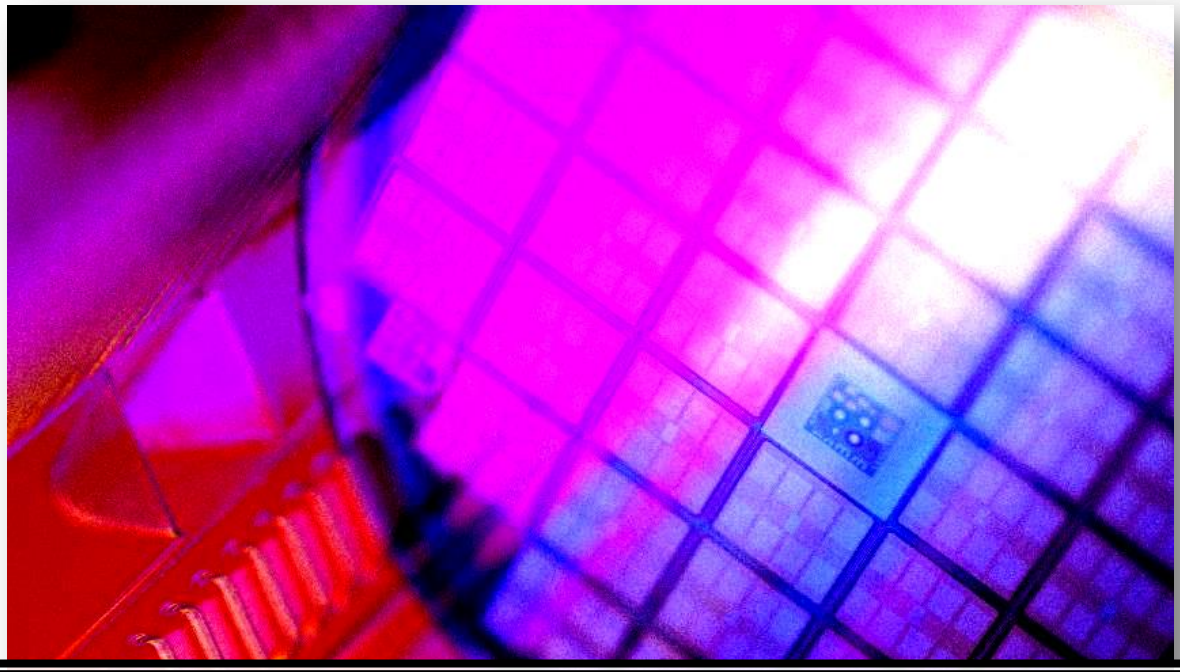
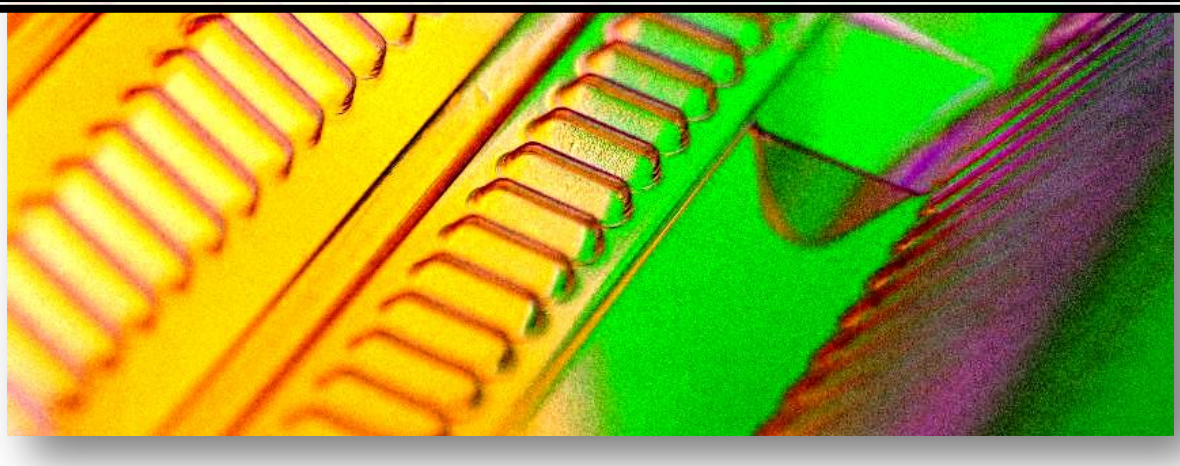
FEC100IT08-V1.0



zrtech

FPGA/CPLD 开发套件实验教程

--仿真，调试，设计篇



WWW.ZR-TECH.COM

实验八、丢掉你的晶振—MAXII 的 UFM Oscillator

实验目的：

通过这个基础实验，使用户了解 UFM Oscillator 的使用方法，并代替晶振完成设计。

实验原理：

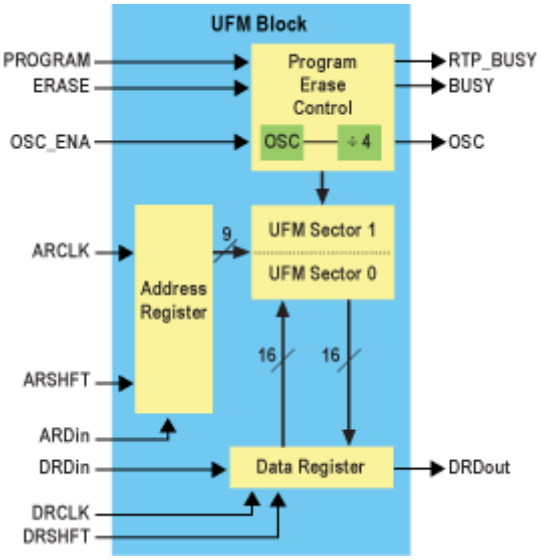
MAX® II 器件的内部振荡器 UFM Oscillator 是用户闪存(UFM) 的组成部分。内部振荡器能够满足很多设计对时钟的需求，避免了使用外部时钟电路。本节主要绍 UFM Oscillator 内部振荡器的例化及其使用。

大部分设计在正常工作时都需要时钟。利用内部时钟， MAX II 器件不再需要外部时钟电路。例如 ,内部振荡器能够满足 LCD 控制器、SM 总线控制器以及其他接口协议的时钟要求，还可以实现脉冲宽度调制器。这不但减少了元件数量，节省了电路板面积，而且还降低了系统总成本。

内部振荡器具有以下特性：

■ 未分频内部振荡器工作频率范围在 13.33 MHz 至 22.22 MHz 之间。振荡器输出频率 OSC 是未分频频率的四分之一，在 3.3 MHz 到 5.5MHz 之间。

■ 不需要例化 UFM 就可以例化内部振荡器。这可以通过使用 Quartus® II 软件的 MAX II 振荡器宏功能来实现。



内部振荡器是程序擦除控制模块的组成部分，该模块控制 UFM 的编程和擦除。数据寄存器保持和 UFM 之间传送的数据。地址寄存器保持数据读取和写入的地址。OSC_ENA 用于使能内部振荡器的信号。OSC 为 内部振荡器输出。振荡器没有工作时，该信号为低电平。实际测试中，发现对 Oscillator 的控制有 2 种方法，直接 Oscillator 使能拉高或者 Oscena Osc 连接成反馈电路，我们本节实验里采用直接使能拉高。

实验结果：

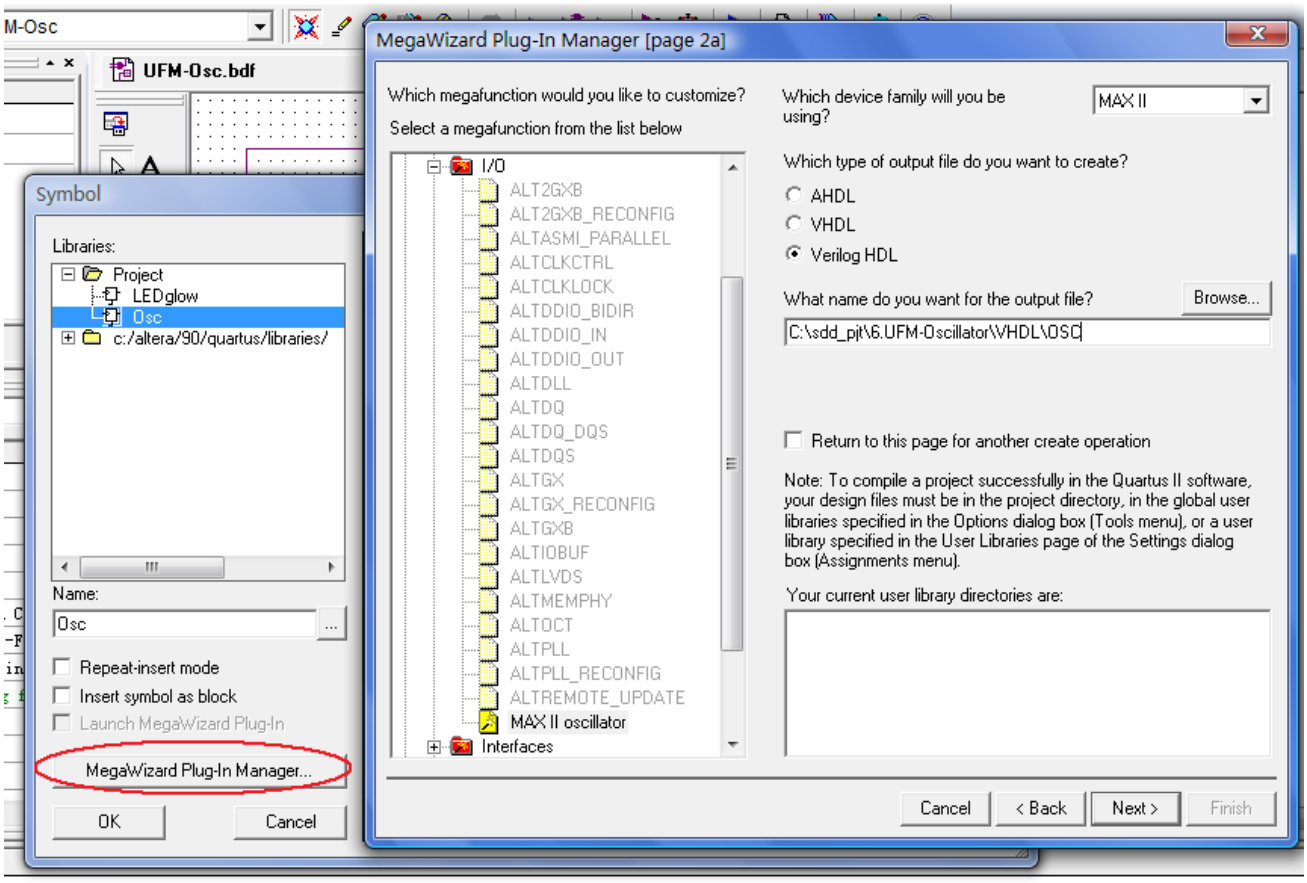
使用 UFM Oscillator 代替晶振，控制 COREC-240U 核心板上 LED 进行闪烁。

具体步骤：

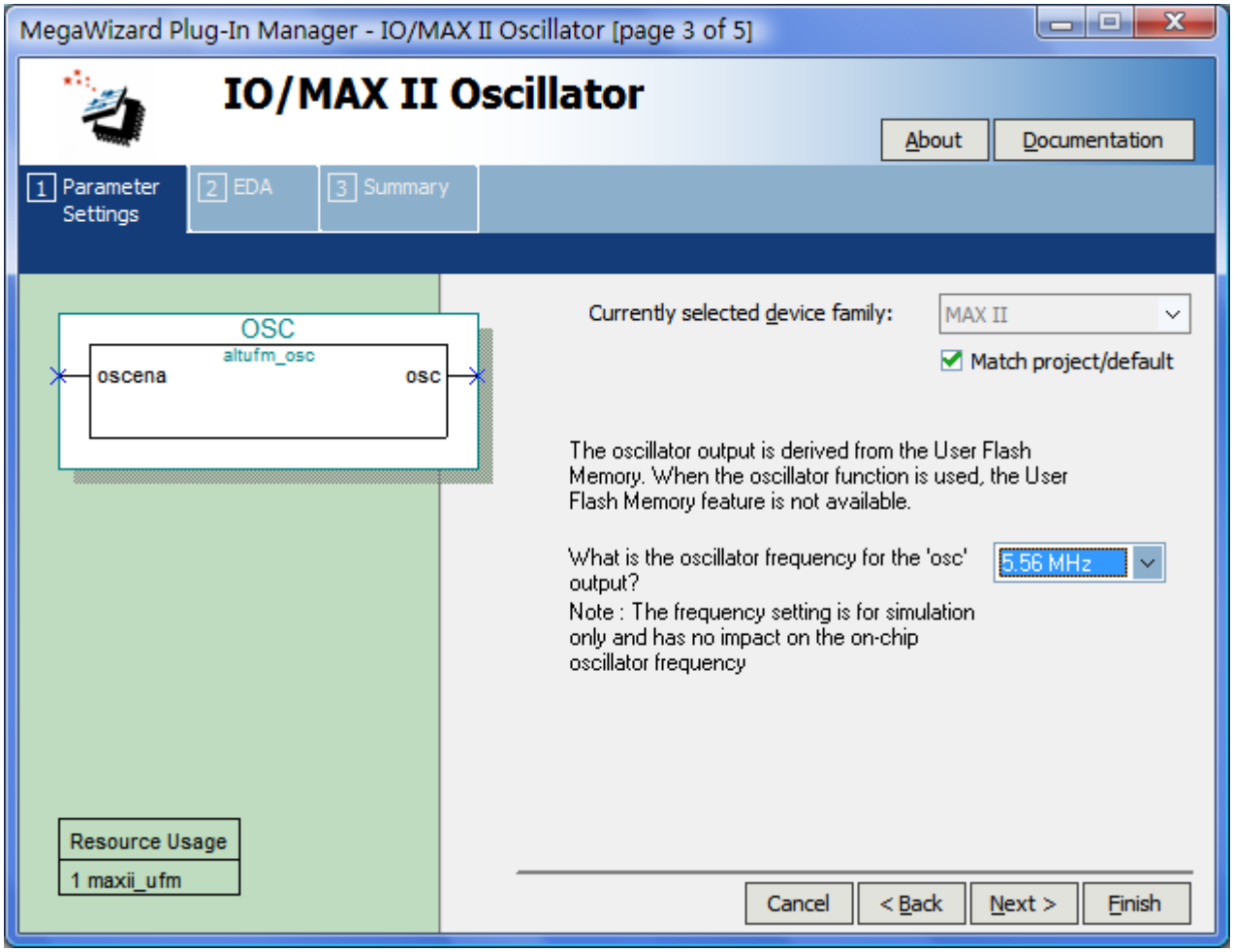
内部振荡器有一路输入 OSC_ENA 和一路输出 OSC。输入引脚 OSC_ENA 用于激活内部振荡器。激活后，输出上将产生频率为 3.3 MHz 到 5.5 MHz 的信号。如果振荡器使能信号 OSC_ENA 被驱动为低电平，振荡器输出保持低电平不变。

首先新建一个工程，创建一个 bdf 文件，保存设置为顶层文件，然后在 Tools 菜单中，单击 MegaWizard Plug-In Manager。

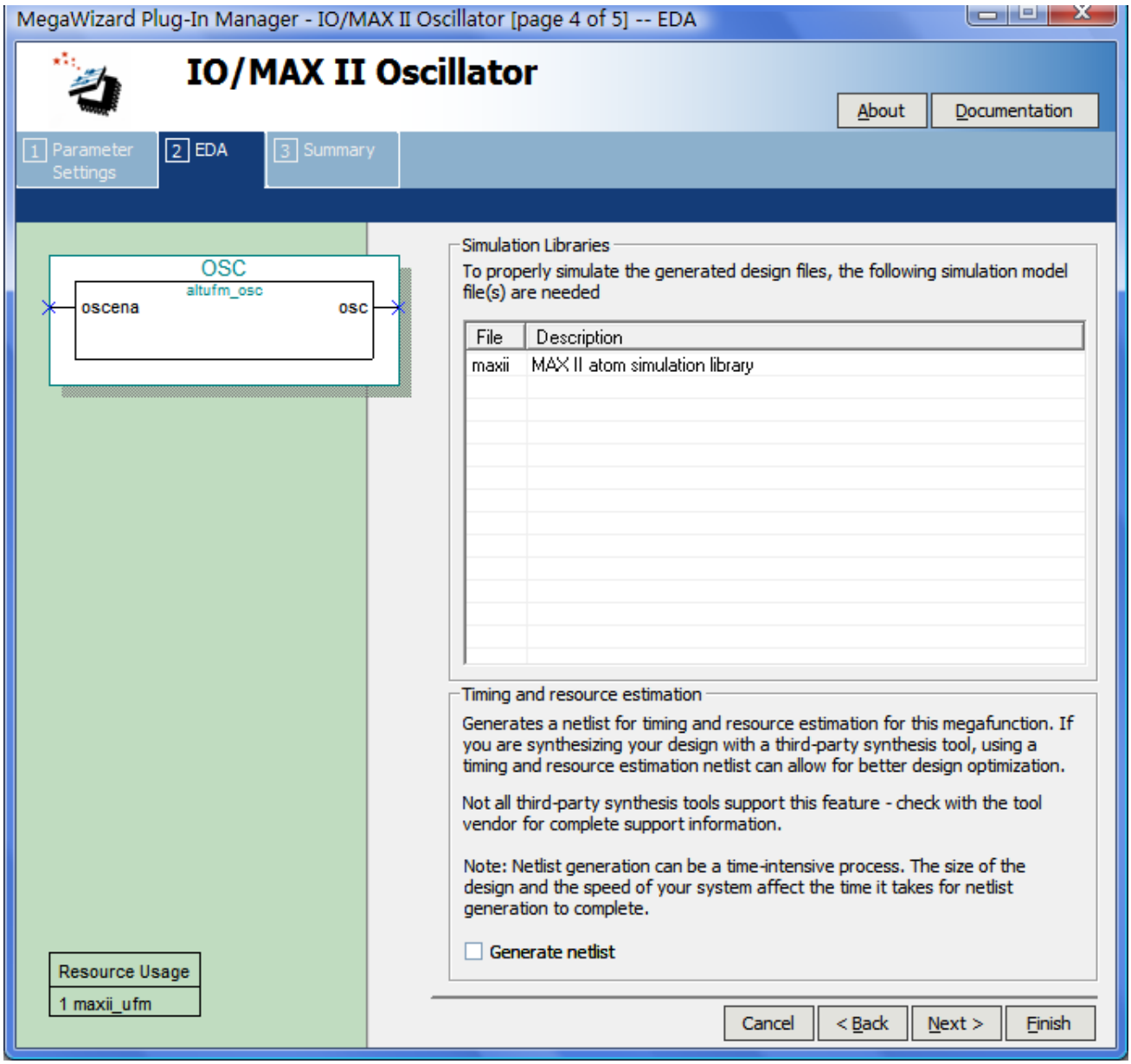
在 MegaWizard 插件管理器的第一页，选择 Create a new custom megafunction variation，然后单击 Next。在 MegaWizard 插件管理器的第 2a 页，选择 MAX II 在宏功能列表中，双击 I/O，然后单击 MAX II oscillator。键入输出文件名称，然后单击 Next。



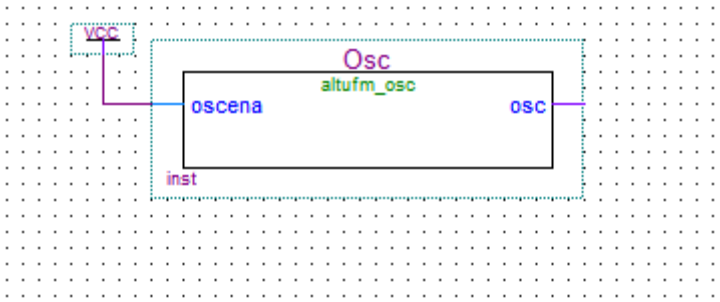
现在可以选择振荡器输出频率，请注意，频率设置只用于仿真，对片内振荡器频率没有影响。频率仅由 CPLD 决定，范围在 3.33 MHz 至 5.56 MHz 之间。



在 Simulation Libraries 中，列出了必须包含的模型文件，如图所示。单击 Next。



选择要产生的文件。单击 Finish。产生所选的文件。一旦将例化代码加入到文件中后，OSC_ENA 输入必须作为连线，这里我们直接分配逻辑值“1”，以使能振荡器。



该设计实例可以采用 EPM240G 或者其他 MAX II CPLD 来实现，这些器件都含有内部振荡器。其具体实现包括将振荡器输出分配给计数器，驱动 MAX II CPLD 上的 GPIO 引脚，以演示内部振荡器的功能。然后，驱动 LED 的 PWM，产生渐亮渐暗效应，演示内部振荡器。

Verilog 版本：

```
module LEDglow(clk, LED);
input clk;
output LED;
reg [25:0] cnt;
always @(posedge clk) cnt<=cnt+1;
wire [3:0] PWM_input = cnt[25] ? cnt[24:21] : ~cnt[24:21];       // ramp the PWM input up and down

reg [4:0] PWM;
always @(posedge clk) PWM <= PWM[3:0]+PWM_input;
```

VHDL 版本：

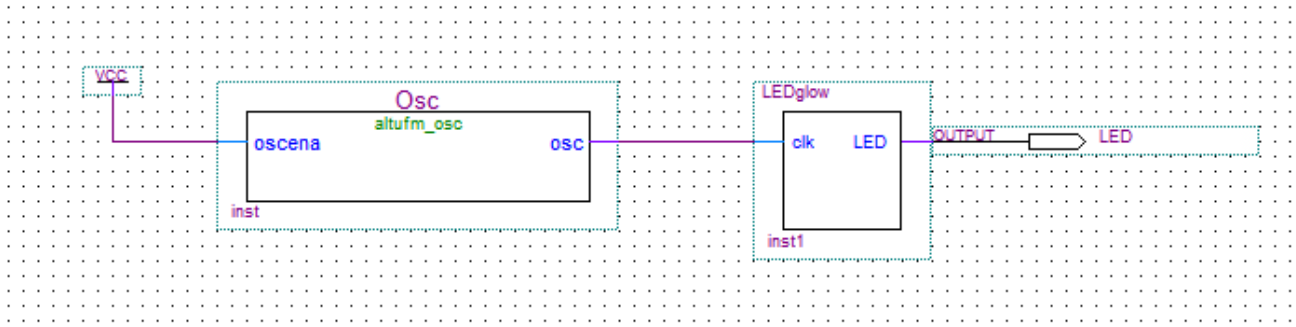
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity LEDglow is
port(clk:in std_logic;
        LED:out std_logic);
end LEDglow;

architecture fun of LEDglow is
signal cnt: std_logic_vector(25 downto 0);
signal PWM_input: std_logic_vector(3 downto 0);
signal PWM: std_logic_vector(4 downto 0);
begin
LED <= PWM(4);
PWM_input <= cnt(24 downto 21) when cnt(25)='1' else not cnt(24 downto 21);
--ramp the PWM input up and down
process(clk)
begin
if(clk'event and clk='1')then
        cnt <= cnt + 1;
        PWM <= '0' & PWM(3 downto 0) + PWM_input;
end if;
end process;
```

这个程序与第五课完全相同，即通过 PWM_input 输入不同的值来控制计数器的一次增加的幅度，从而改变分频比，从而改变 LED 的亮度，每当计数器计满后，就改变一次分频比幅度的趋势，从而实现 LED 的渐亮—渐暗的呼吸灯效果。但是由于内部振荡器大概在 3.33-5.56M 范围内，所以闪烁的 LED 的渐亮—渐暗周期要比使用 50M 晶振长得多。

编译生成 bsf 文件，加入到顶层中。



分配管脚，LED 在 corec-240U 的 38 号 IO 上，编译后，下载到目标板上。可以看到 LED 的渐亮渐暗效果。

实验总结：

MAX II CPLD 虽小，但是五脏俱全，且具有独特的内部振荡器，而且还是低功耗可编程逻辑解决方案。正如节实例所示，对于需要时钟的设计，该器件是非常好的选择，它不需要外部时钟电路，从而节省了电路板面积，降低了成本。非常适用于低成本的工业应用。

课后作业：

将Osc连接成反馈电路（即将使能端与输出端端接），再测试一下本节的程序，下载到板子上，发现LED闪烁的变化了吗？（利用Oscena反馈Osc，会出现二倍频的效果，即能产生接近10M的时钟频率哦！）

文档内部编号：FEC1001T08

编号说明：
首一字母：F-FPGA系列
首二字母：L-理论类 E-实验类 T-专题类
首三字母：C-普及类 Q-逻辑类 S-软核类
数字前两位：代表年度
数字后两位：同类文档顺序编号
尾字母/数字：C目录，T正文，数字表示章节号

修订记录

版本号	日期	描述	修改人
1.0	11.1.2	初稿完成	左超