

# SISTEMI EMBEDDED

SOPC DE2 Basic Computer

Parallel port

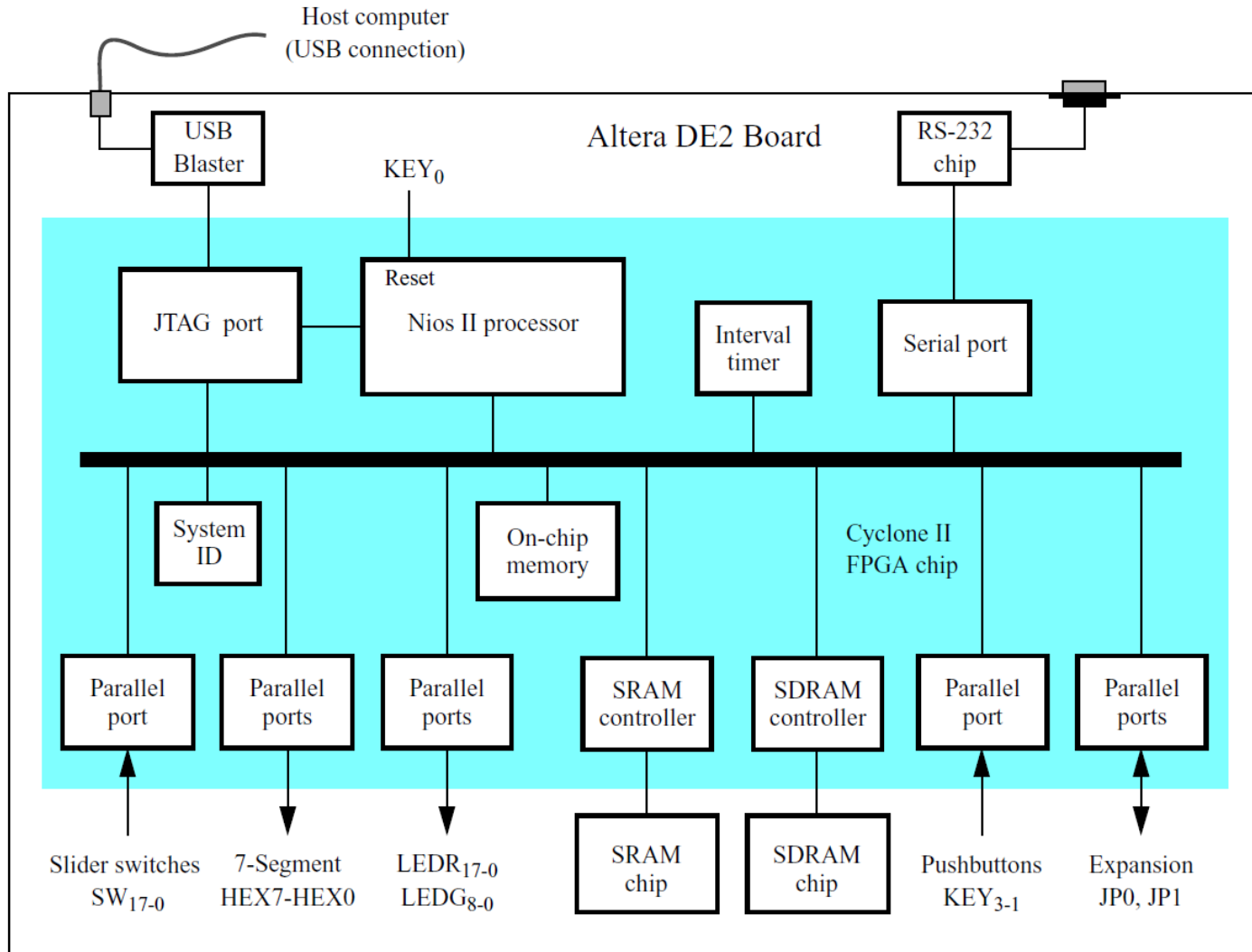
Federico Baronti

Last version: 20170306

# DE2 Basic Computer

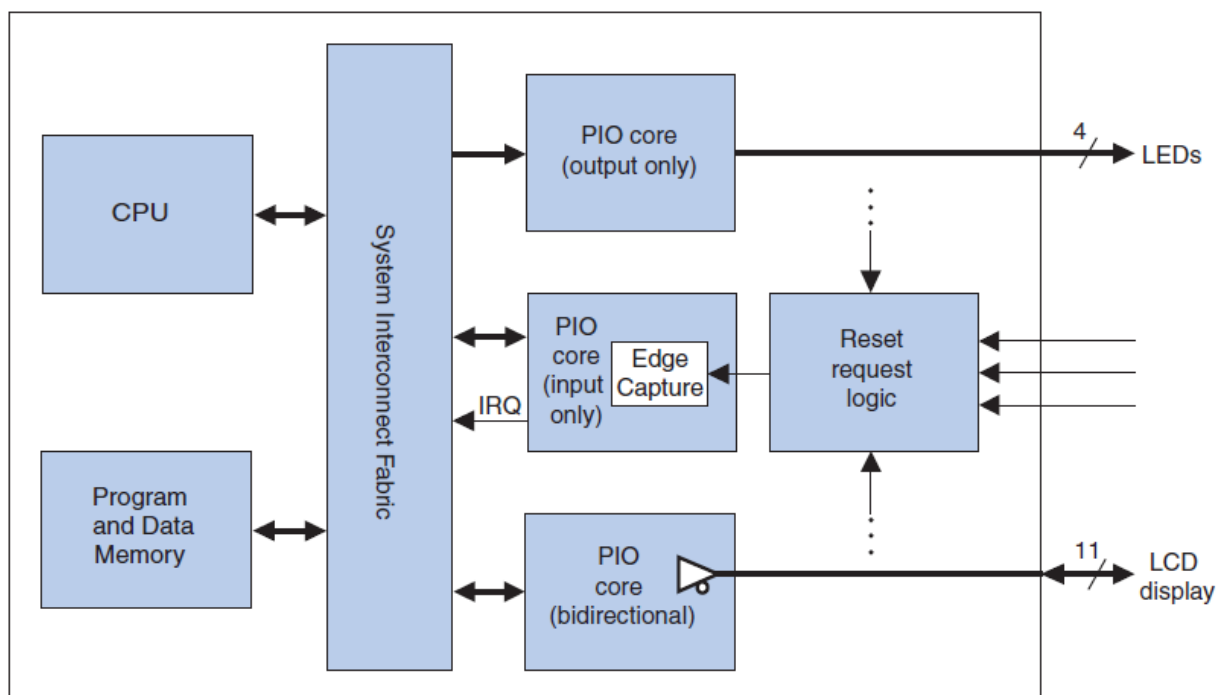
- Computer provided by Altera University Program
  - **Processor:** Nios II/e
  - **Memory:** SDRAM, SRAM, On-chip memory
  - **I/O:**
    - **Parallel ports:** red\_LEDs, Green\_LEDs, HEX3\_HEX0, HEX7\_HEX4, sliders, Pushbuttons, etc.
    - **Other peripherals:** JTAG UART, Serial\_port, Interval\_timer, sysid

# DE2 Basic Computer (cont.)



# Parallel port (1)

- Interface for general purpose I/O
  - Based on Altera's PIO core customized for DE-series boards
  - Controlling LEDs, acquiring data from Switches, etc.



# Parallel port (2)

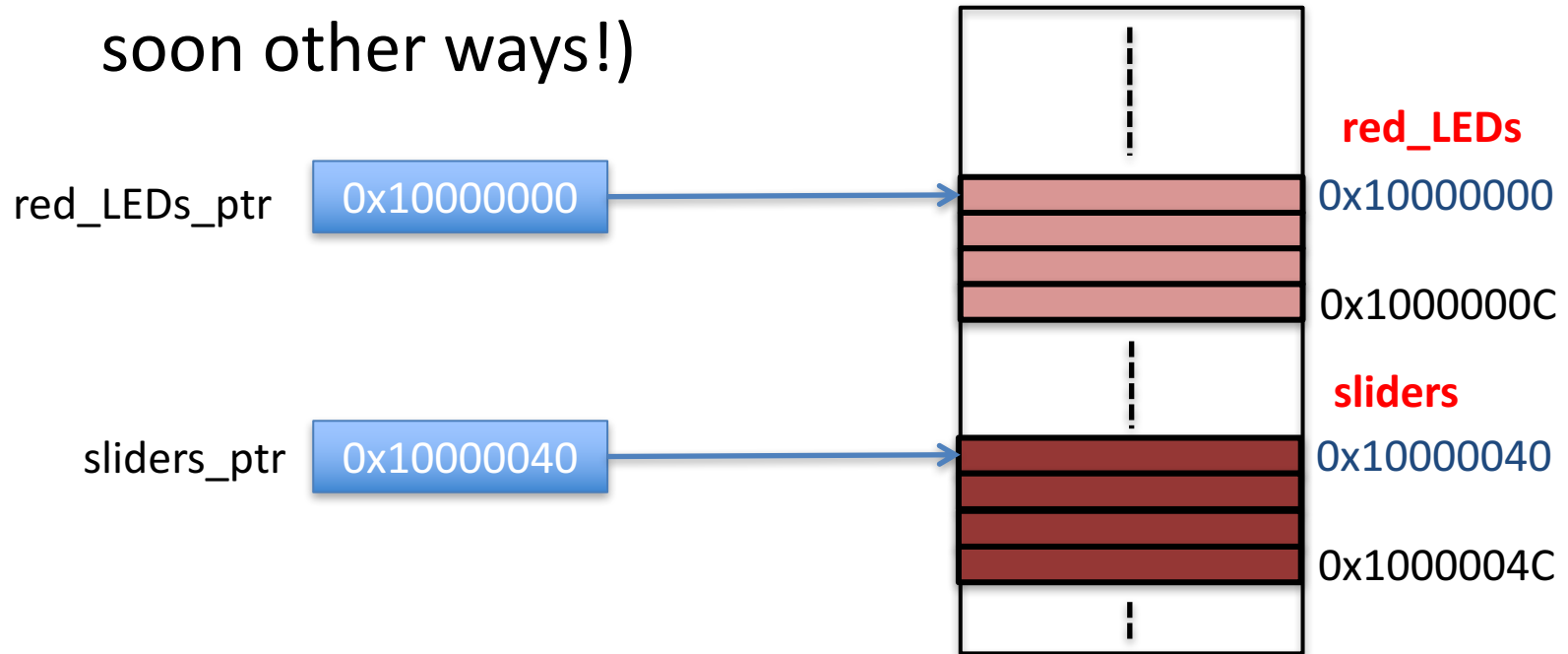
- 4x 32-bit memory-mapped registers
- $n$  actual number of I/O pins

*Table 2. Parallel Port register map*

Offset in bytes	Register name	Read/Write	Bits $(n-1)\dots 0$	
0	data	Input	R	Data value currently on Parallel Port inputs.
		Output	W	New value to drive on Parallel Port outputs.
4	direction	R/W	Individual direction control for each I/O port. A value of 0 sets the direction to input; 1 sets the direction to output.	
8	interruptmask	R/W	IRQ enable/disable for each input port. Setting a bit to 1 enables interrupts for the corresponding port.	
12	edgecapture	R/W	Edge detection for each input port.	

# Parallel port (3)

- **Managing PIO in C program:**
  - Use of pointers to *unsigned int* initialized with PIO base memory address (we'll learn soon other ways!)



```
volatile unsigned int *red_LED_ptr = (unsigned int *) 0x10000000; //red LED address
volatile unsigned int *sliders_ptr = (unsigned int *) 0x10000040;
//SW slider switch address
```

# Parallel port (4)

- **Why volatile attribute?**
  - I/O registers may change even if the program does not modify them!
    - The peripheral hardware may modify their contents
  - **Volatile** tells the compiler do not make any optimization to the code involving an object declared with the **volatile** attribute

# Parallel port (5)

- Reading/Writing I/O registers:

```
*red_LED_ptr = *slider_ptr;
```



# Putting into practice

- Let's start our first program with Nios II processor
  - Control the status of each DE2 red LED through the corresponding slider switch ( $LEDR_i = Sw_i$ )