Lab 4: Character LCD Control

National Chiao Tung University
Chun-Jen Tsai
10/5/2018

Lab 4: Character LCD Control

- In this lab, you will compute the first 25 Fibonacci numbers, and use the standard 1602 character LCD to display the numbers

- The deadline of the lab is on 10/16

1602 Character LCD Display

- The Arty board has only simple I/O devices such as the LEDs, switches, buttons, and UART
- We have designed an expansion board, Arty_IO, that adds three more peripherals to Arty:
  - a 1602 character LCD device (supports only 4-bit mode)
  - a SD card socket (supports only the SPIF mode)
  - a 12-bit color VGA interface

Memory Map of the LCD

- The LCD device can be treated as a 32-byte memory
  - Each memory cell corresponds to a character on the display
  - Writing an ASCII code to a cell will display the character on the corresponding location on the LCD:

Note: the LCD device is slow, you should not update the screen faster than 2 Hz.

- Display data memory (DD RAM) addresses:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

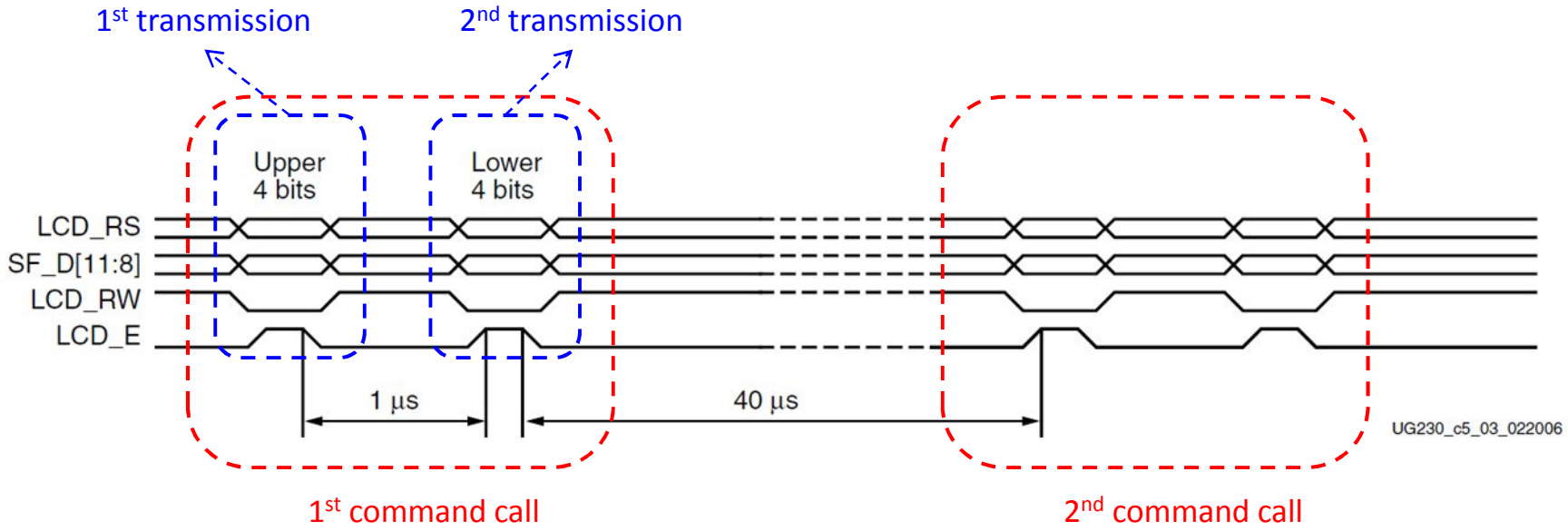
Character LCD Interface (1/2)

- The LCD interface has 8 data wires (DB0 ~ DB7) and 3 control wires (LCD_E, LCD_RS, LCD_RW):
  - LCD_E enable/disable the inputs to the LCD module
  - The rest of the wires are defined depending on the functions:

Function	LCD_RS	LCD_RW	Upper Nibble				Lower Nibble			
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Display	0	0	0	0	0	0	0	0	0	1
Return Cursor Home	0	0	0	0	0	0	0	0	1	-
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S
Display On/Off	0	0	0	0	0	0	1	D	C	B
Cursor and Display Shift	0	0	0	0	0	1	S/C	R/L	-	-
Function Set	0	0	0	0	1	0	1	0	-	-
Set CG RAM Address	0	0	0	1	A5	A4	A3	A2	A1	A0
Set DD RAM Address	0	0	1	A6	A5	A4	A3	A2	A1	A0
Read Busy Flag and Address	0	1	BF	A6	A5	A4	A3	A2	A1	A0
Write Data to CG RAM or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Read Data from CG RAM or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0

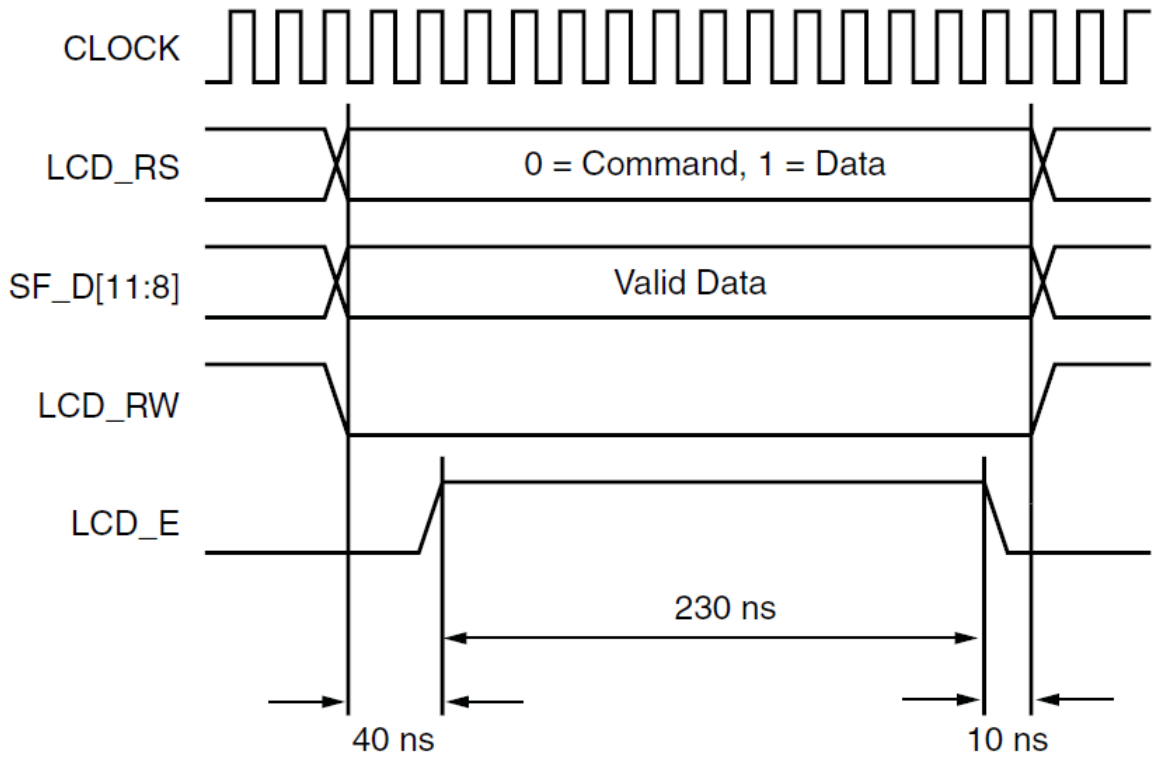
Character LCD Interface (2/2)

- However, the Arty_IO board uses the 4-bit operating mode of the LCD device, that is, only DB4~DB7 are connected to the FPGA
  - Execution of a function will need two transmissions, using only LCD_E, LCD_RS, LCD_RW, and DB4~DB7:



Timing Diagrams for Transmission

- The timing diagram for one transmission in four-bit mode is as follows:
  - Note that execution of a function requires two transmissions



The Sample Circuit of Lab4

- Two Verilog program files will be provided to you:
  - LCD_Module.v – An LCD controller module
  - Lab4.v – a sample top-level module that prints a “Hello, World!” message using the LCD controller module

```
module LCD_module(
    input clk,
    input reset,
    input [127:0] row_A,
    input [127:0] row_B,
    output reg LCD_E,
    output reg LCD_RS,     //register select
    output reg LCD_RW,     //read / write
    output reg [3:0]LCD_D //data
    );
```

top-row of text

bottom-row of text

The Fibonacci Number C-Model

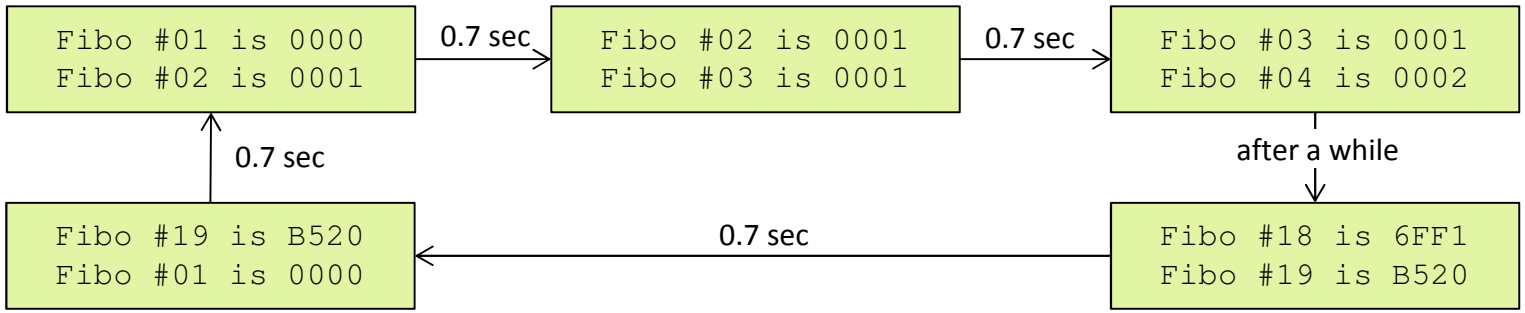
- The first two Fibonacci numbers are 0 and 1. Each remaining number is the addition of the previous two.
- A short C-model that computes the first 25 Fibonacci numbers is as follows:

```
int fibo[25], idx;

fibo[0] = 0, fibo[1] = 1;
for (idx = 0; idx < 25; idx++)
{
    if (idx >= 2)
    {
        fibo[idx] = fibo[idx-1] + fibo[idx-2];
    }
    printf("Fibo #%02x is %04x.\n", idx+1, fibo[idx]);
}
```

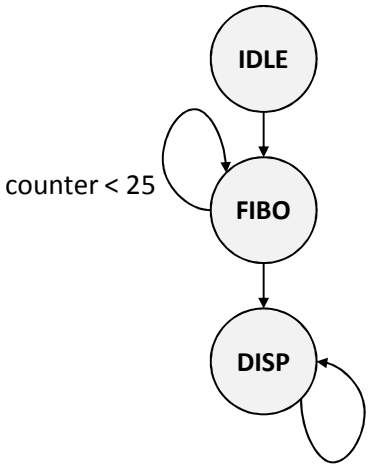
What to Do in Lab 4

- In Lab 4, it is mandatory to do the following things:
  - Design a circuit to compute and store the first 25 Fibonacci numbers in a register array
  - Once they are stored, the LCD will start displaying numbers:
    - Roughly every 0.7 sec, the LCD scrolls up one number cyclically
    - If BTN3 is pressed, the scrolling direction will be reversed (scroll-up becomes scroll-down, and vice versa)
  - Example display: cyclic scroll-up (numbers are hexadecimal)



Things to Try in Lab 4

- It is recommended that you design a simple finite-state machine to sequentialize the tasks




```
localparam [3:1] IDLE=3'b001, FIBO=3'b010, DISP=3'b100;
reg [3:1] Q, Q_next;

always @(posedge clk)
  if (reset) Q <= IDLE;
  else state <= Q_next;

always @* // next-state logic
  case (Q)
  IDLE: Q_next = FIBO;
  FIBO: Q_next = (counter < 25)? FIBO : DISP;
  DISP: Q_next = DISP;
  default: Q_next = Q_next;
endcase
```

- Design a periodic 0.7-second timer to control the scrolling, the tick count for 0.7 second is 7×10^7 clocks