AXI Master and Slave demo

Christopher R. Bowman

3 May 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project implements the most basic AXI Master interface controlled by a AXI lite slave register set. One register controls the read address for the master. Another triggers a read by the master from main memory which value is stored in yet another register that can be read from the slave register set. A git repository is located on github. The project is targeted at the AMD/Xilinx Vivado 2024.2 Design Suite.

Chapter 2

Licensing

Under no circumstances should anything be committed to the project repository containing copyrighted material which is owned by a third party.

At some point this project may move to a more permissive licenses. At present, however, all contributions will have to either donate the code and copyright or include a world wide, unrestricted license including the right to create and distribute derivitive works and sublicense the orignal and derivitive works.

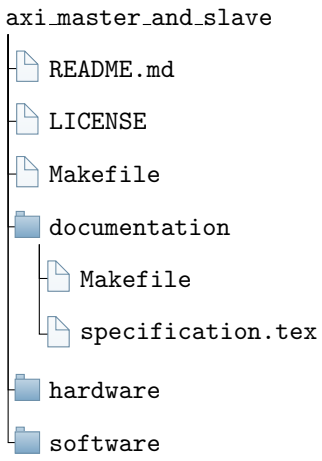
Chapter 3

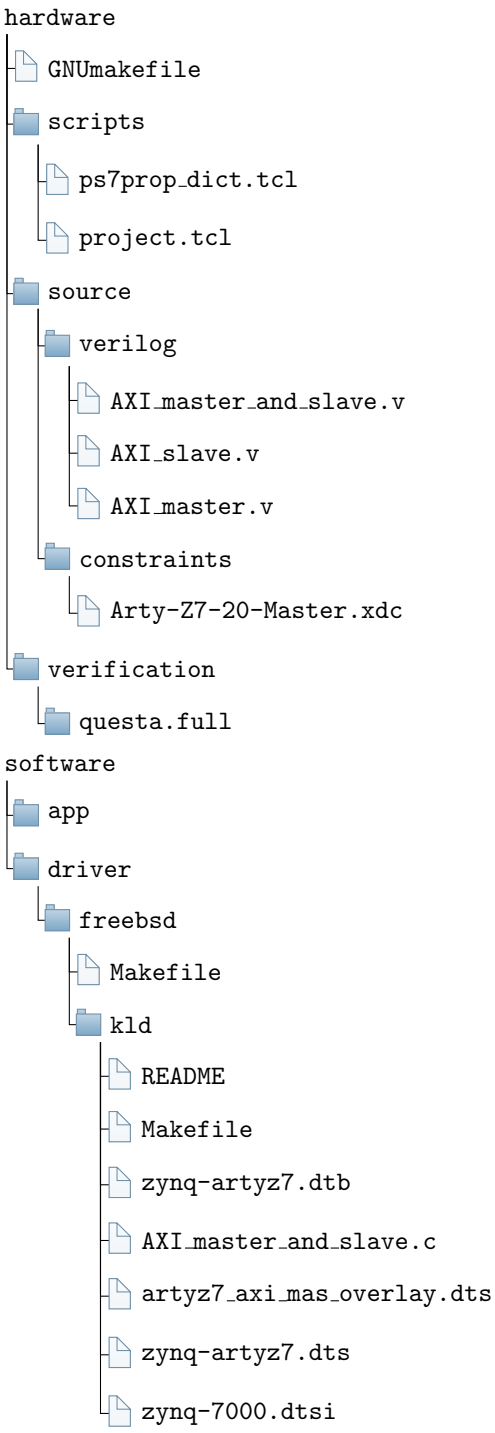
Required Software

- AMD/Xilinx Vivado 2024.2 Design Suite
- Linux (Rocky 9.4)
- FreeBSD (Drivers are QAed for 14.2)
- GNU Make
- git
- wget
- fetch
- Latex
- dvips
- dvipdf
- latex register packge

Chapter 4

Repository File Organization




hardware
GNUmakefile
scripts
ps7prop_dict.tcl
project.tcl
source
verilog
AXI_master_and_slave.v
AXI_slave.v
AXI_master.v
constraints
Arty-Z7-20-Master.xdc
verification
questa.full
software
app
driver
freebsd
Makefile
kld
README
Makefile
zynq-artyz7.dtb
AXI_master_and_slave.c
artyz7_axi_mas_overlay.dts
zynq-artyz7.dts
zynq-7000.dtsi

Chapter 5

Board Design

The board design has Verificaiton IP (VIP) for clock, reset and AXI interfaces. See Figure 5.1 When building the board design these VIPs pass through their signals. In the verification testbench provided in the verification directory, these VIPs are configured to drive their respective signals and thus wrap the master slave block allowing it's verification since the processor subsystem doesn't have a model.

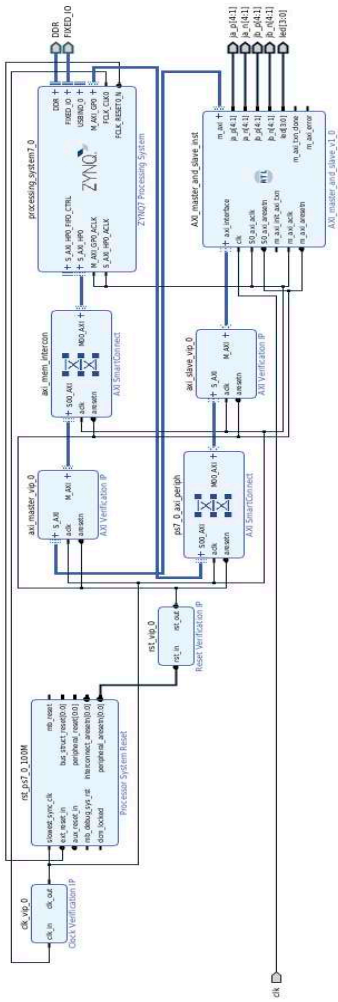


Figure 5.1: Board design

Chapter 6

Verification

A basic verification testbench is provided in `hardware/verification/questa.full` that writes and reads several slave interface registers. Clock and reset VIPs are used to provide clock and reset. The VIP connected to the master interface implements a memory models such that reads from a memory location will return the results of the most recent write to that location.

Chapter 7

Resources

The following paragraphs provide links to important resources of relevance to this project as most materials are copyrighted and cannot be distributed or included in the project git repostories. It is recommended that you visit the links, download the materials and keep a local copy for your use.

The board used for this project is the Digilent Arty Z7-20 available from Digilent. Documentation for this board can be found on the Arty Z7 Reference Manual web page.

The AMD/Xilinx Zynq FPGA contained on the board is documented in a variety of manuals. Of particular interest is the Zynq-7000 SoC Technical Reference Manual UG585.

Chapter 8

Programming

Insert the SD card with the image and at the U-Boot prompt enter the following commands:

```
fatload mmc 0 400000 bit/ssd_top.bit
   fpga loadb 0 400000 size
```

Alternatively one can run the following command as root: `xbit2bin` *`fpga.bit`* where *`fpga.bit`* is the FPGA bitsream file generated by the AMD/Xilinx tools. You can also program from within Vivado on a machine connected via the console/programming port.

Chapter 9

Register Specifcation

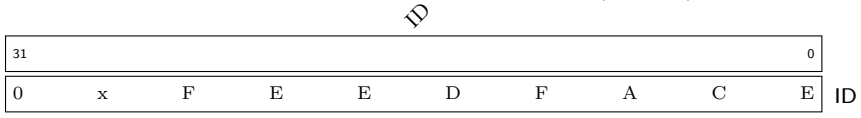
This design has 4, 32-bit registers which identify the hardware and allow control. The first register has a fixed ID code and always reads back the same fixed value. Register 1 sets an address for the master to read from and register 2 receives and stores the value read from the main memory address in register 1. Register 3 when, written, causes the master to read from the main memory and store the results in register 2 which can then be read back.

The regsiter map is shown in Table 9.1 on page 14.

0x0000	IP Identification register
0x0004	Master main memory read fetch address
0x0008	Main memory read fetch result
0x000C	Write only register triggering main memory read

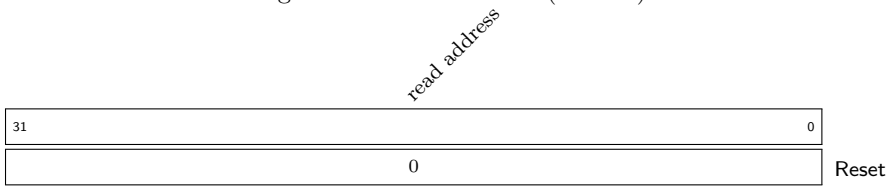
Table 9.1: Register Map

Register 9.1: IP Identification (0x0000)



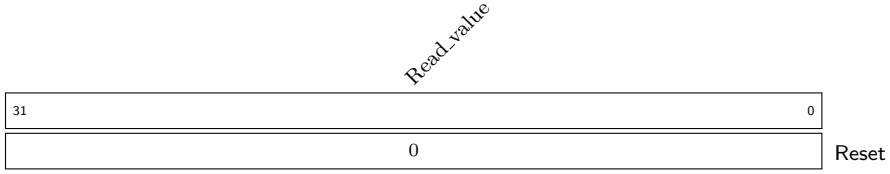
ID (Read only) Contains the constant 0xFEEDFACE

Register 9.2: READ_ADR (0x0004)



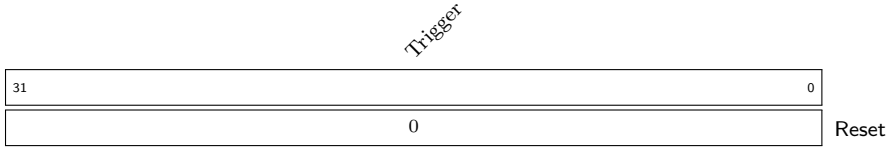
Read_Adr	32 bit address of main memory localtion from which the master should read when triggered. Addresses should be 32-bit aligned

Register 9.3: READ_VALUE (0x0008)



Read_value	When the Master is triggered the eventual result of the read from main memory will be stored here.

Register 9.4: RESERVED (0x000C)



Trigger	Writing any value to this register results in the master interface reading from the main memory address in the Read_Adr register and storing the results of that read in the Read_value register