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 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

```
#!/usr/bin/perl
if (scalar @ARGV < 1) {
    print "usage: perl vpl.pl xxx.x [xxx.x.v]\n";
    exit;
}

$ivpl = shift @ARGV;
$opl = "$ivpl.pl";
if (scalar @ARGV > 0) {
$vfile = shift @ARGV;
}
else {
$vfile = "$ivpl.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";
    foreach $tmp (@ar) {
        unlink ($tmp);
    }
}

exit 0;
```

1.2 example

```
localhost /home/user01 perl ./vpl.pl a.x
generate a.x.v ok!
```

file: a.x

```
module a(
    input clk,

; for ($i=0; $i<10; $i++) {
    input data_$i,
;}

    output data_o
);

; for ($i=0; $i<10; $i++) {
;     $j = 10 - $i;
//  10 - $j = $i;
; }


endmodule
```

output a.x.v:

```
module a(
    input clk,

    input data_0,
```

```
    input data_1,
    input data_2,
    input data_3,
    input data_4,
    input data_5,
    input data_6,
    input data_7,
    input data_8,
    input data_9,

    output data_o
);

//  10 - 10 = 0;
//  10 - 9 = 1;
//  10 - 8 = 2;
//  10 - 7 = 3;
//  10 - 6 = 4;
//  10 - 5 = 5;
//  10 - 4 = 6;
//  10 - 3 = 7;
//  10 - 2 = 8;
//  10 - 1 = 9;


endmodule
```