "00010000";
      when "101" => x <= "00100000";
      when "110" => x <= "01000000";
      when "111" => x <= "10000000";
      when others => x <= "XXXXXXXX";
    end case;
  end process;
end arch;
```

回路図



$a(0) = 1$
 $a(1) = 0$
 $a(2) \neq 0$

$a(0)$
 $a(1)$
 $a(2)$

$a = 0 \rightarrow x(0)$ のみ 1
 $a = 1 \rightarrow x(1)$ のみ 1
 $a = 2 \rightarrow x(2)$ のみ 1

デコーダのHDL記述のポイント

- ✓ この例では入力=3ビット・出力=8ビット ($2^3=8$)
- ✓ process文で、入力aの変化に応じて出力xを決めている
 - ✓ process文のカッコ内の変数・信号(センシティブティ・リスト)が「変化した」時に、process文の中身が実行される
- ✓ case文で場合分け(C言語のswitch文と同様) = 真理値表とそっくり
- ✓ (重要)ビット数を変える必要があっても、それほど大げさにならない:HDLを使うメリット
 - ✓ 回路図だとゼロから設計しなおし

入力が変わると出力も変わる

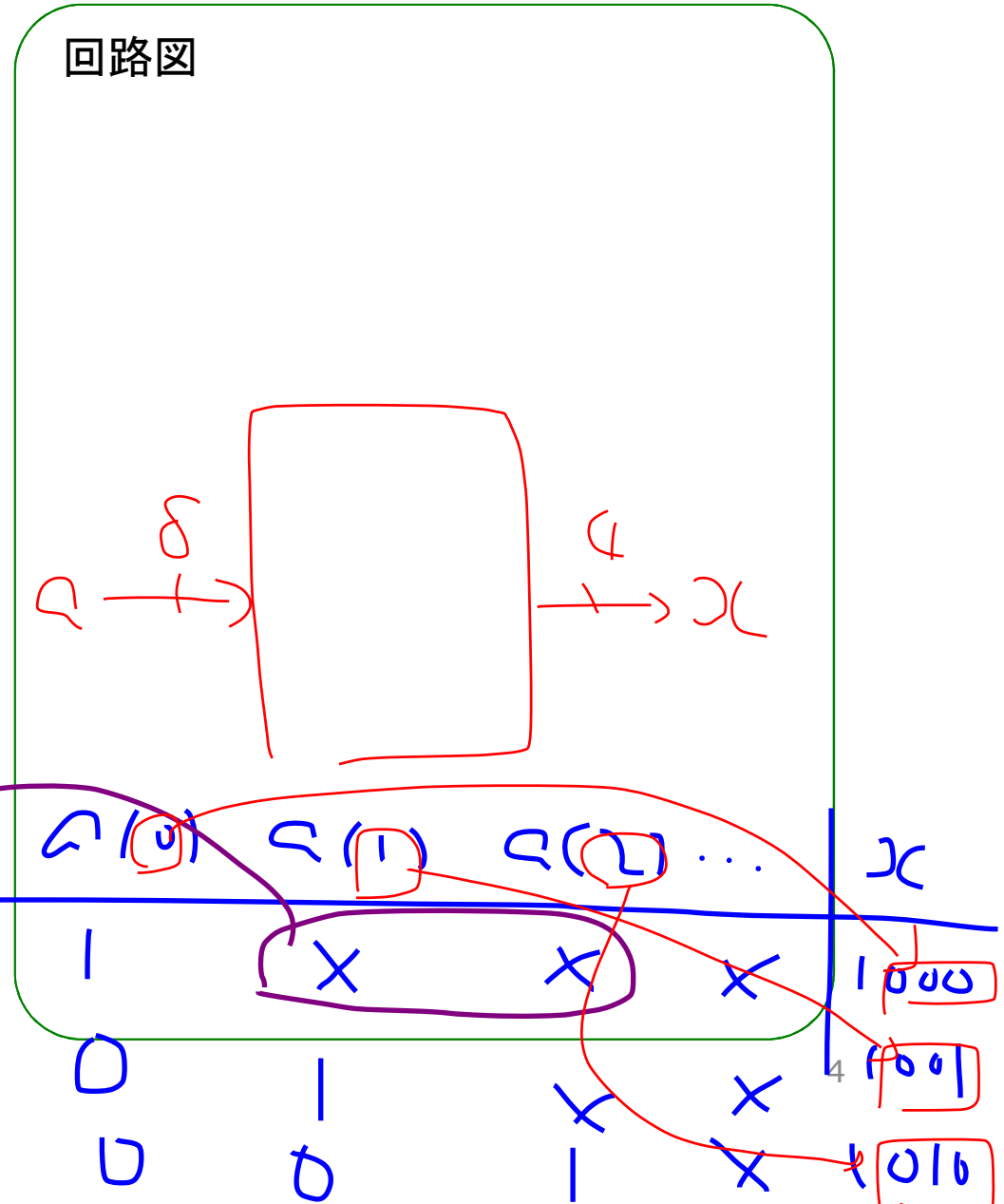
a : 3 a : 4
x : 8 x : 16

HDL記述の例: エンコーダ (p.64)

```
entity enc is
  port (
    a: in std_logic_vector(7 downto 0);
    x: out std_logic_vector(3 downto 0)
  );
end enc;
```

```
architecture arch of enc is
begin
  process (a) begin
    if (a(0) = '1') then x <= "1000";
    elsif (a(1) = '1') then x <= "1001";
    elsif (a(2) = '1') then x <= "1010";
    elsif (a(3) = '1') then x <= "1011";
    elsif (a(4) = '1') then x <= "1100";
    elsif (a(5) = '1') then x <= "1101";
    elsif (a(6) = '1') then x <= "1110";
    elsif (a(7) = '1') then x <= "1111";
    else
      x <= "0000";
    end if;
  end process;
end arch;
```

1番目から
 7番目まで
 (1から7)



エンコーダのHDL記述のポイント

☑この例では、入力=8ビット・出力=3ビット

☑ $2^3=8$

☑if文で、 $a(0) \sim a(7)$ のどれが1になっているかを順に調べ、該当する x の値を決めている

☑最初に $a(0)$ 、次に $a(1) \dots$ の順に調べている
= $a(0) \rightarrow a(1) \rightarrow \dots \rightarrow a(7)$ の順に優先づけ

☑※ただし、上から順に「実行」されるわけではない(プログラムの動作とは異なる)

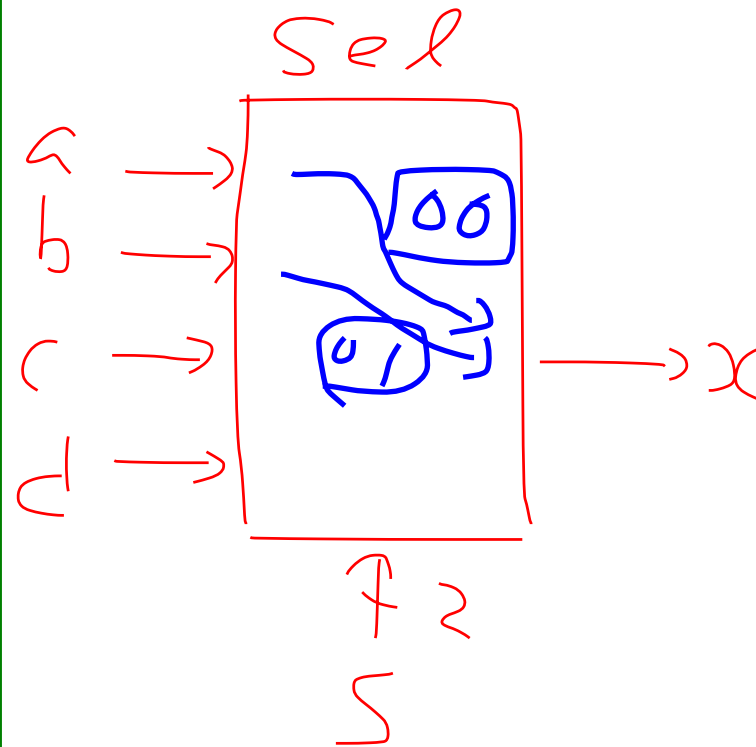
☑あくまでもHDLで書いているのは論理回路

HDL記述の例: セレクタ(p.67)

```
entity sel is
  port (
    a, b, c, d: in std_logic;
    s: in std_logic_vector(1 downto 0);
    x: out std_logic
  );
end sel;
```

```
architecture arch of sel is
begin
  process (a, b, c, d, s) begin
    case s is
      when "00" => x <= a;
      when "01" => x <= b;
      when "10" => x <= c;
      when "11" => x <= d;
      when others => x <= 'X';
    end case;
  end process;
end arch;
```

回路図



x を求める回路図は?

⇒ s, a, b, c, d

例) s = 00 のとき
a, b, c, d
→ x = a

セレクタのHDL記述のポイント

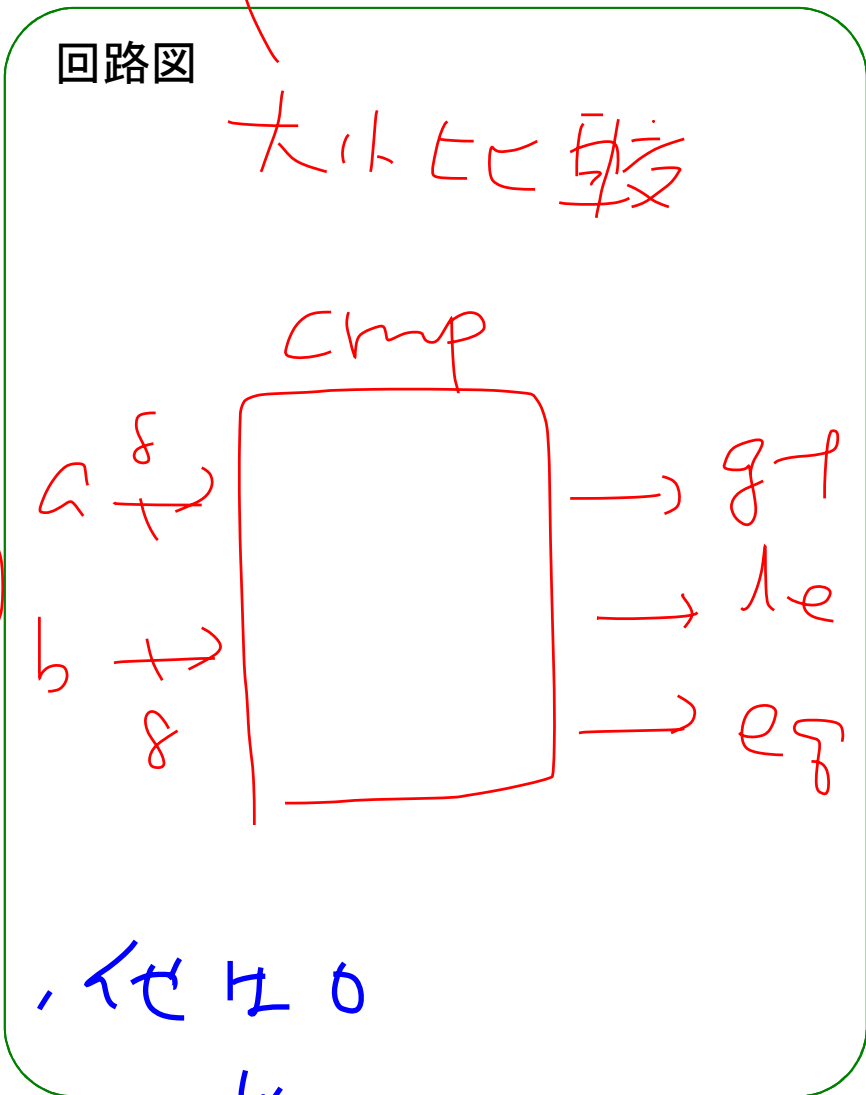
- ✓ 入力sに応じて、入力a,b,c,dのどれかの値が出力xに伝わる
 - ✓ 動作はスイッチのようなイメージ
- ✓ process文・case文で真理値表のように記述
- ✓ センシティブティ・リストに注意！
 - ✓ sだけではダメ(sが変化せずにa~dが変化したとき、xが変化しない回路になってしまう)
 - ✓ 出力xの値に「関係する」変数・信号をすべて書く

HDL記述の例:コンパレータ(p.69)

```
entity cmp is
  port (
    a, b: in std_logic_vector(7 downto 0);
    gt, lt, eq: out std_logic;
  );
end cmp;
```

```
architecture arch of cmp is
begin
  process (a, b) begin
    gt <= '0';
    lt <= '0';
    eq <= '0';
    if (a > b) then gt <= '1';
    elsif (a < b) then lt <= '1';
    else eq <= '1';
    end if;
  end process;
end arch;
```

大小比較の演算子



$$\left\{ \begin{array}{l} a > b \leftrightarrow gt = 1 \\ a = b \leftrightarrow eq = 1 \\ a < b \leftrightarrow lt = 1 \end{array} \right. , \text{ 他は } 0$$

コンパレータのHDL記述のポイント

- ✓ if文で、a,bのどちらが大きいか、によって
gt(a>bの場合)、lt(a<bの場合)、eq(a=bの場合)
のどれかを1にする
 - ✓ ちなみにgt="Greater Than"、"lt"="Less Than"の略
- ✓ まずgtなどに0を代入して、その後、gt=1などを
代入する、というように読めるが、違う
 - ✓ あくまでもHDLで書いているのは論理回路
 - ✓ 上から順番に「実行」されるわけではない
 - ✓ process文が終わった時点での値が最終的な結果

$$\begin{array}{r} \boxed{gt = 0 \quad 0 \vee 1} \\ \hline 1 + \end{array}$$

HDL記述の例: 加算器(p.71)

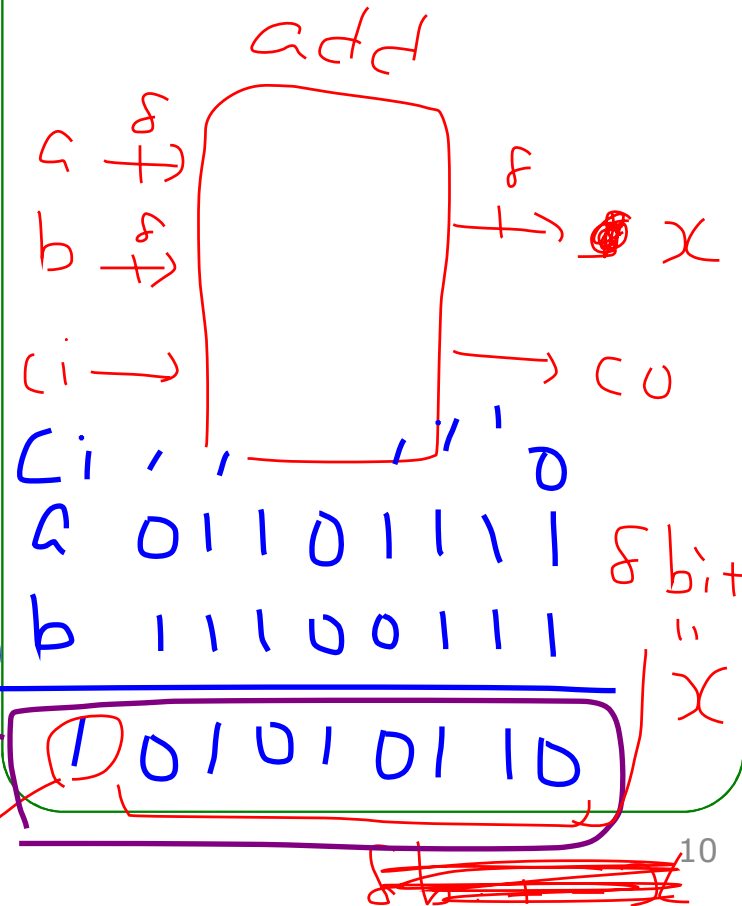
```
entity add is
  port (
    a, b: in std_logic_vector(7 downto 0);
    ci: in std_logic;
    co: out std_logic;
    x: out std_logic_vector(7 downto 0)
  );
end add;
```

```
architecture arch of add is
begin
  process (a, b, ci)
    variable tmp: std_logic_vector(8 downto 0);
  begin
    tmp := ("0" & a) + ("0" & b) + ("00000000" & ci);
    co <= tmp(8);
    x <= tmp(7 downto 0);
  end process;
end arch;
```

加算器

9 bit

回路図



加算器のHDL記述のポイント

- ✓この例では、8ビット加算器を動作記述
 - ✓8ビット+8ビット→9ビット
 - ✓最上位ビットは、次のケタへのケタ上がり

