HI4 ハードウェア実験
VHDLのまとめ

縄田（神崎先生）

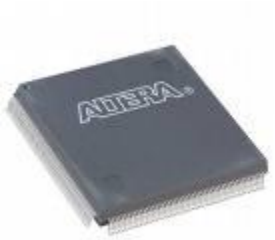
今後の予定

実施日	内容
6/12	第1回実験前半：VHDL座学および情報収集
6/19	第1回実験後半：VHDL演習（実装方法の習得）
6/26	第2回実験前半：組み合わせ回路実験
7/ 3	第2回実験後半：組み合わせ回路実験
7/10	第3回実験前半：順序回路実験
7/17	第3回実験後半：順序回路実験

FPGAとその用途

FPGA (Field Programmable Gate Array)とは‥

- PLD(Programmable Logic Device)の一種
- 製造後に内部の論理回路を自由に変更できる



FPGAの用途

- 液晶テレビ，レコーダなどのAV機器
- 携帯電話の基地局などの無線機器
- ホビー　　他多数

VHDLについて

VHDL :VHSIC Hardware Description Language

- FPGAの内部構造を記述するHDL(ハードウェア記述言語)の一種
- Verilog-HDLと並んでHDLの主流となっている

VHDLコードの構成

```
library ieee;
use ieee.std_logic_1164.all;
```

ライブラリ宣言

必要なライブラリ・パッケージの指定

```
entity test is
  port (
    sw_in : in std_logic;
    led_out : out std_logic);
end test;
```

エンティティ宣言

入出力関係の記述

```
architecture rtl of test is
begin
  led_out <= sw_in;
end rtl;
```

アーキテクチャ宣言

回路構成の記述

エンティティ宣言

- ポートの宣言を並べて入出力の部品を用意する
  - port (ポート名 : モード型　データ型);
- モード型
  - in（入力）, out(出力), inout（双方向）
- データ型（よく使うもの）
  - std_logic　:1ビット
  - std_logic_vector(n downto 0)　: nビット
  - integer :整数型(32ビット)

アーキテクチャ宣言

```
architecture rtl of led_blink is

signal div_counter : std_logic_vector(24 downto 0);
signal div_clk : std_logic;
signal light : std_logic;

begin
  div_clk <= div_counter(22);

  process (clk)
  begin
    if clk'event and clk = '1' then
      div_counter <= div_counter + 1;
    end if;
  end process;

end rtl;
```

信号宣言部

機能宣言部

- 信号宣言部では，signal(内部で用いる信号)などの宣言を並べる
- 機能宣言部では，信号の関係やprocess文を並べる

VHDLコードの構成

```
library ieee;
use ieee.std_logic_1164.all;
```

ライブラリ宣言

必要なライブラリ・パッケージの指定

```
entity test is
  port (
    sw_in : in std_logic;
    led_out : out std_logic);
end test;
```

エンティティ宣言

入出力関係の記述

```
architecture rtl of test is
begin
  led_out <= sw_in;
end rtl;
```

信号代入文

アーキテクチャ宣言

回路構成の記述

リスト1

```
library ieee;
use ieee.std_logic_1164.all;
```

ライブラリ宣言

```
entity NAND_CIRCUIT is
  port (
    A : in std_logic;
    B : in std_logic;
    C : out std_logic);
end NAND_CIRCUIT;
```

エンティティ宣言

```
architecture rtl of NAND_CIRCUIT is
signal ab : std_logic;
begin
  ab <= A and B;
  C <= not ab;
end rtl;
```

アーキテクチャ宣言

リスト2

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
```

ライブラリ宣言

```
entity REI2 is
  port (
    A : in std_logic_vector(7 downto 0);
    B : in std_logic_vector(6 downto 0);
    Z : out std_logic_vector(8 downto 0);
end REI2;
```

エンティティ宣言

```
architecture rtl of REI2 is
begin
  Z <= (‘0’ & A) + (‘0’ & ‘0’ & B);
end rtl;
```

アーキテクチャ宣言

process文

文をシーケンシャル（順番）に実行させたいときに使用する.

センシティビティ・リスト

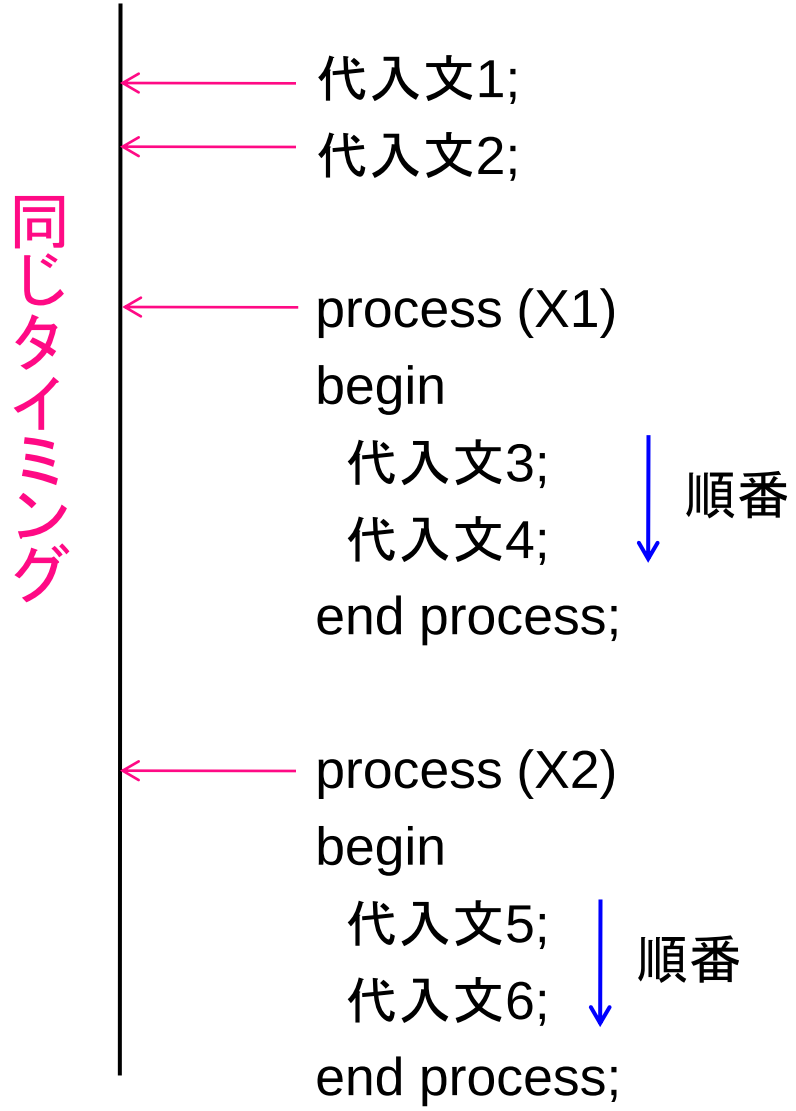
```
process (A, B, C)
begin
  X <= A;
  Y <= B and C;
end process;
```

A,B,Cのどれかに変化があったとき. .

順番に実行

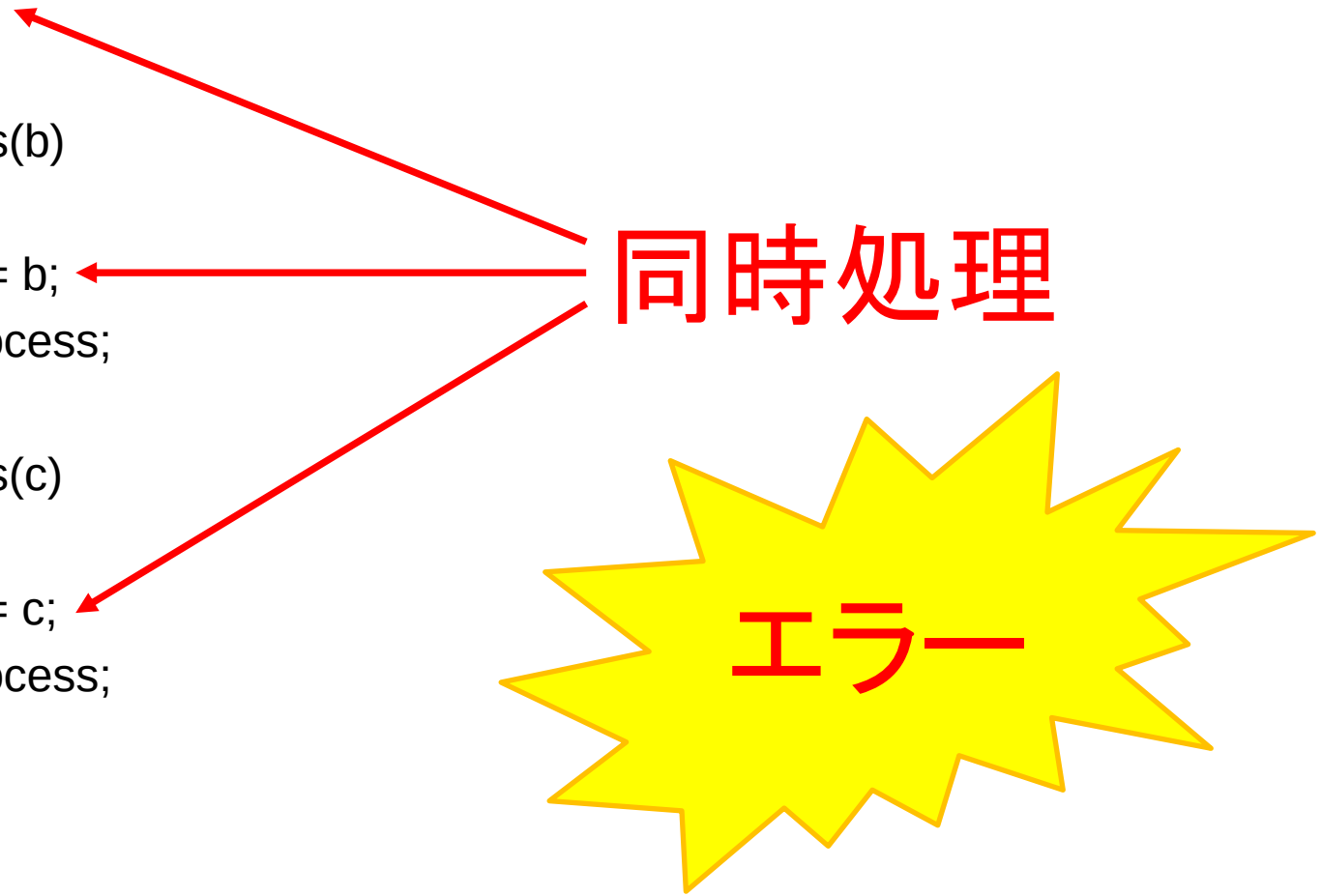
複数のプロセスで，同じ信号への代入はできない.

process文と実行のタイミング



process文と実行のタイミング

```
------
d <= a;
------
process(b)
begin
    d <= b;
end process;
------
process(c)
begin
    d <= c;
end process;
```



リスト3

```
library ieee;
use ieee.std_logic_1164.all;
```

ライブラリ宣言

```
entity REI3 is
  port (
    D : in std_logic_vector(1 downto 0);
    Y : in std_logic_vector(3 downto 0);
end REI3;
```

エンティティ宣言

```
architecture rtl of REI3 is
begin
    process ( D )
    begin
        -- case文による出力の場合分け
        case D is
            when "00" => Y <= "0001";
            when "01" => Y <= "0010";
            when "10" => Y <= "0100";
            when "11" => Y <= "1000";
            when others => Y <= "XXXX";
            end case;
    end process;
end rtl;
```

アーキテクチャ宣言

process文内の代表的な制御文

- if文
- case文
- loop文
  - for-loop文
  - while-loop文

signalとvariable

- signal （信号宣言）
  - architecture, entity内等で宣言.
  - <=で代入する.　例: X <= A + B;
  - 同時処理の対象となる
- variable (変数宣言)
  - process内等で，一時的に用いる変数.
  - :=で代入する.　例: X := A + B;
  - 実行時，即時に反映される

エッジ検出

processの中では，特定の信号の立ち上がり（立ち下がり）のタイミングで処理を行いたい場合が多くある.

- 立ち上がりの検出
  - if(clk'event and clk=‘1’) then..
  - if(rising_edge(clk)) then ..
- 立ち下がりの検出
  - if(clk'event and clk=‘0’) then..
  - if(falling_edge(clk)) then ..

1つのprocess内では，複数回のエッジ検出はできないことに注意する.

エッジ検出

```
architecture rtl of led_blink is

signal div_counter : std_logic_vector(24 downto 0);
signal div_clk : std_logic;
signal light : std_logic;

begin
  div_clk <= div_counter(22);

  process (clk)
  begin
    if clk'event and clk = '1' then
      div_counter <= div_counter + 1;
    end if;
  end process;

end rtl;
```

よくあるエラー

- 複数のプロセスで，同じ信号への代入が行われている．

  プロセスを1つにまとめるなどの修正が必要

- 1つのプロセス内で，エッジ検出が複数回行われている．

  コード設計の修正が必要

実験に関する注意

- Quartusのエラーメッセージや DE2のマニュアルは英語で書かれている．英語に自信のない人は辞書を持参すること．
- 基本的に居残り実験は認めない．予習・復習を十分行い実験中は真剣に取り組むこと．
- 公欠などで実験を休む場合は，原則事前に連絡し，追実験を願い出ること．
- Quartusの環境をインストールしたい人はDVDを貸し出します．ただしボードは貸し出し不可．

来週の実験に関する注意

- レポートには以下の情報を加えること.
  - ピン設定情報
  - 初めて使用した際の感想,今後の注意点など
- 新しいプログラムを作成する際は,プロジェクトから新規に作成すること.