# Beginner VHDL Training Class VHDL初学者入门

Danny Mok

Altera HK FAE

(dmok@altera.com)

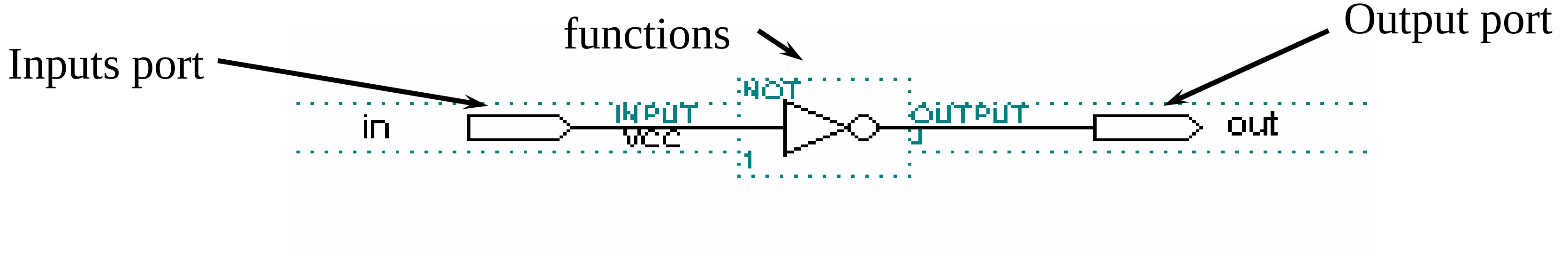
Lucky Sun 译

# What is VHDL(什么是VHDL)

- ***V***ery high speed integrated ***H***ardware ***D***escription ***L***anguage (VHDL) 超高速集成硬件描述语言
  - is an industry standard hardware description language
  - description the hardware in language instead of graphic
    - easy to modify 是一种工业标准硬件描述语言; 用语言
    - easy to maintain 代替图形描述硬件;易修改;易维护
  - very good for 非常适合于
    - complex combinational logic 复杂组合逻辑
      - BCD to 7 Segment converter BCD-七段码转换器
      - address decoding 地址解码
    - state machine 确定机器状态
    - more than you want…….更多你想做的……

## What VHDL Standard means??(VHDL 标准的含义)

- The VHDL is used to describe VHDL用于描述
  - Inputs port -输入口
  - Outputs port -输出口
  - behavior and functions of the circuits 电路的状态和功能



- The language is defined by two successive standards
  - IEEE Std 1076-1987 (called VHDL 1987)
  - IEEE Std 1076-1993 (called VHDL 1993)

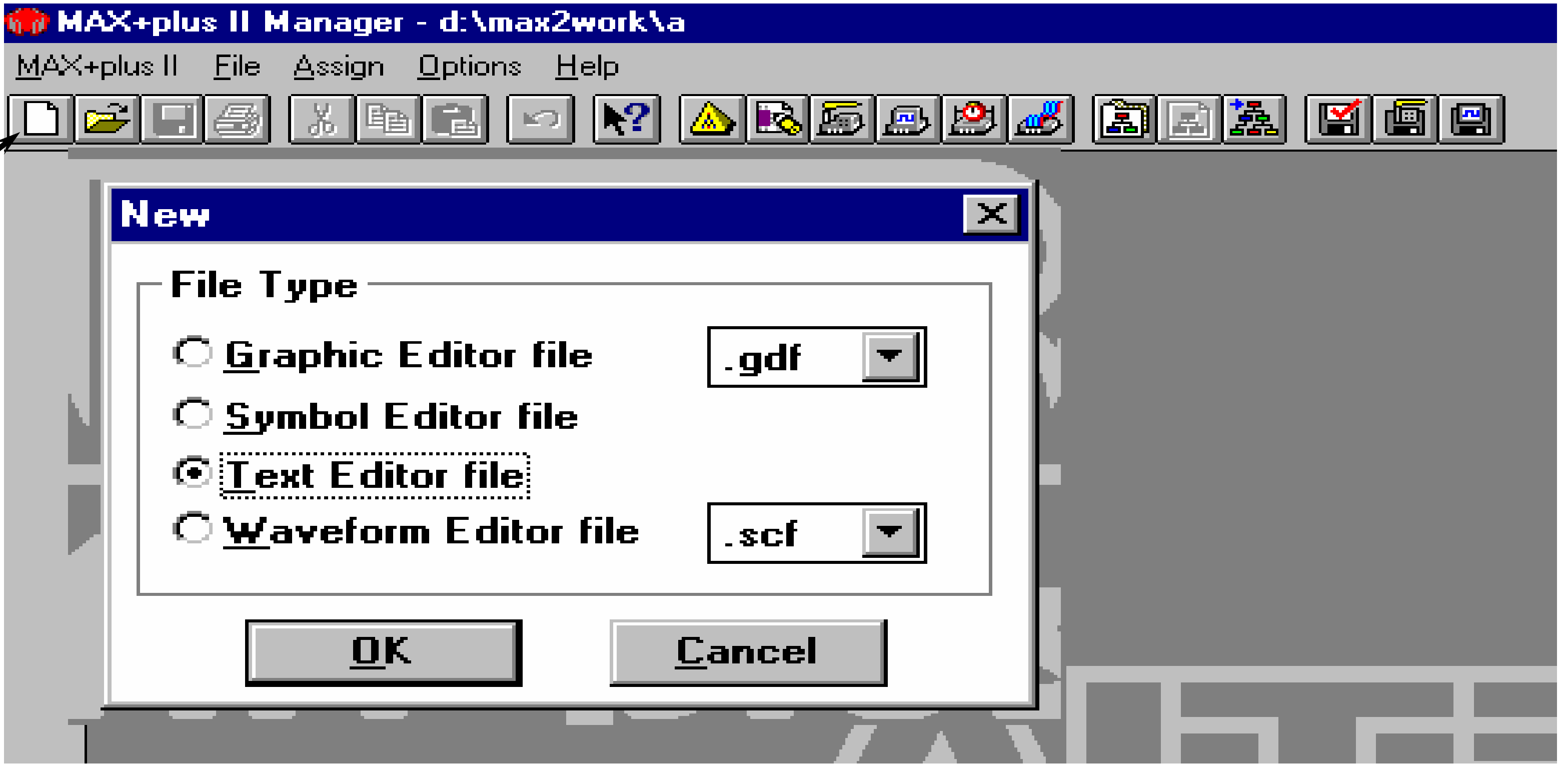
该语言言由上述两种标准连续定义

## Altera VHDL

- Altera Max+Plus II support both VHDL 1987 and 1993   (ALTERA MAX+PLUSII 支持上两种标准)
- Max+Plus II only support SUBSET of the two IEEE standard(MAX+PLUSII 仅支持两个IEEE标准的子集)
- Detail of the support can be referred to Altera Max+Plus II VHDL handbook on page 89 Section 3 (支持的细节请参考Max+Plus II VHDL 手册89页3段)

# How to use the VHDL 如何使用VHDL

- use any text editor to create the file 用任意文本编辑器建立文件
  - Altera Software Max+Plus II provides text editor ALTERA软件也有文本编辑器

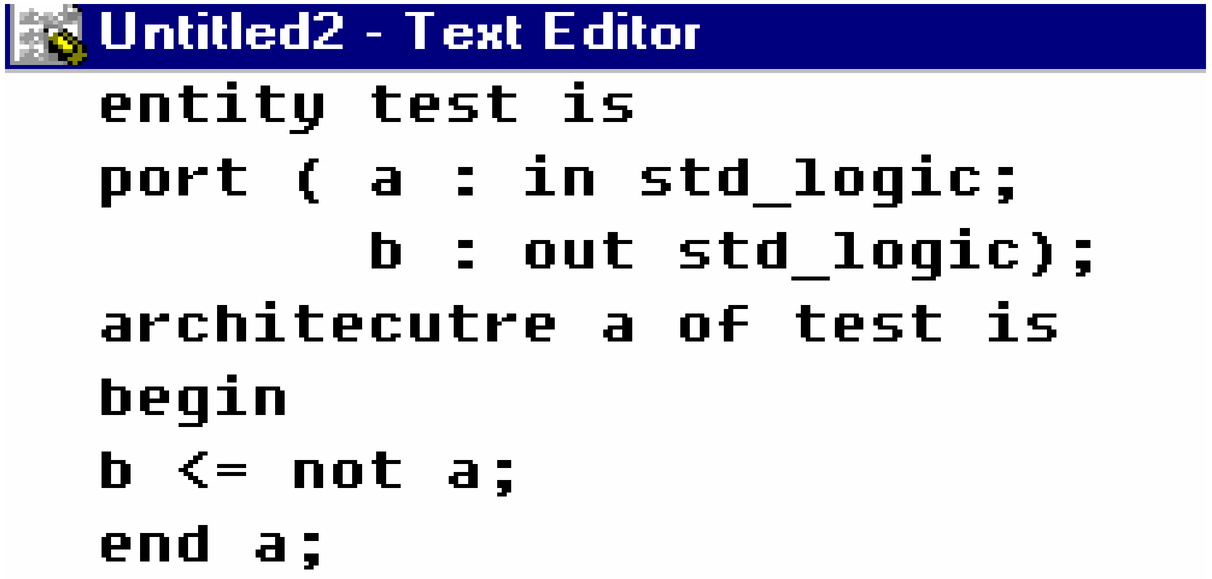


( ) Graphic Editor file
( ) Symbol Editor file
(●) Text Editor file
( ) Waveform Editor file

click at this icon 点击此图标

## 建立自己的VHDL文件

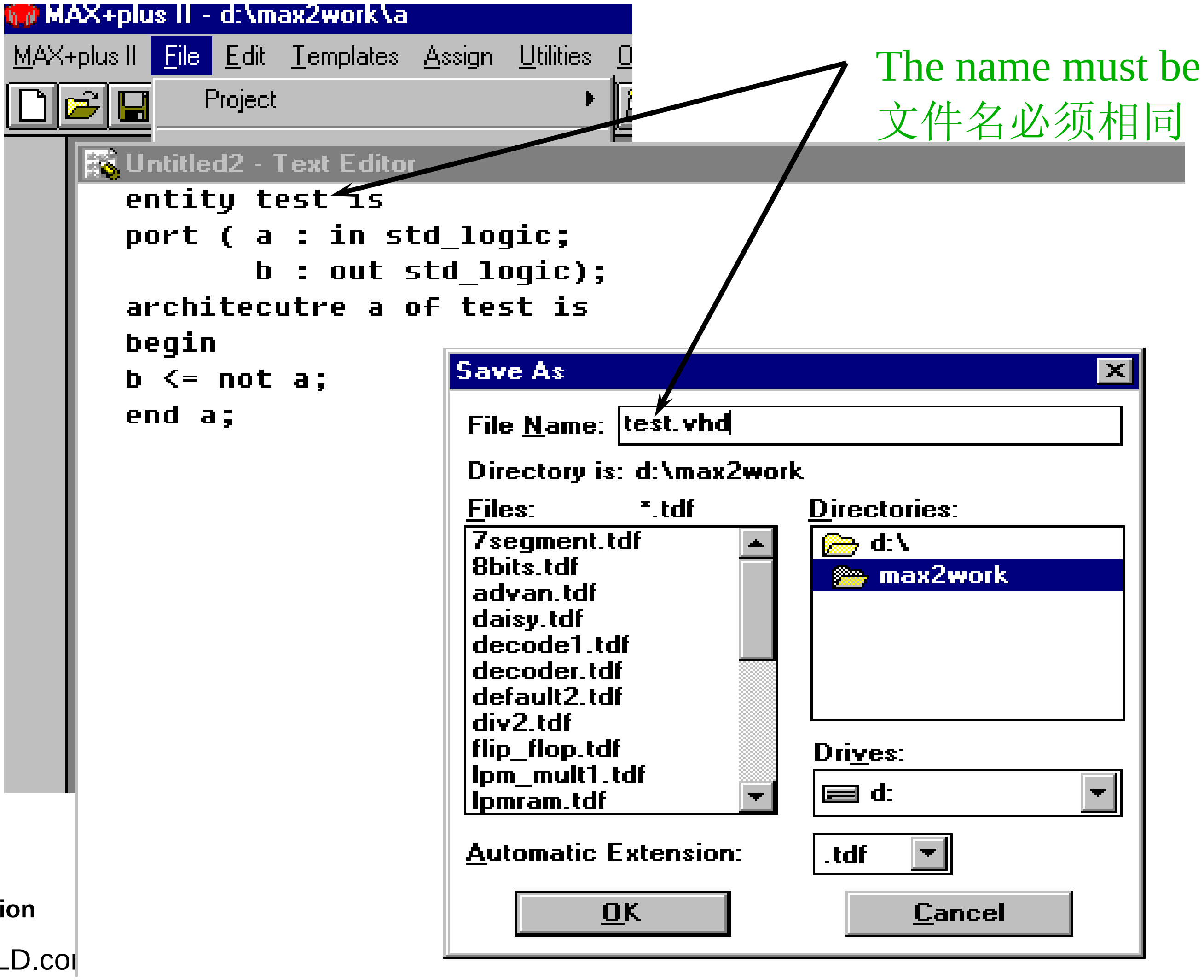
- create your VHDL file



```
entity test is
port ( a : in std_logic;
       b : out std_logic);
architecutre a of test is
begin
b <= not a;
end a;
```

# 把你的文件存为name.VHD

- save your VHDL file as name.VHD



MAX+plus II - d:\max2work\a
MAX+plus II File Edit Templates Assign Utilities O
Project
Untitled2 - Text Editor
Save As

```
entity test is
port ( a : in std_logic;
       b : out std_logic);
architecutre a of test is
begin
b <= not a;
end a;
```

File Name: test.vhd

Directory is: d:\max2work

| Files: *.tdf | Directories: |
|---|---|
| 7segment.tdf | d:\ |
| 8bits.tdf | max2work |
| advan.tdf | |
| daisy.tdf | |
| decode1.tdf | |
| decoder.tdf | |
| default2.tdf | |
| div2.tdf | |
| flip_flop.tdf | |
| lpm_mult1.tdf | |
| lpmram.tdf | |

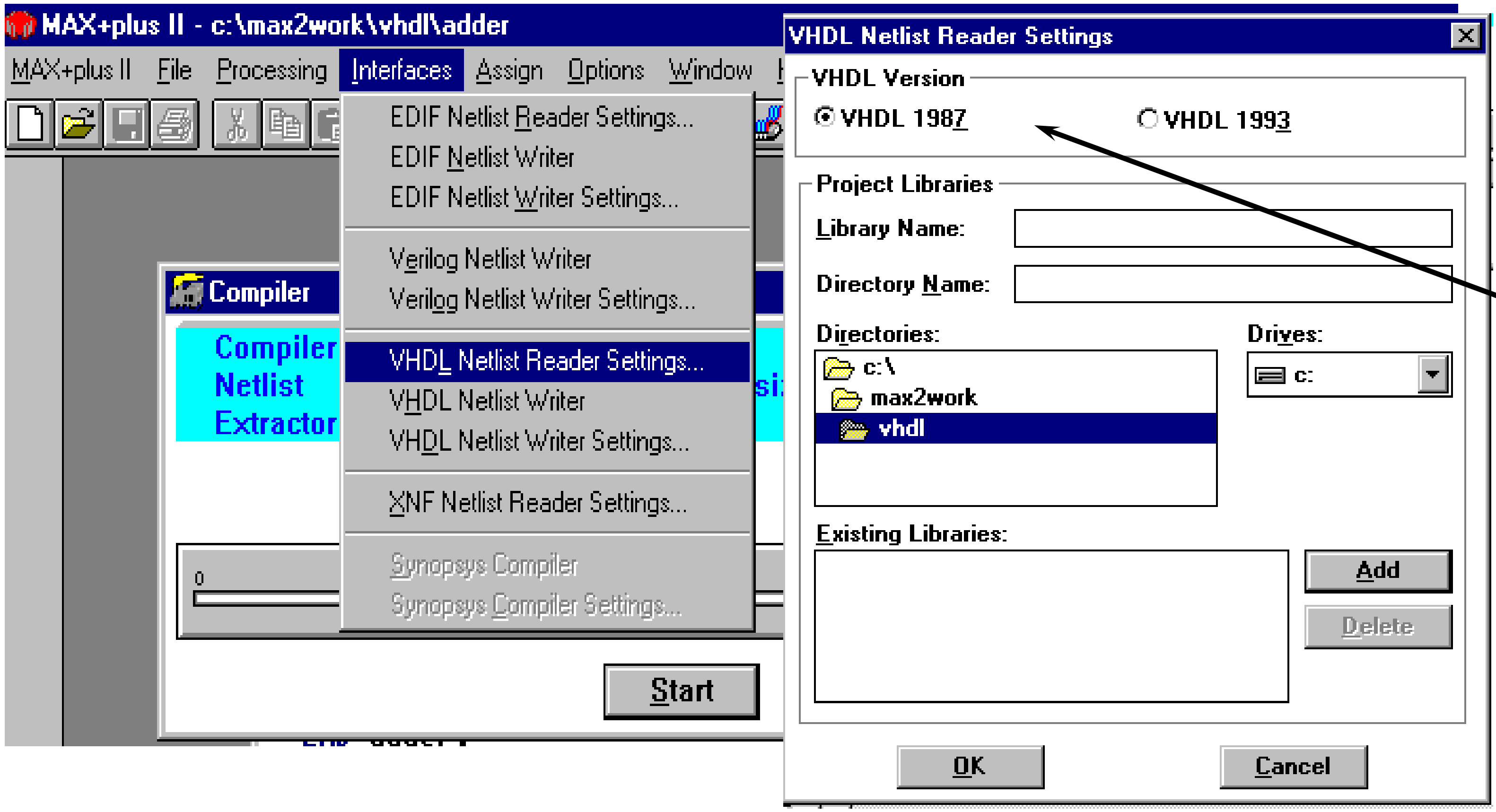
Drives: d:

Automatic Extension: .tdf

The name must be the same
文件名必须相同

## 选择VHDL标准版编码

- Select the Standard Version of VHDL coding
  - 1987 or 1993



MAX+plus II - c:\max2work\vhdl\adder

MAX+plus II
File
Processing
Interfaces
Assign
Options
Window
EDIF Netlist Reader Settings...
EDIF Netlist Writer
EDIF Netlist Writer Settings...
Verilog Netlist Writer
Verilog Netlist Writer Settings...
VHDL Netlist Reader Settings...
VHDL Netlist Writer
VHDL Netlist Writer Settings...
XNF Netlist Reader Settings...
Synopsys Compiler
Synopsys Compiler Settings...
Compiler
Compiler Netlist Extractor
Start

VHDL Netlist Reader Settings

VHDL Version: [✓ VHDL 1987] VHDL 1993

Project Libraries

Library Name:

Directory Name:

Directories: c:\ max2work vhdl

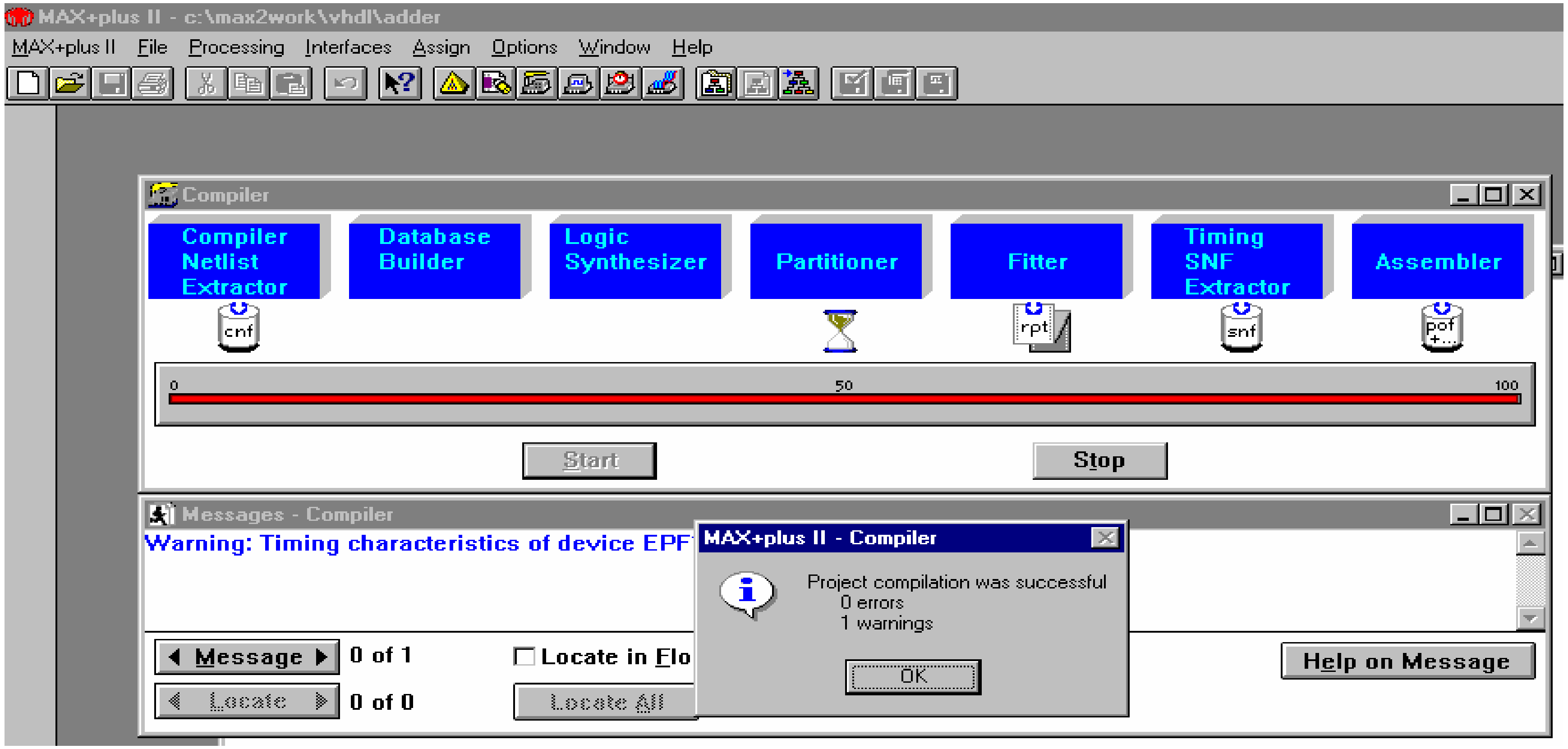
Drives: c:

Existing Libraries:

Select which version you Want
选择你用的版本

# 编译你的VHDL文件

- Compile your VHDL file



Warning: Timing characteristics of device EPF

0 of 1

0 of 0

Project compilation was successful
0 errors
1 warnings

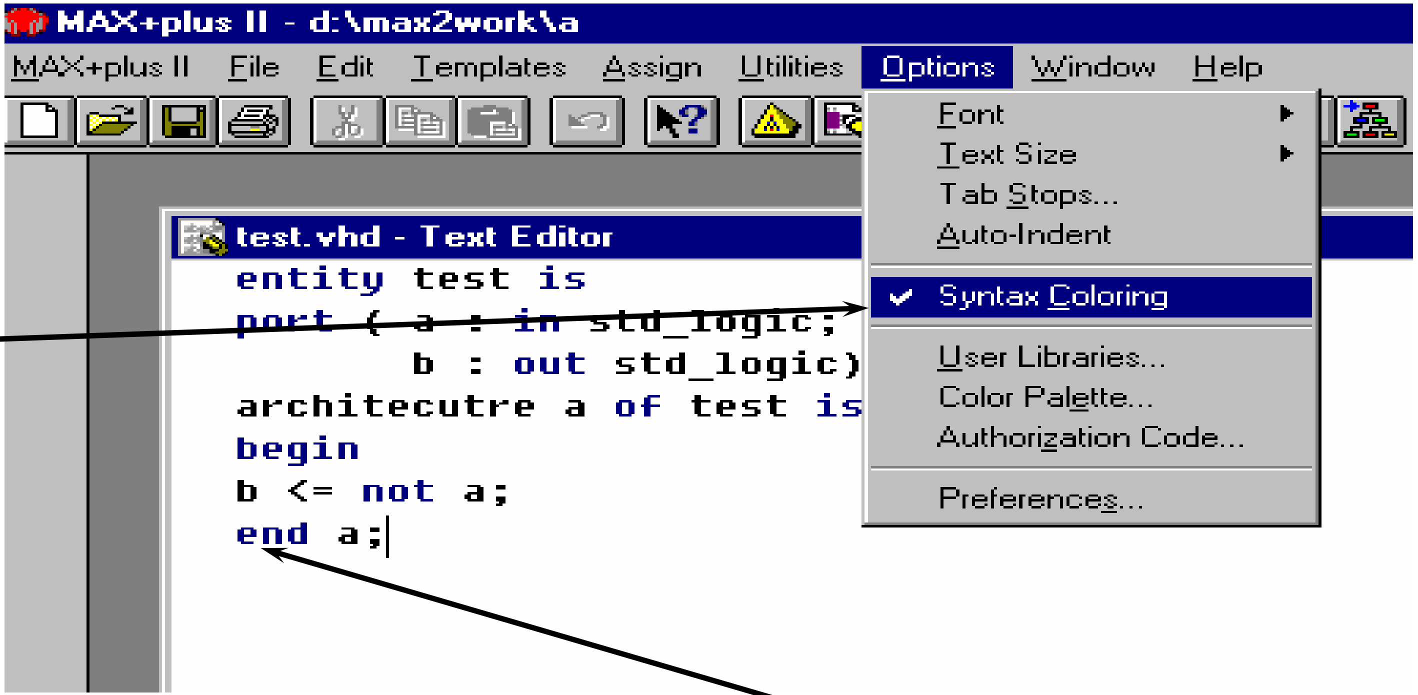
DONE！ 完成!

## Bonus Topic may help for VHDL Design within Max+Plus II

MAX+PLUSII中的获奖论文有助于 VHDL设计

## Turn on some Max+Plus II Option启用选择项

- There are some built-in Option to assist the engineer during the VHDL design stage一些内部选项可帮助设计
  - Syntax Color Option from the Option menu选择菜单中的语句颜色选择



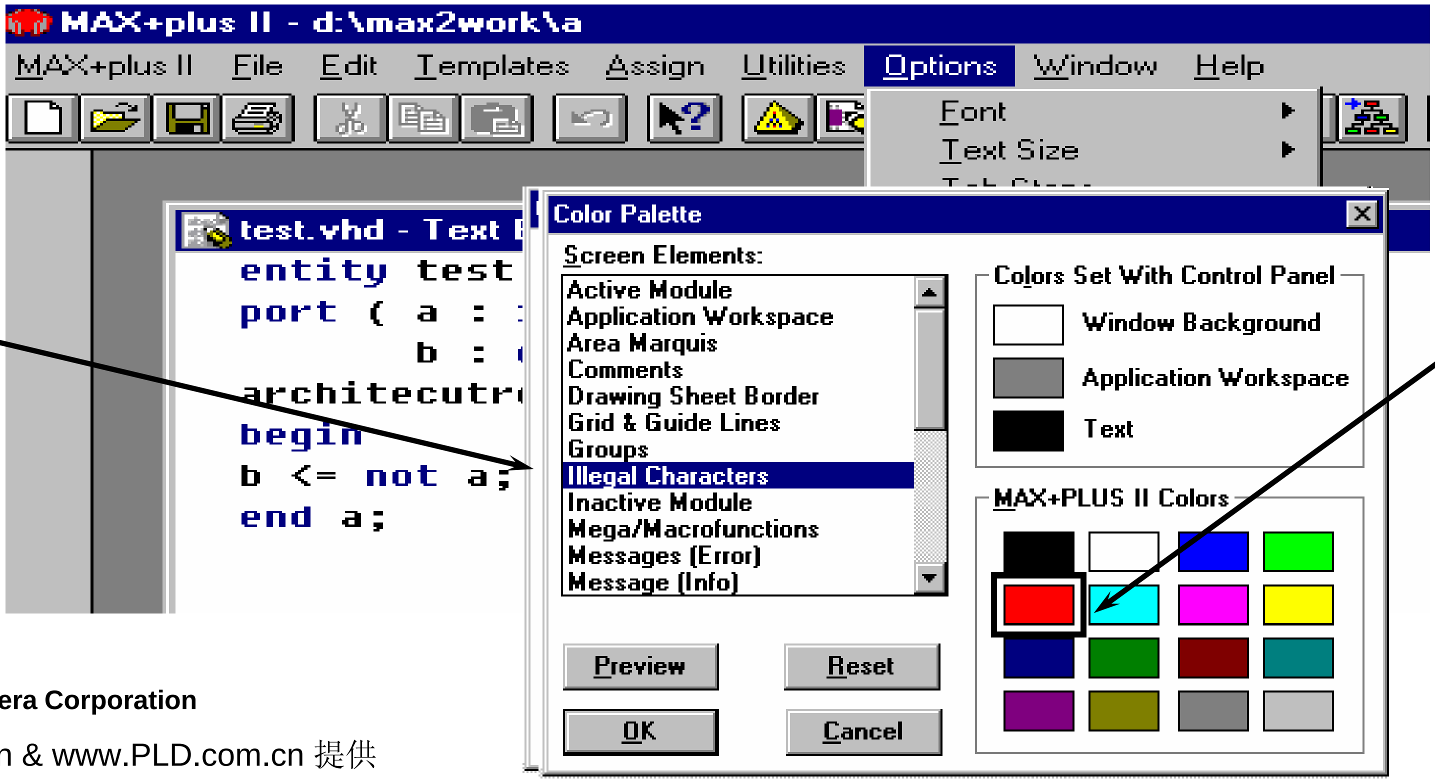
```
entity test is
port ( a : in std_logic;
          b : out std_logic);
architecutre a of test is
begin
b <= not a;
end a;
```

Turn on this option 选定此项

Reserve word in Blue 语句变为蓝色

# User can modify the Color Option用户能修改颜色

- use the Color Palette under Option Menu to  customize the color of   在选择菜单下用调色板设定颜色
  – comments, illegal characters, megafunctions, macrofuncitons….
    注释,非法字符,多功能,宏功能……



```
entity test
port ( a  :
         b  :
architecutre
begin
b <= not a;
end a;
```

Active Module
Application Workspace
Area Marquis
Comments
Drawing Sheet Border
Grid & Guide Lines
Groups
Illegal Characters
Inactive Module
Mega/Macrofunctions
Messages (Error)
Message (Info)

Select Element
选对象

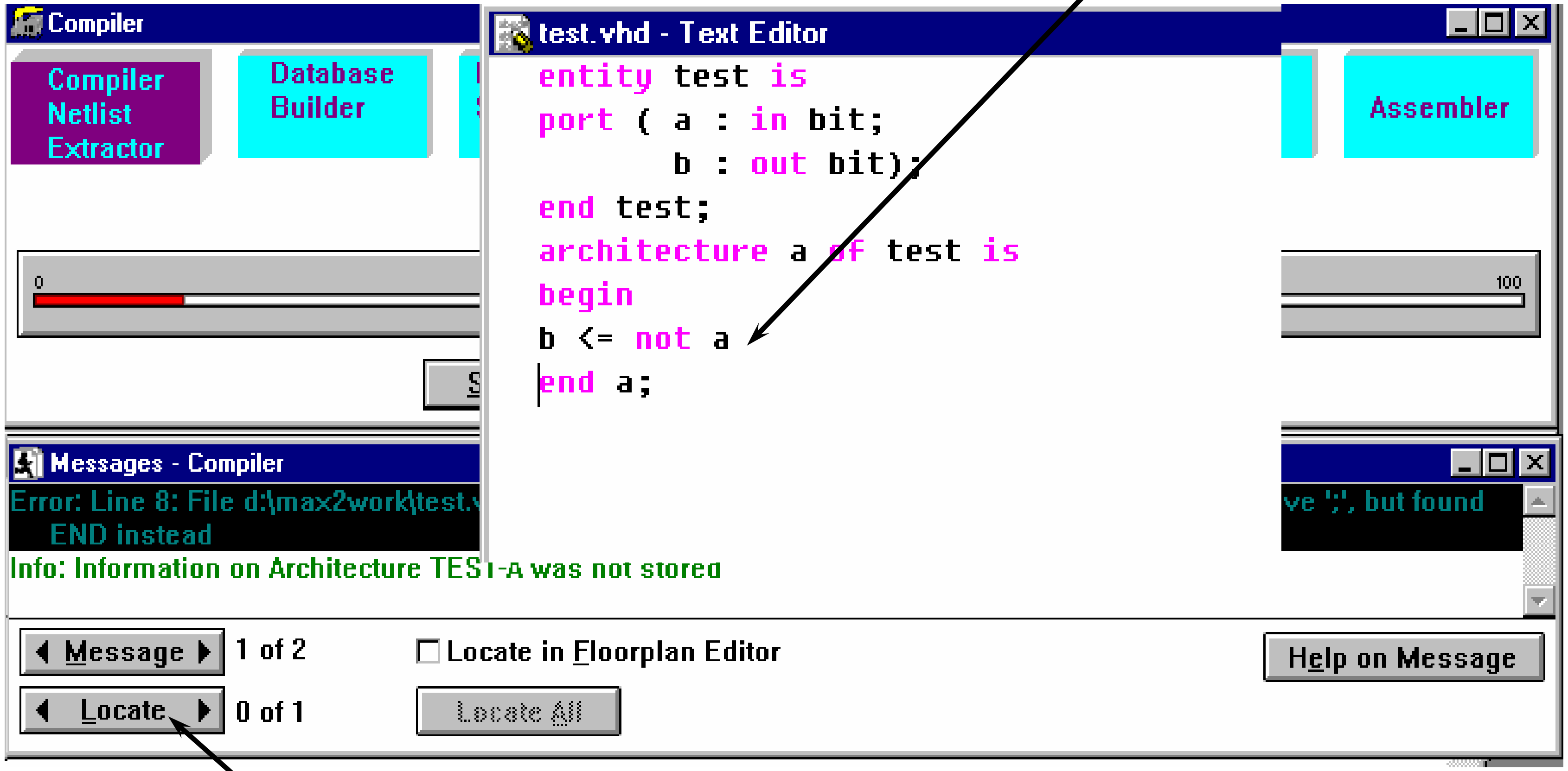
Select Color
选颜色

www.FPGA.com.cn & www.PLD.com.cn 提供

# Error Location during Compilation编译时定位错误

- Easy to locate the error 很容易确定错误的位置

Error location错误位置



```
entity test is
port ( a : in bit;
       b : out bit);
end test;
architecture a of test is
begin
b <= not a
end a;
```

```
Error: Line 8: File d:\max2work\test.v ... ve ';', but found
    END instead
Info: Information on Architecture TEST-A was not stored
```

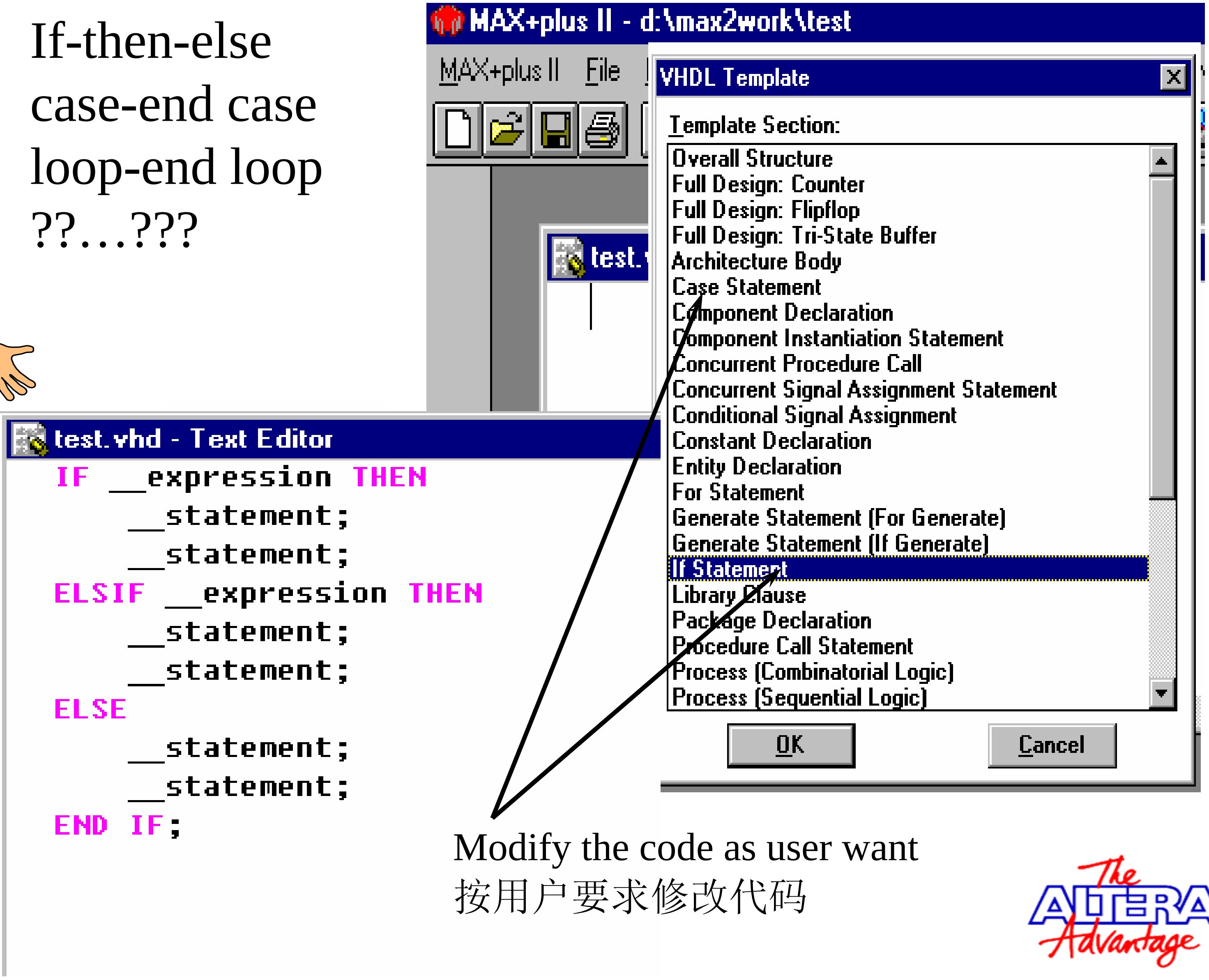
Click on the error message 点击错误讯息

Locate the error确定错误位置

# VHDL Template 模版

I forgot …….忘记了

If-then-else
case-end case
loop-end loop
??…???



Overall Structure
Full Design: Counter
Full Design: Flipflop
Full Design: Tri-State Buffer
Architecture Body
Case Statement
Component Declaration
Component Instantiation Statement
Concurrent Procedure Call
Concurrent Signal Assignment Statement
Conditional Signal Assignment
Constant Declaration
Entity Declaration
For Statement
Generate Statement (For Generate)
Generate Statement (If Generate)
If Statement
Library Clause
Package Declaration
Procedure Call Statement
Process (Combinatorial Logic)
Process (Sequential Logic)

```
IF __expression THEN
    __statement;
    __statement;
ELSIF __expression THEN
    __statement;
    __statement;
ELSE
    __statement;
    __statement;
END IF;
```

Modify the code as user want
按用户要求修改代码

# General VHDL Format 普通VHDL格式

I/O port define Section (I/O口定义段)

Must be the same as the save TEST.VHD File 必须与存贮的VHD文件名相同

Logic Behaviour define section逻辑状态定义段

Key Word关键字

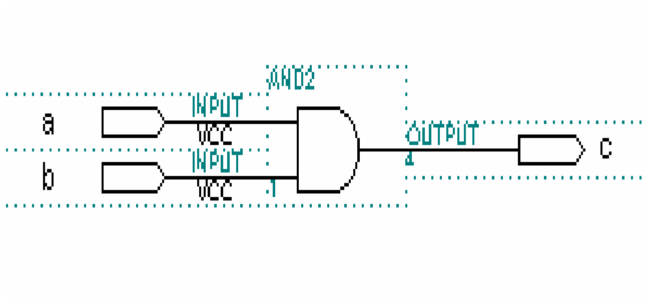
VHDL Format VHDL格式

Logic逻辑

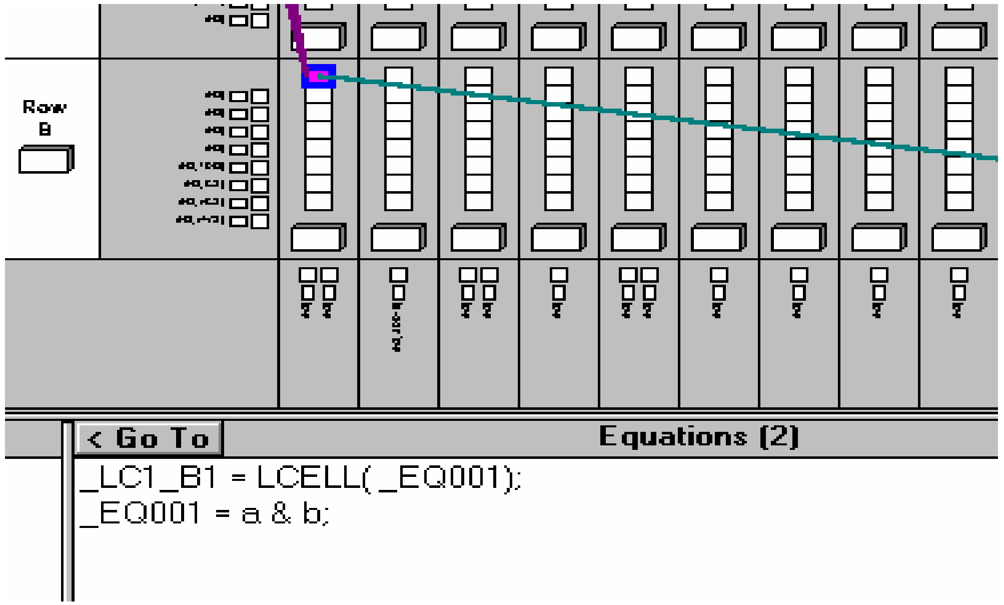
```
ENTITY test IS
PORT ( input_pin_name : IN bit;
       output_pin_name : OUT bit);
END test;
ARCHITECTURE test_body OF test IS
BEGIN
output_pin_name <= input_pin_name;
END test_body;
```

This two must be the same 此二者必须相同

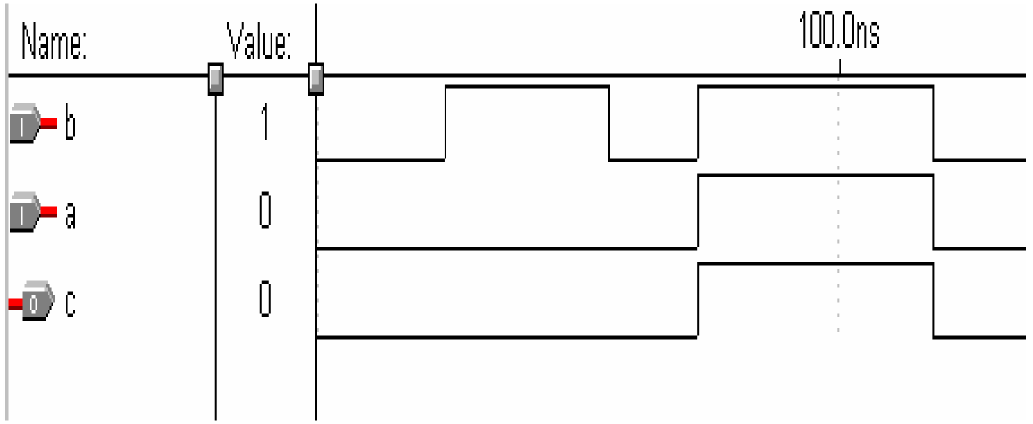
## Your First VHDL design -- 2 input AND gate 自己的第一个设计—2输入端与门



```
Entity simand is
Port ( a, b : in bit;
        c : out bit);
end simand;
architecture simand_body of simand is
begin
c <= a and b;
end simand_body;
```



```
_LC1_B1 = LCELL( _EQ001);
_EQ001 = a & b;
```

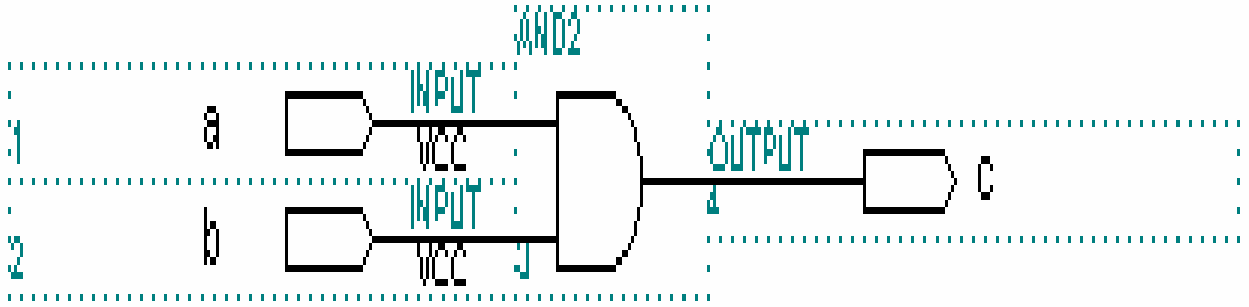


## More detail 细节内容

a, b : in bit

c : out bit

c <= a and b



## Why I use VHDL instead of Graphic 我为什么用VHDL取代图形

- Easy to Modify 容易修改
- It is more powerful than Graphic 功能比图形更强大
- VHDL is a portable language because 称VHDL为便携式语言是因为
  - is device independent 不依赖于设备
  - the same code can be applied to Device manufactured by Company A or Company B 相同的代码可以用于甲公司的设备也可用于乙公司的设备

# 图形与VHDL对比

- Graphic vs VHDL
  - Graphic is what you draw is what you get
  - 图形：需要什么画什么
    - “ tell me what hardware you want and I will give it to you”
    - “告诉我你要什么样的硬件，我会给你的”
  - VHDL is what you write is what functional you get
  - VHDL：把你需要的功能写出来
    - “ tell me how your circuit should behave and the VHDL compiler will give you the hardware that does the job”
    - 告诉我你的硬件如何工作，VHDL编译器会让硬件去做
    - but the designer can not control how the circuit implement
    - 但是设计者不能控制电路的工作

## Learning VHDL must learn
## 学习VHDL前要知道

What is Combinatorial Logic 什么是组合逻辑

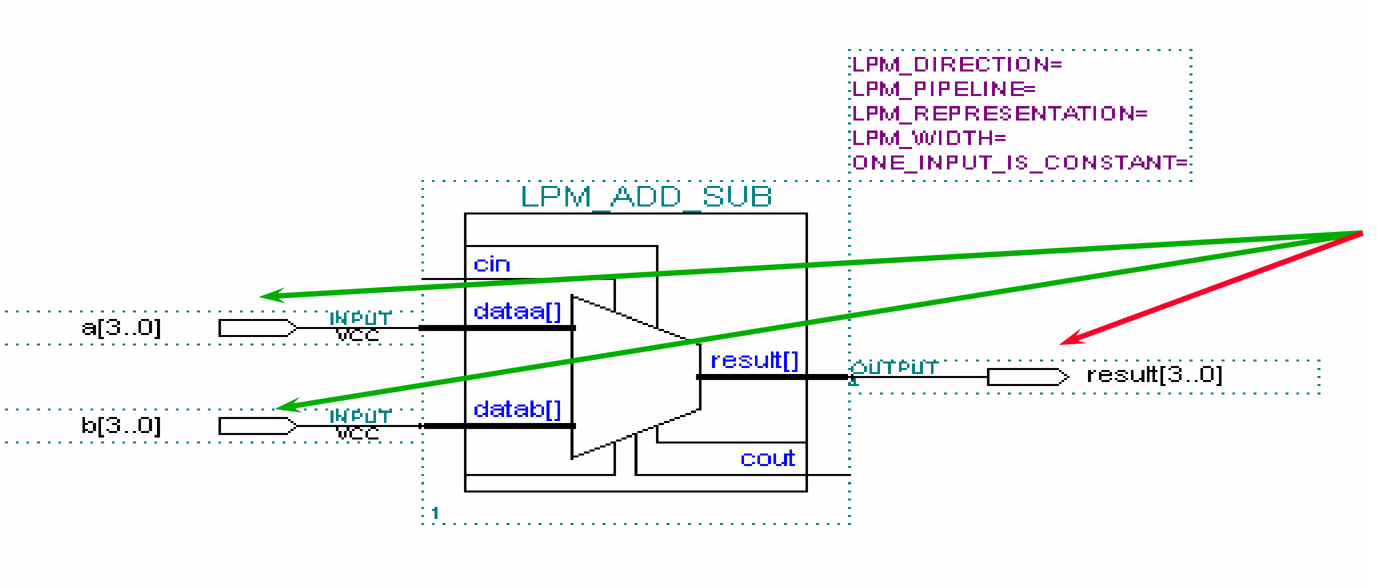
What is Sequential Logic 什么是顺序逻辑

What is Concurrent Statement 什么是并行描述

What is Process Statement 什么是过程描述

## Combinatorial Logic 组合逻辑

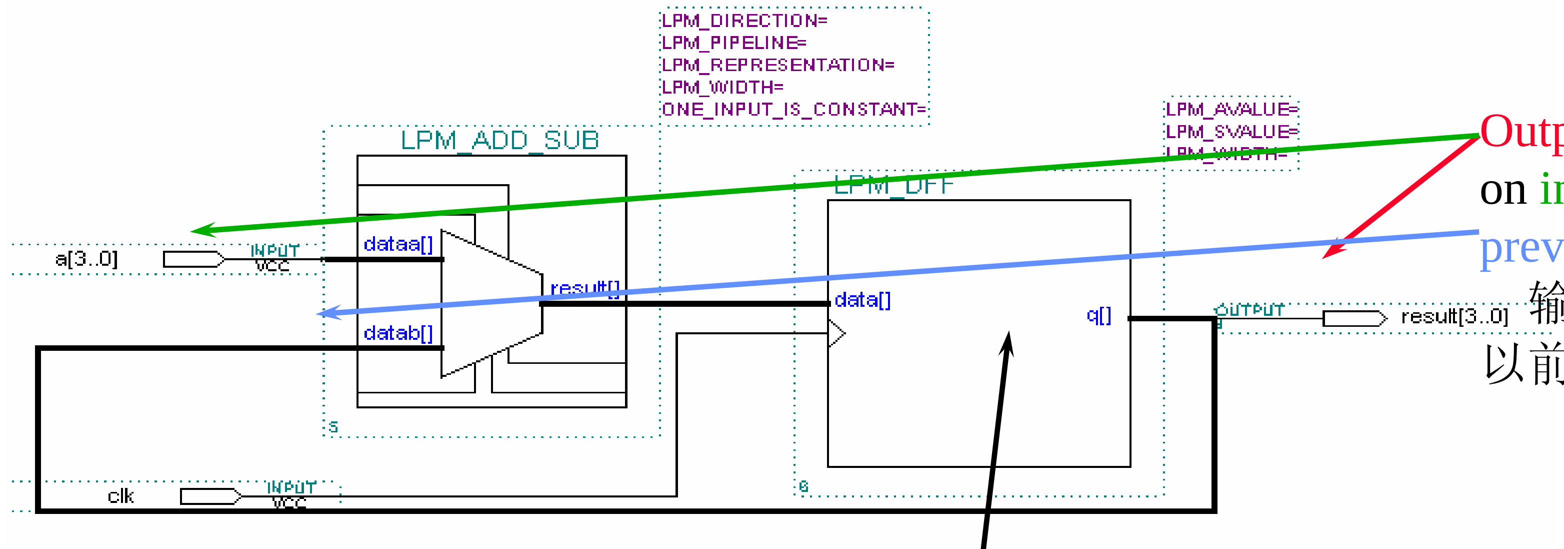
- Combinatorial Logic if 组合逻辑的工作状态
  - Outputs at a specified time are a function only of the inputs at that time 指定时刻的输出仅为此时输入的一个功能
    - e.g. decoders, multiplexes and adders 如解码器，多路器，加法器



Output change instantly when input change
输入变化时输出同时变化

# Sequential Logic 顺序逻辑

- Sequential Logic if 顺序逻辑的工作状态
  - Outputs at a specified time are a function of the inputs at that time and at all preceding times 指定时刻的输出是输入在此时及之前的功能
  - All sequential circuits must include one or more registers顺序电路必有一个以上寄存器。如状态机，计数器，移位寄存器，控制器
    - e.g. State Machine, Counters, Shift Register and Controllers



Outputs depends on inputs and previous output 输出由输入和以前的输出而定

Register is used to hold the previous value 寄存器用来保持以前的内容

## 现在大家应该知道了什么是

- Now everyone should know what is
  - Combinational Logic 组合逻辑
  - Sequential Logic 顺序逻辑

Q : Then how about Concurrent or Process Statement ?
What is it ?

问题：什么是并行描述和过程描述？

## Concurrent Statement 并行描述

- All the Concurrent Statement is executed in parallel 并行描述都是：可以同时工作的
- Concurrent Statement does not care the position within the coding 与在程序中的书写位置无关
- Concurrent Statement is : OUTPUT depends on INPUT only 输出仅与输入有关

Output depends on Input only输出仅取决于输入 without any conditional 没有任何条件限制 constraint

```
Entity test1 Is
Port ( a, b : in bit;
         c, d : out bit);
end test1;
architecture test1_body of test1 is
begin
c <= a and b;
d <= a or b;
end test1_body;
```

```
Entity test1 Is
Port ( a, b : in bit;
         c, d : out bit);
end test1;
architecture test1_body of test1 is
begin
d <= a or b;
c <= a and b;
end test1_body;
```

This two excute in parallel 这两条命令是并行的

Does not care the position within the coding 与在程序中书写位置无关

## THE SAME

```
c <= a and b;
d <= a or b;
```

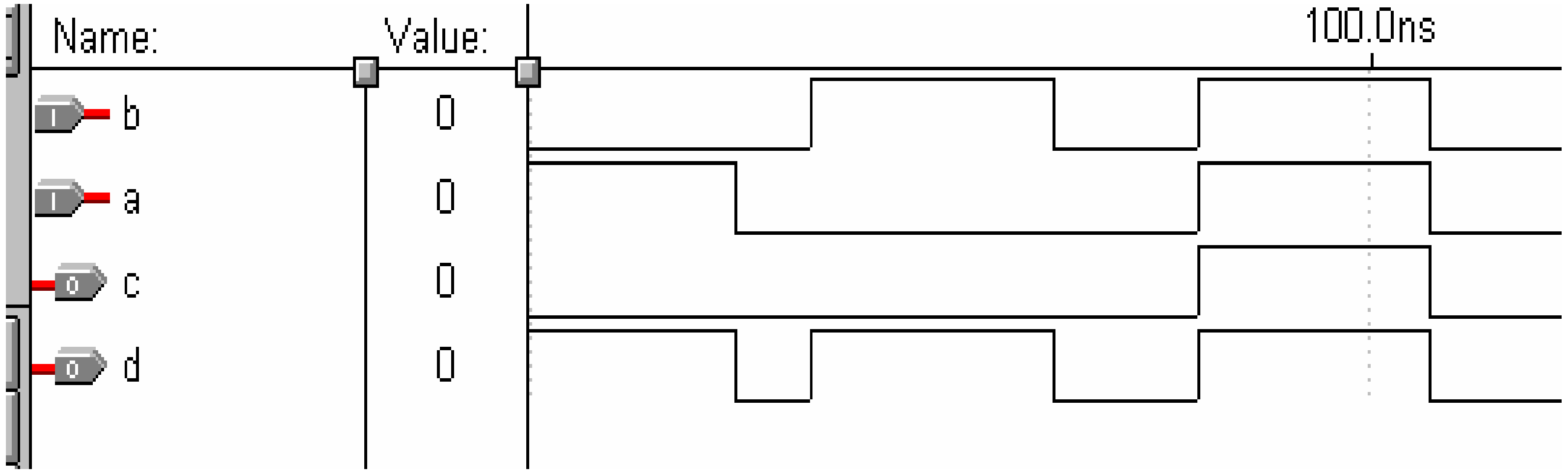
$\longleftrightarrow$

```
d <= a or b;
c <= a and b;
```

执行时一样

C = A and B

D = A OR B



## Process Statement

- All the Process Statement is executed in parallelsequential
- 所有过程描述都是并行执行的
- Within the Process Statement, the coding is execute in sequential
- 在过程描述内，代码是顺序执行的
- Process Statement is : OUTPUT depends on INPUT with Sensitivity List to control the event happen
- 输出取决于与敏感度表有关的输入，敏感度表控制事件的发生

```
Entity test1 is
Port ( clk, d1, d2 : in bit;
          q1, q2 : out bit);
end test1;
architecture test1_body of test1 is
begin
Process (clk, d1)
begin
if (clk'event and clk = '1') then
q1 <= d1;
end if;
end process;
Process (clk, d2)
begin
if (clk'event and clk= '1') then
q2 <= d2;
end if;
end process;
end test1_body;
```

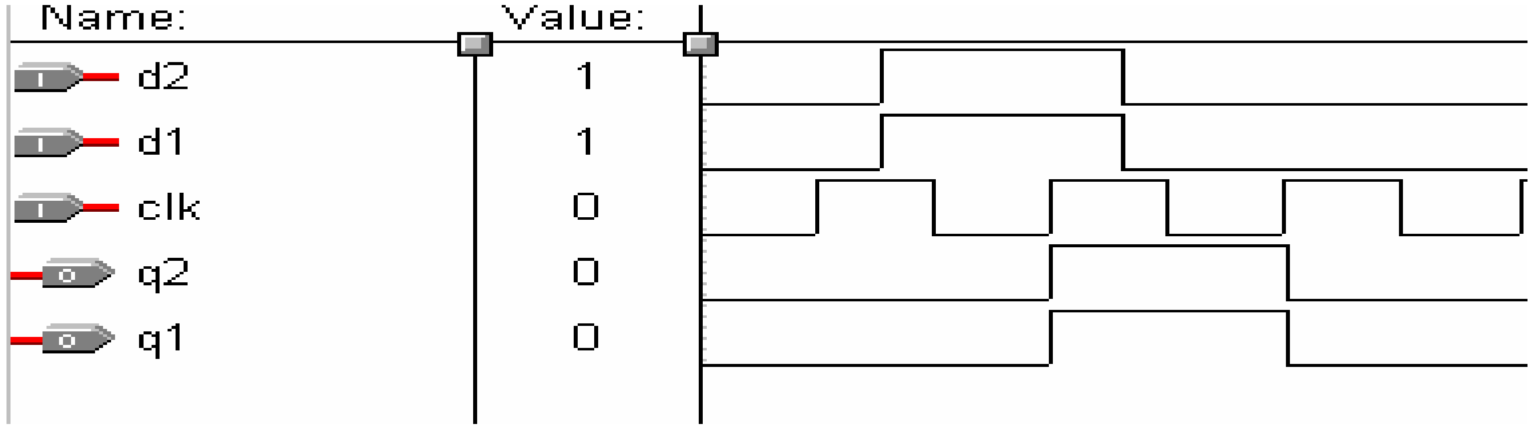
The coding is execute in sequential within the process在过程中代码是顺序执行的

```
Entity test1 is
Port ( clk, d1, d2 : in bit;
          q1, q2 : out bit);
end test1;
architecture test1_body of test1 is
begin
Process (clk, d2)
begin
if (clk'event and clk = '1') then
q2 <= d2;
end if;
end process;
Process (clk, d1)
begin
if (clk'event and clk= '1') then
q1 <= d1;
end if;
end process;
end test1_body;
```

The output depends on input with conditional constraint 输出取决于有限制条件的输入

This two processes execute in parallel 这两个过程并行执行

The two process statement execute in parallel
这两个过程描述并行执行



- Now, I know what is　现在，我们知道了什么是
  - combinational logic　组合逻辑
  - sequential logic　顺序逻辑
  - concurrent statement　并行描述
  - process statement　过程描述

Q : What is the usage of this in VHDL ?问：它们在VHDL中怎么用？

A : Engineer can use the mixture of 答：工程师可混合使用这些内容

combinational logic, sequential logic, concurrent statement and process statement

to do the design　来进行设计

## How to ... ?　　关于...?

- Now I know what is Combinational Logic but?

Q : How to implement of Combinational Logic in VHDL?问:在VHDL中如何实现组合逻辑

- Combinational Logic can be implemented by 可实现,通过
  - Concurrent Signal Assigment Statements并行信号分配描述
  - Process Statement that describe purely combinational behaviour i.e. behaviour that does not depends on any CLOCK EDGE
  - 过程描述纯粹表达组合行为，即与任何CLOCK EDGE无关的行为

# Concurrent Statements for Combinational Logic

组合逻辑中的并行描述

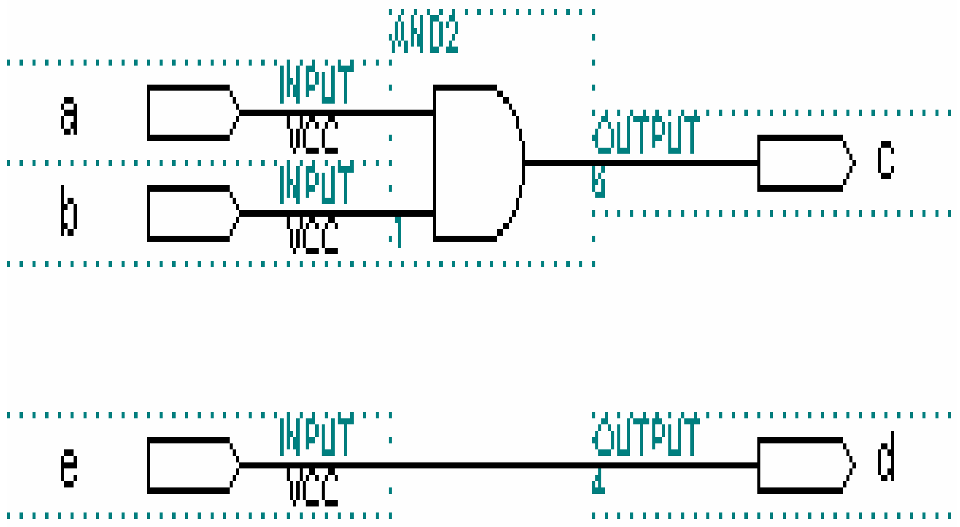
## Concurrent Statements 并行描述

- There are several different kinds of Concurrent Statements几种不同类型的并行描述
  - (1) Simple Signal Assigments  简单信号分配
  - (2) Conditional Signal Assigments  条件信号分配
  - (3) Selected Signal Assignments    选择信号分配

# (1) Simple Signal Assigment 简单信号分配

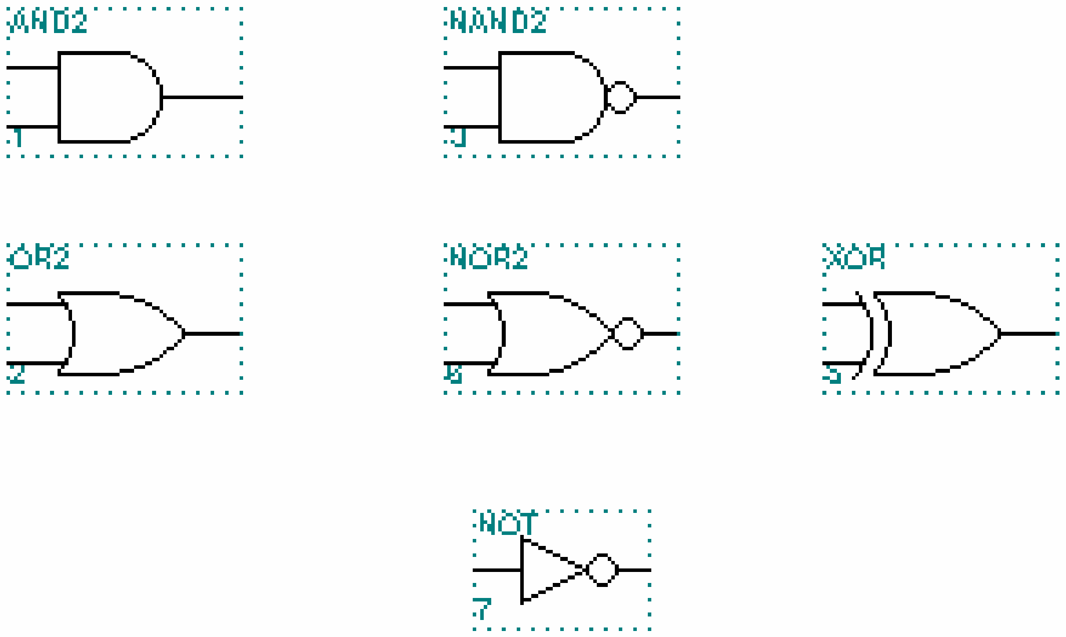
- These kind of statements are executed in Parallel
- 这种描述是并行执行的

```
Entity test1 is
port ( a, b, e : in bit;
          c, d : out bit);
end test1;
architecture test1_body of test1 is
begin
c <= a and b;
d <= e;
end test1_body;
```



## What kind of logic support l逻辑运算种类

- AND 与
- NAND 与非
- OR 或
- NOR 或非
- XOR 异或
- NOT 非
- more .......更多



# I want 5 Input AND Gate设计5输入与门

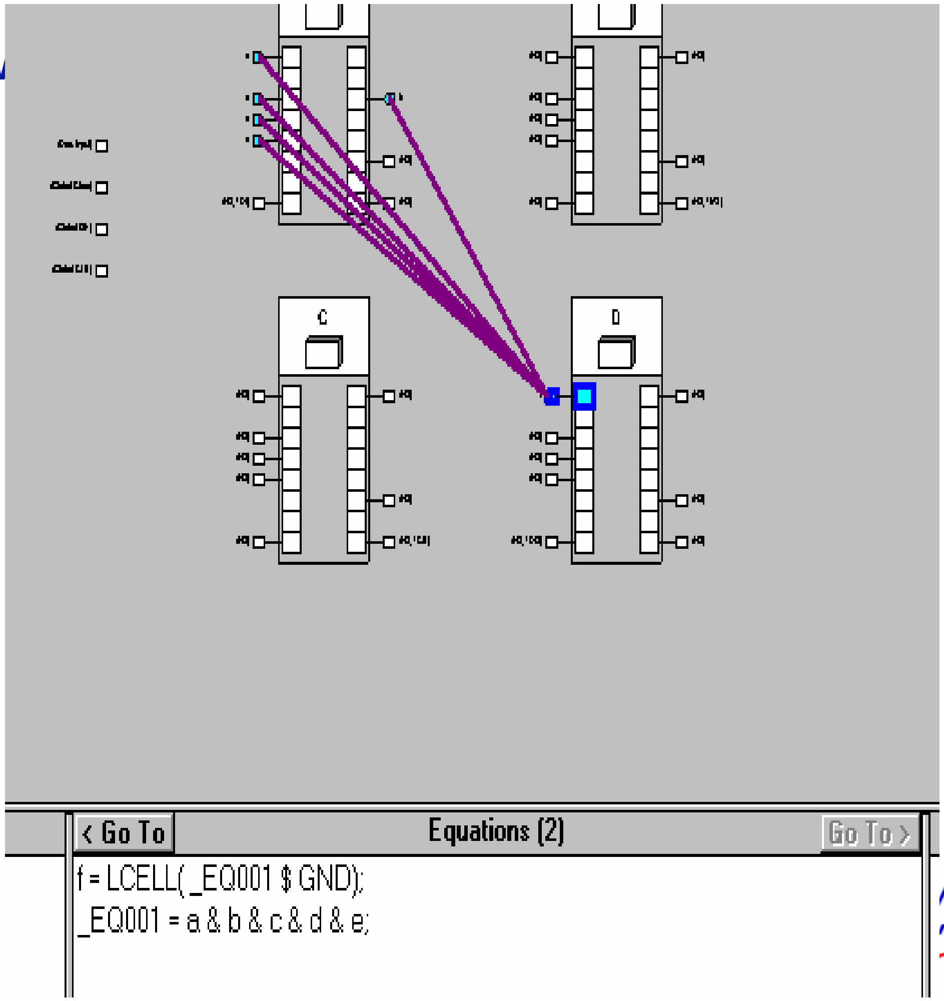
Q :AND is only a two input, if I want more input, what can I do ?

A : It is easy, we are due w

问：与门只有两个输入，如果想要更多的输入怎么办?

答：好办。用语句就可实现。

```
Entity test1 is
port ( a, b, c, d, e : in bit;
        f : out bit);
end test1;
architecture test1_body of test1 is
begin
f <= a and b and c and d and e;
end test1_body;
```



```
f = LCELL( _EQ001 $ GND);
_EQ001 = a & b & c & d & e;
```

# LAB 1
# 实验1

## Design a 7 Input OR gate
## 设计7输入端或门

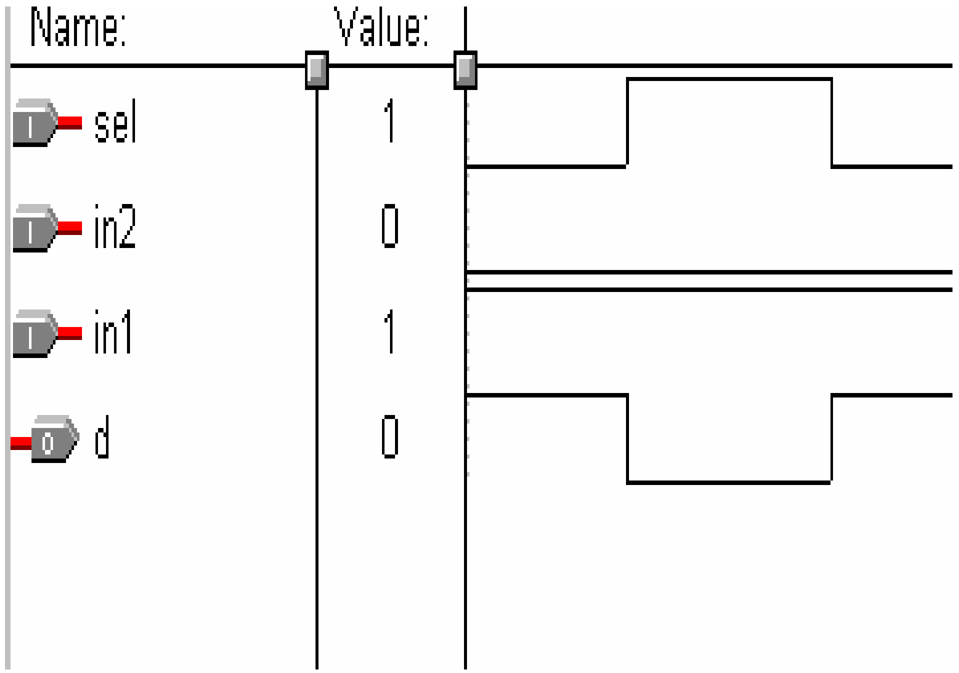
# Sampling Coding代码示例

```
Entity test1 is
port ( a, b, c, d, e, f, g  : in bit;
          h : out bit);
end test1;
architecture test1_body of test1 is
begin
h <= a or b or c or d or  e or f or g;
end test1_body;
```

## (2) Conditional Signal Assigments条件信号分配

- The output get the value when the condition is true条件为真时则有输出
  - e.g. 2 to 1 multiplexer 如 2选1多路开关

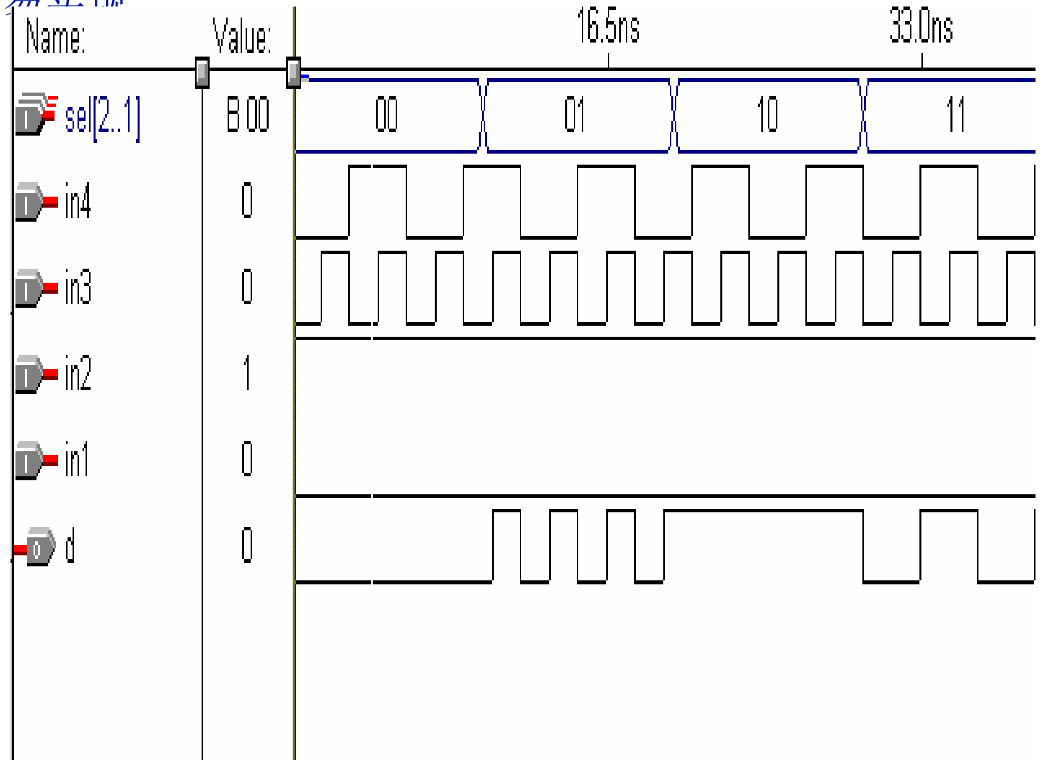
```
Entity test1 is
port (in1, in2, sel : in bit;
        d : out bit);
end test1;
architecture test1_body of test1 is
begin
d <= in1 when sel = '0'
                else in2;
end test1_body;
```



# If I want more -- 4 to 1 Mux 输入增加，4选1多路开关

- Once again, you are due with Language not Graphic, so it is easy 由于使用语句完成，很容易实现

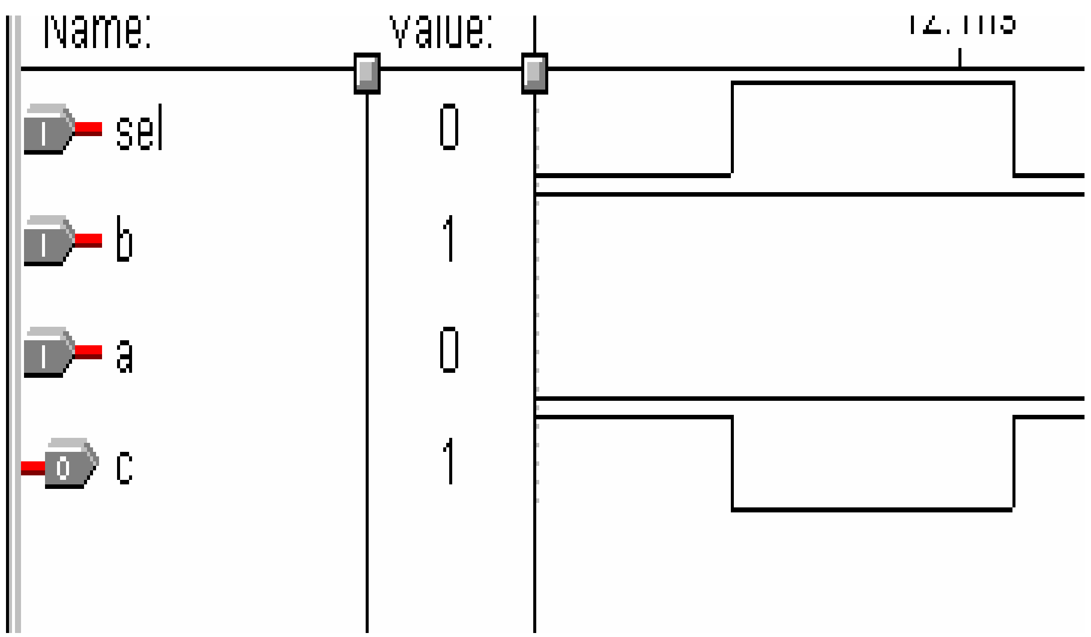
```
Entity test1 is
port (in1, in2, in3, in4 : in bit;
          sel1, sel2 : in bit;
          d : out bit);
end test1;
architecture test1_body of test1 is
begin
d <= in1 when sel1 = ‘0’ and sel2 = ‘0’ else
          in2 when sel1 = ‘0’ and sel2 = ‘1’ else
          in3 when sel1 = ‘1’ and sel2 = ‘0’ else
          in4;
end test1_body;
```



## (3) Select Signal Assignments选择信号分配

- The output get value when matching with the selected item
- 当与选择条件匹配时产生输出

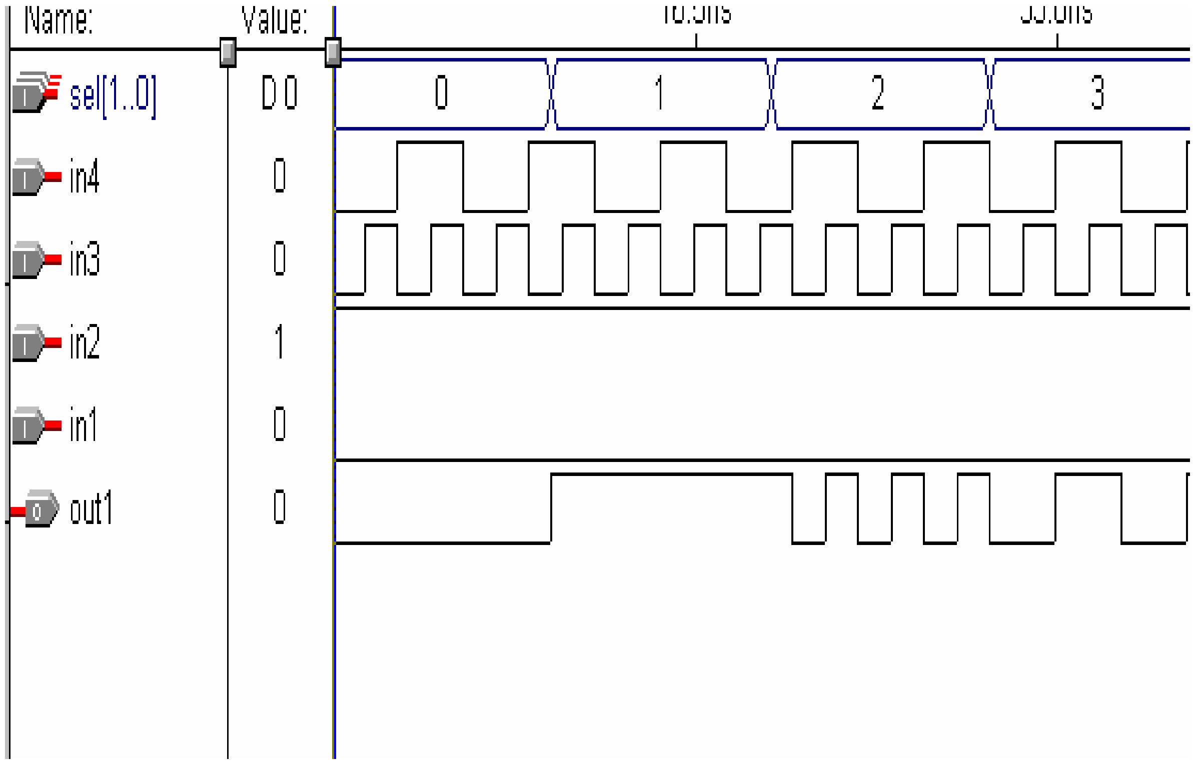
```
Entity test1 is
port (a, b: in bit;
        sel  : in bit;
        c : out bit);
end test1;
architecture test1_body of test1 is
begin
with sel select
   c <= a when '1',
           b when '0';
end test1_body;
```



# If I want more choice --- 如果选项增多时

- It is easy 也很容易

```
Entity test1 is
port (in1, in2, in3, in4 : in bit;
        sel  : in integer;
        out1 : out bit);
end test1;
architecture test1_body of test1 is
begin
with sel select
    out1 <= in1 when 0,
                  in2 when 1,
                  in3 when 2,
                  in4 when 3;
end test1_body;
```

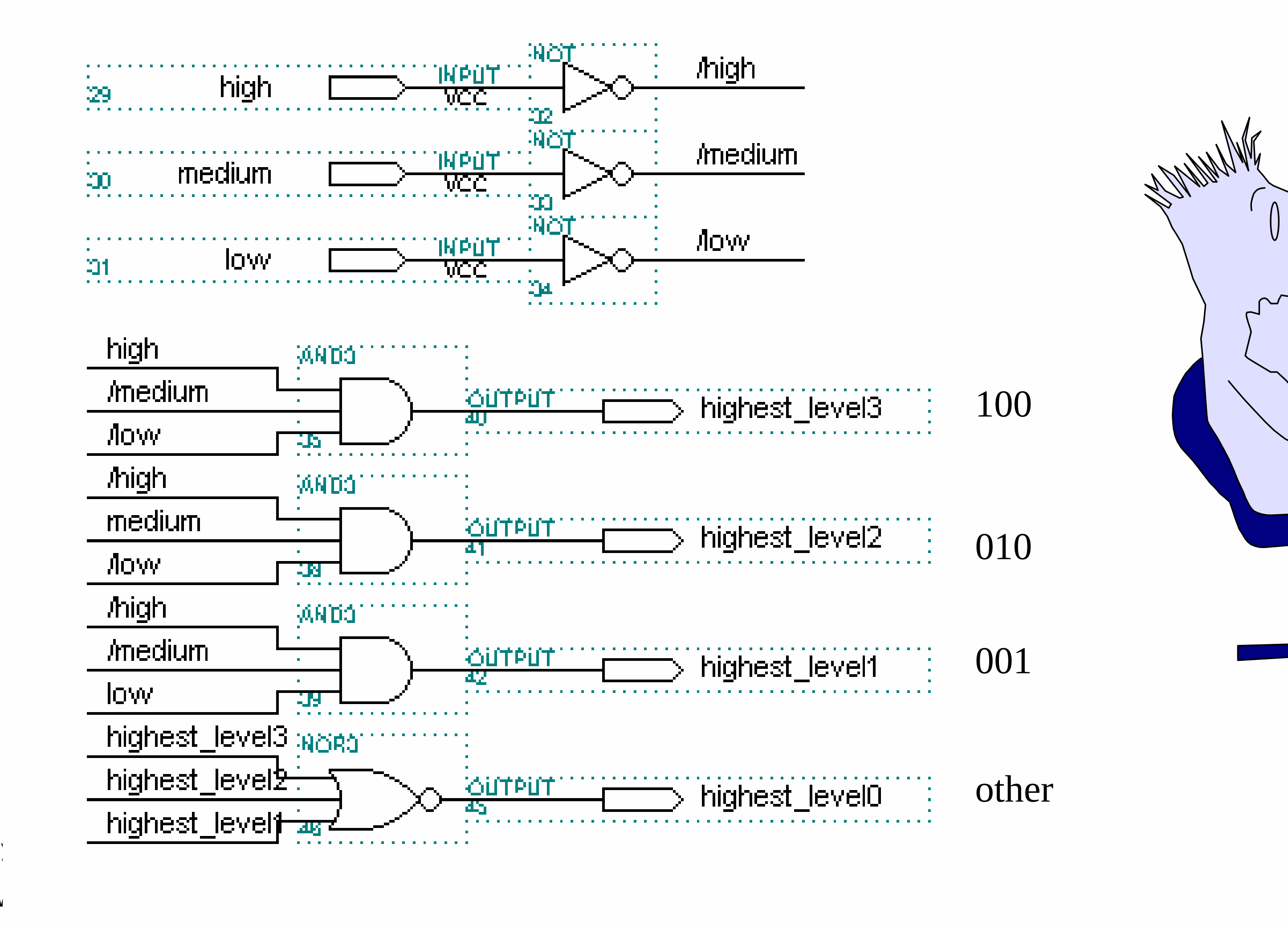


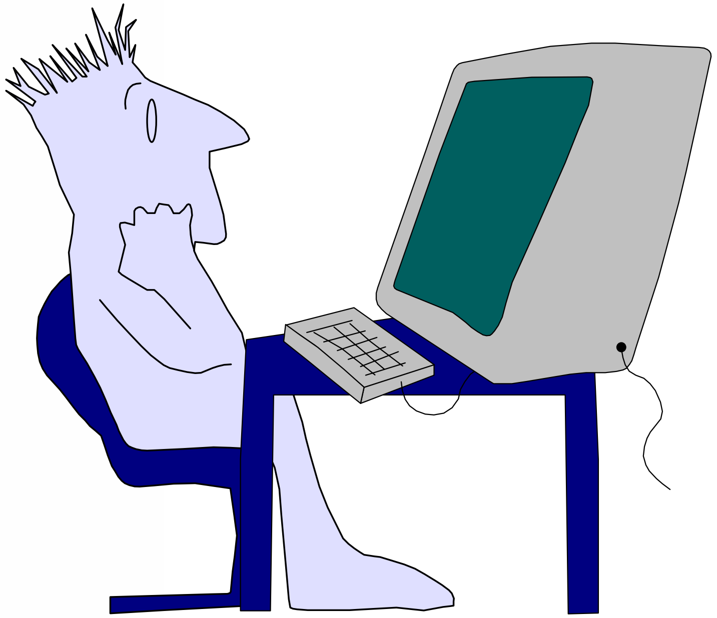
# LAB 2
# 实验2

## Convert the design from Graphic to VHDL
## 从图形向VHDL转换设计

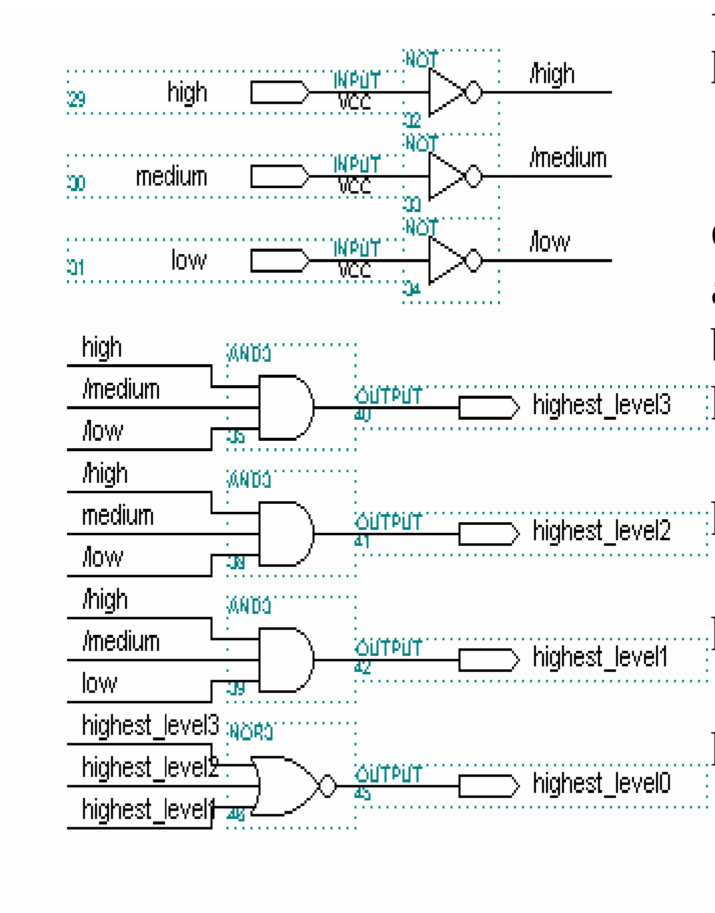
# Design Constraint 设计限制

- Use the WHEN-ELSE statement to finish your design
- 用WHEN-ELSE 语句完成你的设计



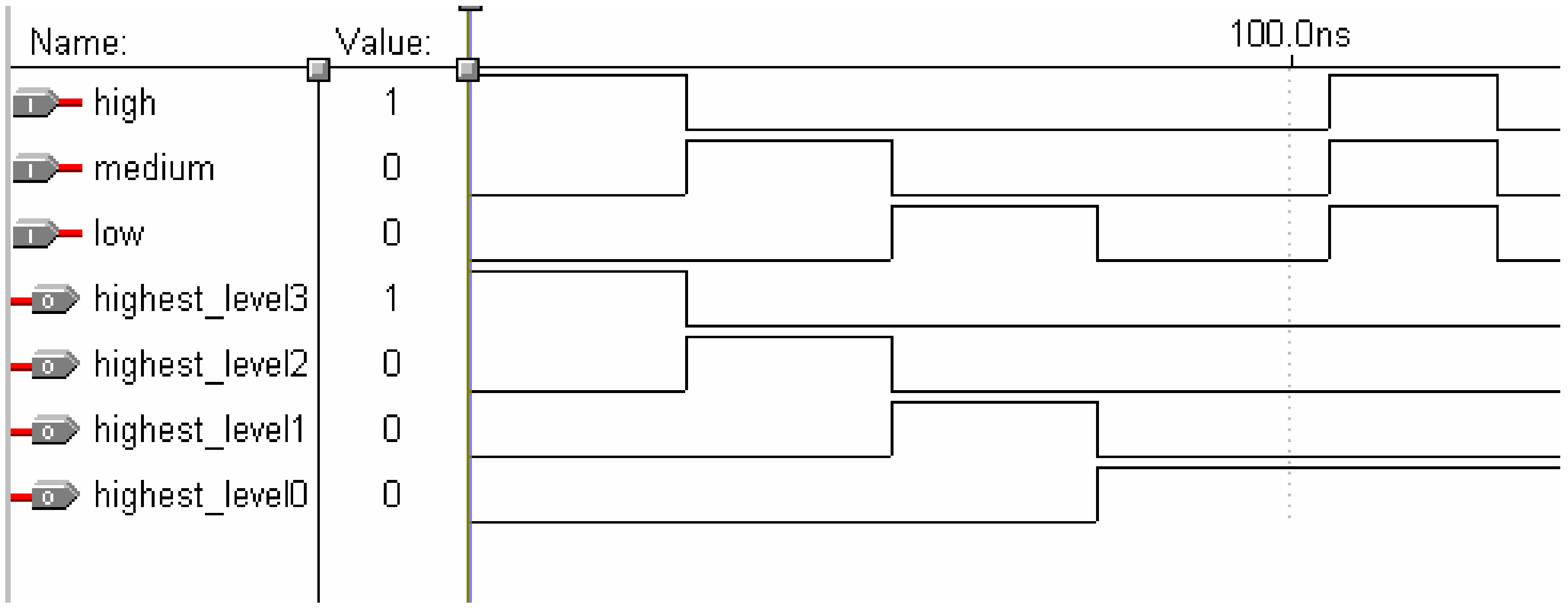


# Sampling Coding 代码示例



```
Entity test1 is
port (high, medium, low : in bit;
        highest_level3, highest_level2 : out bit;
        highest_level1, highest_level0 : out bit);
end test1;
architecture test1_body of test1 is
begin
highest_level3 <= ‘1’ when high=‘1’ and medium=‘0’ and low=‘0’ else
                  ‘0’;
highest_level2 <= ‘1’ when high=‘0’ and medium=‘1’ and low=‘0’ else
                  ‘0’;
highest_level1 <= ‘1’ when high=‘0’ and medium=‘0’ and low=‘1’ else
                  ‘0’;
highest_level0 <= ‘0’ when high=‘1’ and medium=‘0’ and low=‘0’ else
                  ‘0’ when high=‘0’ and medium=‘1’ and low=‘0’ else
                  ‘0’ when high=‘0’ and medium=‘0’ and low=‘1’ else
                  ‘1’;
end test1_body;
```

# Simulation 模拟运行结果



## Process Statement for Combinational Logic

组合逻辑中的过程描述

# Process Statement 过程描述

- There are some rules for the Process Statement usage　使用过程描述的一些规则
  - any kind of Process Statement must have SENITIVITY LIST
  - 过程描述中必须有敏感度表
    - sensitivity list contains the signals that cause the process statement to execute if their values change
    - 敏感度表中的信号，当其值变化时使过程描述执行
  - the statement within the Process Statement will be execute STEP-BY-STEP 过程描述中的语句是一条接一条执行

Q : What does it all means ? 问：这意味着什么?

A : Follow me…….答：接着听…..

## Template for Process Statement过程描述的模版

Using the “SENSITIVITY LIST”

```
name : PROCESS (sensitivity_list)
begin
  sequential statement #1
  sequential statement #2
  .......
  sequential statement # N
END PROCESS name;
```

General format for a Process Statemtent
过程描述的格式

```
Process (   )
begin
..
end process
```

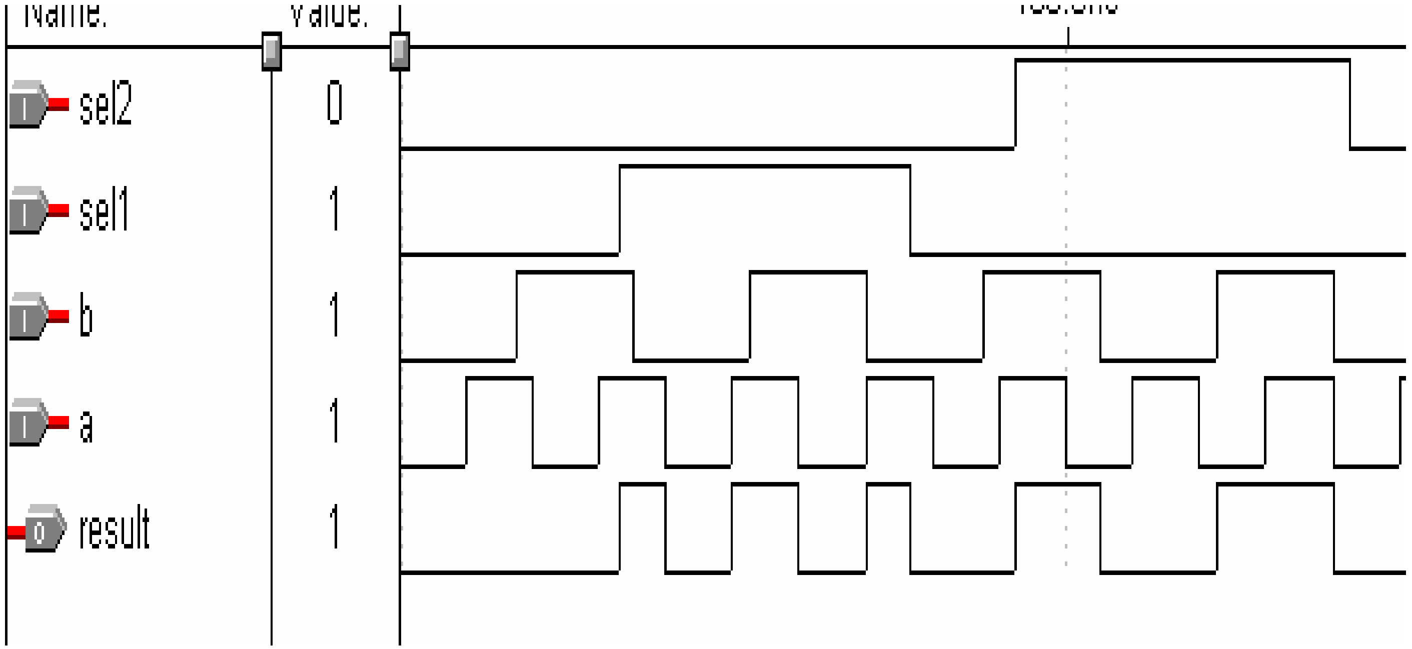
Sensitivity List敏感度表

Sequential Statement顺序描述
The sequential statement execute one by one and follow the order, ie. finish the #1 then #2 then #3
它们一条接一条地顺序执行,即按#1,#2,#3…

name is optional名字是可选的

# Example will be more clear用实例更清楚

```
Entity test1 is
port (a, b,  sel1, sel2 : in bit;
          result : out bit);
end test1;
architecture test1_body of test1 is
begin
process (sel1, sel2,a, b)
begin
if (sel1 = '1') then
result <= a;
elsif (sel2 = '1') then
result <= b;
else
result <= '0';
end if;
end process;
end test1_body;
```



Result change if sel1, sel2, a or b change the value
如果sel1,sel2,a,b的值改变则结果改变

Wait : I can do the same join with Concurrent Statement

## Concurrent Statement并行描述

```
Entity test1 is
port (a, b,  sel1, sel2 : in bit;
          result : out bit);
end test1;
architecture test1_body of test1 is
begin
result <= a when sel1 = ‘1’ else
                 b when sel2 = ‘1’ else
                 ‘0’;
end test1_body;
```

## Process Statement过程描述

```
Entity test1 is
port (a, b,  sel1, sel2 : in bit;
          result : out bit);
end test1;
architecture test1_body of test1 is
begin
process (sel1, sel2,a, b)
begin
if (sel1 = ‘1’) then
result <= a;
elsif (sel2 = ‘1’) then
result <= b;
else
result <= ‘0’;
end if;
end process;
end test1_body;
```

Same function but different way to do the coding编程方法不同但结果相同

Q : What is the different between Concurrent and Process Statement 问:并行和过程描述有什么不同?

A : For this simple example, both Concurrent and Process can do the same job.  But some function must use Process Statement to do

答:对于此例,两种方法可完成同一工作.但是,有些功能必须用过程描述完成.

## How to ... ? 如何…

- Now I know what is Sequential Logic but

Q : How to implement of Sequential Logic in VHDL?

问:在VHDL中如何实现顺序逻辑?

- Sequential Logic can be implemented by
  - Process Statement describe the logic with some CLOCK signal

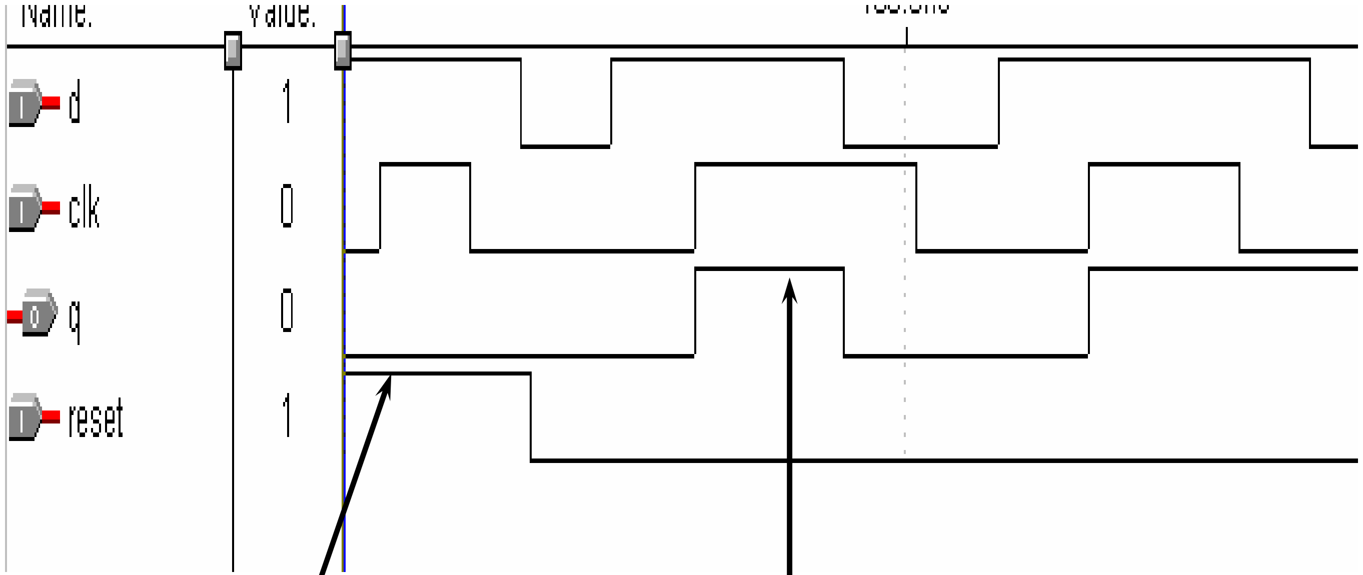
  可以用带有时钟信号的过程描述来完成.

## Process Statement for Sequential Logic

顺序逻辑中的过程描述

# How to do the Latch 如何进行锁存

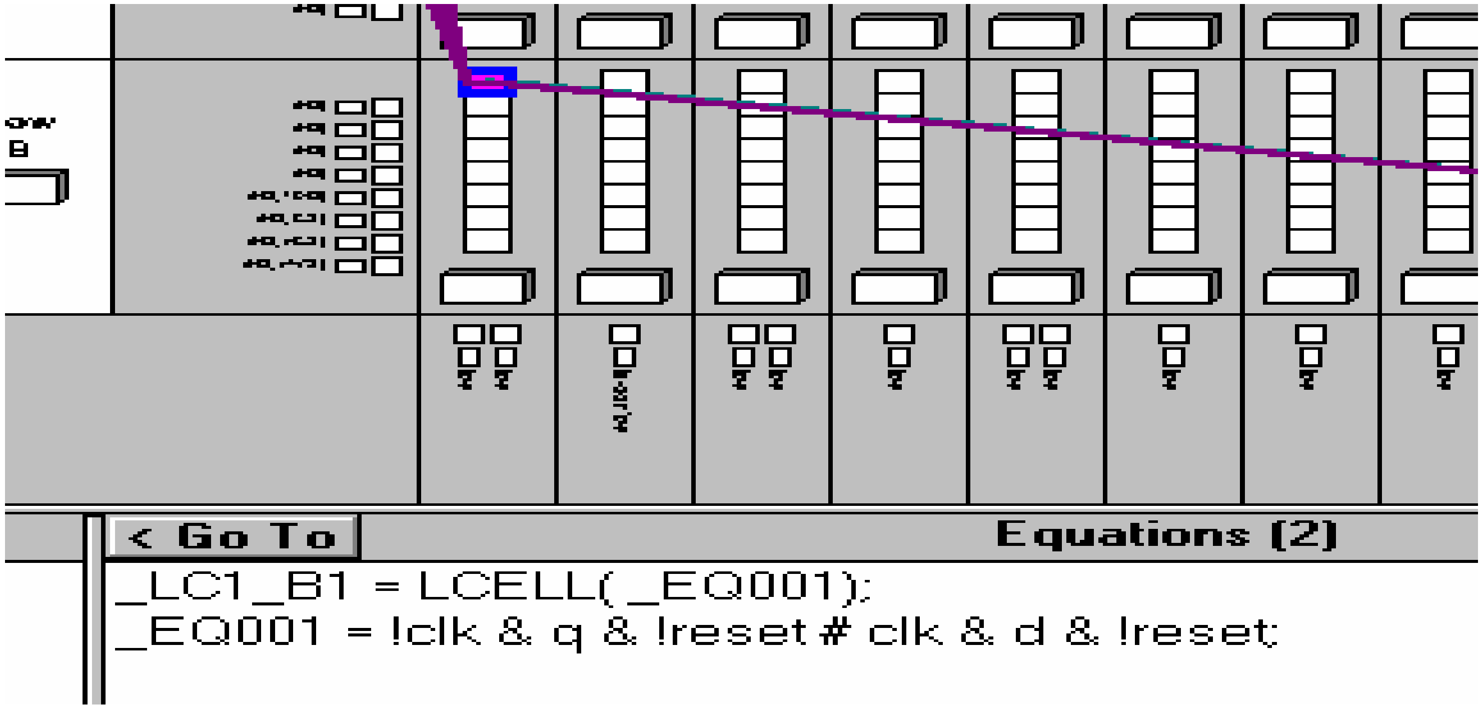
```
Entity test1 is
port (clk, d , reset : in bit;
        q : out bit);
end test1;
architecture test1_body of test1 is
begin
process (clk, d, reset)
begin
  if (reset = '1') then
     q <= '0';
  elsif (clk = '1') then
     q <= d;
  end if;
end process;
end test1_body;
```



Reset take over the control first
复位首先接管控制

Clk take the control Second时钟接着进行控制

Within the process excute in step-by-step在过程中顺序执行

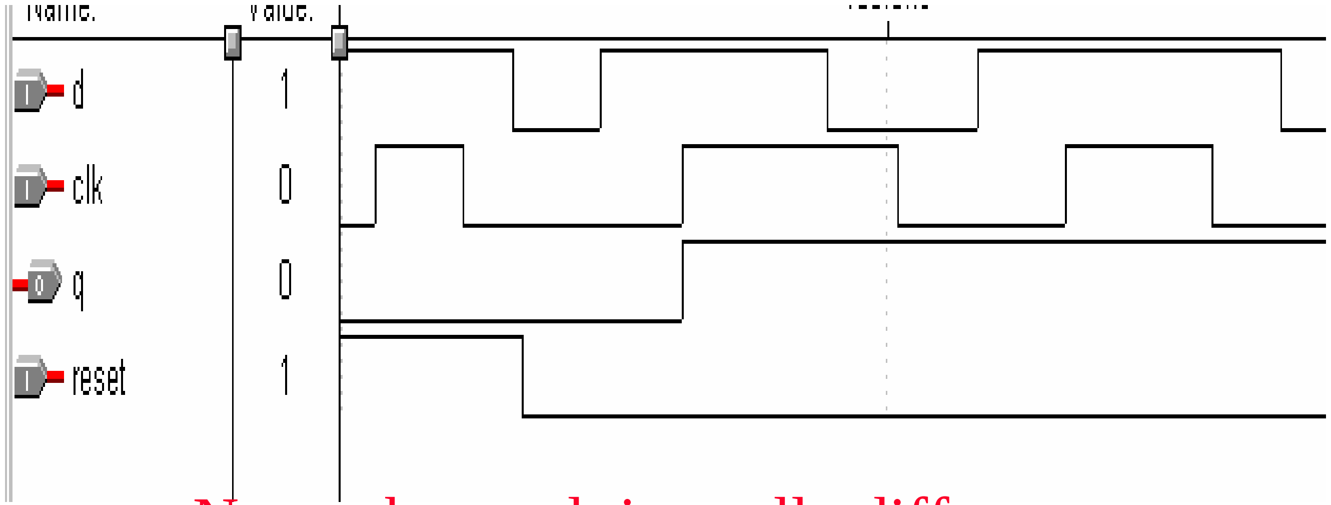


```
_LC1_B1 = LCELL(_EQ001);
_EQ001 = !clk & q & !reset # clk & d & !reset;
```

This is a LATCH

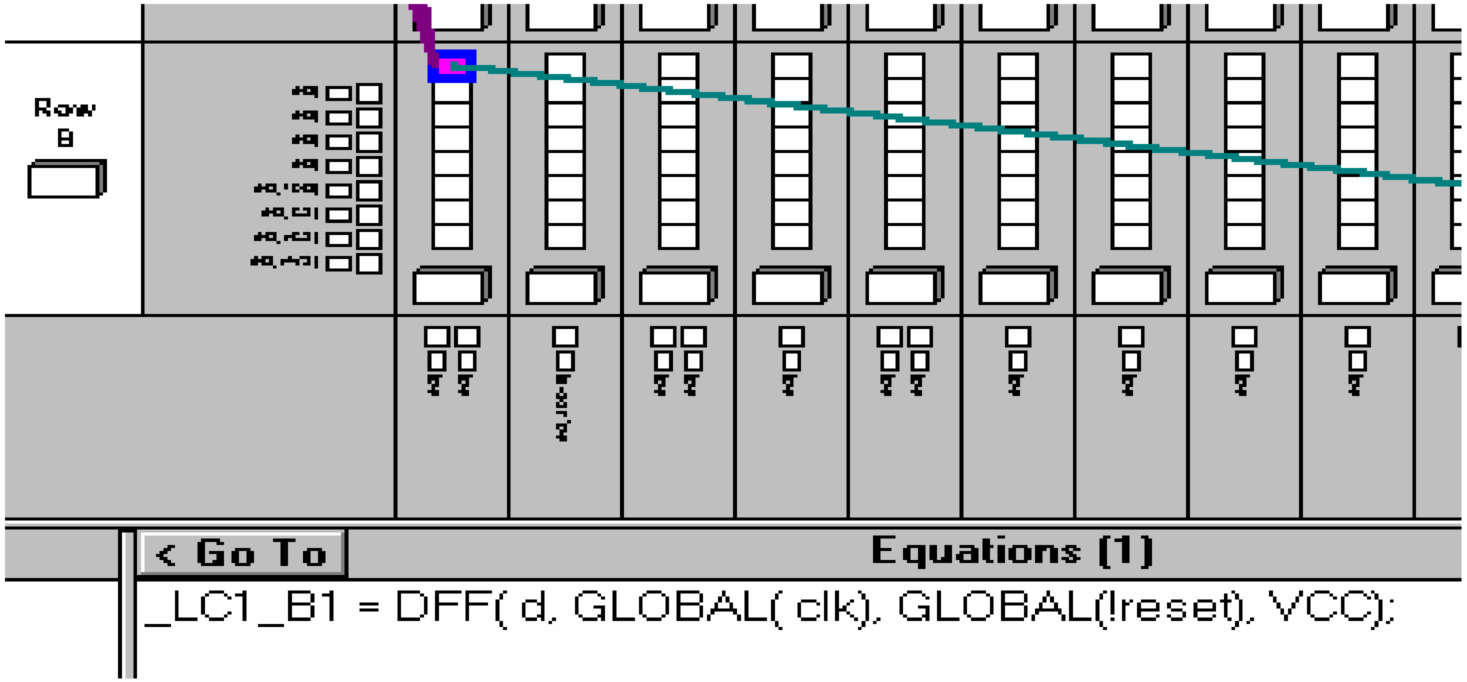
# If I modify the code to...如果修改代码来...

```
Entity test1 is
port (clk, d , reset : in bit;
         q : out bit);
end test1;
architecture test1_body of test1 is
begin
process (clk)
begin
   if (reset = '1') then
       q <= '0';
   elsif (clk = '1') then
       q <= d;
   end if;
end process;
end test1_body;
```



Note : the result is totally different
注意:结果完全不同

What is it ?
为什么?



_LC1_B1 = DFF( d, GLOBAL( clk), GLOBAL(!reset), VCC);

I get a Flip-Flop not a LATCH

这是 触发器,不是锁存器

## Why I have a Flip-Flop not a Latch
为什么成了触发器而不锁存器?

- Latch with a Sensitivity list 锁存器的敏感度表如下

  process (clk, d, reset)

- Flip-Flop with a Sensitivity list触发器的敏感度表这样

  process(clk)

Q : What is the Sensitivity list use for ?

问:什么是适用的敏感度表?

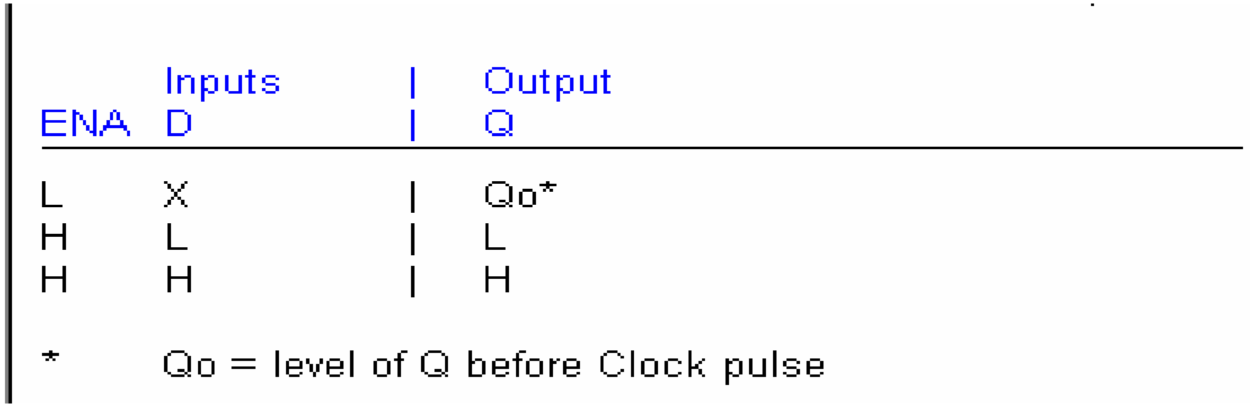
A : The OUTPUT change when the Sensitivity list change

答:当敏感度表改变时输出也改变.

## More Detail更多细节

- process (clk, d, reset)
  - this say that the OUTPUT change when either clk, d or reset change, if clk, d or reset not change, then maintain the output
  - what kind of device will provide this function ?
  - 当CLK,D,RESET中任一个改变时,输出也改变. 如果它不改变,则输出不变
  - 什么样的设备可提供这种功能?

LATCH锁存器

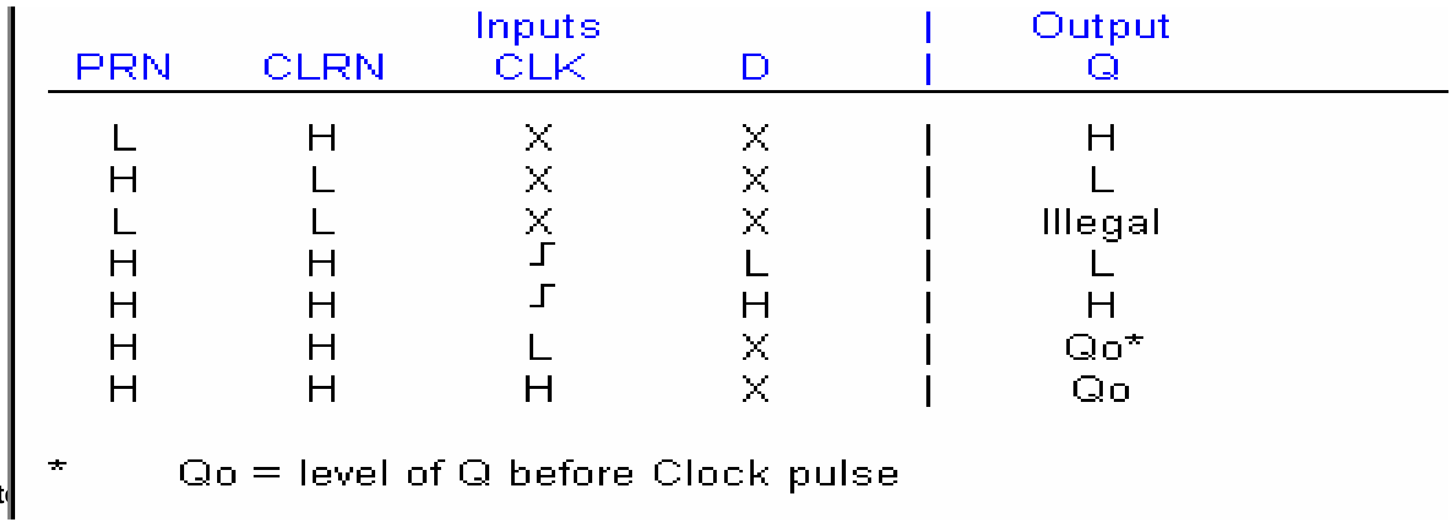


|  | Inputs | Output |
|---|---|---|
| ENA | D | Q |
| L | X | Qo* |
| H | L | L |
| H | H | H |

* Qo = level of Q before Clock pulse

- process (clk)
  - this say that OUTPUT change when CLK change, if clk does not change, maintain the output
  - what kind of device will provide this function ?
  - 当CLK改变输出改变,CLK不变时输出不变
  - 有这种功能的是什么类型的器件?

Flip-Flop触发器



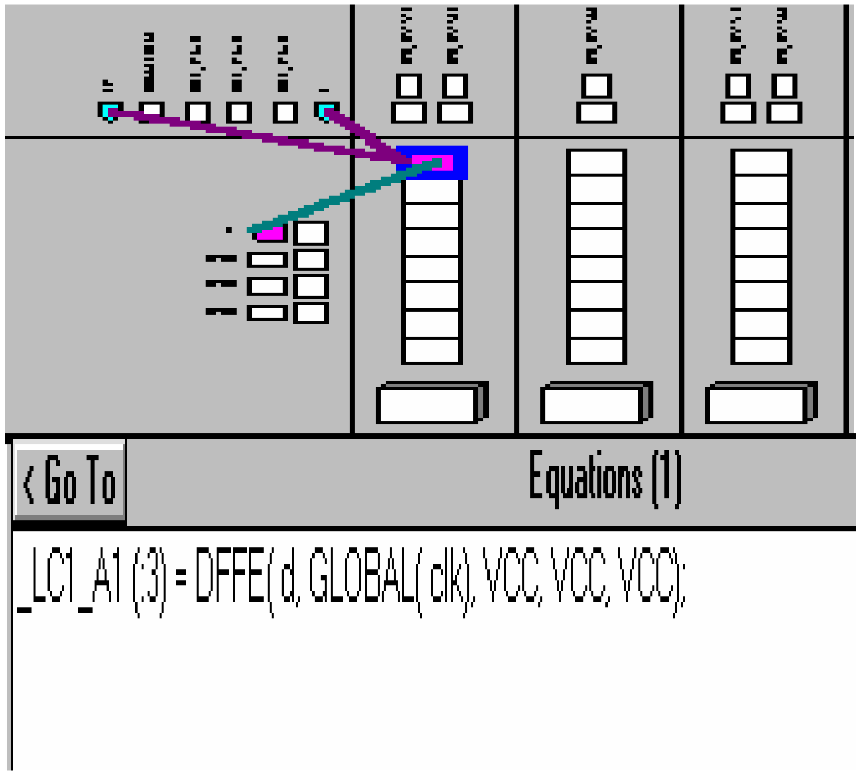
| | | Inputs | | Output |
|---|---|---|---|---|
| PRN | CLRN | CLK | D | Q |
| L | H | X | X | H |
| H | L | X | X | L |
| L | L | X | X | Illegal |
| H | H | ↑ | L | L |
| H | H | ↑ | H | H |
| H | H | L | X | Qo* |
| H | H | H | X | Qo |

* Qo = level of Q before Clock pulse

- Now you can see VHDL is very powerful, but if you don’t know what you are doing, you may not get what you want
- 现在,你已知道VHDL功能很强,但是如果你不知道你在做什么,还是得不到你想要的东西
  - e.g. you want a latch but actually you get a Flip-Flop
  - 比如,你想要一个锁存器,可是实际却得到一个触发器

# The other way of coding其他编程方法

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY tdff IS
PORT(clk, d: in std_logic;
     q : out std_logic);
END tdff;
ARCHITECTURE behaviour OF tdff IS
BEGIN
PROCESS
BEGIN
wait until clk = '1';
q <= d;
END PROCESS;
END behaviour;
```



_LC1_A1 (3) = DFFE( d, GLOBAL( clk), VCC, VCC, VCC);

## Compare IF-THEN-ELSE vs WATI UNTIL    比较IF-THEN-ELSE和WATI UNTIL

```
Entity test1 is
port (clk, d : in bit;
        q : out bit);
end test1;
architecture test1_body of test1 is
begin
process (clk)
begin
   if (clk = '1') then
   q <= d;
   end if;
end process;
end test1_body;
```

```
Entity test1 is
port (clk, d : in bit;
        q : out bit);
end test1;
architecture test1_body of test1 is
begin
process (clk,d)
begin
   if (clk = '1' and  clk'event) then
   q <= d;
   end if;
end process;
end test1_body;
```

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY tdff IS
PORT(clk, d: in std_logic;
     q : out std_logic);
END tdff;
architecture  behaviour OF tdff IS
BEGIN
PROCESS
BEGIN
wait until clk = '1';
q <= d;
END PROCESS;
END behaviour;
```

They are all the same DFF

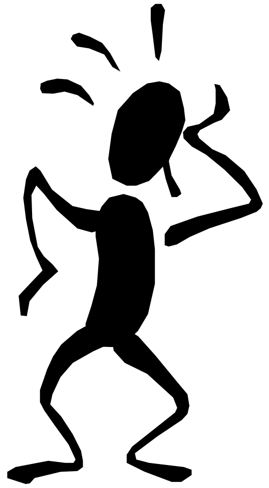
## Review

- Concurrent Statement for　并行描述
  - combinational logic (without Flip-flop circuit)组合逻辑(非触发器)
    - eg. decoder, multiplixer, multiplier, adder如解码器,多路器,加法器
- Process Statement for 过程描述
  - combinational logic (without Flip-Flop circuit)组合逻辑
  - Sequential logic (with Flip-Flop circuit)顺序逻辑
    - e.g. State machine, counters, controller如状态机,计数器,控制器

Combinational Circuit

Sequential Circuit

Process Statement



Concurrent Statement

Is this OK ????

```
ENTITY test1 IS
        PORT( clk, a,b,c : in bit;
        d           : out bit);
END test1;
architecture test1_body of test1 is
begin
d <= ((a and b) xor c);
end test1_body;
```

Combinational Circuit

## Concurrent Statement for Combinational Circuit

Is this OK ????

```
ENTITY test1 IS
        PORT( clk, a,b,c : in bit;
        d, e : out bit);
END test1;
architecture test1_body of test1 is
[illegible]
[illegible]
[illegible]
begin
d <= ((a and b) xor c);
e <= ((a or b) nand c);
end process;
end test1_body;
```

## Process Statement for Combinational Circuit

```
ENTITY test1 IS
PORT( a,b,c : in bit;
        d : out bit);
END test1;
architecture test1_body of test1 is
begin
process(a,b,c)
begin
d <= ((a and b) xor c);
end process;
end test1_body;
```

Is this OK ????

## Process Statement for Combinational Circuit

Is this OK ????

```
ENTITY test1 IS
PORT( a, b, c : in bit;
          d,e : out bit);
END test1;
architecture test1_body of test1 is
begin
process(a,b,c)
begin
d <= ((a and b) xor c);
e <= ((a or b) nand c);
end process;
end test1_body;
```

```
ENTITY test1 IS
        PORT( clk, a,b,c : in bit;
       d, e : out bit);
END test1;
architecture test1_body of test1 is
begin
if (clk'event and clk='1') then
d <= ((a or b) and c);
end test1_body;
```

Is this OK ????

No !! Sequential Statement must be within Process Statement

## Process Statement for Sequential Circuit

```
ENTITY test1 IS
            PORT( clk, a,b,c : in bit;
          d : out bit);
END test1;
architecture test1_body of test1 is
begin
process(clk)
begin
if (clk'event and clk='1') then
d <= ((a or b) and c);
end if;
end process;
end test1_body;
```

Is this OK ????

## Process Statement with mixing Sequential and Concurrent Circuit

Is this OK ????

```
ENTITY test1 IS
PORT( clk, a,b,c : in bit;
            d,e : out bit);
END test1;
architecture test1_body of test1 is
begin
process(clk,a,b,c)
begin
if (clk'event and clk='1') then
d <= ((a xor b) and c);
end if;
e <= ((a or b) nand c);
end process;
end test1_body;
```

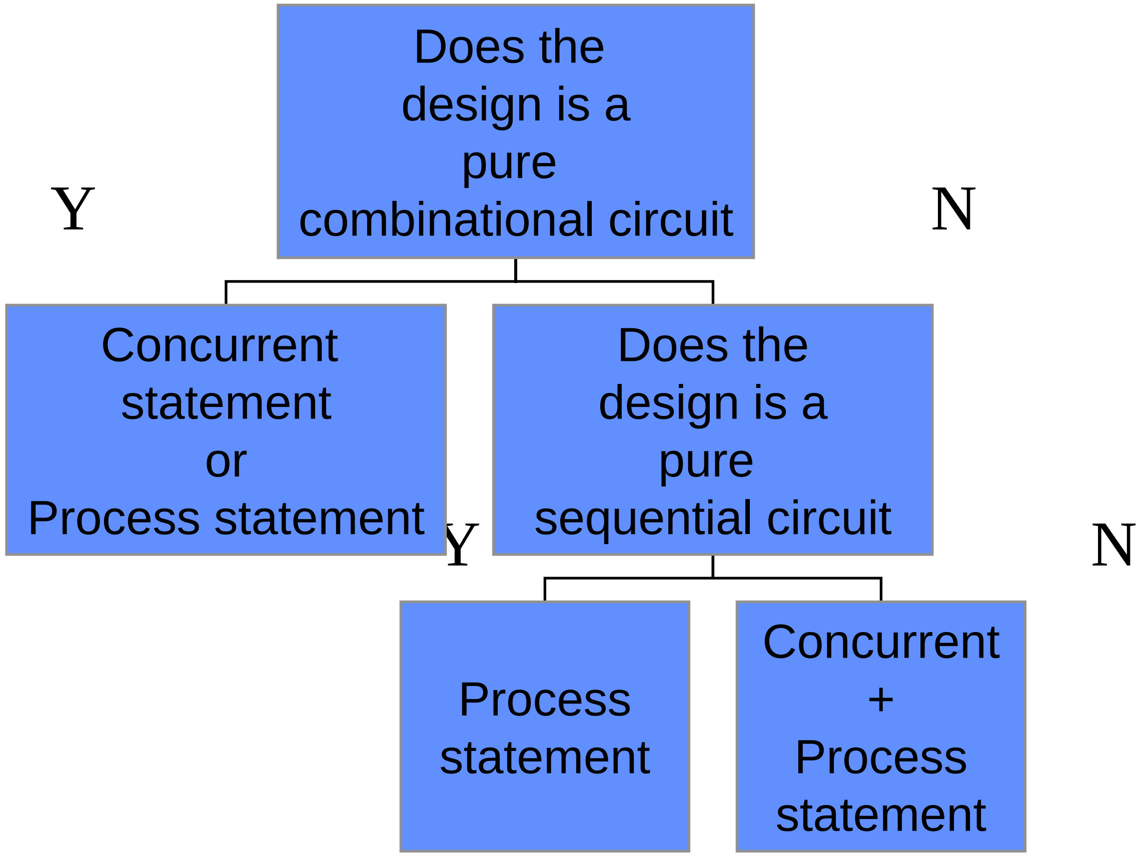
```
ENTITY test1 IS
PORT( clk, a,b,c : in bit;
            d,e,f : out bit);
END test1;
architecture test1_body of test1 is
begin
process(clk,a,b,c)
begin
if (clk'event and clk='1') then
d <= ((a xor b) and c);
end if;
if (clk'event and clk='1') then
e <= ((a or b) nand c);
end if;
if (a = '1') then
f <= (a or b);
end if;
end process;
end test1_body;
```

Is this OK ????

It is OK but not a good coding style

## Seat down an think of your design
## 坐下来想一想你的设计

Design Flow设计步骤


Does the design is a pure combinational circuit
Y
N
Concurrent statement or Process statement
Does the design is a pure sequential circuit
Y
N
Process statement
Concurrent + Process statement

## Compare of AHDL vs VHDL 比较

**AHDL**

**VHDL**

```
Variable

temp : dff


temp.clk = clk;
temp.d = data
out = temp.out
```

```
if (clk'event and clk='1')then
out <= d;
end if;
```

Both give me the DFF, and timing, functional同样给出DFF are all the same, there is no difference between!!!并且时域,功能都一样,没有不同!!!

# Closer look of the AHDL走近AHDL

Variable

temp : dff $\longleftarrow$ I want a DFF and named it as “temp”设计一个DFF,名为temp

temp.clk = clk; $\longleftarrow$ The DFF’s clock input pin is connected to “clk”其时钟端连到clk

temp.d = data $\longleftarrow$ The DFF’s data input pin is connected to “data”其数据端到data

out = temp.out $\longleftarrow$ The DFF’s out output pin is connected to “out”
其输出端连到out

You want a DFF and tell the Max+Plus II how to connect all the input and output pin, so Max+Plus II will follow your instruction and ***give you a DFF*** and ***connect all the pin*** in order to give you what you want. You will get a DFF today, tomorrow or 1000 years later, because you tell me you want a DFF
想要一个DFF,请告诉MAX+PLUSII所有输入输出引脚如何连接,它会按你的指令构成一个DFF并连接所有引脚.

## Closer look at the VHDL 走近VHDL

```
if (clk’event and clk=‘1’)then
out <= d;
end if;
```

When there is a “clk” transaction and stable at “1” then perform the following function, how to connect the pin, I don’t know!!!
当存在CLK处理且稳态为1时则执行下面函数 ,至于如何连接引脚,我不知道!

I want the “out” equal to the “d”, but how to connect the pin, I don’t know!!!!!!我要求out等于d,但是如何连接引脚,我不知道!

So now, base on what the function you want, the VHDL compiler find out that a DFF will meet your requirement, so it give you a DFF. Since the DFF is given to you by VHDL compiler, so you may get a DFF today, but get a latch tomorrow, or get a JKFF 1000 years later -- because you just tell me what is the function you want and how to implement it is really up to the VHDL compiler, as far as VHDL give you the function you want is OK!!
因而,在你要求的功能的基础上,VHDL编译器求得的DFF可满足你的要求,使你有一个DFF.由于DFF是VHDL编译器做出的,因此你今天得到一个DFF,明天有一个LATCH,一千年后有一个JKFF

## Conclusion of learning 学习总结

- Learned what is Combinational Logic/Circuits
- 学习了什么是组合逻辑/电路
- Learned what is Sequential Logic/Circuits
- 学习了什么是顺序逻辑/电路
- Understand the different between Concurrent Statement and Process Statement
- 理解了在并行描述和过程描述之间的差别

- Understand the usage of the 理解了以下方法
  - Concurrent Statement并行描述
    - for Combinational Logic 用于组合逻辑的
      - simple signal assignment statement简单信号分配描述
      - conditional signal assignment statement条件信号分配描述
      - selected signal assignement statement选择信号分配描述
  - Process Statement过程描述
    - for Combinational Logic用于组合逻辑的
    - for Sequential Logic用于顺序逻辑的
      - if-then-else structure for Latch对于LATCH的IF-THEN_ELSE结构
      - if-then-else structure for Flip-Flop对于Flip-Flop的IF-THEN_ELSE结构
      - wait until structure for Flip-Flop对于Flip-Flop的等待-直到结构

- All you learn is the basic concept of the VHDL language以上所学的都是VHDL的基本概念
- This is the Beginner VHDL Training Class
- 这是VHDL的入门培训教程
- The detail of the Circuit design will be covered at the Intermediate VHDL Training Class
- 电路设计的细节请看Intermediate VHDL 培训教程
- Advance VHDL language will be covered at the Expert VHDL Training Class
- 高级VHDL语言请看高手VHDL培训教程

## The most important thing of VHDL Language VHDL语言要点

" Tell me how your circuit should behave and the VHDL compiler will give you the hardware circuit that does the job"

"告诉我你的电路打算如何工作,VHDL编译器会给出满足要求的硬件电路"

- e.g. Latch and Flip-Flop is a good example
- 例如LATCH和FLIP-FLOP

# VHDL
# with
# Graphic Interface

具有图形界面的VHDL

## I have some bug free Graphic designs .

Q : I have some bug free Graphic designs, how can I re-use it with my new VHDL design file ?

A : You can convert your Graphic design to VHDL coding design manually, but

- it takes time
- human conversion error

OR

A : You can convert your new VHDL coding design to Graphic and integrate with your old Graphic design

- it is fast and easy
- Max+Plus II handle the conversion, so no error guarantee

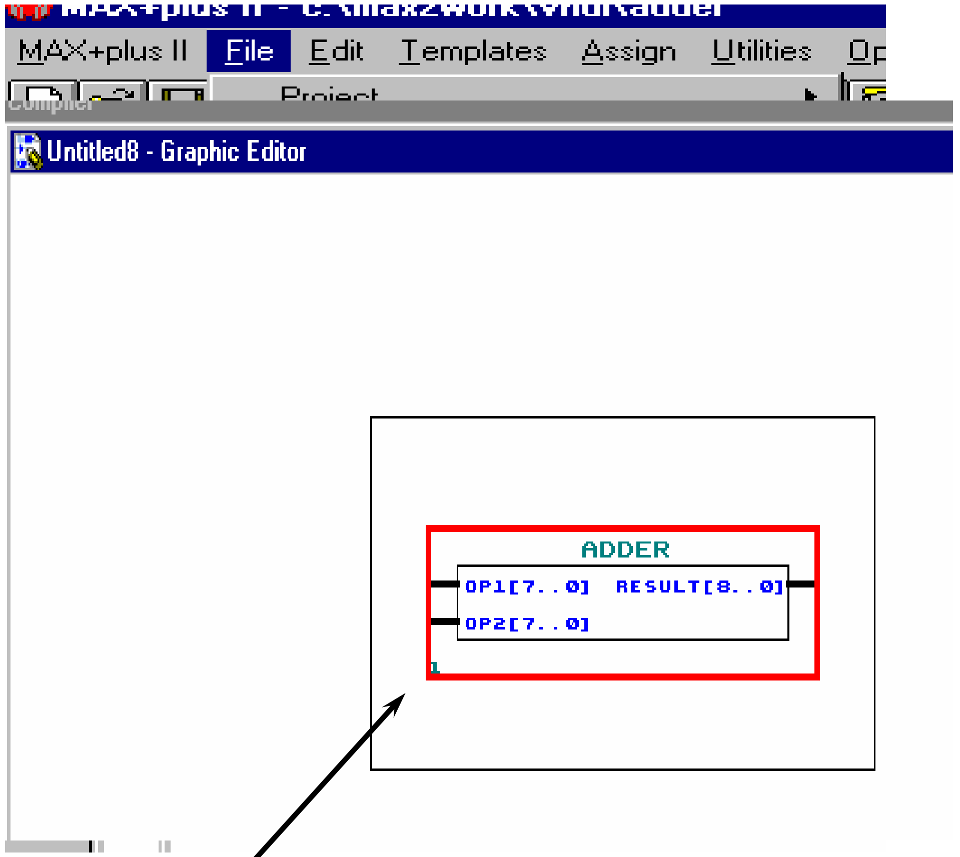
# Good but How ???

It is easy, just follow me

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY adder IS
PORT (op1, op2 : IN  UNSIGNED(7 downto 0);
          result : OUT INTEGER);
END adder;

ARCHITECTURE maxpld OF adder IS
BEGIN
  result <= CONV_INTEGER(op1 + op2);
END maxpld;
```


MAX+plus II File Edit Templates Assign Utilities Op
Project
Untitled8 - Graphic Editor
ADDER
OP1[7..0]
RESULT[8..0]
OP2[7..0]

Use it as normal Graphic Symbol

## See you at the Intermediate VHDL training class