暑期实践实验设计报告

基于 Nios ii 软核的频率计实现

姓　　名　李文涛

学　　号　320200928101

专　　业　电子信息基地班

年　　级　2020 级

第一部分 实验依赖

1.1 实验目的

本次实验项目使用 Quartus II、SOPC Builder（Qsys）、Nios II 等从零开始构建一个能够在 DE2-70 实验平台上运行的 uC/OS-II 操作系统的 Nios II 系统，并在此基础上实现同时适用于高频和低频的频率计。由于 DE2-70 实验平台实验平台有困难，后使用野火的 EP4CE10F17C8 开发板实现。

1.2 实验环境

Quartus II 13.0 + 野火征途 Altera EP4CE10F17C8

（1）Quartus II 13.0: 此版本的 Quartus II 较新，没有内置库，需要提前自行下载。对于野火的 EP4CE10F17C8，下载 Cyclone IV E 库即可；对于 DE2-70，需要下载 Cyclone II 库，由于太古老，安装库失败，建议使用 Quartus II 9.0（已内置库），但使用过程中出现问题较多

（2）野火征途 Altera EP4CE10F17C8：简单介绍此次开发板上我们需要用到的部分板块。

A. EP4CE10 主芯片：该芯片为开发板的主芯片，即 FPGA 芯片，其型号为 EP4CE10F17C8。该芯片拥有 10k 的逻辑单元，179 个可配置的 I/O 口，414kbit 的嵌入式 RAM 资源（每 9kbit 容量为一个块，每块为一个嵌入式存储单元，即有 46 个嵌入式存储单元），两个独立 PLL 锁相环，10 个全局时钟网络。

B. SDRAM：板载 SDRAM 芯片，SDRAM 是一个同步动态随机存储器。这里我们使用的 SDRAM 芯片型号为 W9825G6KH-6，容量为 256Mbit。在设计时不能超过其最大容量。

C. 下载接口 (JTAG)：FPGA 下载器通过该接口与开发板连接，用于程序的下载、固化以及调试。

D. 数码管：征途 Pro 开发板上配置了六位八段数码管，同时搭载了两块 74HC595 芯片，74HC595 具有串行输入，并行输出的功能。使用该芯片的四位控制信号即可输出 14 位的数码管控制信号，这样可以大大地节省 IO 口资源。

E. LED 显示灯：板载四个 led 显示灯（蓝灯），这四个 led 灯可以作为程序的状态显示灯。可以设计通过 led 灯来判断程序是否正确执行，在调试时可以起到辅助作用。

1.3 实验原理

本次设计实现的前提是构建 Nios II 系统作为一个简单的单片机结构，并在此基础上增添频率计。

Nios II 是 Altera(2015 年被 Intel 收购) 为其 FPGA 所设计的一种 RSIC 架构的嵌入式软核处理器。

RSIC：精简指令集。

软核处理器：使用 FPGA 的硬件资源搭建起来的处理器。

硬件处理器：芯片内集成的硬件电路所形成的处理器。

为了使频率计既可以测定高频又可以测定低频，本次频率计中一共使用了两种不同的测定方法：

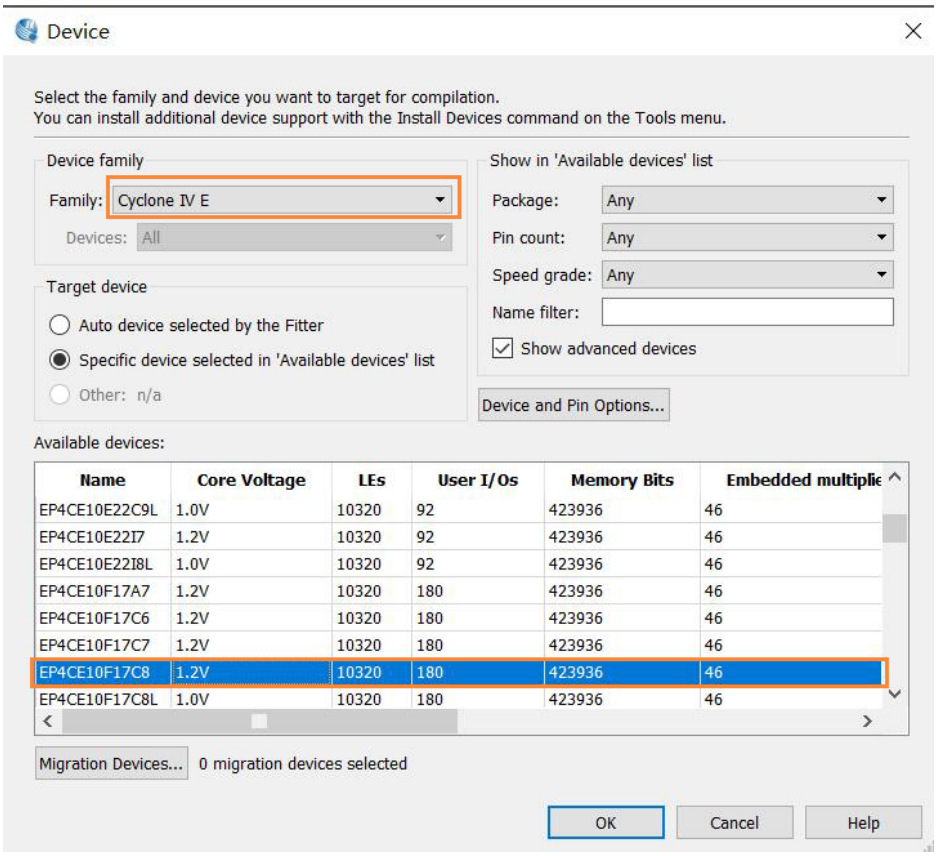
（1）频率测量法（计频法）：单位时间内，信号周期变化的次数。假设时间为 T，那么我们在 T 时间内周期数为 N，那么我们就可以计算出频率，即 f=N/T。此种方法适用于高频信号的测量，并且会出现 §1 的误差，对于低频误差会更大。

（2）周期测量法（计时法）：先测量被测信号的时钟周期 T，然后根据 f=1/T，求出被测时钟的频率，先测量第一个上升沿的时间，然后测量第二个上升沿的时间，两个相减，就是时钟周期 T，然后取倒数就是时钟频率。当频率比较低时，测量两个上升沿的时间是比较容易的，但是频率较高时，就比较难满足要求。因此这种方法适用于低频信号的测量。

第二部分　实验步骤

2.1　新建工程

FileNew Project Wizard，填写好工程的路径和名称点击 Finish，选好芯片，完成工程的创建。芯片型号选择如下图



2.2　搭建 SOPC 系统

A. ToolsQsys

B. 保存文件：FileSave

C. 设置系统时钟频率为 50MHz

D. 添加 Nios II Processor、jtag、sdram、PLL 等等，并将 IP 核连接起来（连接时钟信号、复位信号、数据总线、取指令端口、复位端口、中断信号），并更改对应的名字；添加片 System ID Peripheral 核，保持默认选项即可。

E. 添加 PIO 接口，分别命名为 frq_dt、frq_cs、ext_int。

F. 进行地址分配，点击 Qsys 主界面菜单栏中的“system”下的“Assign Base Address”。

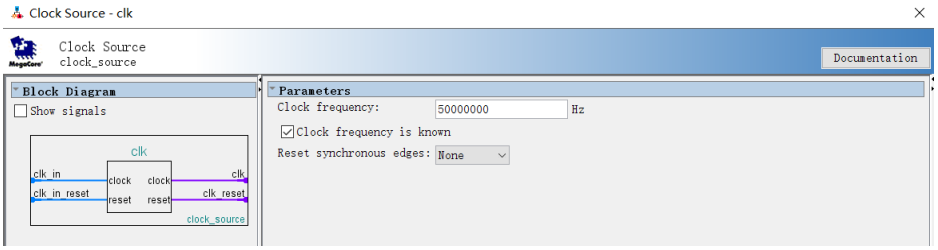


图 2.1　C. 设置系统时钟频率为 50MHz

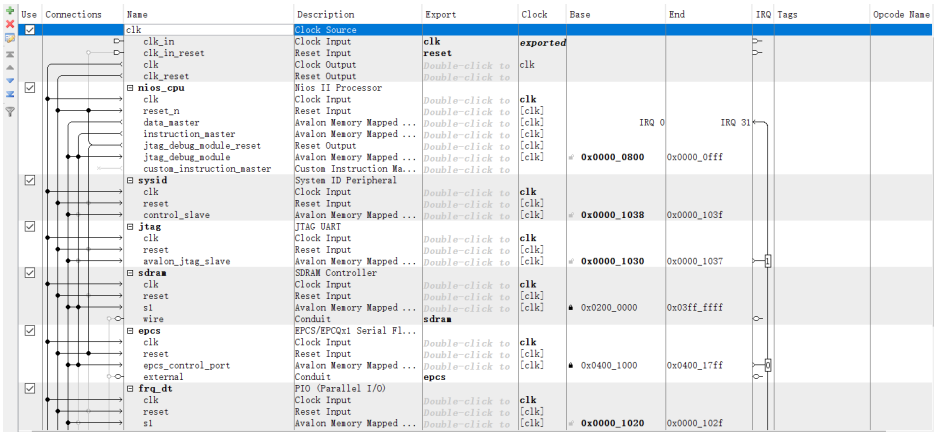


图 2.2　D. 添加 Nios II Processor、jtag、sdram、PLL

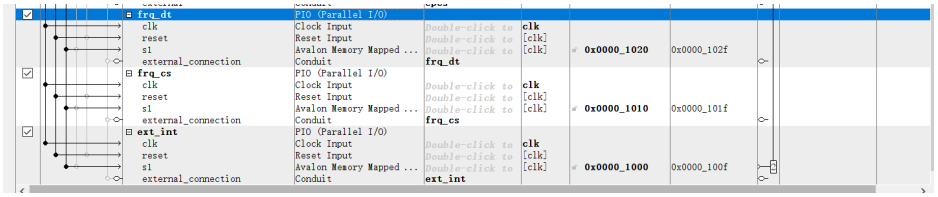


图 2.3　E. 添加 PIO 接口，分别命名为 frq_dt、frq_cs、ext_int

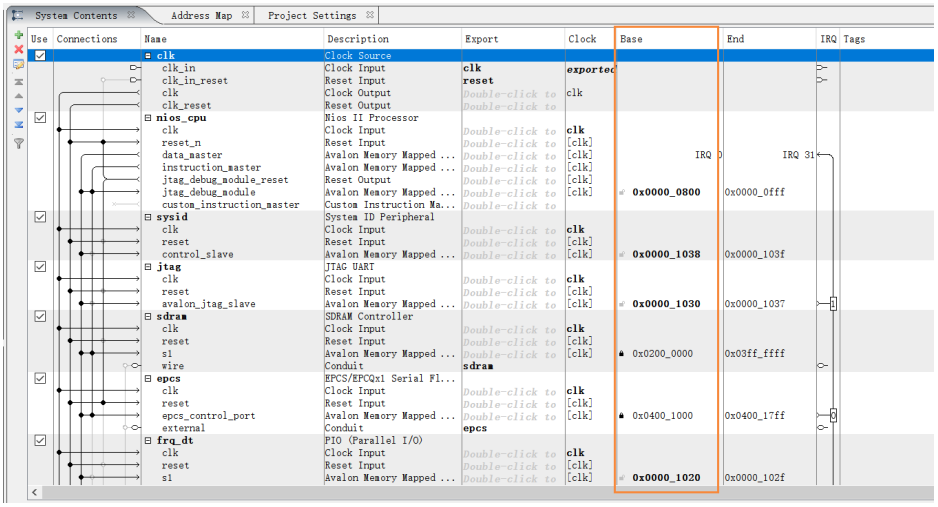


图 2.4　F. 进行地址分配

G. 分配中断号，在“IRQ”标签栏下点选“Avalon_xxx”和 IRQ 的连接点就会给对应设备添加中段号。

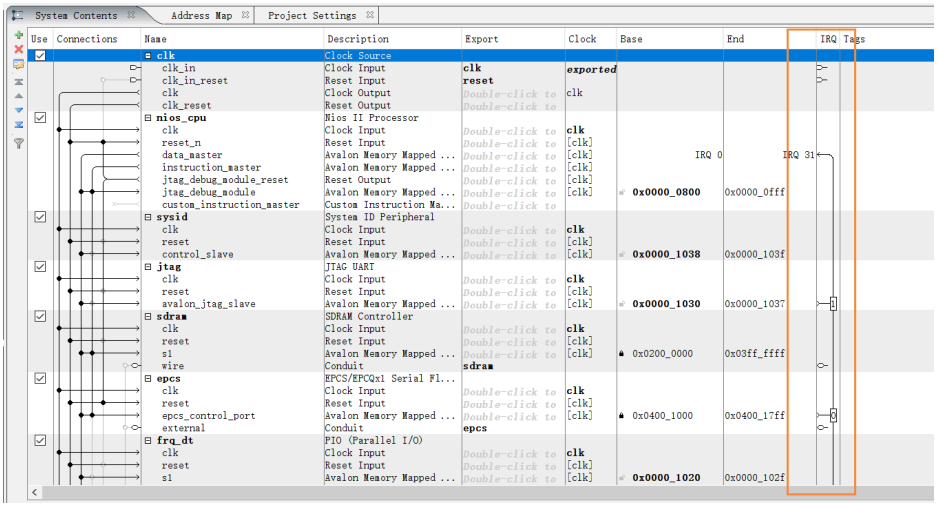


图 2.5　G. 分配中断号

H. 指定 NIos II 的复位和异常地址，从”System Contents”标签栏双击建立好的 cpu 进入 Nios II Processor 的配置界面，配置 Reset Vector 和 Exception Vector 为下图中所示，点击 Finish。

I. 点击 Qsys 主界面菜单栏中的”System”下的”Create Global Reset Network”。完成后会自动连接所有复位端口。

J. 生成 Qsys 系统，点选”Generation HDL”标签栏中 Generate 按钮生成 Qsys 系统。

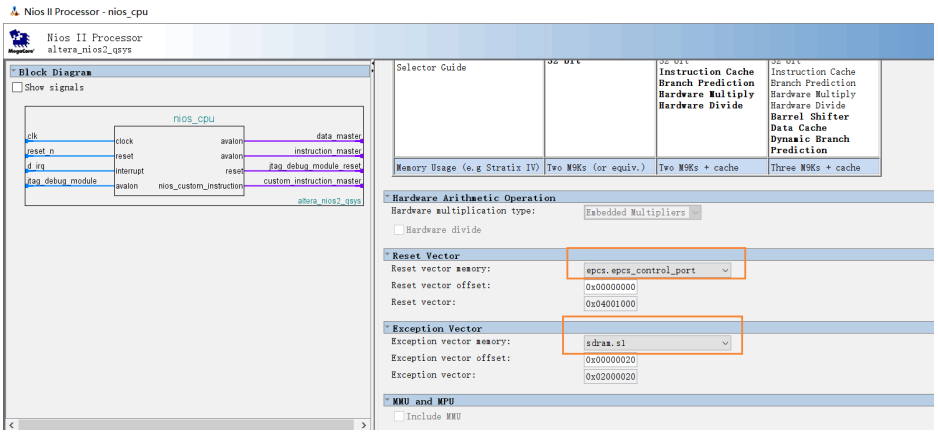


图 2.6　J. 生成 Qsys 系统

2.3　其它模块的设计

A. 时钟分频器

在该模块中，使用了两个寄存器 r 和 r1 来计数时钟周期。寄存器 r 是一个 25 位的计数器，用于计算时钟信号的周期。寄存器 r1 是一个 1 位的寄存器，用于控制时钟分频。使用 always (posedge clk or negedge n_rst) 声明了两个时序块。这些时序块在时钟上升沿和异步复位边缘触发时执行。在第一个时序块中，当异步复位信号 n_rst 为低电平时，计数器 r 被清零。否则，在时钟上升沿时，计数器 r 递增。当计数器 r 达到 25 位的最大值 24_999_999 时，计数器被重新设置为零。在第二个时序块中，当异步复位信号 n_rst 为低电平时，寄存器 r1 被清零。否则，在时钟上升沿时，如果计数器 r 达到 25 位的最大值 24_999_999，则寄存器 r1 取反。最后，使用 assign 语句将 r1 寄存器的值赋给输出端口 tm_scd。

```
module div25m(clk, n_rst, tm_scd);
input clk, n_rst;
output tm_scd;
reg [24:0]r;
always @(posedge clk or negedge n_rst)
begin
if( n_rst)
r<=25'd0;
else
begin
if(r==25'd24_999_999)
r <=25'd0;
else
r <= r + 1'b1;
end
end
reg r1;
always @(posedge clk or negedge n_rst)
begin
if( n_rst)
r1<=1'd0;
else
begin
if(r==25'd24_999_999)
r1 <=  r1;
else
r1 <= r1;
```

end

end

assign tm_scd = r1;

Endmodule

B. 锁存器模块

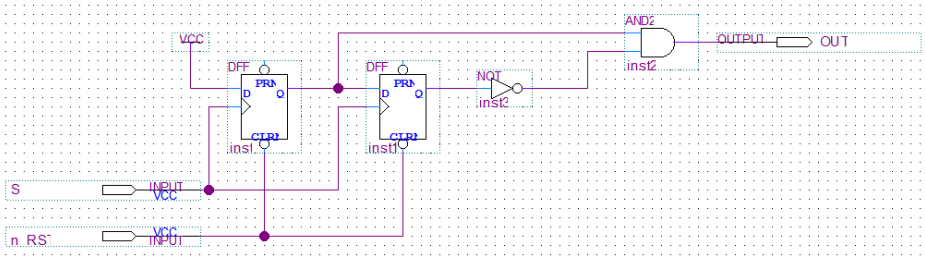


图 2.7 B. 锁存器模块

C. 频率选择器该模块实现了一个频率选择器（Frequency Selector），根据选择信号 sel 的值来选择不同的输入信号 pa 或 pb 输出到 pout。使用了一个三目运算符?: 来根据选择信号 sel 的值选择要输出的信号。如果 sel 为真，则输出 pa，否则输出 pb。使用 assign 语句将选择的输入信号赋给输出端口 pout。

module frq_sel(pa, pb, sel, pout);

input [23:0]pa, pb;

input sel;

output [23:0]pout;

assign pout = (sel) ? pa :pb;

Endmodule

2.4 整体模块设计

整个频率计的工作流程如下：

A. 输入信号经过 Latch 模块进行捕获，触发有效信号。

B. 有效信号触发 FRQ_SEL 模块，选择相应的计时器模块开始计时。

C. 选择的计时器模块开始计时，测量输入信号的持续时间。

D. 当计时器模块完成计时后，将结果保存到适当的寄存器中。

E. FRQ_SEL 模块再次接收有效信号，根据有效信号的状态选择另一个计时器模块。

F. 重复步骤 C 至步骤 E，以测量另一状态（高电平或低电平）的持续时间。

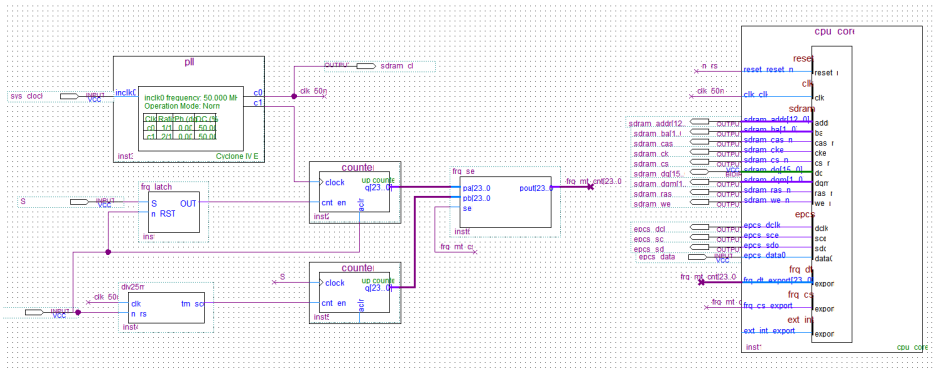


图 2.8　整体模块设计

2.5　软件部分设计

A. 启动 Nios II SBT，点击 Nios II Software Build Tools for Eclipse 打开 Nios II SBT for Eclipse 创建工程。

B. 在”SOPC Information File name”窗口中选择 kernel.sopcinfo 文件，将生成硬件配置信息和软件应用关联。

C. 编写软件程序——C 代码

```
#include <stdio.h>
#include <unistd.h>
#include <altera_avalon_pio_regs.h>
#include <sys/alt_irq.h>
volatile int counter = 0;
// 中断处理函数，在计数器每次溢出时触发
void isr(void* context)
counter++; // 每次溢出计数器加 1
int main()
alt_ic_isr_register(TIMER_IRQ_INTERRUPT_CONTROLLER_ID, TIMER_IRQ, isr, NULL, 0);
// 初始化计数器
IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0x07); // 启用计数器
IOWR_ALTERA_AVALON_TIMER_PERIODL(TIMER_BASE, 0xFFFF); // 设置计数器最大值
// 等待一段时间让计数器计数
usleep(1000000);
// 停止计数器并禁用中断
IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0x00);
// 计算频率
```

```
float frequency = (float)counter / 1000000;
printf(" 频率:
return 0;
```

第三部分　实验设计总结及反思

本次项目成功构建了一个 Nios ii 系统，并在野火的 EP4CE10F17C8 开发板实现了流水灯、数码管等简单例程。由于时间限制以及前期使用 DE2-70 开发板耽误较多时间，后面频率计只能自行课下设计，未能有机会上板测试验证。

这次暑期学校，通过使用 Quartus II 和 SOPC Builder（Qsys），我学到了如何进行基于 FPGA 的硬件设计和开发，也了解了如何选择和配置外设、设置处理器参数以及创建系统连接，这为进一步深入研究和开发 FPGA 项目打下了基础。这个项目为我提供了一个综合性的学习机会，涵盖了硬件设计、嵌入式软件开发以及调试技巧等方面。通过实践和解决问题，积累了宝贵的经验，并提高自己在 FPGA 和嵌入式系统开发方面的能力。ς