ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.17, No.19 July 2021
E-mail: wltx@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

基于Vivado HLS的硬件设计效能评估

戴源,白雨鑫,张伟,陈鑫*

(南京航空航天大学电子信息工程学院,江苏 南京 210016)

摘要:本文为了研究面向FPGA芯片的高层次综合工具Vivado HLS在硬件设计中的性能,分别利用C++语言与Verilog语言设计移位寄存器,通过比较两种设计方法在不同输出位宽下,其时序、功耗、PDP以及资源使用量上的差别来评估HLS工具在硬件电路设计上的效率与功能性。实验结果表明,虽然HLS工具综合得到的Verilog代码表现不如手工直接编写的Verilog代码,但其以高级语言作为输入的特性还是能满足让设计师在不需要掌握硬件描述语言的情况下利用FPGA实现算法加速的目的。

关键词:FPGA;高层次综合;高级语言;Vivado HLS;Verilog;PDP

中图分类号:TP311 文献标识码:A

文章编号:1009-3044(2021)19-0001-04

开放科学(资源服务)标识码(OSID):

DOI:10.14004/j.cnki.ckt.2021.1845

Effectiveness Evaluation of Hardware Design Based on Vivado HLS

DAI Yuan, BAI Yu-xin, ZHANG Wei, CHEN Xin*

(College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: In order to study the performance of vivado HLS, a high-level synthesis tool for FPGA chips, the shift registers are designed by using C ++ language and Verilog language respectively. The efficiency and functionality of HLS tool in hardware circuit design are evaluated by comparing the differences of timing, power consumption, PDP and resource usage between the two design methods under different output bit widths. The experimental results show that although the performance of Verilog code synthesized by HLS tools is not as good as that of the Verilog code written directly by hand, its high-level language as input can still meet the purpose of using FPGA to speed up the algorithm without mastering the hardware description language.

Key words: FPGA; High Level Synthesis; High Level language; Vivado HLS; Verilog; PDP

集成电路伴随摩尔定律发展至今,其复杂性已经逐渐超过人类可以手工管理的范畴。如:一颗拥有百万门级的SoC,其代码量约为20万行,完成一次规范审查和逻辑综合的时间分别为6.5和8小时[1]。因此,完全使用RTL级的逻辑抽象设计当代芯片是不现实的,复杂的代码不光开发耗时长,还大大增加了编码错误的概率,且调试和验证也非常困难[2]。

高层次综合(High Level Synthesis, HLS)技术就是一种将高级语言转换成硬件描述语言的技术,这项技术能够帮助工程师克服直接在寄存器传输层(RTL)进行开发的困难[3]。设计师们在高层次综合流程下需要注重的是系统的运行模式,HLS工具会负责生成RTL级代码。但不幸的是,HLS工具作为一种编译器,其可靠性难以得到保证,例如,Yang等人就在一些成熟、使用广泛的编译器中发现了数百个以前未知的错误[4]。况且即便是高级语言,其算法中也会存在诸如数据依赖等问题,因此想要利用HLS工具实现高性能的硬件设计,还需要从高级语言代码的编写和HLS工具优化等角度进行大量的工作[5]。

Vivado HLS是FPGA芯片公司Xilinx在2012年发布的集成开发环境中的一款高层次综合工具[6]。近年来,学术和工业界利用Vivado HLS开展了大量工作,其中张俊涛等人设计了FFT IP核[7];王春江等人设计了一套运动目标检测系统[8];齐乐等人设计了一套实时图像去雾系统[9]。然而,HLS工具综合结果与手工Verilog输入之间差异性的对比却鲜有人研究。所以本文提出在Vivado HLS工具中使用C++语言设计移位寄存器,将HLS工具综合出来的Verilog代码与手工设计的Verilog代码进行对比,通过两者之间的差异来评估HLS工具在硬件设计上的效率与功能性。因为功耗/能量效率是评估HLS工具的最重要指标之一[10]。所以,我们将需要对比两种设计统一放到测试系统中进行时序、功耗和资源使用量的对比,以达到评估其效能的目的。测试的软件平台为Vivado 2019.2,并选用Xilinx Kintex-7系列FPGA。

收稿日期:2021-01-12

基金项目:本课题由模拟集成电路重点实验室基金项目(No.61428020304);国家自然科学基金项目(No.61106029、61701228);航空科学基金(No.20180852005)资助

作者简介:戴源(1998—),男,广西梧州人,学士,研究方向:数字集成电路设计;白雨鑫(1999—),女,吉林四平人,学士,研究方向:数字集成电路设计;张伟(2000—),男,安徽铜陵人,学士,研究方向:数字集成电路设计;陈鑫,男,江苏句容人,副教授,博士,研究方向:数字集成电路设计。

实验结果证明，HLS综合出来的设计整体表现与手工设计的Verilog设计基本持平，但因为HLS工具综合的设计使用的是状态机逻辑，所以其在时序和资源使用的表现上略有不足。然而，HLS工具所具备的加速设计周期的优点以及降低硬件开发门槛的特点，仍能在复杂的电路设计中发挥其作用。

1 Vivado HLS工作流程

Vivado HLS的工作流程如图1所示：

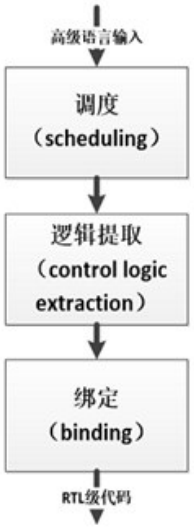

图1　Vivado HLS 工作流程

其流程分为调度、逻辑提取和绑定三个步骤：首先，在调度阶段，HLS工具将对如加法、减法等操作所需要的时钟周期进行安排，以确定整个设计所需要的时钟周期数；随后在逻辑提取阶段，通过生成有限状态机（FSM）来控制寄存器存储数据以及I/O端口控制信号的状态；最后在绑定阶段将高层代码与底层硬件进行对应，确定最终的硬件资源使用量。

2 被测电路的设计

2.1 移位寄存器的选择

移位寄存器功能是数字电路中常用的逻辑功能之一[11]，其工作原理如图2所示，在每一个时钟周期内，寄存器D_i将其内容(0或1)向下一级D_{i+1}传递[12]。基于移位寄存器，可以构建如乘法器、M序列发生器等更复杂的电路。因此，选择其作为被测电路，能够从底层的逻辑反映HLS工具的效能，并以此预测HLS工具在更复杂电路设计中的表现。

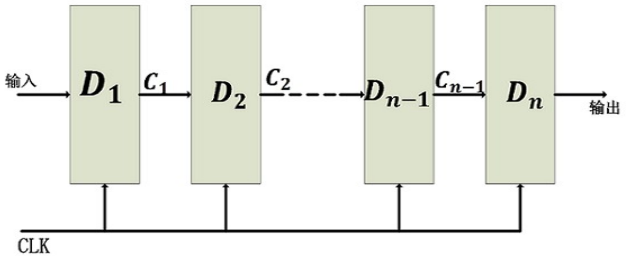

图2　移位寄存器工作原理

2.1 移位寄存器设计

移位寄存器在每个时钟周期CLK内移位一次，即根据输出位宽的不同，我们设计了从4位开始以倍数形式增长到64位的移位寄存器。

下面给出了Verilog输入的4位移位寄存器代码：

```
module shift4
#(
    parameter Width = 4
)
(
  input clk,
  input rst,
  input In_V,
  output reg [Width-1:0] ap_return
);
 always@(posedge clk)begin
  if(rst!=0)begin
  ap_return<={ap_return[Width-2:0],In_V};
  end
  end
endmodule
```

端口clk、rst、In_V分别是时钟、复位以及数据输入端口，在时钟控制下完成数据左移。

对于C++输入的设计，我们保持了与Verilog输入设计一样的设计思路与端口结构，下面给出的是C++输入的4位移位寄存器代码：

```
#include "ap_int.h"
ap_uint<4>shift4(ap_uint<1> In)
{
#pragma HLS INTERFACE ap_none port=In
#pragma HLS INTERFACE ap_ctrl_none register port=return
 static ap_uint<4> mem;
  mem=mem*2;
  mem=mem+In;

  return mem;
}
```

调用ap_int.h库以支持任意位宽的数据，并且#pragma HLS INTERFACE ap_none port=In 和 #pragma HLS INTERFACE ap_ctrl_none register port=return 两条指令能确保输入输出端口不需要复杂的握手信号。

3 仿真验证及测试方案

本文从时序、功耗以及资源使用等多个方面对HLS工具的综合结果进行评估，为了确保测试数据的可靠性，我们设计了一个通用的测试平台。由于两种设计方案均采用同样的端口结构，在外围电路的结构和参数保持一致的情况下，只需要将被测的模块的输入输出端口与外围测试电路的接口对应起来。图3给出了该通用测试平台的原理框图：

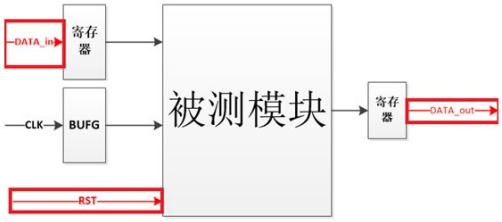

图3　测试平台系统框图

由于测试电路是基础的算术逻辑，只测试单个模块得到的数据可能会过小而不利于分析，因此我们在测试的时候选择同时例化三个相同的模块，并分别在被测的设计与输入输出端口之间都添加了一级寄存器，将图中方框选中的数据路径在约束文件中设置为“假路径”，SAT工具将不会对该路径进行分析[13]，在生成时序报告的时候也不会报告上述路径。接着将输入输出的最大路径延时设置为0，那么对于上述路径的约束是最宽松的，不会产生时序约束违规的情况。为了避免产生时钟树延时，我们还手动在时钟输入端口与设计模块之间添加一级缓冲器(BUFG)。

测试过程统一在Vivado软件中完成对两种不同设计的Verilog代码的调用。由于在设计中设置了“假路径”，如果选择时序仿真将得不到正确的结果。因此，我们仿真验证选择的是实现后功能仿真(Post-Implementation Functional Simulation)，且仿真的输入数据均使用$random函数产生随机输入，并将仿真生成的开关行为文件(.saif)作为功耗分析的输入文件。仿真后，所有的数据都在implementation-design中生成。

4 实验结果的评估与比较

4.1 时序比较与分析

根据Vivado软件报告的时序结果，两种设计中最差的路径延时值随位宽的变化情况如表1所示，并将数据绘制成图4，我们可以发现：HLS工具综合出来的设计普遍都比手工Verilog输入的设计延时要大，且随着位宽的不断增大，前者的设计延时显著地增大。所以，在对时序要求严格的设计中，使用HLS工具所得到的设计将无法满足要求。

表1　移位寄存器最大延时

位宽	4	8	16	32	64
HLS延时/ns	0.688	0.714	0.717	0.743	1.131
Verilog延时/ns	0.686	0.714	0.717	0.714	0.758

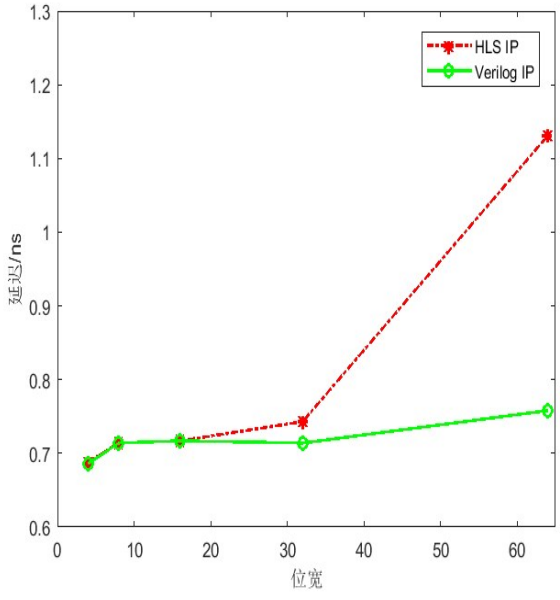

图4　移位寄存器最大延迟

4.2 功耗比较与分析

将Vivado软件报告的功耗结果记录在表2。可以发现，HLS工具综合出来的设计与手工输入Verilog的设计在功耗上差别并不大，前者并没有因为延时的增加而降低功耗。为了进一步的探究，我们进行功耗延时积(PDP)的数据指标分析。

表2　移位寄存器功耗

位宽	4	8	16	32	64
HLS功耗/W	0.329	0.355	0.405	0.506	0.697
Verilog功耗/W	0.328	0.354	0.405	0.506	0.692

4.3 PDP比较与分析

电路延时和功耗的乘积就是功耗延时积(PDP)[14]，它是目前普遍认同的综合性衡量电路的综合性能的指标。表3列出了移位寄存器的PDP，并将表*中的数据绘制成图5。由于HLS工具综合出来的设计在延时上的表现不如手工输入Verilog的设计，这就造成了随着位宽的不断增大，其PDP要比手工输入Verilog的设计大。因此，在电路的性能上，HLS工具综合出来的设计的表现比手工输入Verilog的设计要差。

表3　移位寄存器PDP

位宽	4	8	16	32	64
HLS的PDP	0.2264	0.2535	0.2904	0.3760	0.7883
Verilog的PDP	0.2250	0.2528	0.2904	0.3613	0.5245

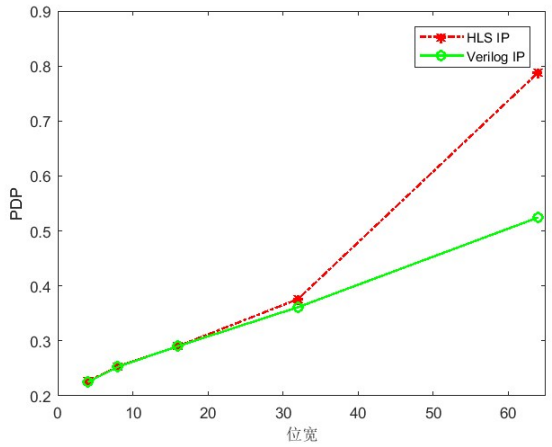

图5　移位寄存器PDP

4.4 资源使用量比较与分析

Kintex-7系列的FPGA一个可编程逻辑块(CLB)拥有一对Slice，每一个Slice中包含4个6输入查找表(LUT)与8个寄存器[15]。对于不同的设计，其使用LUT或寄存器的数量之间没有一个很明显的联系，无法很好地比较做出结论。因此，此处仅比较它们的Slice使用量。

表4　移位寄存器slice使用情况

位宽	4	8	16	32	64
HLS资源使用量	17	25	37	64	148
Verilog资源使用量	8	23	38	68	138

表4列出了移位寄存器和加法器随着位宽而变化的资源使用情况，可以发现HLS工具综合出来的设计普遍比手写Verilog的设计使用更多的资源。仅16位和32位的HLS工具综合出来的移位寄存器设计却比手工输入Verilog的设计Slice使用量要少。造成此结果的原因是Vivado在进行布局布线的时候会进行优化，有一些Slice内部的LUT和移位寄存器并没有用完就启用新的Slice。

经过对比两者的代码后发现，不同于按照数电架构设计的手工Verilog输入电路，HLS工具综合出来的代码完全采用状态机的逻辑实现，其需要更多的逻辑资源来完成对状态机的控制。这一点可以通过比较两者的原理图可以发现，图6给出了4位移位寄存器原理图对比，图中左侧的是Verilog设计，右侧为HLS设计。

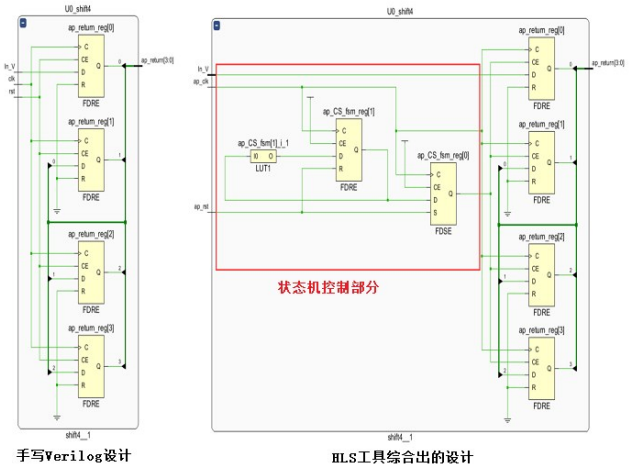

图6　两种设计原理图对比

5 结论

本文通过设计一个通用的测试平台，从时序、功耗、PDP和资源使用这几个方面对比了HLS工具综合出来的与手工输入Verilog的移位寄存器的表现。经过对比实验数据以及各自的代码，我们得出以下结论：

（1）手工输入Verilog的设计在时序、功耗以及资源使用上的表现均好于HLS工具综合的设计。

（2）HLS综合出来的设计完全采用状态机的逻辑实现套路，case分支经常没有用全，容易造成面积过大；if else没有用全，不适合ASIC实现，特别是dft流程会报错，造成测试覆盖率下降。

（3）HLS工具更适合利用高级语言实现较复杂的逻辑控制，比如状态转换等等，不适合做比较复杂的数据逻辑运算。

虽然HLS工具综合出来的设计有着上述的种种弊端，但其利用高级语言作为输入这一特点不仅能大大降低硬件设计的入门门槛，还能加快电路的研发周期，并且HLS工具与手工输入Verilog的混合使用，软硬件协同设计的方法能让设计出来的电路功能更完善。鉴于此，其仍能在复杂的电路系统的设计中发挥其作用。

参考文献：

[1] 邸志雄.HDL代码质量评估方法关键技术研究与电路性能优化[D].西安:西安电子科技大学,2013.

[2] 李雪.基于HLS编译器的FIR滤波器的设计实现与优化[D].西安:西安电子科技大学,2019.

[3] D. E. Thomas, E. D. Lagnese, R. A. Walker, J. A. Nestor, J. V. Rajan, and R. L. Blackburn, Algorithmic and Register-Transfer Level Synthesis:
The System Architects Workbench. The Kluwer International Series in Engineering and Computer Science 85, Springer US, 1 ed., 1990.

[4] X. Yang, Y. Chen, E. Eide, and J. Regehr, "Finding and understanding bugs in C compilers," in PLDI '11.

[5] 李大琳.基于FPGA的高性能算法实现的设计模式及其应用研究[D].长春:吉林大学,2020.

[6] 党宏社,王黎,王晓倩.基于Vivado HLS的FPGA开发与应用研究[J].陕西科技大学学报(自然科学版),2015,33(1):155-159.

[7] 张俊涛,付芳琪,曹梦娜.基于Vivado HLS的FFTIP核设计与实现[J].电子器件,2016,39(0):374-378.

[8] 王春江,李鹏.基于ZYNQ的运动目标检测系统设计[J].电子科技,2020,33(5):82-86.

[9] 齐乐,张小刚,姚航.基于HLS的实时图像去雾实现[J].计算机工程,2016,42(5):224-229.

[10] Lin C Y,Jiang Z H,Fu C,et al.FPGA High-level Synthesis versus Overlay[J]. ACM SIGARCH Computer Architecture News,2017,44(4):92-97.

[11] 金煜涵.一款基于FPGA的可编程逻辑块的设计[D].沈阳:辽宁大学,2019.

[12] 陈澄,张茂青,崔秀美.基于FPGA的可编程M序列发生器的设计[J].工业控制计算机,2013,26(6):15-16.

[13] 刘晟.FPGA静态时序约束的策略研究及探讨[J].通信技术,2019,52(8):2038-2043.

[14] 邵擎.FinFET电路设计[D].宁波:宁波大学,2018.

[15] 7 Series FPGAs CLB User Guide UG474 (v1.8)[M].Xilinx, Inc.2016.

【通联编辑:梁书】