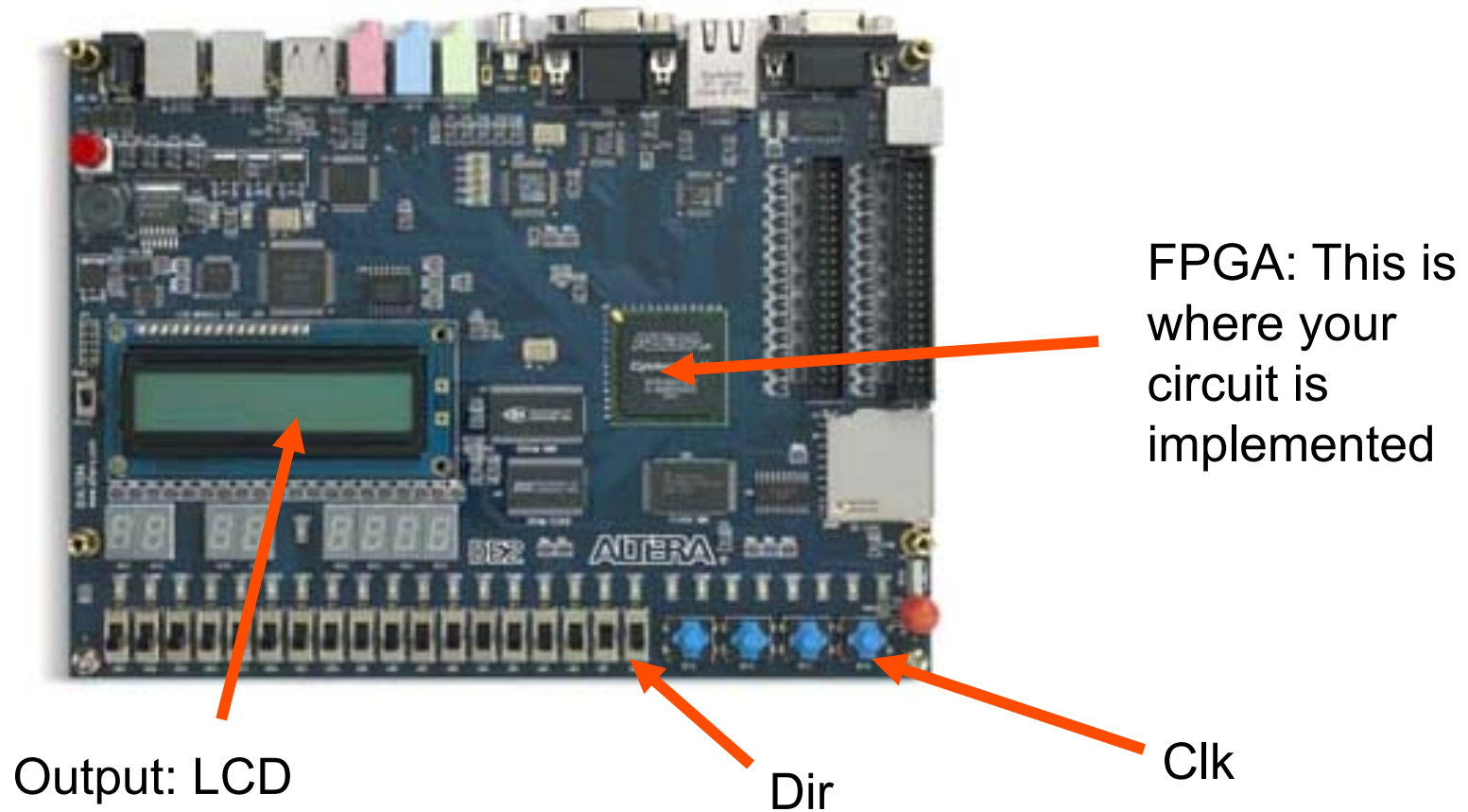


EECE 353: Digital Systems Design

Introduction to Lab 2

Lab 2:



What your circuit will do

Each clock cycle, your circuit will send one character to the LCD

- You will cycle through the first five characters of your name (and repeat)

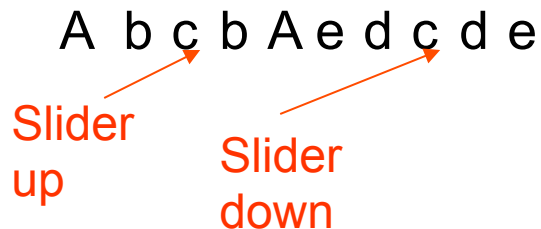
A b c d e A b c d e

If the DIR input is “up”, the LCD displays your name backwards (starting with the first character)

A e d c b A e d c b

DIR can be switched any time

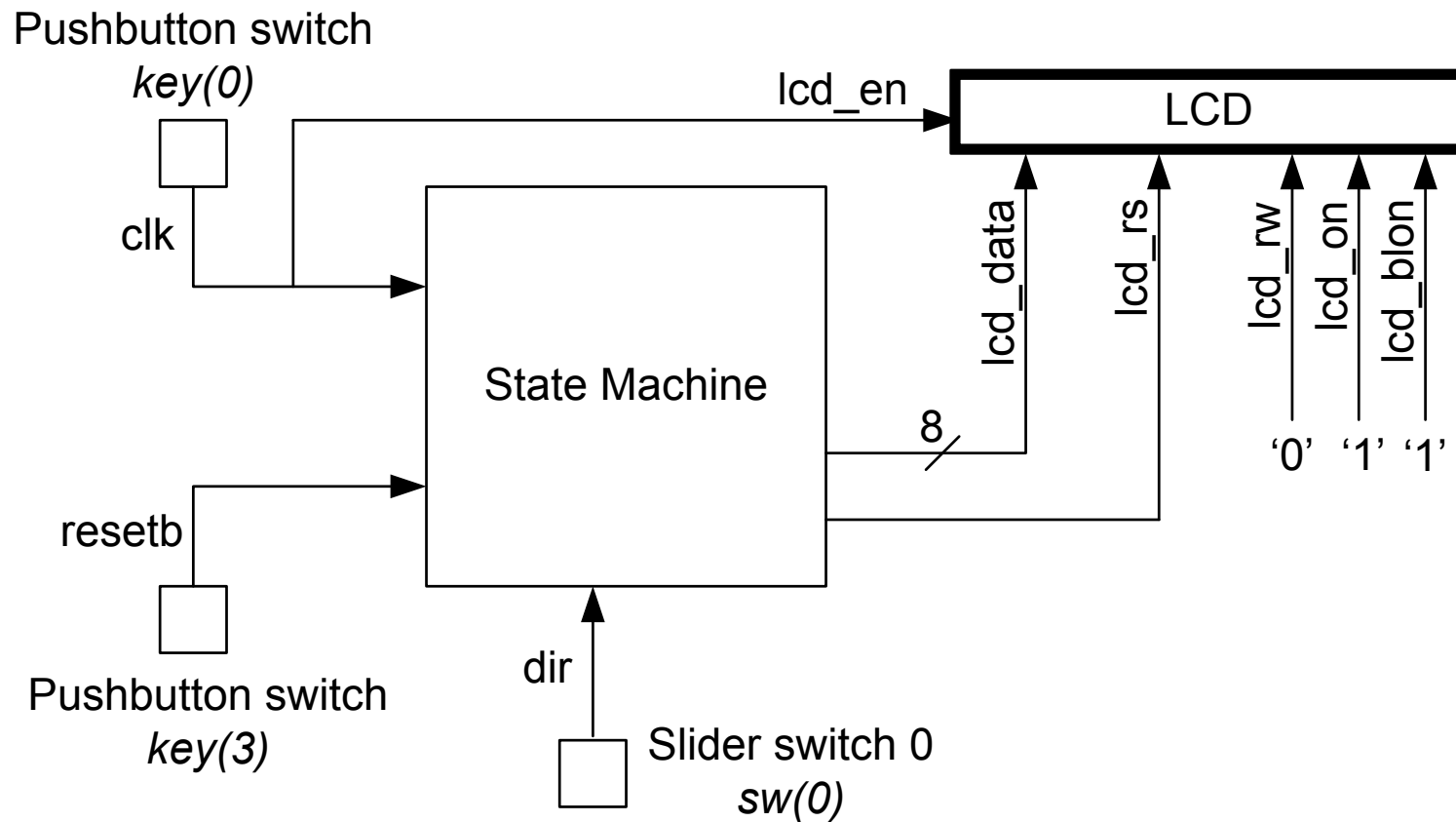
A b c b A e d c d e



Slider up Slider down

The diagram shows a sequence of characters: A b c b A e d c d e. Two orange arrows point from the text 'Slider up' to the third character 'c' and from 'Slider down' to the eighth character 'c'. This indicates that the DIR input is 'up' for the first three characters (A b c) and 'down' for the next five characters (b A e d c d e).

Structure of your circuit



Task 1: Learn how to use the LCD

Task 2: Implement the circuit using a key
as the clock

Task 3: Use the free-running 50MHz clock along
with a frequency divider

Challenge task: counting characters and clearing
the screen when it is full

LCD

In one clock cycle you can send one of:

- An eight-bit instruction, or
- an eight-bit piece of data

Instructions tell the LCD to do things like clear the screen, move the cursor, etc.

If you send a data byte, the corresponding character is displayed on the LCD.

The **lcd_data** input (8 bits) is used to send either the instruction or data

The **lcd_rs** input is: 0 if we are sending an instruction

1 if we are sending data

The full table of characters is in the lab handout:

Character	Code		Character	Code		Character	Code	
	Binary	Hex		Binary	Hex		Binary	Hex
Space	00100000	20	@	01000000	40	`	01100000	60
!	00100001	21	A	01000001	41	a	01100001	61
"	00100010	22	B	01000010	42	b	01100010	62
#	00100011	23	C	01000011	43	c	01100011	63
\$	00100100	24	D	01000100	44	d	01100100	64
%	00100101	25	E	01000101	45	e	01100101	65
&	00100110	26	F	01000110	46	f	01100110	66
'	00100111	27	G	01000111	47	g	01100111	67
(00101000	28	H	01001000	48	h	01101000	68
)	00101001	29	I	01001001	49	i	01101001	69
*	00101010	2A	J	01001010	4A	j	01101010	6A
+	00101011	2B	K	01001011	4B	k	01101011	6B
,	00101100	2C	L	01001100	4C	l	01101100	6C
-	00101101	2D	M	01001101	4D	m	01101101	6D
.	00101110	2E	N	01001110	4E	n	01101110	6E
/	00101111	2F	O	01001111	4F	o	01101111	6F
0	00110000	30	P	01010000	50	p	01110000	70
1	00110001	31	Q	01010001	51	q	01110001	71
2	00110010	32	R	01010010	52	r	01110010	72
3	00110011	33	S	01010011	53	s	01110011	73
4	00110100	34	T	01010100	54	t	01110100	74
5	00110101	35	U	01010101	55	u	01110101	75
6	00110110	36	V	01010110	56	v	01110110	76

Full set of instructions is in the LCD Datasheet

Instruction	Instruction Code										Description	Execution time (fosc=270KHz)
	R/S	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "00H" to DDRAM and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	—	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39 μ s
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and blinking of cursor (B) on/off control bit.	39 μ s
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	—	—	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39 μ s
Function Set	0	0	0	0	1	DL	N	F	—	—	Set interface data length (DL:8-bit/4-bit), numbers of display line (N:2-line/1-line)and, display font type (F:5 × 11 dots/5 × 8 dots)	39 μ s
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 μ s
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39 μ s
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 μ s
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43 μ s
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43 μ s

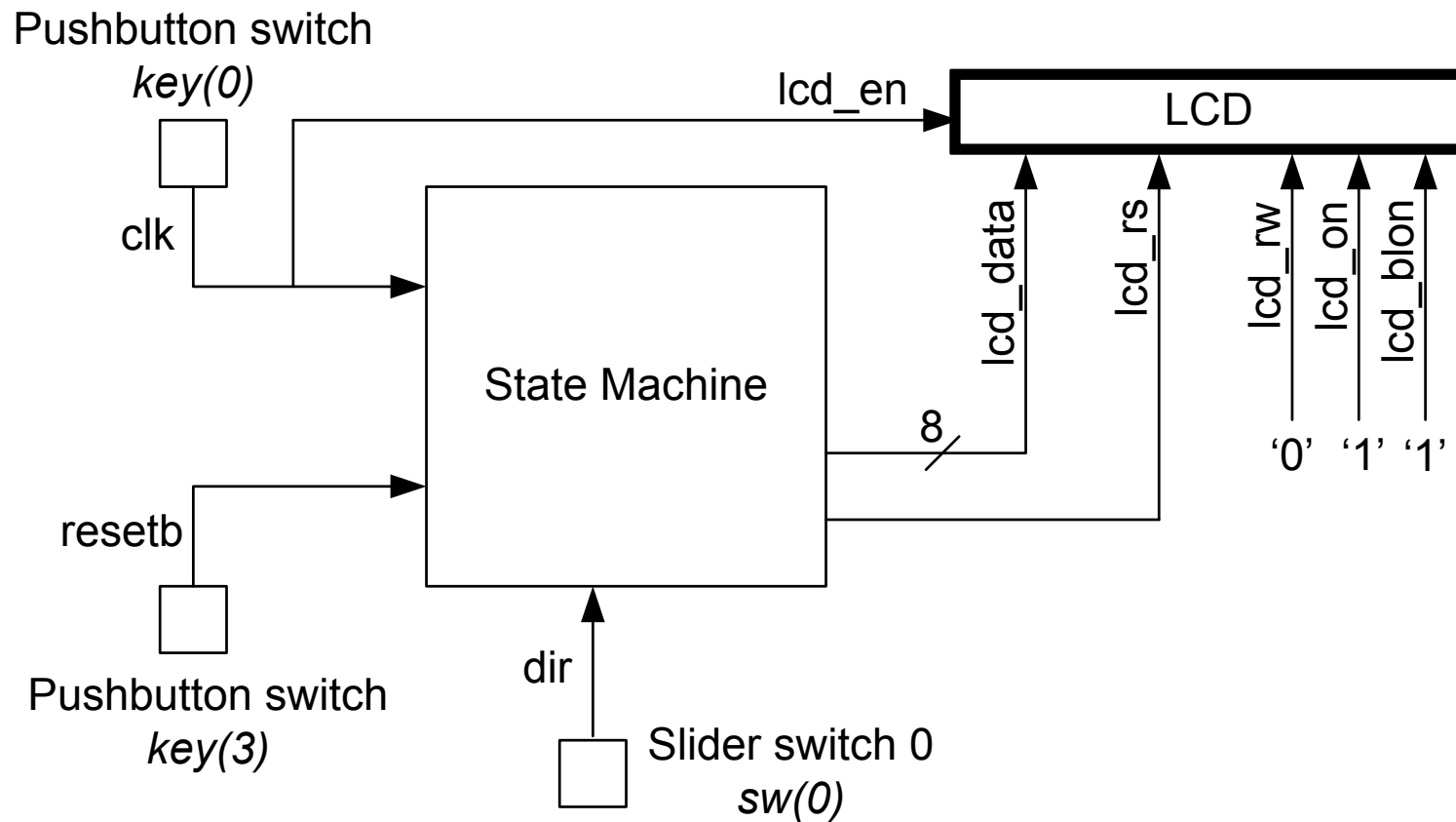
The instructions we care about:

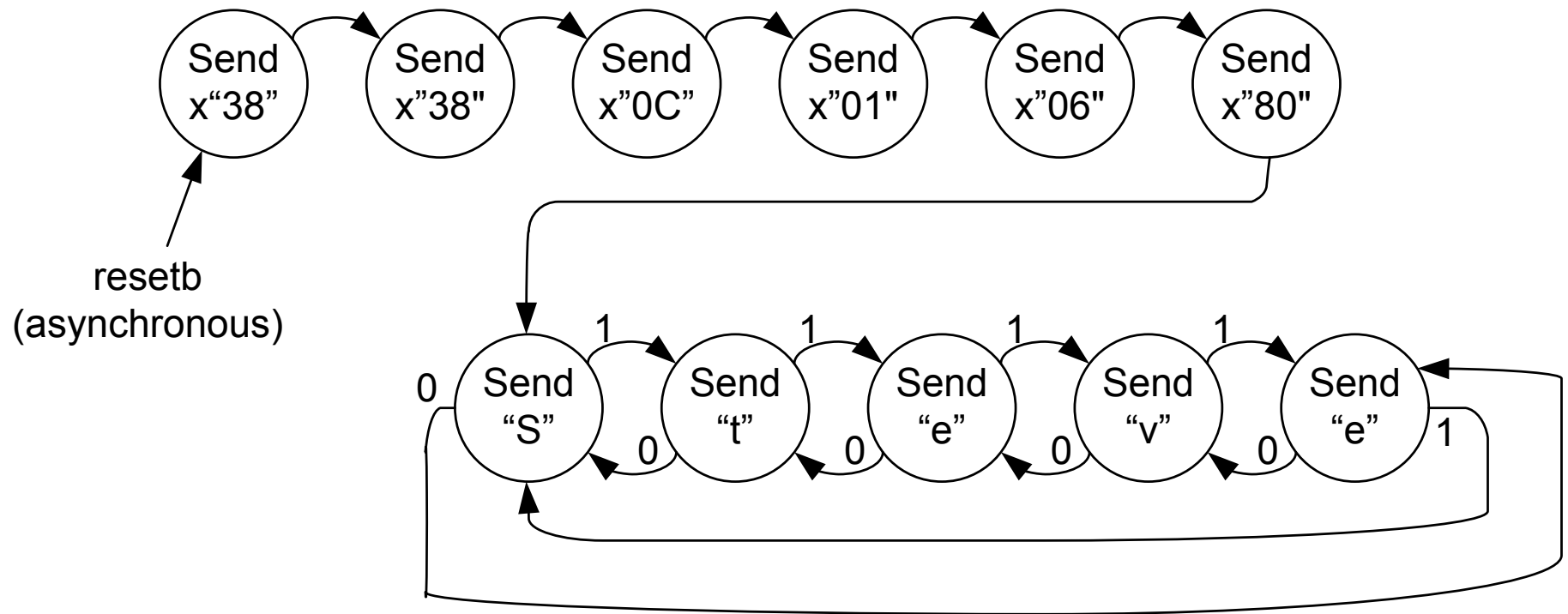
To initialize the LCD, clear the screen, and set it in the proper mode, using the following 6 instructions (in this order):

00111000 (hex "38") ← Actually, send this one twice
00001100 (hex "0C")
00000001 (hex "01")
00000110 (hex "06")
10000000 (hex "80")

So, the first thing your state machine should do is send these instructions. Send one per cycle. After the six cycles, your LCD is set up, and your state machine can send it data to display.

Structure of your circuit





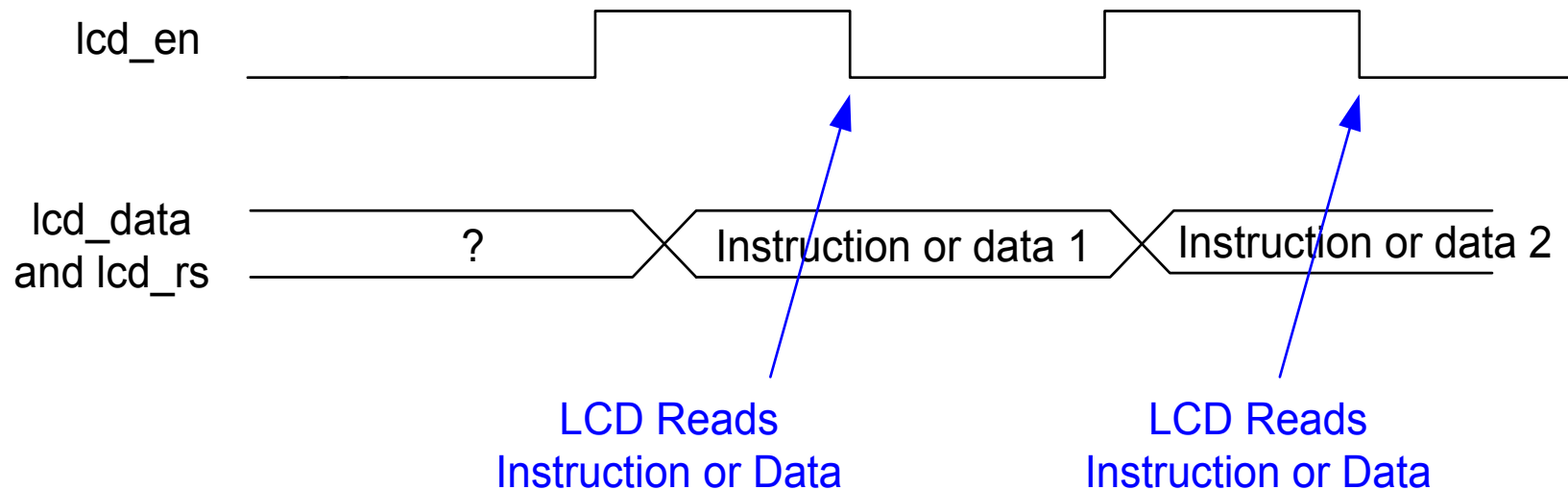
This is a Moore machine – you can design it with one process.

- States change on the rising clock edge
- Therefore, outputs change on the rising clock edge

One minor wrinkle...

Timing of the LCD inputs

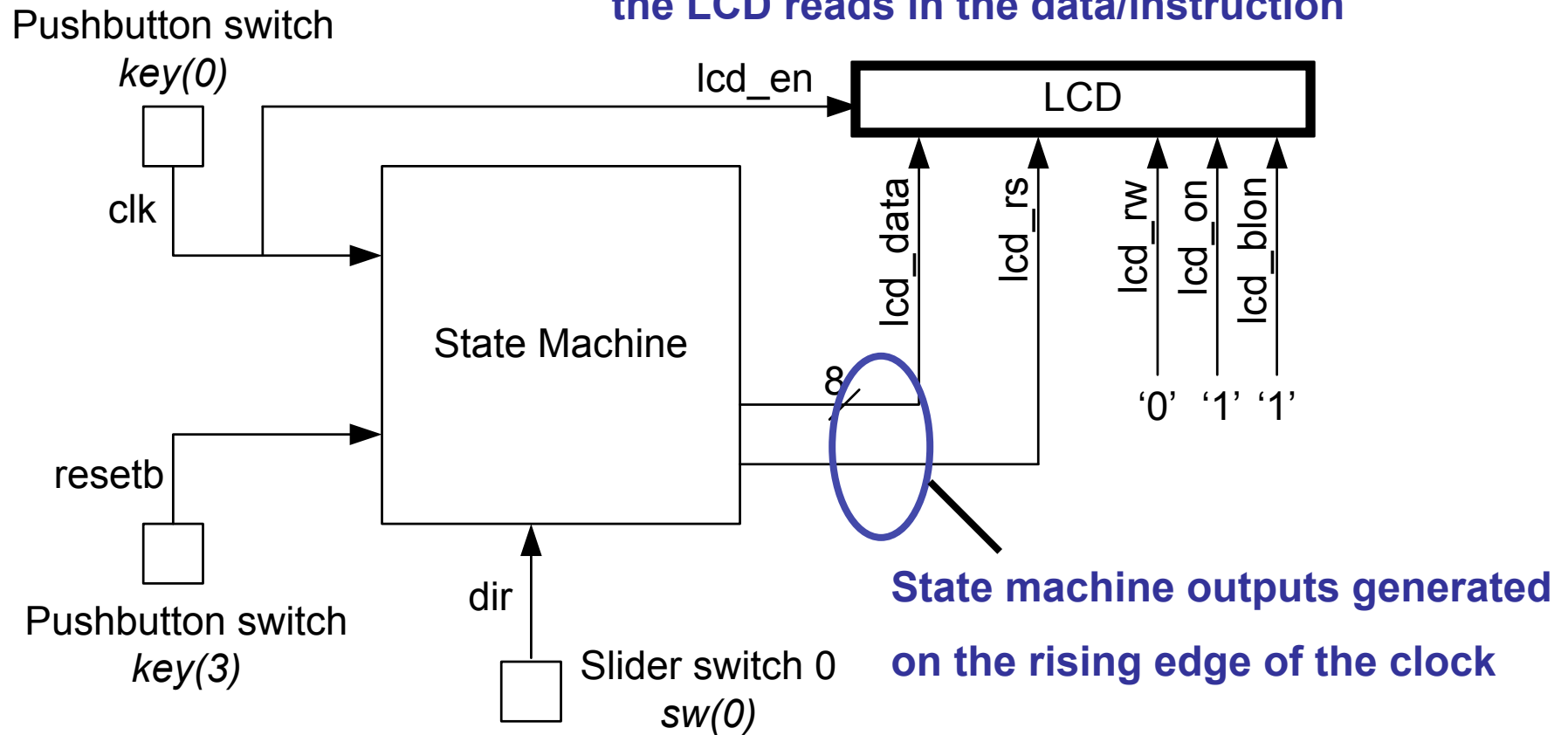
LCD Accepts data or instructions on the *falling edge* of **lcd_en**.



In Lab 2, our state machine will generate instructions/data words on the rising edge of the clock.

The LCD will then read them on the next falling edge.

**Half a cycle later, on the falling edge,
the LCD reads in the data/instruction**



Don't be confused: your task is to design a state machine that generates its outputs on the rising edge of the clock.

- This is like any other state machine you have ever designed

The fact that the LCD reads them on the falling edge really doesn't matter for your design.

test_lcd.vhd

You can download this from the web site. It will help you understand how the LCD works:

architecture behavioural of test_lcd is

begin

LCD_BLON <= '1'; -- backlight is always on

LCD_ON <= '1'; -- LCD is always on

LCD_EN <= KEY(0); -- connect the clock to the lcd_en input

LCD_RW <= '0'; -- always writing to the LCD

LCD_RS <= SW(8); -- connect to slider switch 8

LCD_DATA <= SW(7 downto 0); -- connect to slider switches 7-0

LEDG(0) <= KEY(0); -- send the clock to a green light

end behavioural;

Hints for this lab

1. Start by downloading **test_lcd.vhd** and spend a bit of time understanding how to use the LCD
 - Remember the push-button switches are active-low (0 = pushed in). This matters for your clock and reset inputs
2. Make sure you have your state machine designed and working (in simulation) before trying to download it to the board.
3. Send “important” values in your state machine to output signals, and connect these output signals to green lights.
 - For example, send the state bits to green lights, so you always know what state you are in. Makes debugging much easier!
4. Don't get too stressed ... remember, this is fun!

Challenge Task

Keep track of the number of characters printed and clear the screen when the screen is full

- The reason this is hard is because you don't know exactly what characters you will have displayed (because the user can change the sequence using SW(0) switch)

