Question 31 Laboratory project

1. Explain the LCD controller or Camera controller design which you realized as part of the mini-project on master interfaces during the laboratory sessions.

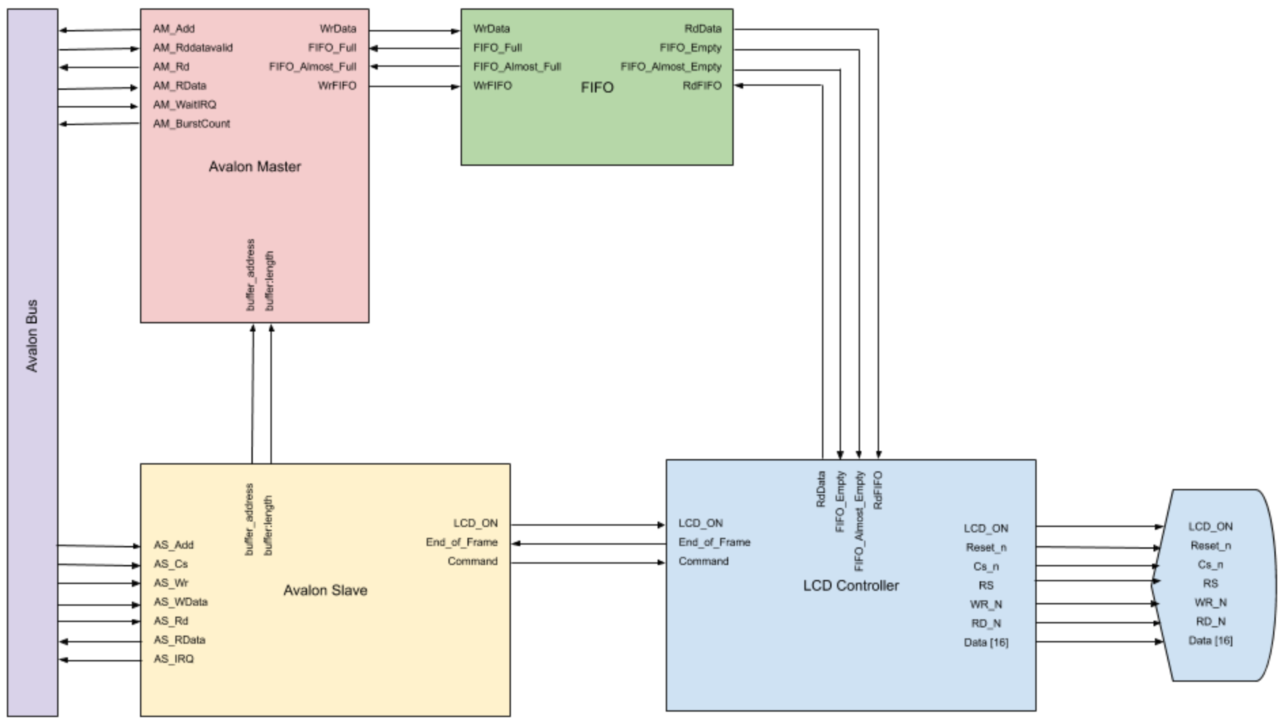
2. Explain the internal registers made available to the processor by your slave programmable interface and why these registers are needed. What do they do?

3. Explain the architecture of the general system in which your programmable interface is used and explain how the different components interact together. Would this be possible if the different interfaces were not programmable?

4. Explain the Avalon slave/master transfer protocol used in your system.

- LCD Controller :

Avalon Slave used for communication with the NIOS-2 processor. Avalon Master used as a DMA to fetch the pixels directly in the memory without using the NIOS-2. LCD Controller used to send data/commands to the ILI9341 with the correct timings. FIFO buffer to allow the non-synchronous transfer of the pixels from the Avalon Master to the LCD Controller.



- Internal Registers :

Name	Offset Interface	Offset uP	Data Width	Description
buffer_address	0	0	32bits	Base address of the frame stored in memory
buffer_length	1	4	32bits	Length of the frame stored in memory
LCD_command	2	8	8bits	Command sent to the ILI9341 by the NIOS-2
LCD_data	3	12	16bits	Data sent to the ILI9341 by the NIOS-2
burstcount	4	16	8bits	Size of each burst transfers for the Avalon Master read
finished	5	20	1bit	Read only : Status register indicating if last frame is finished
interrupt_enable	6	24	1bit	Enable the interrupt when the Master has finished reading a frame
LCD_on	7	28	1bit	Turn the LCD ON or OFF

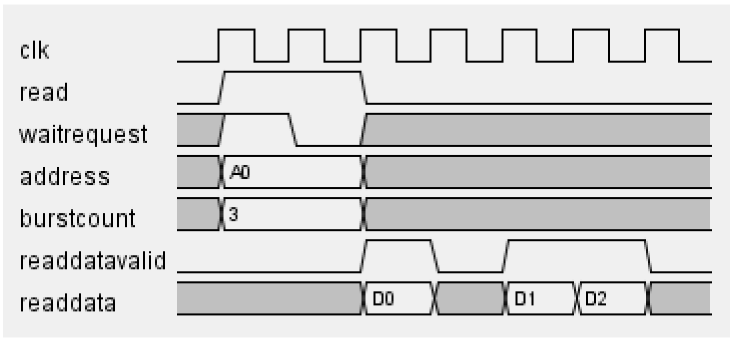
- General architecture :

Camera writes in memory in buffer 1, LCD displays the frame in buffer 2. When both are finished (polling), we swap the memory buffers.

We need programmable interfaces to send the addresses and the commands !

Non programmable interfaces would not make this project possible.

- Avalon slave/master protocol :



1. Idle. Go to 2 if a buffer_length is given.
2. Request read : assert AM_address, AM_burstcount, AM_read. Go to 3 if AM_waitrequest = 0
3. Acquire data : If AM_readdatavalid = 1, read what's on AM_readdata and put it in the FIFO. Go back to two if there are more pixels to display.
4. When a frame has been finished, go back to idle and change the finished signal.