实验四 多周期MIPS处理器

实验内容：

（1） 编写 Verilog 代码，完成《数字逻辑与处理器基础》理论课程布置的多周期 CPU 大作业，并综合实现、仿真并上板验证。

（2） 仿真执行代码请运行大作业中三.5 节给出的汇编程序：

MIPS Assembly

```
0       addi $a0, $zero, 5
1       xor $v0, $zero, $zero
2       jal sum
    Loop:
3       beq $zero, $zero, Loop
    sum:
4       addi $sp, $sp, -8 5 sw $ra, 4($sp)
5       sw $a0, 0($sp)
6       slti $t0, $a0, 1
7       beq $t0, $zero, L1
8       addi $sp, $sp, 8
9       addi $a0, $zero, 5
10      jr $ra
    L1:
11      add $v0, $a0, $v0
12      addi $a0, $a0, -1
13      jal sum
14      lw $a0, 0($sp)
15      lw $ra, 4($sp)
16      addi $sp, $sp, 8
17      add $v0, $a0, $v0
18      jr $ra
```

（3） 将 PC 寄存器值的最低 8bit，显示在 LED7~LED0 上；

（4） 利用 SW1 和 SW0 选择将$a0、$v0、$sp 和$ra 的最低 16bit，以 16 进制的形式显示在 7 段数码管上；

（5） 可以将时钟频率改为 1~10Hz 或者将时钟管脚绑定在按键上（注意按键消抖），来观察代码执行的情况，和仿真的情况进行对比。

（6） 实验报告中要分析单周期处理器的时序性能（最高工作频率，限制最高工作频率的关键路径是什么，路径上有哪些单元，哪些单元是最耗时的），FPGA 资源消耗情况等。

实验提示：

（1） 时序性能的分析对象是 MIPS 处理器，请建立一个不添加时钟分频或按键消抖单元的专门用来进行时序性能分析的版本。

（2） 在上层模块需要连线下层单元时，如 CPU 顶层模块中例化了 RegisterFile，需要获得 register_file1 中 RF_Data[2]这个信号时，在 Vivado 2017 及以后的版本中可以使用操作符“.”向上索引(Upwards name referencing)的办法：

```
wire [31:0] v0;
RegisterFile register_file1(.reset(reset), .clk(clk), ……);
assign v0=resgister_file1.RFData[2];
```

详见网络学堂上上传 Verilog 标准 IEEE1364 的 12.6 节 Upwards name referencing（p193），请留意一些老的 Vivado 版本对此不支持。

（3） 选择$a0， $v0， $sp 和$ra 连到电路的输出可以避免因缺少输出，工具将大量电路优化掉。

（4） 在 FPGA 中，寄存器（触发器和 RAM）的值可以在 Verilog 代码进行设置，具体可以参考网络学堂上上传的 UG901（综合和可综合的代码）中，第四章中“Initializing RAM Contents”中相关内容。务必注意在一般的 ASIC 设计中，不能使用这样的方法。

（5） 为了优化性能，Vivado 工具可以将层次化的设计打平，然后突破 HDL 的描述层级进行优化。跨层的信号，以及对信号的重命名会给调试带来不便，因此，可以将这个优化选项关掉。如下图所示。