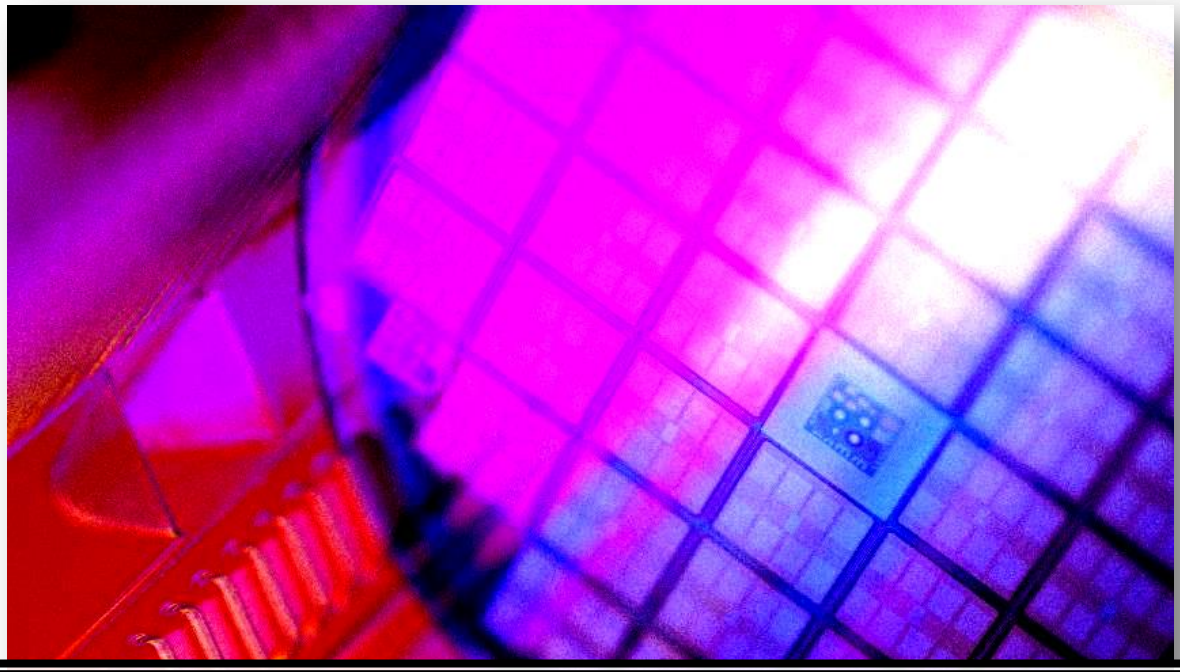
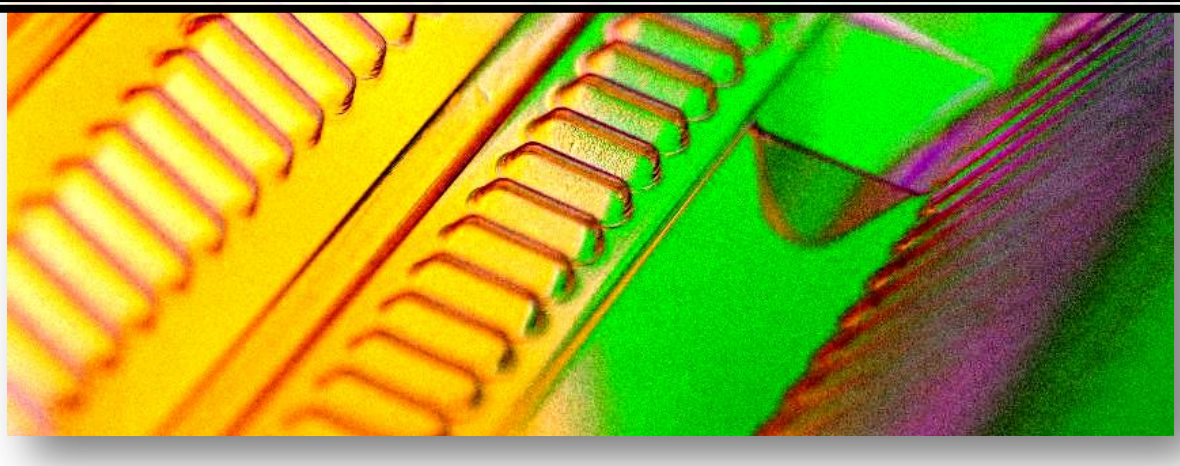
FEC100ITOG-V1.0



zrtech

FPGA/CPLD 开发套件实验教程

--仿真，调试，设计篇



WWW.ZR-TECH.COM

实验六、巧用 ISMCE 调试系统

实验目的：

通过这个基础实验 ,使用户了解 In-System Memory Content Editor 的使用方法 ,熟悉 SignaltapII 结合 In-System Memory Content Editor 协同进行调试。

实验原理：

QuartusII 提供工具实时修改存储器中的存储值，这就是 In-System Memory Content Editor。使用该工具可以实时更改 RAM 或者 Rom 中的数值，特别是在配置调整软件算法的系数时候非常实用。如滤波器参数调整，图像处理算法的阈值设置等等。因为 In-System Memory Content Editor 支持在线调试，这就意味着可以与 Signaltap 甚至你的硬件接收装置协同调试，获得满意的结果，而不必拘泥于理论值。使用该工具的要求，必须利用 Jtag，目前只支持实时修改常数（constant）、单口 RAM 以及 ROM。另外此工具最好只是用来调试，因为器件掉电重启以后 Memory 还是会加载其初始值（除非工程重新更改初始值并重新编译）。

实验结果：

使用 SignaltapII 与 Intent Memory Content Editor 在线修改片上的常量（constant）数据并在线调试测试工程。

具体步骤：

使用 ISMCE 的步骤十分简单，主要分为如下四步：

1）在例化 Constant、RAM 或者 ROM 的时候首先要使能 In-System Content Editting；

2）重新编译加载器件；

3）启动 In-System Memory Content Editor；

4）对常数或者存储器内容进行读写操作。

由于我们并还没有介绍 LPM 的具体使用方法，今天就以最简单的 constant-常量 LPM 来作为实例进行演示。首先我们新建立一个 quartus 工程，加入一段 verilog 或者 vhdl 程序：

Verilog 版本：

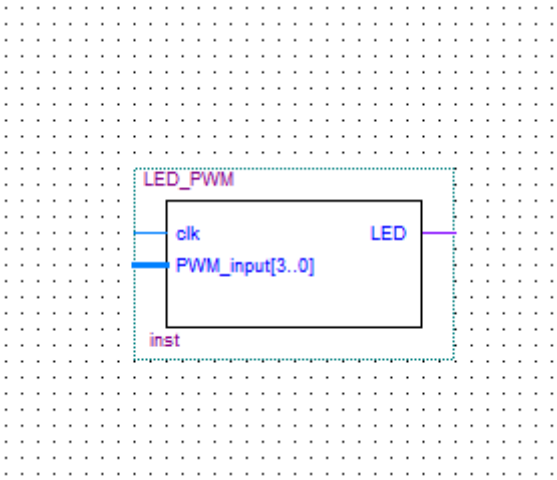
```
module LED_PWM(clk, PWM_input, LED);
input clk;
input [3:0] PWM_input;          // 16 intensity levels
output LED;
reg [4:0] PWM;
always @(posedge clk) PWM <= PWM[3:0]+PWM_input;

assign LED = PWM[4];
endmodule
```

VHDL 版本：

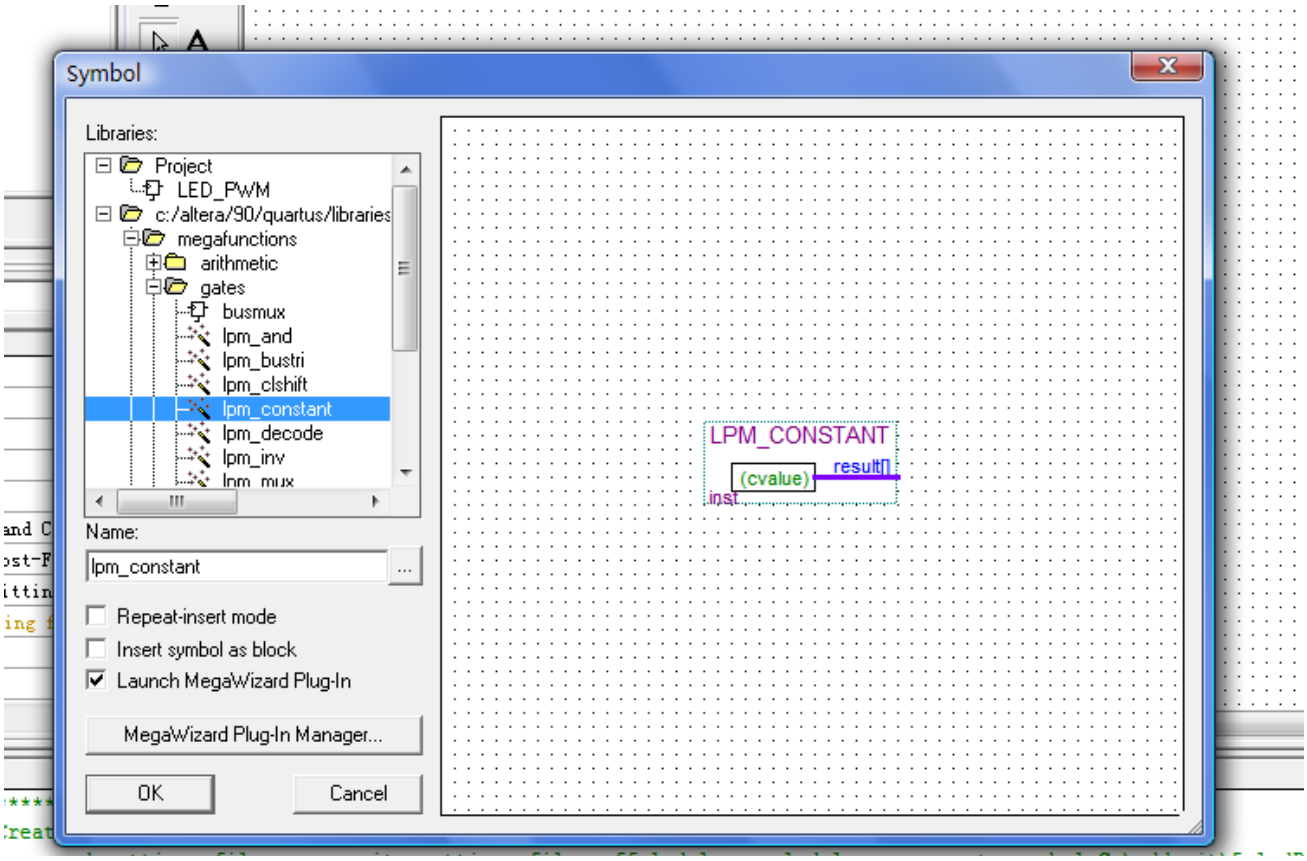
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity LED_PWM is
port(clk:in std_logic;
     PWM_input:in std_logic_vector(3 downto 0); --16 intensity levels
     LED:out std_logic);
end LED_PWM;

architecture fun of LED_PWM is
signal PWM: std_logic_vector(4 downto 0);

begin
LED <= PWM(4);

process(clk)
begin
if(clk'event and clk='1')then
     PWM <= '0' & PWM(3 downto 0) + PWM_input;
end if;
```

这个程序不难理解，就是上一节程序做了一个简单的修改，实现了一个外部可控的 LED 的 PWM 控制器，我们可以通过 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，共分 16 级。

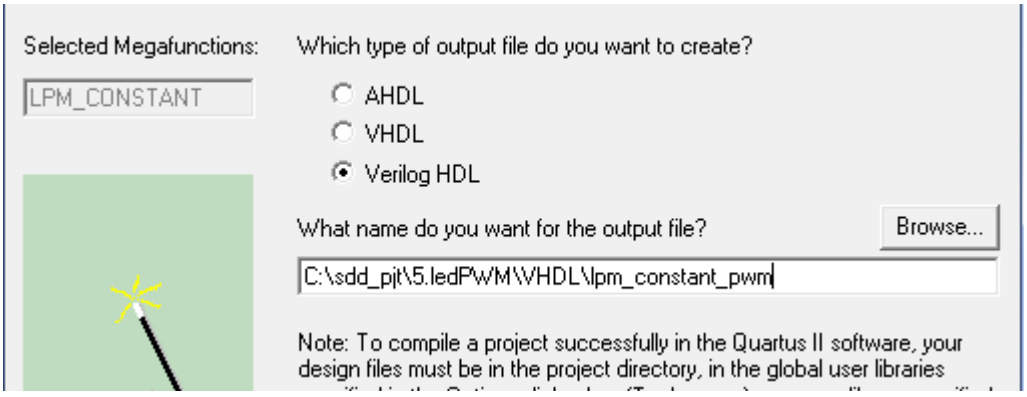
将以上文件编译通过并生成一个 bsf，并在 quartus 中新建一个 bdf 文件，将生成的模块加入原理图，并置为顶层文件。



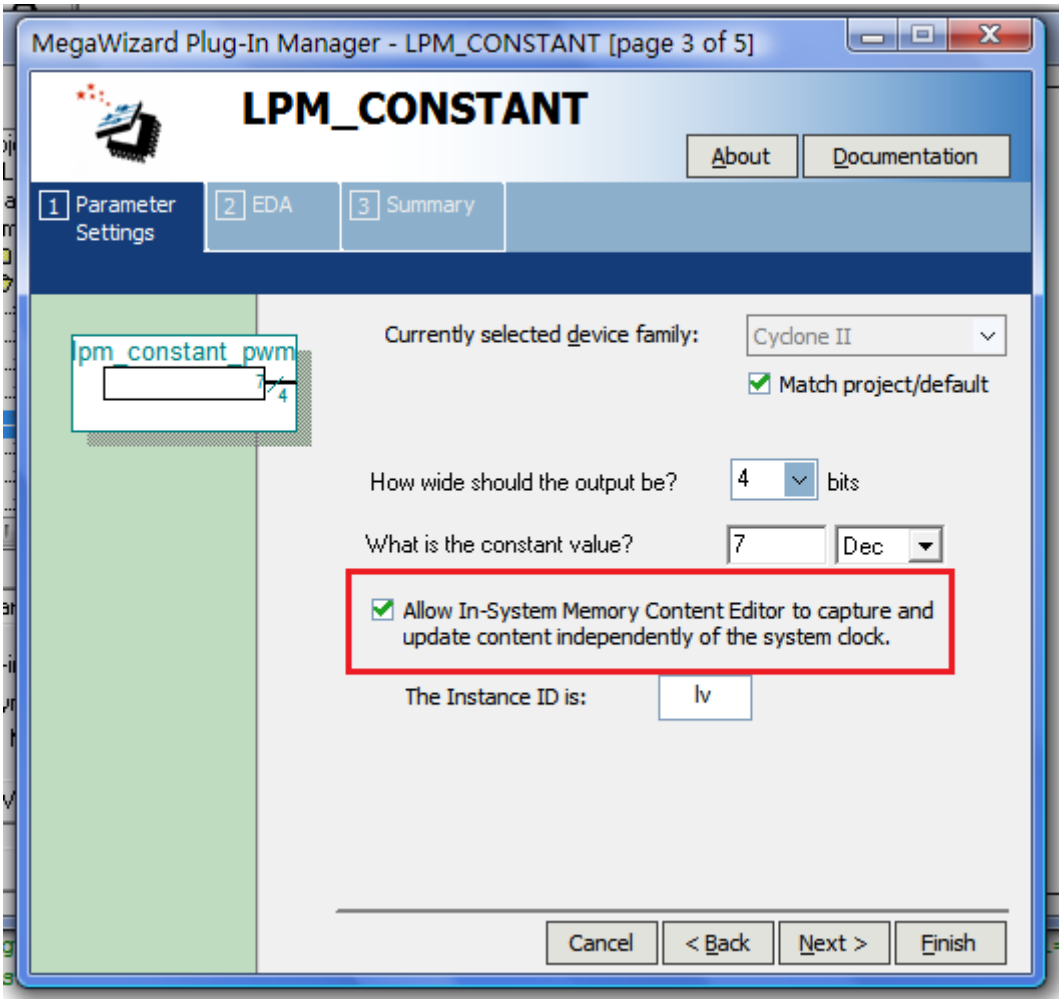
双击原理图的空白处，添加一个常量 LPM



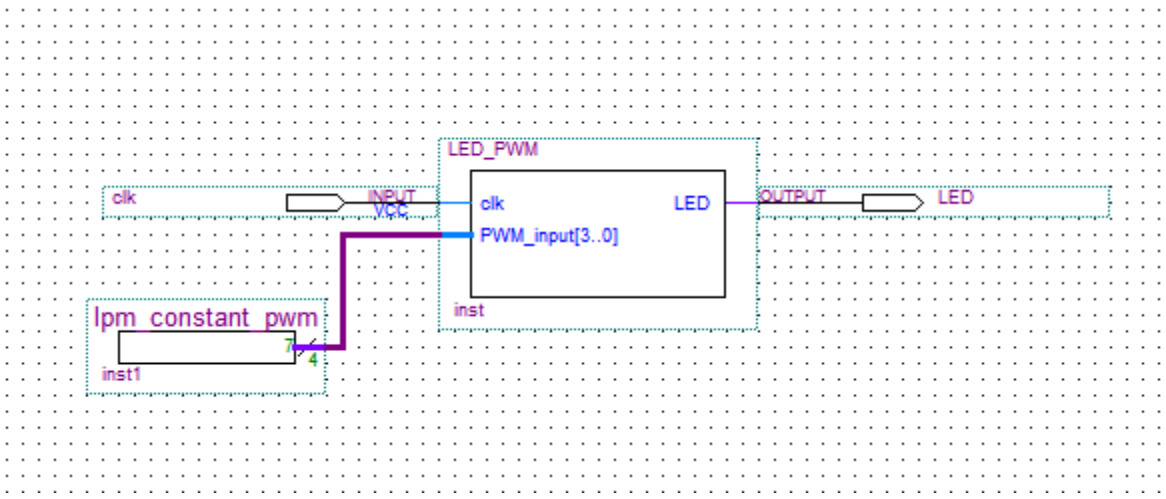
修改名称后，点击下一步



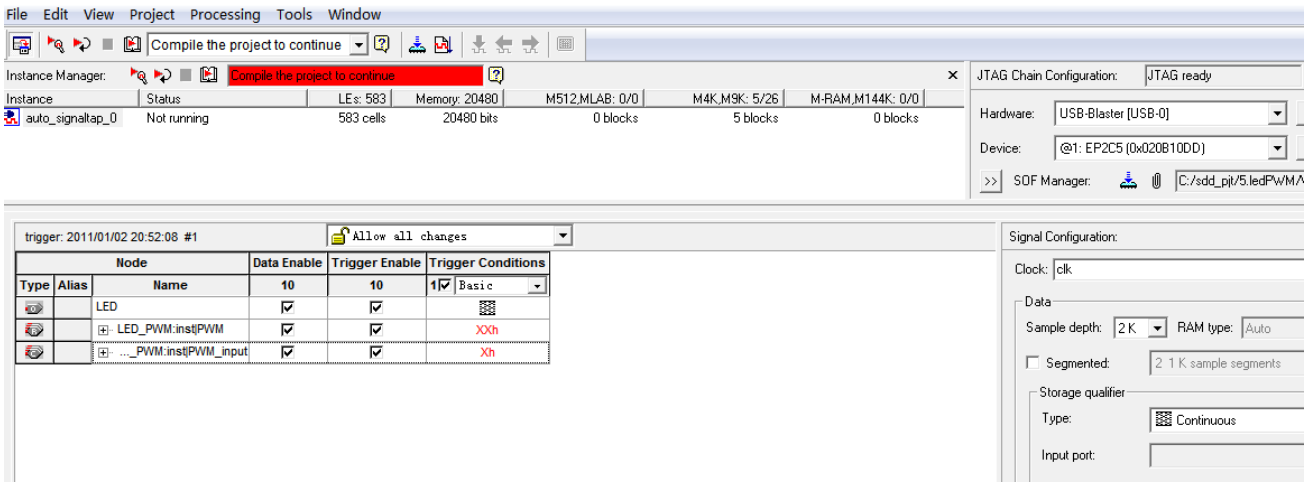
下图将该 4bit 的常数的初值设置为十进制的 7，并开启 ISMCE，当时你 ISMCE 的时候最好给实例取一个 Instance ID，因为在实际设计中可能存在多个实例需要区分，这里我们将该常数 ID 设为“lv”。



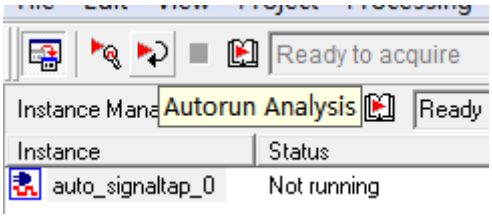
设置完毕后，如下连接好 Constant 与 IO 管脚：



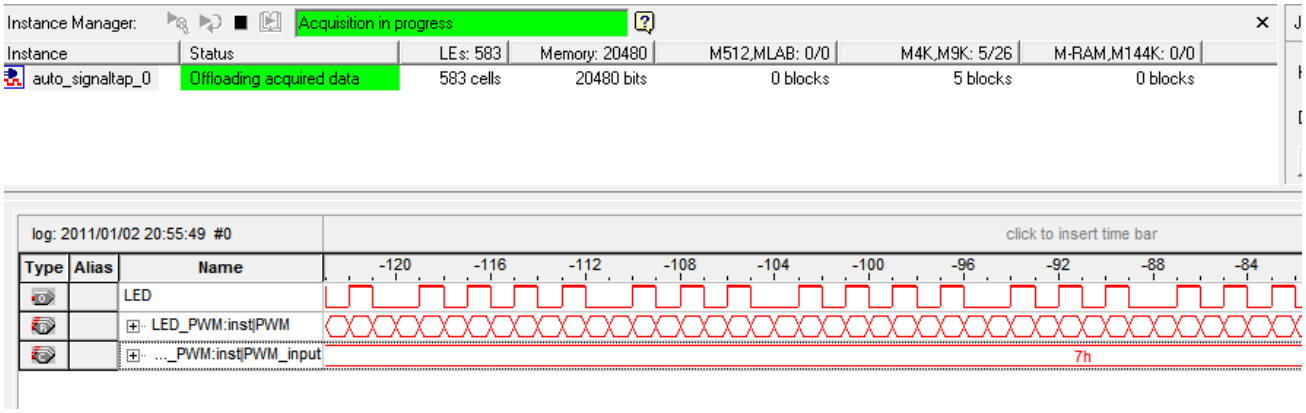
下面重新编译一下工程，分配好管脚，也可以用 tcl 文件进行分配。然后打开 signaltap，和上一课类似，添加时钟与观测信号。



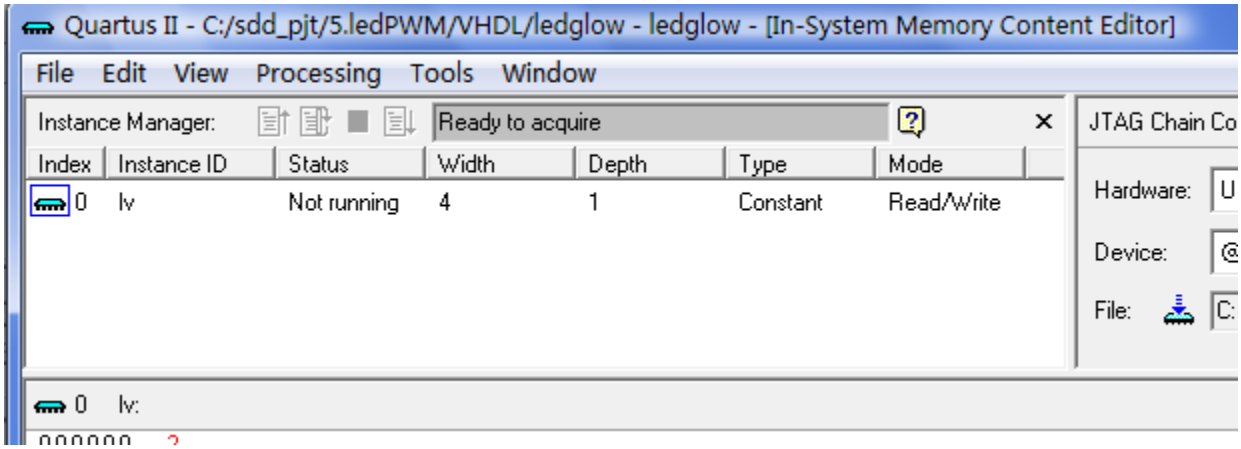
添加完毕后启用 signaltap，重新编译一下工程。完毕后下载程序，点击 Autorun ayalysis 开始采样。



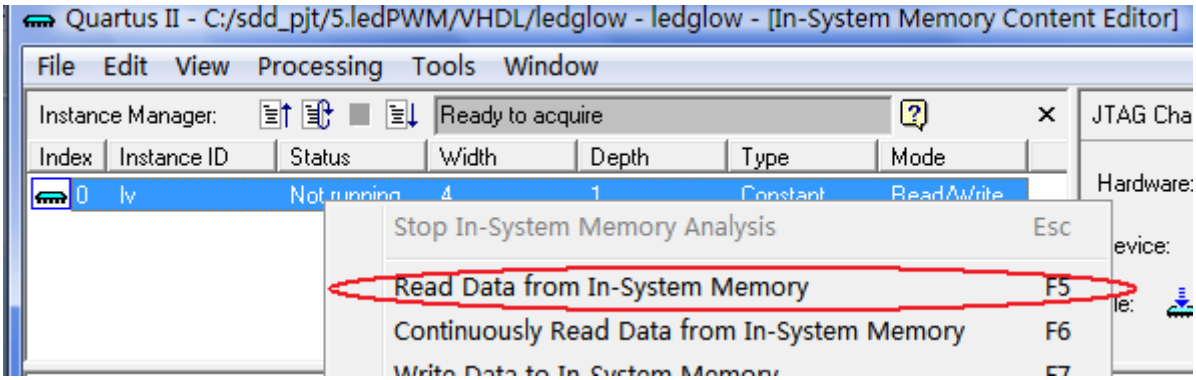
显然采到的数据中 PWM_input 为 7，这也是我们设置的值。再看看板子上的 LED，额….亮的比较惨淡



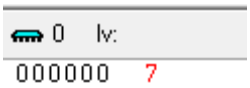
下面进入我们的正题，ISMCE，先离开 signaltap，在 Quartus 中选择“Tools”/“In system Memory Content Editor”选项，打开如图的 ISMCE 主窗口



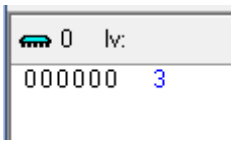
上图中包含 1 个实例，这就是我们例化的常数（其 ID 为 lv）。我们注意到这例化常数的时候我们设置的初始值为 7，但请注意当未执行“read data from In-system memory”命令的时候我们看到的值是一问号，我们点击 read data from In-system memory



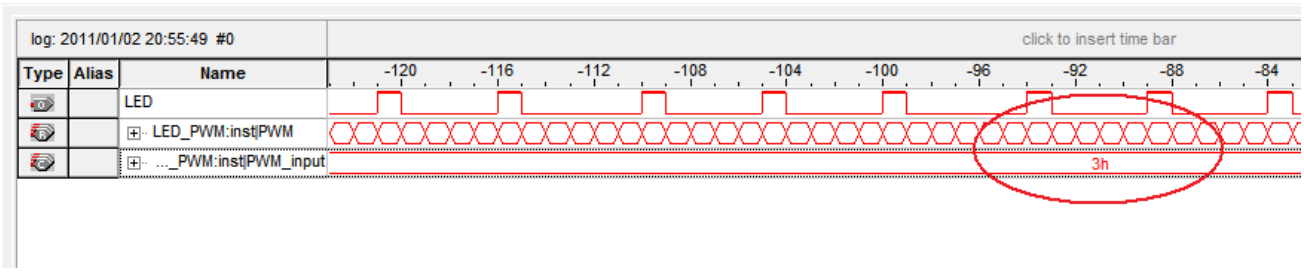
这时候可以在下面的窗口看到已经读出了 7



这时候我们将其编辑一下，改为 3



然后通过“Write data to In-system memory”命令写入到器件，快捷键为 F7，此时可以同时通过 SignalTapII 来观察数值是否已被更新。



再看看那 LED，这下是不是亮了很多呢。大家可以试着再改一些值，看看 signaltap 的数值与板上 LED 的亮度变化。

实验总结：

通过以上的讲解，相信各位两大调试软件signaltap，ISMCE很熟悉了。并可以用来调试自己的程序工程了吧！有的朋友也许会说，ISMCE也没什么用啊，我设置个按键，写段代码，也能改变某些参数啊。但是您是否想过，如果您的工程非常庞大，应用又非常特殊，比如算法某些参数的取值需要在板上仿真得到，而有的参数的位宽又有20位甚至更多，那么您调试的时候要多少个按键来控制这些参数呢？况且您也不希望您的实际项目中多出这些按键作为累赘吧！此外ISMCE还可以用来实时更新存储器的值，这也是非常有用的。大家在课后可以多操作几遍熟悉一下这些步骤，并可以查阅Quartus官方手册，把那些没有详细介绍的功能有个大致的了解。相信这两个工具会在您今后的设计调试中让您如虎添翼。

课后作业：

选择一个你感兴趣的例程，利用ISMCE结合signaltap调试它。

文档内部编号：FEC1001T06

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