

引用格式:裴希杰,田泽,郑新建,等. 一种流水处理图元建立电路的设计与实现[J]. 微电子学与计算机,2019,36(8):10-13.
[PEI X J, TIAN Z, ZHENG X J, et al. Design and implementation of a pipeline primitives assembly circuit [J]. Microelectronics & Computer, 2019, 36(8): 10-13.]

一种流水处理图元建立电路的设计与实现

裴希杰¹, 田 泽^{1,2}, 郑新建^{1,2}, 张 骏^{1,2}, 许宏杰^{1,2}, 刘宁宁^{1,2}

(1 中国航空工业集团 西安航空计算技术研究所, 陕西 西安 710068;

2 集成电路与微系统设计航空科技重点实验室, 陕西 西安 710068)

摘 要: 为了提高图形处理器的图形绘制能力, 本文设计了一种能够流水处理的图元建立电路, 实现了 OpenGL 定义的 9 种图元到点、线和三角形简单图形的转换, 可有效降低图形流水线后续单元任务的复杂度, 提高图形绘制性能. 通过虚拟仿真和 FPGA 原型验证确认, 实现了基本图元的建立功能, 频率能够达到 400 MHz 以上, 三角形建立峰值可达 380 M/s.

关键词: 图形流水线; OpenGL; 图元建立

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2019)08-0010-04

Design and implementation of a pipeline primitives assembly circuit

PEI Xi-jie¹, TIAN Ze^{1,2}, ZHENG Xi-jian^{1,2}, ZHANG Jun^{1,2},
XU Hong-jie^{1,2}, LIU Ning-ning^{1,2}

(1 Aeronautics computing technique research institute, Aviation industry corporation of china, Xi'an 710068, China)

2 Key Laboratory of Aviation Science and Technology on Integrated Circuit and
Micro-System Design, Xi'an 710068, China)

Abstract: Primitives is a basic unit for graphic processing. To improve the performance for GPUs a primitive assembling circuit is presented. The circuit has a simple structure, pipelining and can convert all primitives defined by OpenGL to separate point, line or triangle. The circuit can reduce the complexity of the flowing unit task and improve the performance of drawing. The verification of the circuit is completed by module level simulation verification and FPGA prototype verification, the results show that the design is function and the frequency can reach 400MHz and the peak value of triangle assembly can reach 380M/s.

Key words: Graphics Pipelining; Open Graphic Language; Primitives Set up

1 引言

在图形处理过程中首先需要将形状各不相同的图形使用简单的基本图元进行表示, 然后才能由图形处理器完成最终的图形绘制^[1]. OpenGL 是目前使用最为广泛的图形标准接口之一, 标准共定义了

10 种图元, 应用层软件将各种复杂的图形拆分为这些基本图元, 然后使用顶点命令下发至图形处理器进行绘制^[2].

GPU 在图形绘制时, 任务种类越少、电路结构越简单, 越利于图形绘制性能的提升, 因此在图形流水线设计时, 首先需要使用特定的图元建立算法将

10 种图元拆分成点、线和三角形等简单图形,然后交由后续流水单元处理,这就是图元建立过程^[3-5]. 在 GPU 设计中,图元建立单元是图元由分离顶点向简单图形转换的关键模块,该模块的处理性能将直接关系到整个 GPU 的图形处理能力,因此适合硬件实现的高效图元建立算法是实现高性能图元建立单元的关键^[6-7]. 目前,国内图元建立算法的研究主要面向软件应用,而国外 GPU 设计公司也由于各自技术方面的因素尚未公开相关算法^[8]. 本文基于 OpenGL 标准,结合硬件实现设计了一套完图元建立电路^[9].

2 电路设计及实现

2.1 OpenGL 定义的 10 种图元

OpenGL 是一种“图形硬件的软件接口”,从本质上讲,它是一个 3D 图形的模型库,具有可移植性强、速度快等特点^[10]. OpenGL 定义了如图 1 所示的 10 种图元,使用这些图元或其组合可以绘制出各种复杂的 3D 图形^[11]. 图形流水线中的图元建立单元的主要任务就是将接收到的这 10 种图元拆分为点、线和三角形.

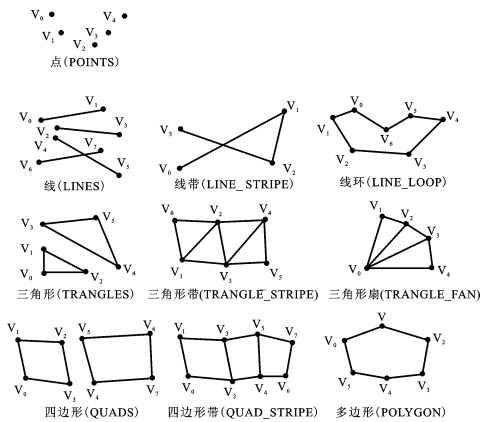


图 1 图元类型

2.2 图元建立算法

10 种图元的按特性进行分析,可以分成三类,一类是离散性图元,包括点、线、三角形、四边形;第二类是连续性图元,包括线带、线环、三角形带、三角形扇,四边形带;第三类是多边形.

2.2.1 离散型图元的装配算法

离散型图元的特点是每个图元具有固定的顶点数目,且不同图元间不存在共用的顶点. 基于这两点该类图元可采用类似的算法实现. 该类图元最多包含 4 个顶点,在该类图元建立过程中最多需要 4 组存储单元保全部的顶点信息. 为了便于描述,假设

r_0 、 r_1 、 r_2 、 r_3 四个存储单元,且每个存储单元可存储 1 个顶点的全部信息.

2.2.1.1 点、线

对于点图元,将接收到的顶点 V_0 写入 r_0 ,并以当前最新属性值作为顶点属性,完成装配. 对于线图元,接顶点 V_0 和顶点 V_1 依次写入 r_0 、 r_1 后开始装配. 若为 SMOOTH 着色模式,则将 r_0 、 r_1 所存储的顶点作为线段的起始点和结束点输出;若为 FLAT 模式,则使用 r_1 中的颜色改写 r_0 中颜色后将 r_0 、 r_1 所存储的顶点作为线段的起始点和结束点输出. 在顶点接收时,如果在接收到 V_0 后就接收到 glEnd 命令,则抛弃已经接收的顶点,删除存入 r_0 顶点数据,重新等待接收新的顶点数据.

2.2.1.2 三角形

对于三角形,接收到顶点 V_0 、 V_1 和 V_2 ,并依次写入 r_0 、 r_1 、 r_2 后开始装配. 若为 SMOOTH 着色模式,则将 r_0 、 r_1 、 r_2 中的顶点作为三角形的起始点、中间点和结束点输出;若为 FLAT 模式,则使用 r_2 中的颜色修改 r_0 、 r_1 中的颜色后,将 r_0 、 r_1 、 r_2 中的顶点作为三角形的起始点、中间点和结束点输出,完成三角形装配.

2.2.1.3 四边形

对于四边形,接收到顶点 V_0 、 V_1 、 V_2 和 V_3 ,依次写入 r_0 、 r_1 、 r_2 、 r_3 后开始装配,如图 2,首先确定四边形的拆分方向,然后将 r_0 、 r_1 、 r_2 中的顶点作为组成第一个三角形进行装配,完成后将 r_2 中的顶点转移到 r_1 中,将 r_3 中的顶点转移到 r_2 ,随后将 r_0 、 r_1 、 r_2 中的顶点作为组成第二个三角形进行装配,2 个三角形颜色属性采用与三角形同样的方式进行装配.

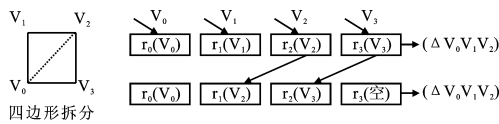


图 2 四边形装配示意图

2.2.2 连续型图元的装配算法

连续型图元的特点是图元是由多个连续的图形组成,且相邻的图形存在公共顶点,因此在图元绘制过程中需要对存储器中的顶点信息按照一定的算法进行刷新,下面就对线带、线环、三角形带、三角形扇及四边形带的算法进行描述.

2.2.2.1 线带与线环

线带与线环的装配过程如图 3 所示,当接收完成 2 个顶点 V_0 、 V_1 写入 r_0 、 r_1 后开始装配,将 r_0 、 r_1

中存储的顶点组成一条线段输出。随后将 r_1 中顶点转移到 r_0 , 新的顶点 V_2 写入 r_1 后, 将 r_0 、 r_1 中存储的顶点组成一条新的线段输出, 后续每个顶点依次与前一个顶点装配成一条直线。当接到 glEnd 命令后, 如果是线带结束装配; 如果是线环, 则将 r_1 中的顶点转移到 r_0 , 并将 V_0 (在接收 V_0 时要单独存储) 写入 r_1 随后将 r_0 、 r_1 中存储的顶点组成一条线段输出后结束线环的装配。在每条直线颜色属性装配时需考虑着色模式进行相应处理。

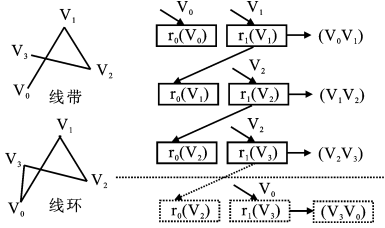


图3 线带装配示意图

2.2.2.2 三角形带

三角形带装配过程如图4所示, 将顶点 V_0 、 V_1 、 V_2 依次写入 r_0 、 r_1 、 r_2 后开始装配。将 r_0 、 r_1 、 r_2 所存的顶点组装成一个三角形输出; 随后 r_2 中的顶点转移至 r_0 中, 将下一个顶点 V_3 时将其写入 r_2 , 随后将 r_0 、 r_1 、 r_2 所存的顶点组装成一个三角形输出。后续每一个顶点与其之前的2个顶点组装一个新的三角形输出, 直到接收到 glEnd 命令结束三角形带的装配。

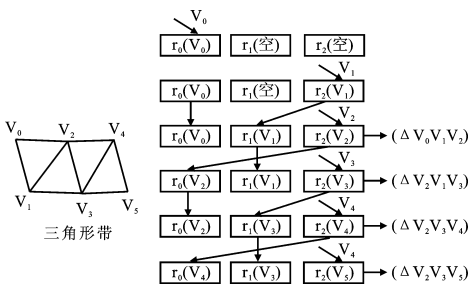


图4 三角形带的组装过程

2.2.2.3 三角形扇

三角形扇的装配过程如图5所示。将顶点 V_0 、 V_1 写入 r_0 、 r_2 , 然后将 r_2 中的顶点数据转移到 r_1 , 再将新接到的顶点 V_2 写入 r_2 后将 r_0 、 r_1 、 r_2 中的顶点组装成三角形输出。完成后将 r_2 中的顶点数据转移到 r_1 , 再将 V_2 写入 r_2 , 然后将 r_0 、 r_1 、 r_2 中的顶点组装成第二个三角形输出。采用此方法依次接收后续顶点, 并组装成三角形输出, 直至接到 glEnd 后结束三角形扇的装配。

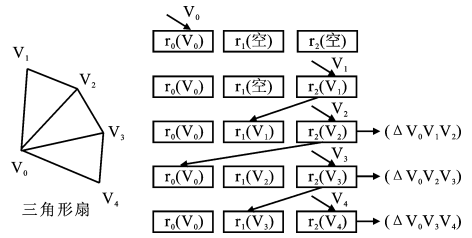


图5 三角形扇的装配过程示意图

2.2.2.4 四边形带

四边形装配过程如图6所示。将顶点 V_0 、 V_1 、 V_2 、 V_3 写入 r_0 、 r_1 、 r_2 、 r_3 , 然后将 r_2 中, 将 r_0 、 r_1 、 r_2 中的装配成一个三角形输出。随后将 r_1 中的顶点转移到 r_0 , r_3 中的顶点转移到 r_1 , 然后将 r_0 、 r_1 、 r_2 中的装配成一个三角形输出, 完成第一个四边形的装配。后续接收到2个顶点与前2个顶点组成新的四边形, 采用与第一个四边形相同的方法完成装配, 直到接收到 glEnd 命令结束四边形带的装配。如果在接收到 glEnd 命令时, 仅接收到1个顶点, 则抛弃该顶点, 并结束四边形带的装配。

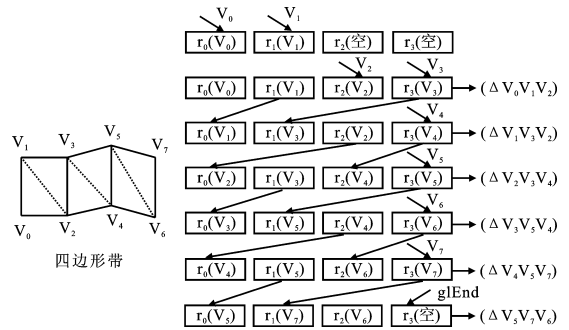


图6 四边形带装配过程示意图

2.2.3 多边形图元的装配算法

由于一个多边形顶点个数不确定, 无法明确接收到多少的顶点才能开始对其进行装配, 所以不利于电路的设计及实现。通过对多边形图元的分析, 一个多边形图元可拆分为其它9种图元, 拆分产生的内部边可使用 glEdgeflag 处理即可, 因此多边形图元的装配可在应用层将其拆分成9种其它图形, 然后由 GPU 完成其绘制^[12]。

3 仿真实验

为了保证设计功能的正确性和完整性, 需要对其进行全面的仿真实验。该电路的设计主要分为2模块级仿真实验和基于FPGA原型的验证。

3.1 模块级仿真实验

使用虚拟仿真工具搭建了图元建立的仿真平

台. 平台结构如图 7 所示,由命令产生模型加载图形绘制命令并同时发送至图元建立单元和功能模型,最终由对比模型完成仿真结果判断^[13].

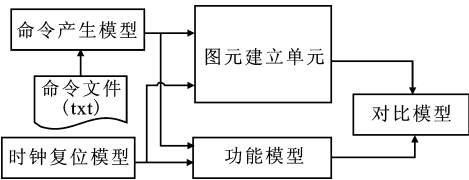


图 7 模块级仿真平台

该阶段仿真需要对图元建立单元的功能进行全面验证. 为此,命令文件以图元建立单元功能相关的 OpenGL 指令进行组合,确保绘制任务与系统任务尽可能保持一致. 基于该平台,使用 400 MHz 时钟进行性能测试. 在图元建立单元处于可以接收新顶点的状态线,以单周期输入 1 个顶点的速度,连续完成 1000 个顶点的处理,并统计建立完成的图形个数和所使用的周期数,完成性能的测试. 表 1 给出了部分图元的图元输出性能.

表 1 图元建立单元性能测试结果

类型	三角形	三角形带	三角形扇	四边形	四边形带
顶点个数	1 000	1 000	1 000	1 000	1 000
周期数	1 040	1 052	1 044	1 040	1 048
图形个数	333	998	998	500	998
生成率 M/s	128.1	379.5	382.4	192.3	380.9

3.2 FPGA 原型验证

FPGA 原型验证可以为设计提供真实的运行环境^[14]. 基于该平台可以对设计进行资源的评估,并可以运行完整的图形绘制任务,对图形处理器的图形绘制性能及效果进行测试.

在完成图元建立单元虚拟仿真验证后,将其集成到图形处理器,搭建 FPGA 原型验证平台^[15]. 使电路综合工具对 RTL 代码进行逻辑综合,然后通过编程电缆将综合好的程序烧写到 FPGA 验证板. 通过主机端运行图形绘制程序,并通过板级总线将命令下发至图形处理器进行图形绘制,最终绘制的图像可通过图像输出设备进行观测.

图元建立的 RTL 代码综合后,资源占用情况如表 2 所示,最高时钟频率可达到 400 MHz 以上.

表 2 综合后的资源占用情况

资源名称	LUT	寄存器
占用情况	53 212	20 196

元. 绘制结果与 INTER、AMD、NVIDIA 等商用显卡的绘制结果一致. 表明该设计能够正确绘制 OpenGL 定义的 9 基本图元,多边形拆分成三角形、三角形扇、三角形带等图元进行绘制.

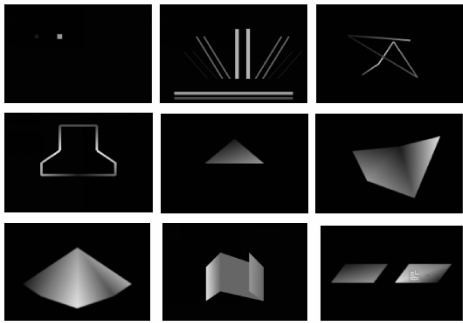


图 8 9 种图元的绘制真结果

4 结束语

本文基于 OpenGL 定义的 10 种基本图元提出了一种可流水处理的图元建立电路及实现方法,并经过了虚拟仿真和 FPGA 原型验证. 该电路可用于图形处理器图形绘制管线的设计. 验证结果表明该图元建立电路可实现 10 种基本图元的绘制,资源使用量较少,频率能够达到 400 MHz 以上,每秒种可建立 380M 三角形.

参考文献:

[1] Xilinx. Vivado Design Suite User Guide Using Constraints UG903 [S]. America: Xilinx , 2017.

[2] WU J, HUA G X, LIU H J. SoC Research AND Application Of Verification Methodology[J]. Space Control Technology And Application. 2012, 38(5) : 27-33.

[3] QIAO L F. Verilog HDL Digital system design and verification[J]. Electronic Industry Press, 2009(4) : 134-146.

[4] OWENS J D, LUEBKE D, GOVINDARAJU NE-TAL. A survey of general-purpose computation on graphics hardware[J]. Computer Graphics Forum, 2007, 26(1) : 80-113.

[5] 李攀, 田文娟, 李娟, 等. FC 协议处理芯片设计与实现[J]. 电子技术与应用, 2016, 42(9), 147-151.

[6] 邓艺, 田泽, 韩立敏. 一种负载均衡的 3D 引擎任务调度策略[J]. 电子技术与应用, 2017, 43(5) : 55-59.

[7] 章元智. 嵌入式 GPU 设计与实现[D]. 山东: 山东大学, 2014.

图 8 为 FPGA 原型验证平台绘制出的 9 种图

辑电路并进行逻辑操作. 将存储在忆阻器的数据读出并将其作为输入逻辑变量, $M21 = (((A \text{ IMP } B) \text{ IMP } (B \text{ IMP } A)) \text{ IMP } 0) \text{ IMP } 0$, 实现了 A 异或 B .

4 结束语

为了使用更少的忆阻器元件实现完备的逻辑功能电路, 本文设计了一款电压控制可重构的基于忆阻器异步时序电路. 本文提出的电路通过 4 个输入信号编码, 采用 1 个忆阻器和 2 个模拟开关可实现大部分的基本逻辑功能; 基于忆阻器的实质蕴涵与非逻辑, 实现了与非、或非、异或等其它逻辑功能; 并通过异步时序设计, 克服了电路逻辑的误操作, 增强了电路的稳定性. P-SPICE 软件仿真结果证明了逻辑电路的可行性. 然而, 存储数据是以电阻值的状态存储于忆阻器中, 如何能将电阻逻辑转变为电压逻辑输出到下一级仍然是值得研究的问题.

参考文献:

- [1] CHUA L. Memristor—the Missing Circuit Element. [J] IEEE Transactions on Circuit Theory. 1971, 18(5): 507-519.
- [2] STRUKOV D B, SNIDER G S, STEWART D R, et al. The Missing Memristor Found. [J] Nature, 2008, 453(7191): 80-83.
- [3] CHEN Q, WANG X P, WAN H B, et al. A Circuit Design for Multi-inputs Stateful OR Gate. [J] Physics Letters A. 2016, 380(38): 3081-3085.
- [4] ZHANG Y, SHEN Y, WANG X P, et al. A Novel Design for Memristor Based Logic Switch and Cross-

bar Circuits. [J] IEEE Transactions on Circuit and System-I Regular Papers. 2015, 62(5): 1402-1411.

- [5] KHALID M, SINGH J. Memristor Based Unbalanced Ternary Logic Gates. [J] Analog Integrated Circuits & Signal Processing, 2016, 87(3): 399-406.
- [6] WANG Z R, SU Y T, LI Y, et al. Functionally Complete Boolean Logic in 1T1R Resistive Random Access Memory. [J] IEEE Electron Device Letters. 2017, 38(2): 179-182.
- [7] NISHIL T, GUPTA S, MANE P, et al. Logic Design within Memristive Memories Using Memristor Aided Logic. [J] IEEE Transactions on Nanotechnology. 2016, 15(4): 635-650.
- [8] K VATINSKY S, STATAT G, WALD N, et al. Memristor-Based Material Implication(IMPLY) Logic: Design Principles and Methodologies. [J] IEEE Transactions on very large scale integration(VLSI) systems. 2014, 20(10): 2054-2066.
- [9] VOURKAS I, BATSOS A, SIRAKOULIS G C. SPICE modeling nonlinear memristive behavior. [J] International Journal of Circuit Theory and Applications. 2015, 43(5): 533-565.

作者简介:

付迅 男, (1994-), 硕士研究生. 研究方向为忆阻器电路设计.

彭菊红(通讯作者) 女, (1978-), 硕士, 讲师. 研究方向为记忆电路的研究与设计. E-mail: juhongpeng@hubu.edu.cn.

杨维明 男, (1969-), 博士, 教授. 研究方向为射频器件与电路.

(上接第 13 页)

- [8] 饶志恒. 图形处理器图形管线的研究与实现[D]. 长沙: 湖南大学, 2011.
- [9] 张淮声, 余莉. 一种三维绘制中可编程的图元设置着色方法[J]. 集成电路应用. 2017, 34(6): 9-13.
- [10] 武凤丹, 阙恒, 王渊峰. 一种新的低功耗高性能绘图芯片 GPU 设计方案[J]. 集成电路应用. 2017, 34(6): 33-36.
- [11] 袁扬智, 韦明, 曾献军. 2D 图像硬件加速引擎的设计优化[J]. 微电子学与计算机, 2009, 26(4): 241-248.
- [12] 韩俊刚. 一种图形加速器和着色器的体系结构. 计算机辅助设计与图形学学报[J]. 2010, 2(3): 363-372.
- [13] 谭显强. 基于 FPGA 的 3D 图形处理器 IP 核的设计与实现[D]. 南京: 南京航空航天大学, 2010.
- [14] 韩伟. 图形处理芯片的研究与设计验证[D]. 上海: 上

海交通大学. 2008

- [15] 黎小玉, 田泽, 郭亮, 等. 符合 OpenGL 标准的国产化显卡研究与实现[J]. 计算机技术与发展, 2014(5): 173-175.

作者简介:

裴希杰 男, (1984-), 硕士, 工程师. 研究方向为数字集成电路设计与验证、图形处理器设计.

E-mail: peixj2008@163.com.

田泽 男, (1965-), 博士, 研究员. 研究方向为 VLSI 设计、SoC 设计、航空专用集成电路设计.

郑新建 男, (1980-), 博士, 工程师. 研究方向为图形处理器系统结构、微处理器系统结构.

张骏 男, (1978-), 博士后, 高级工程师. 研究方向为计算机体系结构、GPU 设计、微处理器设计、SoC 设计.