VeriSim_Compiler
以Logisim为目标平台的verilog编译器

孟繁瑞	ZY2006239
郝 浩	ZY2006202

设计背景

《计算机组成原理》是计算机类各专业本科生必修的硬件课程中的重要核心课。在北航，**计算机组成原理**、**操作系统**、**编译原理**，因其**高标准、高难度、高强度**的课程设计，一直是六系众人肩上的三座大山。自有记载以来，无数仁人志士日以继夜、夜以继日，挑灯夜读、埋头苦干。
说回《计组课程设计》，作为目前专业分流后的第一门核心专业课的动手环节，目前采用的是Logisim-verilog-mips的理论预习顺序 & logisim-verilog 的CPU构建顺序实现的。作为面向特定领域的语言（Domain Specified Language），verilog实现了将特定的语言对应到硬件模型上。如果学习了verilog，并且有一定的实践经验的编程人员能强烈地感受到，verilog和C/C++等编程语言有着本质且明显的差别。但对于初学者而言，仅在课堂上听高老板一句高屋建瓴的“编程时做到心中有电路”，难以对应到实践中。
因此课程设计中先用logisim完成简单电路的设计和单周期CPU的设计，帮助学生建立电路构造的 “feel” 后，再进行verilog的较复杂CPU设计。这一安排是很合理且成功的，但是在担任计组朋辈助教期间，我负责辅导的同学的课程进度多卡在**Logisim_CPU——Verilog_CPU**之间，说明对于一些同学而言，有限时间内思维方式的快速转换仍有难度。是否有其他方式可以解决这一问题，在这门课上我想到一个解决方案，**反其道而行之**，基于可综合的Verilog生成Logisim电路。

相关工作调研

硬件描述语言verilog开发效率很低。于是，出现了HLS，chisel，CHDL等，要么是期望用软件编程思路编写硬件如高级综合HLS,如Matlab,要么是代码转换如chisel，bluespec,spinal hdl,Chdl等函数式编程，前一种主要由编译器自动推导生成硬件，自动化程度高，但是对复杂的算法生成的硬件性能成问题，生成代码不可读后一种是由编译器帮着写代码，硬件架构还得自个把握，这个自动化很大程度是冲着减轻编码者负担的，生成代码也是可读性差，但是应该可以解决，硬件性能有保证。按说，这两方案目的相同，但路子不同。这些都是在解决编程特别是软件编程与硬件编程间的GAP问题。就目前看，软件与硬件编程还没有得到统一。这二者在计算机发展初期是统一的，后面分离，现在随着软硬件，语言的发展，又有统一的趋势，这种统一也会带来一些新的变化。