

C6 – Délais, Delta Délais et variables

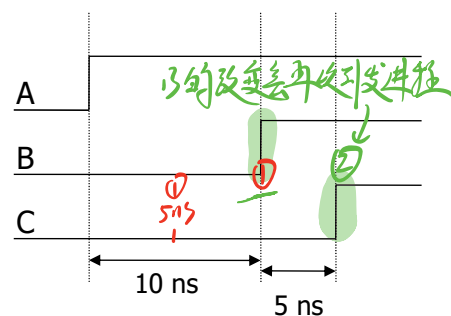
Yann DOUZE

1

Les délais

chronogramme de A ,B et C ?

```
process (A,B)
begin
  B <= A after 10 ns;
  C <= B after 5 ns;
end process;
```



- Les délais sont utilisés pour :
 - Générer des stimuli pendant la simulation (stimuli generated during simulation) $\Rightarrow th$
 - Modéliser les retards dû à la technologie (modeling delays due to technology)
- Les délais sont ignorés à la synthèse. (delays are ignored at synthesis.)

2

产生脉冲

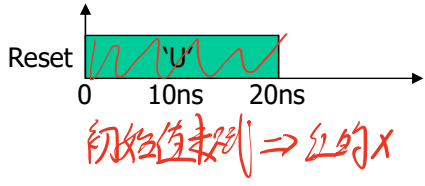
Génération d'une impulsion (1)

- 1. 没有敏感信号列表（只用于仿真），只执行一遍；
- 2. process结束后完成一次赋值；
- 3. 多重赋值，只执行最后一次赋值语句。

```
process
begin
  reset <= '0';
  reset <= '1' after 10 ns;
  reset <= '0' after 20 ns;
  wait;
end process;
```

每个信号 有提回信号
脉冲后接有延迟后并执行的
指令. 即 wait for xxns.

Mauvaise syntaxe !



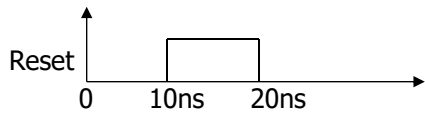
wait 绝对时间 -> 遇到此执行. 相当于长语句了.
after 相对时间 -> 相对开始的时间

Génération d'une impulsion (2)

```
Reset <= '0', '1' after 10 ns, '0' after 20 ns;
```

```
process
begin
  reset <= '0';
  wait for 10 ns;
  reset <= '1';
  wait for 10 ns;
  reset <= '0';
  wait;
end process;
```

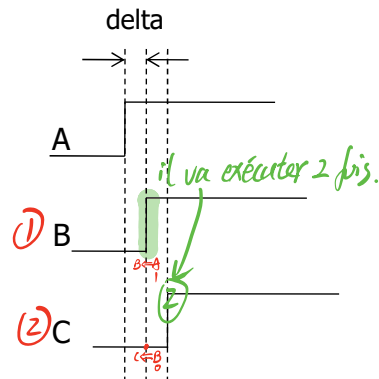
Bonne syntaxe !



延时 (temps de propagation dans les portes logiques)

Delta Délais

```
process (A, B)
begin
  B <= A; ①
  C <= B; ②
end process;
```



- En jargon VHDL, un délai delta = délai infinitésimal non nul *非零无穷小延迟*
- Un signal prend sa nouvelle valeur après un délai delta.
- Une variable prend sa nouvelle valeur immédiatement.

信号在delta延时后更新值, (即结束进程后的delta)
变量会立即更新值. *信号传输 (认为一般信号传输为即时传输)*
变量不传输 (变量在进程中的赋值不点时间)

5

变量 Les Variables

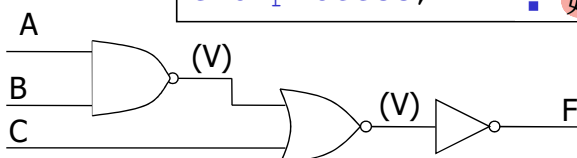
- Les variables ne peuvent être déclarées et n'existent que dans un process. *变量只能在进程中使用.*
- L'affectation d'une variable est immédiate : la valeur affectée à V à la première ligne peut directement être réutilisée à la deuxième. *变量的赋值是立即的, 第一行赋值给V的值, 可在第二行直接用.*

```
process (A, B, C)
-- zone de déclaration d'une variable
variable V: STD_LOGIC;
begin
  V := A nand B;
  V := V nor C;
  F <= not V;
end process;
```

对变量赋值 (V := A nand B;)

对信号赋值 (F <= not V;)

- 变量只能声明, 并且只存在于进程中。
- 变量的赋值是立即的: 在第一行赋值给v的值, 可以在第二行直接重用。
- 如果V表示为信号, 第一行赋值不起作用。执行第二行赋值。



Redessinez le schéma si on considère V comme un signal ?

6

```

signal V: STD_LOGIC;
process (A, B, C)
-- zone de déclaration d'une variable
begin
V <= A nand B;
V <= V nor C;
F <= not V;
end process;

```



Redessinez le schéma si on considère V comme un signal ?

Exemple : parité impaire

```
Entity parite is
PORT ( a : IN std_logic_vector(0 TO 3) ;
      s : OUT std_logic );
END entity;
architecture behaviour of parite is
begin
    process(a)
        variable parite : std_logic ;
    begin
        parite := '1' ;
        FOR i in 0 to 3 LOOP
            if a(i) = '1' then
                parite := not parite;
            end if;
        END LOOP;
        s <= parite;
    end process;
END architecture;
```

检验偶数。初始P为1;
来一个1, 设为0;
来第二个1, 设为1;

偶数 奇数
偶数为偶数 => 真
奇数 => 假

如用 when(, 0) 会有很多寄存器 (同步电路)
(完整条件语句)

变量只能在过程内使用 => 对值值信号。

7

Exercice

```
process (A, S)
    variable V: STD_LOGIC;
begin
    V := A;
    S <= V;
    V := S;
    T <= V;
end process;
```

S=1 <= 延迟

T=0 <= 延迟

第一遍 第二遍

过程内的每一句都是顺序执行!!!
(虽然变量的赋值可能作不点时间, 但也有先后顺序)

在过程内:

对同一变量的多次赋值, 不等于信号的多次赋值
赋值均有效

相当于同一信号有多个赋值, 但最后的赋值是接收数据的最后数据
还以到过程结束 (还没到且执行) 时赋值了。
最后赋值有效, 除延迟并执行指令 (wait for) 外。

变量立即赋值。

当在同一进程中, 同一信号赋值目标有多个赋值源时, 信号赋值目标获得的是最后一个赋值源的赋值, 其前面相同的赋值目标不作任何变化。

本应Process结束赋值。最终V=S, S再次被赋给自己。

立即赋值, 使得之前V:=A无用。

process结束时, T被赋为V。

- Supposer que S vaut '0', et A change de l'état '0' à l'état '1'.

8

Questions

1. Qu'elle est la valeur de S à la fin du process, avant le delta délai ? 0
2. Qu'elle est la valeur de V à la fin du process, avant le delta délai ? 0
3. Après l'exécution du process et après le delta délai, S et T prennent leurs nouvelles valeurs, que valent S et T ?
4. Après la première exécution du process et après le delta délai, que se passe-t-il ?

第一遍执行

在第一次执行之后(的那次执行),
也是第2次
执行了.hhh