Bern University
of Applied Sciences

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

MSE - TSM_EmbHardw

Optimizing memory access with DMA

TABLE OF CONTENTS

Introduction

The goal of this document is to guide you through the design process of a complete SoPC system where memory accesses are optimized through a DMA access. For achieving so, there is a simple preliminary step to be performed: an additional timer must be included in order to measure and compare system performance.

- Add a timer or counter for measuring performance (with or without DMA)
- Extend the LCD ctrl by a DMA interface (as for testing use DMA core from IP Library)
- Realize your own proper DMA interface and extend the LCD Avalon slave ctrl,
- Add the camera IP (the course responsible give it to you) and extend the firmware to transfer data from camera to LCD

1 Measuring system performance

The main goal of using a DMA access is to accelerate memory transfers. Because of this, we need to somehow be able to measure system performance. If we recognize acceleration its nice but we can not contend we improved system performance without results through an adequate profiling.
Include a new timer in the SoPC system. It could be a performance counter or an interval timer. For more details on these peripherals refer to the document Embedded Peripherals IP User Guide, available in the Moodle website.
It's up to you to select how to initialise the peripheral and how to control it from the firmware (C application).

2 DMA access from your LCD controller

2.1 Adapt the peripheral architecture

The goal is to optimize the data transfer performed when displaying an image. Up to now it is the processor that performs the transfer. Instead of transferring we would like the processor to configure the LCD controller (initiate the transfer) in order to be involved in each single data transaction itself. This allows the processor to handle other tasks while the data get transferred. This is what we call a Direct Memory Access (DMA).
For doing so follow the steps below.

- Identify the inputs-outputs of the system. Keep in mind that in addition to the LCD interface and Avalon slave interface additionally an Avalon master interface is needed.
- Define a register model allowing to keep the same functionality as before and additionally the required registers to make the DMA access parameterizable to indicate the base address of the image to be read, and the size of the image[1].
- Include also an IRQ output generating an interruption at the end of a transfer, in order to inform the Nios II processor that the transaction has been completed.
- Include the required control register(s) accessible through the slave interface in order to at least start the DMA transfer and manage interruptions.
- **ON A PAPER**: propose an architecture for an LCD controller with DMA access (i.e. Master access to the bus Avalon).
- Describe the circuit by using VHDL.

2.2 Simulation

Verify the functionality described by your VHDL model. Two options are proposed for this.

- You can perform a manual simulation of the VHDL model in which you must identify and define the input time diagram of some key use cases. For instance the configuration of a DMA transfer.
- A test-bench is available in the Moodle website in order to verify the functionality of the DMA-LCD controller.

Read the file `howto_modelsim.txt` and follow the instructions in order to use it.

[1] Based on this information the DMA ctrl knows how-many data will be transferred.

2.3 Plugging and using your DMA-LCD

In Qsys, attach the designed LCD-DMA controller to the SoPC, following a similar procedure as the one of the previous lab.
Back to eclipse, configure the registers in order to program a transfer from the SDRAM memory to the LCD controller. For doing so pre-load an image to an array (by using a for loop), and then send the pointer pointing to the table to the DMA. Use the timer in order to compute the time required for:

- copying a complete image when using the DMA access
- the time when not using it (doing everything from within processor)

Some question concerning system performance profiling:

- In average, how many clock cycles do we use per memory access?
- Is it coherent with the estimations?
- What speed-up do we achieve by using DMA access?

3 Attach the Camera (Optional)

Two new interfaces are needed for accessing the Camera :

- An I2C bus is required for initializing the Camera registers in order to setup the configuration well suited for our system.
- After initialized data is sent through a parallel interface, by using a synchronous VGA-like protocol.

These two interfaces can be downloaded from the Moodle webpage. You just have to make sure to correctly integrating them to the SoPC.
Moreover there is a library allowing to initialize the camera, and to setup the Camera interface for writing the input video frames to an array. This provides an example of gathering camera data and sending them to the LCD through the DMA. Your task will be to integrate the libraries to Eclipse project in order to transfer to the SDRAM an image acquired from the camera.