/**
*文件名： README.pdf
*作者： 李程鹏
*/

## 1. 基本描述：

这是 17~18 秋季学年期末， 实现的一个以 verilog 为硬件描述语言， 基于 mips 指令集， 运行在 Xilinx Nexyz4 开发板上的单周期 cpu.

主要功能模块包括: 控制器, 运算器, 显示器, 以及总线模块. 实现的 mips 指令一共有 20 条, 包括 8 条 R 型指令, 8 条 I 型指令, 2 条 J 型指令, 2 条自定义指令.

用户可以运用这 20 条 mips 汇编指令， 实现逻辑,算数,分支,循环以及输入输出等基本程序功能.

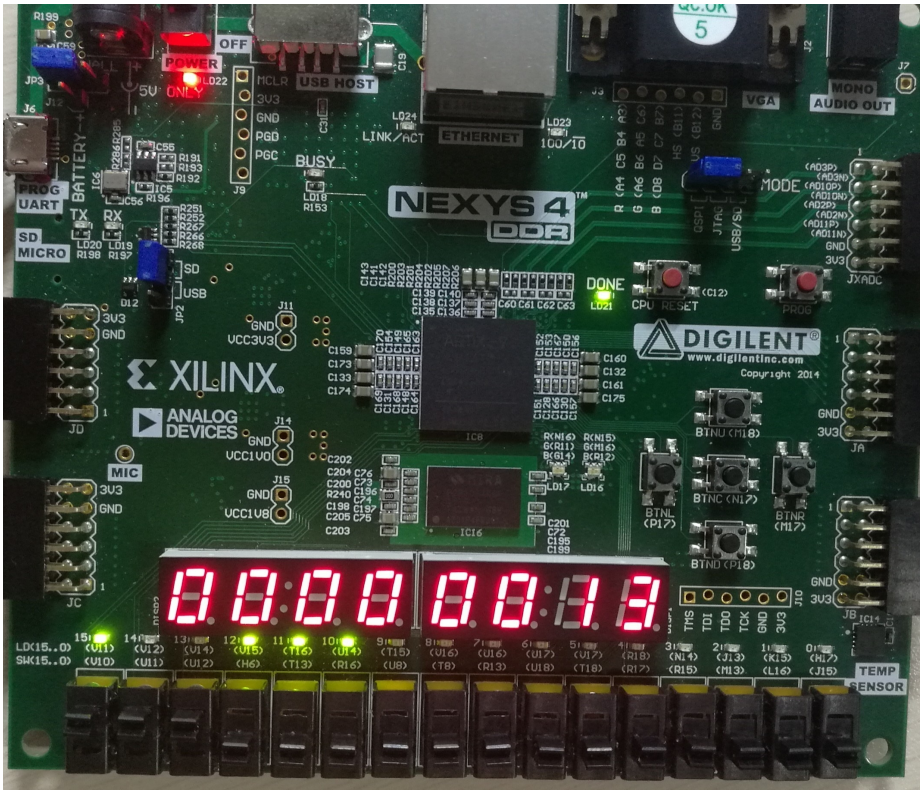
## 2. 使用手册：

### (1) 接口说明：

如下图所示的 N4 开发板， 最下面为开关(sw),显示数字的区域为七段数码管,2 者之间为 led 灯,开关和 led 灯从左到右编号为 15~0. 开关向上拨为 1,在下面时为 0.

我使用 sw15 作为模式选择开关， sw15 为 0 时,表示调试模式,sw15 为 1 时,表示时钟模式. sw14~sw13 用于控制显示到数码管的内容， sw14~sw13=00， 表示显示数据存储器的内容， 01 表 示显示寄存器堆的内容， 10 表示显示指令存储器的内容， 11 表示显示当前的 pc 值. 我设置的数据 存储器,寄存器堆,指令存储器的大小都是 32， 所以统一用 sw4~sw0 用于控制显示的位置.

led15~led0 用于显示当前指令的控制信号(判断表见 data/0.opAndFunc.txt). N17 按 钮用于调试模式时的单步， C12 按钮用于复位操作.




### (2) 举例说明：

比如用户输入一条： add $0,$0,$1 的指令， 初始时$0=10,$s1=3， 我们设置 sw15=0,sw14~sw13=01， 即为调试模式， 并显示寄存器内容. 当我们按下 N17， 就执行这条指令， 此时把 sw4~sw0 置为 0， 就可以看到 0 号寄存器显示的结果为 13， 并且 led 灯显示当前的控制信号.

## 3. 文件说明：

根目录下面 2 个文件夹：codes,data 和 pictures.

(1) 对于 codes:

sources 文件夹存放设计文件，sim 文件夹里面存放仿真文件，constrs 文件夹里面存放限制文件.

(2) 对于 data:

用于存放过程记录文件.

(a) 0.opAndFunc.txt：以类似 json 文件的格式存放需要实现的 20 条汇编指令，及其对应的 operand 字段和 funct 字段. 并判断出每条指令的控制信号，为方便阅读，每个字段均以 16 进制的形式保存.

(b) 1.codes.txt：测试这 20 条指令所设计的 mips 汇编程序

(c) 2.results.txt：对每一条测试指令执行后，相应的寄存器,数据存储器,pc 值的变化

(d) 3.interface.txt：接口说明

(3) 对于 pictures:

用于存放结果文件. clockMode 和 debugMode2 个文件夹分别存放时钟模式和调试模式下的结果.

## 4. 设计流程:

(1) 查阅 mips 文档(http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html)，把需要实现的指令按 R 型,I 型,J 型和自定义类型归类，分解出每条指令的 Operand(31:26)字段和 funct(5:0)字段，记录在文件 data/0.opAndFunc.txt 中.

(2) 设计包含所有实现指令的测试代码,记录到文件 data/1.codes.txt 中.

(3) 设计 cpu 需要的主要模块(文件的组织并不完全按照逻辑模块的形式):

(a) 指令存储器堆：存放用户需要指行的指令

(b) 寄存器堆：存放 0 到 31 号寄存器

(c) 数据存储器：存放 0 到 31 号数据存储器的内容

(d) 控制器：用于根据指令的 operand 字段，选择需要进行的操作

(e) 运算器：根据输入和控制信号，执行相应的计算，根据控制信号，决定运算结果的返回位置

(f) 显示器：对输入的数据，显示对应内容到 7 段数码管上.

(g) 总线：连接以上各个模块.

(4) 一条指令的执行流程:

我们以 add $0,$0,$1 为列，首先，在时钟上升沿时,根据当前的 pc 值，从指令存储器中取出指令，通过控制器分解出 operand 和 funct 字段，判断出为 add 指令，然后从寄存器堆中取出 rs,rt 和 rd 寄存器，将 rs 和 rt 寄存器放入运算器模块参与运算，结果写入数据存储器. 在时钟下降沿时，把结果写回 rd 寄存器，并更新 pc 值，开始寻找下一条要执行的指令.