[Lab 8] Direct Memory Access

Jinha Jeong, Bogyeong Park, and Junhyuk Choi

arch23-ta@hpcs.snu.ac.kr

Overview

In this project, you will implement a simple DMA engine on your previous CPU design and analyze the differences in performance.

Goal

- Understand DMA and its impact on performance
- Understand how devices communicate with the CPU and the memory
- Implement a simplified DMA logic on your pipelined CPU with a cache

The assignment for this week is to implement a simple DMA design for education. In this design, only the device initiates the data transfer. Also, this simple DMA engine supports only fixed address and fixed size write operations.

Assignment Description

Assignment 8: Direct Memory Access

The DMA controller should support the following functions.

1. It receives and parses a DMA command from the CPU, and gets the memory buses exclusively using Bus Request and Bus Granted signals.
2. It reads the data from the external device and writes the data to the memory.
3. It releases the data memory buses after all DMA transactions are done

DMA target address (0x1F4) and length (12 words) are fixed in this project.

For a detailed description of the project, please refer to the pdf file, skeleton code, and test bench.

Assignment 8 - extra credit: Cycle stealing

In this design, the DMA engine releases the bus after sending 4 words.

As a result, the stalled-CPU can execute instructions.

If the CPU does not use the memory bus, the DMA controller should retake the bus as soon as possible.

Grading

Functionality

Your CPU should pass every test in the previous testbench. Its functional correctness must not be affected by the DMA.

Code review

Your code should be easy to read. If not, you may get some penalties.

- You should provide explanations of the module's inputs, outputs, and functionality in the comments.
- All the variables, parameters, and file names should be self-explanatory. It should be neither "reg a", "wire wire1" nor "helloWorld.v".
- You should stick to a consistent code style (e.g., indentation, naming).
- Do not mix blocking and non-blocking assignments in a single always block.
- Do not use delay (e.g., #10) to correct the timing. Delays are only for testbench.

Report

Basically, please refer to “How_to_write_your_report.pdf”.

We'll follow the previous guideline for this week.

In addition to this, you should prove the correctness by providing the proper waveforms. Also, you need to analyze the difference in performance to get the perfect score.

If you implement cycle stealing, you need to compare the performance to the CPU without it.

Submission

Due: 05/29 (Mon.) 19:00

- Late submission
    - 10% penalty per day until 06/03 (Sat.) 19:00
    - NO points after 06/03 (Sat.) 19:00

Upload a single zip file named LabXX_groupXX_20XX-XXXXX including your (1) project folder, (2) report on the eTL board.

- For example, if your student number is 2023-12345, and your group number is 67, submit Lab08_group67_2023-12345.zip
- Check “copy sources into project” option in “add or create design sources”.
- **Make sure your Vivado project contains your source code.**
    - Check the Verilog files inside “[vivado_project_name].srcs/sources_1/new/” before submission.
- Do not delete files in the project folder if unnecessary. (e.g., .xpr file)