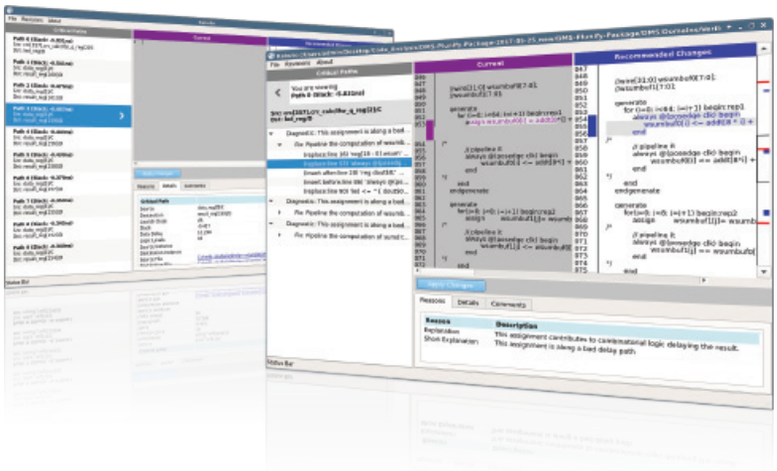
PLUNIFY

Kabuto Beta 数据资料表



Kabuto 是一款使用机器学习技术的专业软件，它可以帮助 RTL 设计师写出更好的代码来解决时序问题，进而提升设计性能。它的核心是一个模式识别引擎，该引擎可以检测出“不良”的RTL代码，也就是导致时序或性能问题的 Verilog 或 VHDL 代码。Kabuto 优先处理设计中的关键路径，并且提供如何修改相关的 RTL 部分的智能建议。设计师最终决定是否接收这些建议，也可以手动修改这些建议。Kabuto 同时支持和 InTime 协同工作。

不同厂商的 RTL 综合以及布局布线工具处理代码的方式也都不尽相同。因此，某家厂商工具的“不良”代码可能不会在另一家厂商的工具上导致性能问题。基于这个事实，不同设备系列的种类可能会要求不同的的 RTL 代码风格。而 Kabuto 可以识别并自动对这些差异做出说明。

工作原理

Kabuto 会分析一个已编译设计的时序报告，并且关注关键路径。在关键路径的数据中，Kabuto 专注在与失败路径相关的 RTL 代码。Kabuto 可以发现每一个特定代码部分的问题是什么，进而为设计师提出必要的代码修改建议。设计师可以评估这些意见的可行性，然后选择接受还是修改这些意见，最后保存这个设计。

读取时序报告 → 抽取关键路径 → 确认关键代码 → 推荐 RTL 代码修改 → 设计师采纳修改意见

主要特色与优势

- 智能模式识别引擎可以检测复杂的情形，比如流水线处理（pipelining）
- 分析和推荐引擎可以提供教导和指南
- 代码风格校正可以使新设计和新工具链的转换更加容易，起到帮助作用
- 可自定义的模式识别引擎可以整合具有不同设计目标的定制的 RTL 指导

检测内容

Kabuto 可以检测多种 RTL 问题。下面是 Kabuto 检测内容的示例：

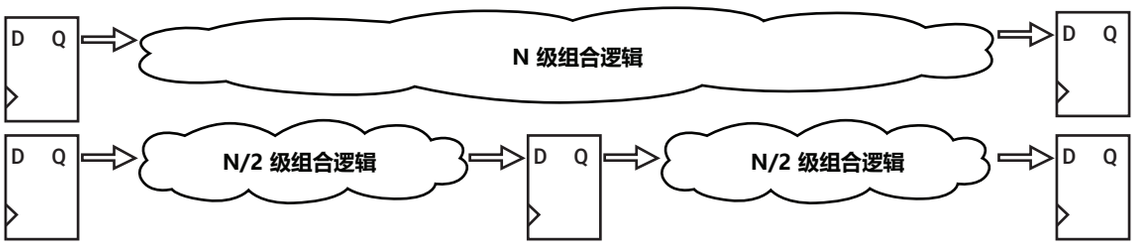
A: 与常数相乘

Kabuto 使用一种转换算法。该算法利用切换与加减运算（shift and addition/subtraction）的操作来替换与常数相乘的操作，从而产生了一种新的表达方式。从时序的角度来讲，这种方法对综合更加友好。

检测内容	Kabuto 修改建议
Out1 = in1 * 60;	Out1 = (in1<<6) – (in<<2);

B: 增加管道阶段

流水线处理使长延时路径（long delay path)细化成了更小的管道阶段（pipeline stages)，这些阶段被寄存器分离。 Kabuto 同时也检测并更正流水线的依赖关系(dependencies)来降低风险。



检测内容	Kabuto 修改建议
assign wsumbuf0[i] = addt[8*i] + addt[8*i+1] + addt[8*i+2] + addt[8*i+3] + addt[8*i+4] + addt[8*i+5] + addt[8*i+6] + addt[8*i+7];	always @(posedge clk) begin wsumbuf0[i] <= addt[8*i] + addt[8*i+1] + addt[8*i+2] + addt[8*i+3] + addt[8*i+4] + addt[8*i+5] + addt[8*i+6] + addt[8*i+7]; end

检测内容（流水线相关信号）	Kabuto 修改建议

always @(posedge clk) begin
if (!rst)
led <= 'b0;
else
led <= ^{dout,sumd};
end
	reg dout$0;
	
	always @(posedge clk) begin
	dout$0 <= dout;
	end
	
	always @(posedge clk) begin
	if (!rst)
	led <= 'b0;
	else
	led <= ^{dout$0,sumd};
	end

C: If 缺少 else 而导致的非故意锁存 (latch)

锁存可能导致时序问题，甚至竞态条件(race conditions)，所以编译器通常会在这一事项上发出警告。导致非故意锁存的一个重要原因是在一个“if”语句中缺少“else”子句。

检测内容	Kabuto 修改建议

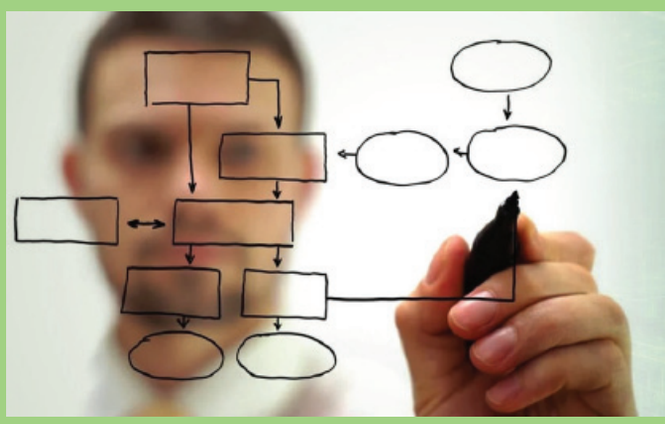
process(vi, en)
begin
if en = '1' then
for i in vi'range loop
vo(i) <= vi(nbits-1 - i);
end loop;
end if;
end process;
	process(vi, en)
	begin
	if en = '1' then
	for i in vi'range loop
	vo(i) <= vi(nbits-1 - i);
	end loop;
	else
	for i in vi'range loop
	vo(i) <= 'X';
	end loop;
	end if;
	end process;

可高度定制的识别引擎

Kabuto 具有一个高度复杂 RTL 代码识别引擎，该引擎可以通过定制来确认多种 RTL 问题，比如资源占用率，功耗等。

很多机构拥有内部的编程指导或培训，以遵循某些特定的标准。这些标准对于不同的平台来说可能不尽相同，比如 FPGA 和 ASICs，或者更新版本的 FPGA 器件系列所带来的新代码风格。

而 Kabuto 可以成为设计师的绝佳助手，帮助他们写出更好的 RTL 代码，以针对不同的平台和器件。



系统要求

- 至少1GB RAM, 以及至少4GB虚拟内存
- 至少100MB 硬盘空间用于 Kabuto 软件
- 处理器: Intel i3 CPU 或类似性能的型号
- Java: Java Runtime Environment (JRE 1.6 或更高版本)

许可证

- 一年期许可证的价格,根据最大并行用户数/电脑数而定

技术规范

- 所支持的操作系统： Windows 7 及以上版本，Ubuntu 和 Centos
- 所支持的 FPGA 工具：Quartus 15 及以上版本， Vivado 2015.4 及以上版本

关于 Plunify

Plunify 通过使用大数据分析和机器学习，帮助芯片公司解决时序优化问题。Plunify 是一家总部位于新加坡的跨国公司。Quartus 是 Intel 有限公司的注册商标。Vivado 是 Xilinx 有限公司的注册商标。Kabuto 是 Plunify 有限公司的注册商标。

Plunify 有限公司
电子邮件:
tellus@plunify.com

新加坡
82, Lorong 23 Geylang,
Atrix Building,
#05-14, Singapore
(388409)

美国
4962 El Camino Real #225
Los Altos, CA 94022
USA

中国
成都市，武侯区，人民南路四段三号，来福士广场二号楼，17层，1737号
邮编：610041