h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Prof. A. Upegui

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Optimizing memory access with DMA : LCD and Camera interfaces

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 prelimiary step to be performed : a timer must be included in order to measure and compare system performance.

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.

Include a new timer in your 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 your C code.

2 DMA access from your LCD controller

2.1 Adapt your peripheral architecture

The goal here is to optimize the data transfer performed when displaying an imge. Up to now it is the processor who performs the transfer. Instead of this we would like the processor to configure the LCD controller in order to allow this last one to directly read by itself the data from memory. This is what we call a Direct Memory Access (DMA).

For doing so :

- Identify the inputs-outputs of your system. Keep in mind that in addition to the LCD interface and avalon slave interface, you also need now an avalon master interface.
- Define a register model allowing to keep the same functionalities as before and additionally the required registers to make the DMA access parametrizable to indicate the base address of the image to be read, and the size of the image.
- 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).
- Write your VHDL code.
- **NOTE :** in order to simplify your work, you can find in the moodle webpage a compressed file including a vhdl file which contains an entity with all the input and output ports required to build your DMA-LCD interface and a testbench.

2.2 Simulation

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

- You can perform a manual simulation of your VHDL code 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 testbench is available in the Moodle website in order to verify the functionality of the DMA-LCD controller.
  In order to use it follow the next steps :
  - Uncompress the files.
  - Open modelsim and change the working directory to $PATH/simulation/vhdl/modelsim/
  - Run 'do start.do' in modelsim console to run the testbench
  - If you haven't yet described your architecture you will get many error messages. This is normal. If you have already written your architecture you might have some error messages. Have fun debugging them ! !
  - The resulting simulation waves should look like the following figure :

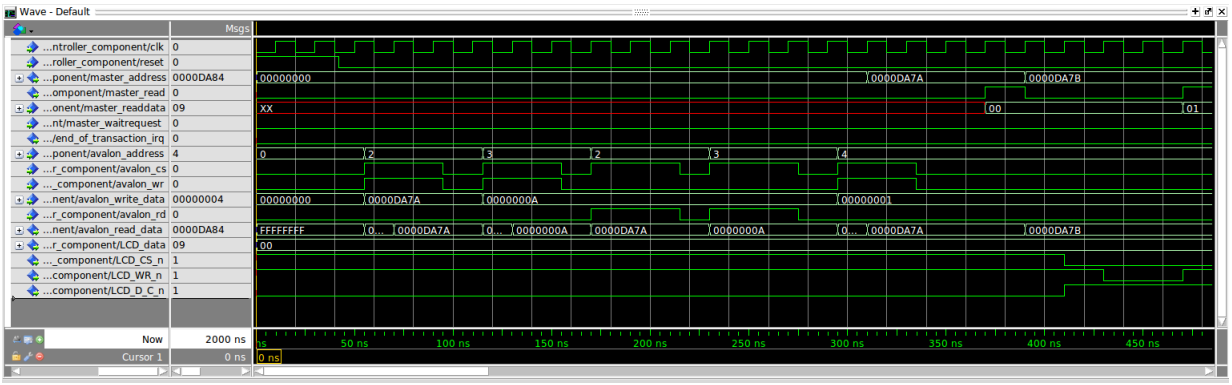

FIGURE 1 – Initially the DMA transfer must be configured through the avalon slave interface. The value 0xDA7A is written to the register address 2 for indicating the source address. The value 0xA is writen to the address 3 for indicating the number of transfers to execute. Then, both addresses are read for verification purposes. Finally, the value 0x1 is written to the register 4 (control register) for starting the transfer.

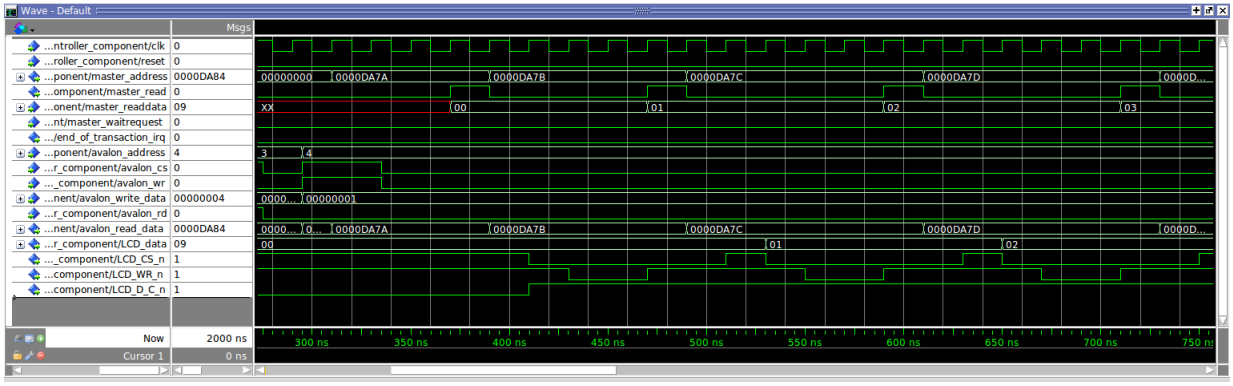

FIGURE 2 – After starting the transfer a series of 10 consecutive reads are performed from the address 0xDA7A, address is further auto-incremented for each consecutive address. Each value read through the avalon master interface is further presented in the LCD_data output and the corresponding output LCD_WR_n and LCD_CS_n signals are generated. **NOTE :** this example performs transfers on 8 bits, addresses are thus increased by 1. 16-bit transfers require an increase of 2.

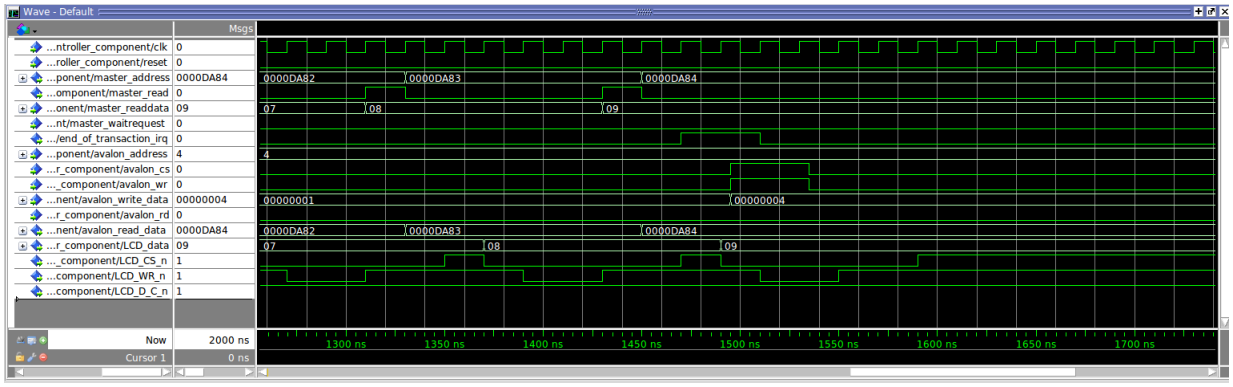

FIGURE 3 – At the end of a complete transfer, an irq output signal is generated by the DMA. The irq must be maintained until an acknowledge is performed from the processor. The acknowledge is completed by writing a 1 to the third bit of the control register (writing the value 4 to register 4)

2.3 Plugging and using your DMA-LCD

In Qsys, integrate the newly created LCD-DMA controller to your SoPC, following a similar procedure as the one followed in the last lab.

Back to eclipse, access your registers in order to program a transfer from the SDRAM memory to the LCD controller. For doing so preload an image on a table (with a `for` loop), and then send the pointer pointing to the table to your DMA.

Use the timer included in the first part of this document in order to compute the time required for copying a complete image when using the DMA access and the time when not using it (doing everything from the processor).

In average, how many clock cycles do you use per memory access ? Is it coherent with your estimations ?

What speed-up do you achieve with the DMA access ?

3 System reuse : Camera interface (Optional)

Two new interfaces are needed for acessing the Camera :

An I2C bus is required for initialising the Camera registers in order to setup the configuration well suited for our system.

After initialised 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 your SoPC.

Moreover you also find a library allowing to initialise the camera, and to setting up the Camera interface for writing the input video frames to a pointer defined by the programmer. Your work is to integrate these libraries to your Eclipse project in order to tranfer to the SDRAM an image acquired from the camera.

The same pointer used as camera image destination is to be used as LCD image source.