1. 用perl产生verilog文件

主要是perl可以很方便支持类似1..3, a..c, A..C的 语法，对生成规律性的verilog语句很有帮助。 --
而python没有这样的语法。

1.1 t.pl

直接写perl脚本来产生代码；

- 缺点是打印语句要写print, 和后面的\n符号，有点点麻烦；
- 好处是直接上手perl语法，方便，容易debug

t.pl

```
@ar = (1..3, a..c, A..C);
foreach $i (@ar) {
    print "hello0${i}good\n";
}

print "\n";
@ar = reverse @ar;
foreach $i (@ar) {
    print "hello1${i}good\n";
}
```

执行效果：

```
>perl t.pl


hello01good
hello02good
hello03good
hello0agood
hello0bgood
hello0cgood
hello0Agood
hello0Bgood
hello0Cgood

hello1Cgood
hello1Bgood
hello1Agood
hello1cgood
hello1bgood
hello1agood
```

```
hello13good
hello12good
hello11good
```

1.2 vpl.pl

专门写一个统一的脚本来处理，通过读取分析源文件，来产生需要的代码。

- 以行首;分号开始为perl语句
- 其它情况全部当做是perl的print语句，不用自己再去敲print和\n这样的东西，相对来说还是比较简单，容易使用。

vpl.pl

```
if (scalar @ARGV < 1) {
    print "usage: perl vpl.pl xxx.vpl\n";
    exit;
}

$ivpl = shift @ARGV;
$opl = $ivpl;
$opl =~ s/\.\w+$/.opl/;
if (scalar @ARGV > 0) {
$vfile = shift @ARGV;
}
else {
$vfile = $ivpl;
$vfile =~ s/\.\w+$/.v/;
}

open(fh, "$ivpl") || die "can not open $ivpl";
open(ofh, ">$opl") || die "can not open $opl";

print ofh "open (vfh, \">$vfile\");\n";
while($line = <fh>) {
    chomp $line;

    if ($line =~ /^;(.*)/) {
        # is perl program
        $perl_line = $1;
        $perl_line =~ s/print\s*"/print vfh "/;
        print ofh "$perl_line" ."\n";
    }
    else{
        $line =~ s/\\/\\\\/g;
        $line =~ s/"/\\"/g;
        print ofh "print vfh \"$line\\n\"" .";\n";
    }
}
```

```
close fh;
close ofh;

# excute .opl file, and then delete tmp .opl file
$syscmd = "perl $opl";
$ret = system("$syscmd");
if ($ret == 0) {
    print "generate $vfile ok!\n";
    @ar = glob "$opl*";
    #print "will delete @ar\n";
    foreach $tmp (@ar) {
        unlink ($tmp);
    }
}

exit 0;
```