**PCIe to OCP Bridge**

Michael Walton

Benjamin Huntsman

Peter Depeche

Kevin Bedrossian

Table of Contents

[1 Introduction 4](#_Toc377042136)

[2 Design Plan 5](#_Toc377042137)

**Abstract**

PCI Express (PCIe) was created in 2004 and became into wide spread use soon after that due to the great benefits over PCI. It allows for increased bandwidth and flexibility that could not be achieved before. Revisions of the PCIe protocol continue to be developed giving increased bandwidth and functionality to the already robust communications bus. Open Core Protocol (OCP) is an openly licensed, core-centric protocol intended to meet system level integration challenges. It is an independent bus interface for on-chip systems for communications. These two protocols will be used to develop a System On Chip (SOC) that will allow for direct communication of the PCIe bus with a Virtual File System (VFS) supporting all memory transactions. This is to be accomplished by an IP block that will transparently bridge these two standards. The target device will be a Xilinx Spartan-6.

# Introduction

A PCIe to OCP bridge IP block will be implemented to allow communication between a VFS. The design will be validated by performing Direct Memory Access (DMA) communication with an EMMC memory device connected to the FPGA platform.

# Design Plan

The following is the project plan that will be used in developing the bridge using a Hardware Design Language (HDL) such as Verilog and System Verilog.

1. Design
   1. Pseudo-code of core and test bench
   2. Write HDL/Verilog for simulation that is technology independent
   3. Analyze for functionality
   4. Revise HDL/Verilog design
2. Optimization
   1. Map HDL to Xilinx Vertex 6 logic circuits and functional blocks
   2. Optimize for technology dependence
      1. Prepare for synthesis
   3. Timing Analysis
      1. Check for speed, setup, and hold time
      2. Form constraints
   4. Revise HDL
3. Place & Route
   1. Map(place) netlist to FPGA
   2. Route the structures (cores) on the FPGA to interconnect and perform the desired function.
   3. Timing Analysis
      1. Apply further constraints if needed
      2. Move blocks around on the FPGA using a floor planning tool if needed to help timing.
   4. Revise HDL
4. Verification
   1. Check design functionality
   2. Check if design meets performance goals