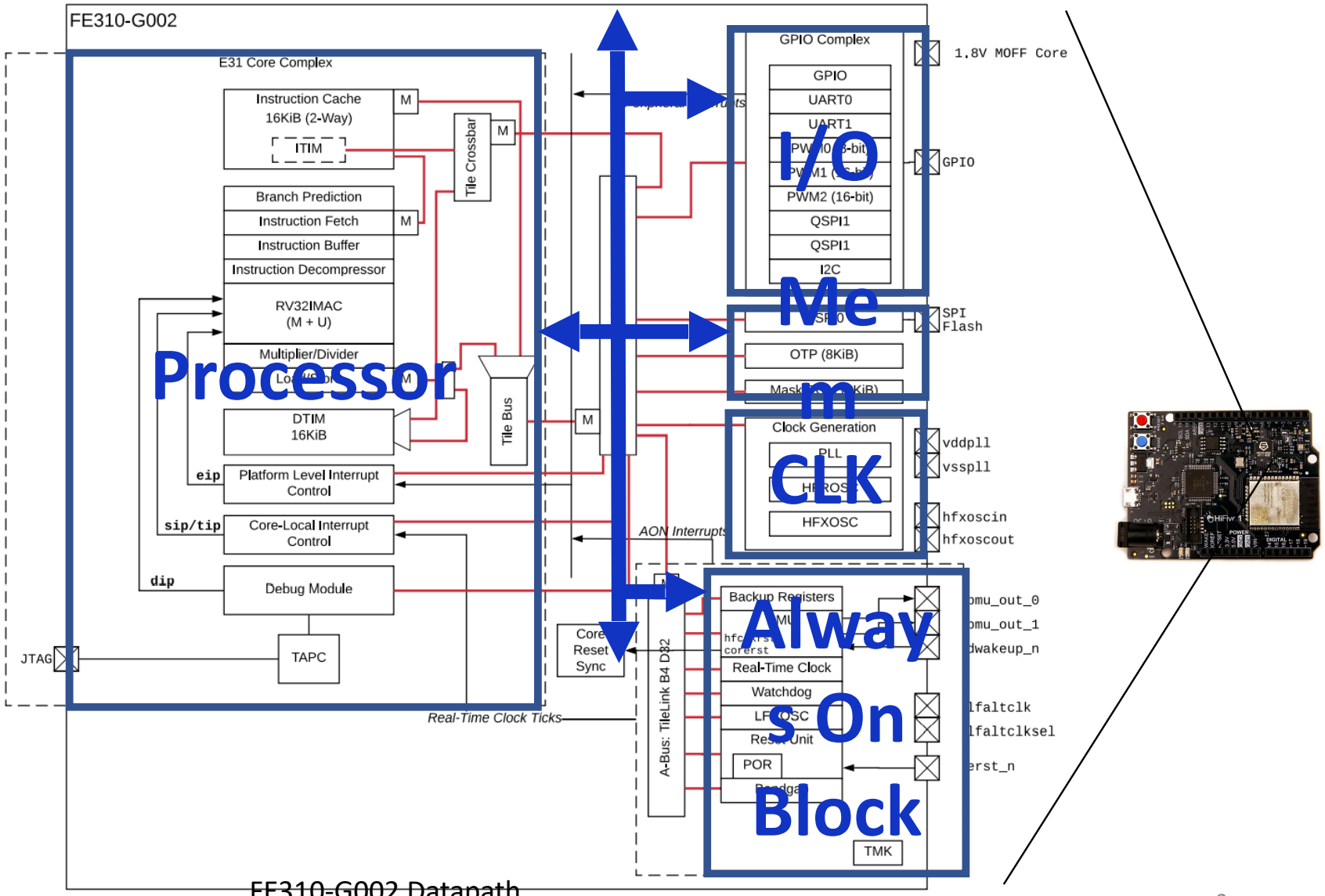
I/O Interface

EECS388 Spring 2023

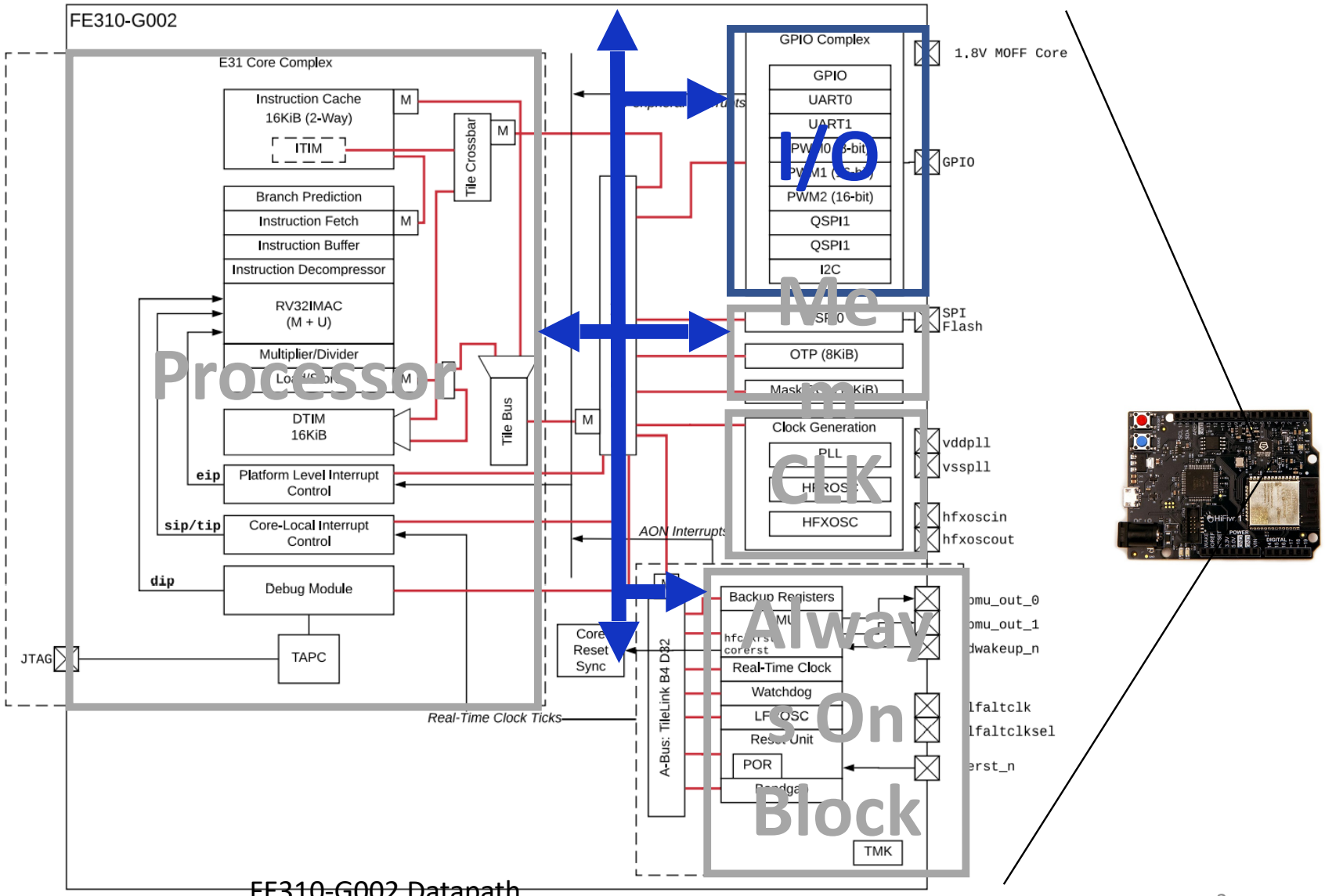
Lecture notes based in part on slides created by Dr. Mohammad Alian and Dr. Heechul Yun

System Bus



FE310-G002 Datapath

System Bus



FE310-G002 Datapath

3

How Does CPU Talk to Devices?

Port-mapped I/O

- Use a separate address space for I/O devices and use special instructions to access the I/O memory

Memory Mapped I/O

- I/O memory is mapped into the CPU address space
- Use load/store instruction to communicate with I/O devices

How Does CPU Talk to Devices?

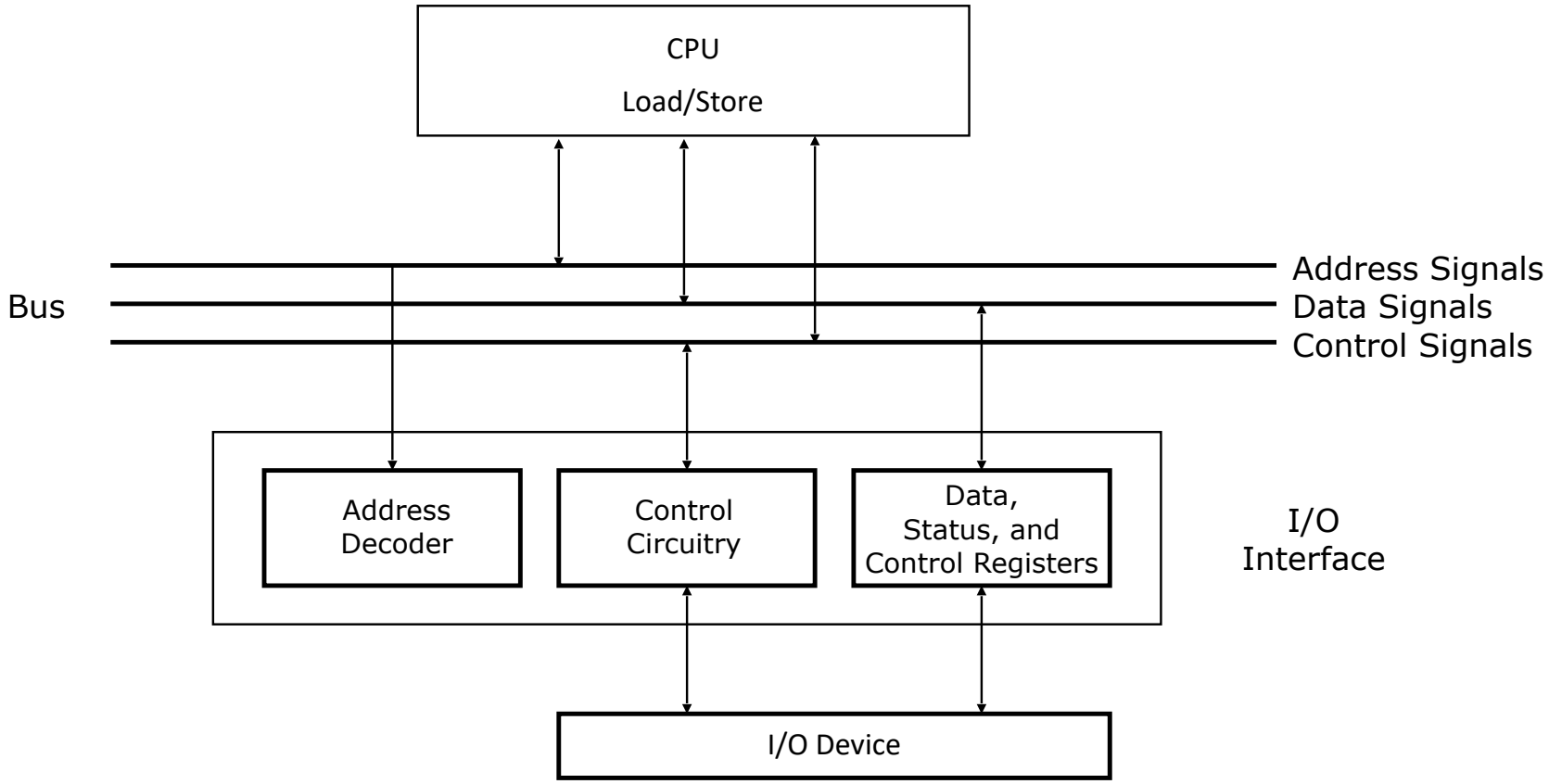
Port-mapped I/O

- Use a separate address space for I/O devices and use special instructions to access the I/O memory

Memory Mapped I/O

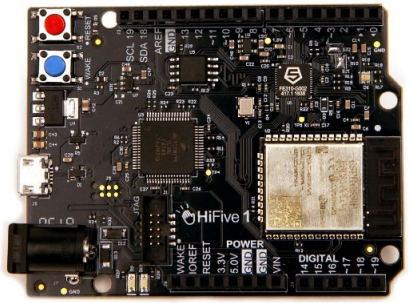
- I/O memory is mapped into the CPU address space
- Use load/store instruction to communicate with I/O devices

Memory Mapped I/O



Memory Map of SiFive FE310

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x0001_2000	0x0001_FFFF		Reserved	
0x0002_0000	0x0002_1FFF	R XC	OTP Memory Region	
0x0002_2000	0x001F_FFFF		Reserved	
0x0200_0000	0x0200_FFFF	RW A	CLINT	On-Chip Peripherals
0x0201_0000	0x07FF_FFFF		Reserved	
0x0800_0000	0x0800_1FFF	RWX A	E31 ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF		Reserved	
0x0C00_0000	0x0FFF_FFFF	RW A	PLIC	
0x1000_0000	0x1000_0FFF	RW A	AON	
0x1000_1000	0x1000_7FFF		Reserved	
0x1000_8000	0x1000_8FFF	RW A	PRCI	
0x1000_9000	0x1000_FFFF		Reserved	
0x1001_0000	0x1001_0FFF	RW A	OTP Control	
0x1001_1000	0x1001_1FFF		Reserved	
0x1001_2000	0x1001_2FFF	RW A	GPIO	
0x1001_3000	0x1001_3FFF	RW A	UART 0	
0x1001_4000	0x1001_4FFF	RW A	QSPI 0	
0x1001_5000	0x1001_5FFF	RW A	PWM 0	
0x1001_6000	0x1001_6FFF	RW A	I2C 0	
0x1001_7000	0x1002_2FFF		Reserved	
0x1002_3000	0x1002_3FFF	RW A	UART 1	
0x1002_4000	0x1002_4FFF	RW A	SPI 1	
0x1002_5000	0x1002_5FFF	RW A	PWM 1	
0x1002_6000	0x1003_3FFF		Reserved	
0x1003_4000	0x1003_4FFF	RW A	SPI 2	
0x1003_5000	0x1003_5FFF	RW A	PWM 2	
0x1003_6000	0x1FFF_FFFF		Reserved	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	



Memory mapped I/O regions

Memory Map of SiFive FE310

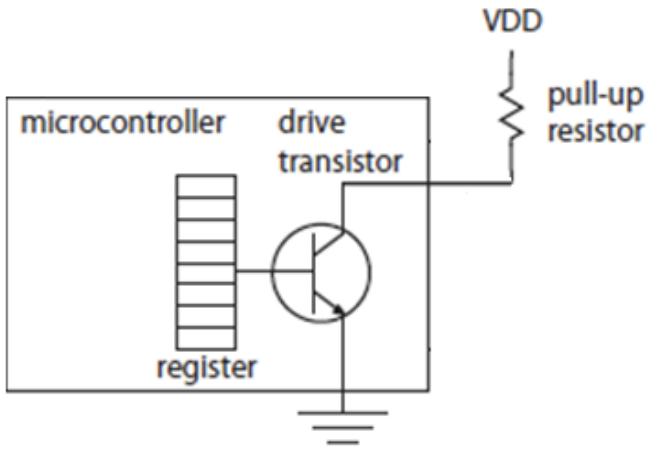
Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x0001_2000	0x0001_FFFF		Reserved	
0x0002_0000	0x0002_1FFF	R XC	OTP Memory Region	
0x0002_2000	0x001F_FFFF		Reserved	
0x0200_0000	0x0200_FFFF	RW A	CLINT	On-Chip Peripherals
0x0201_0000	0x07FF_FFFF		Reserved	
0x0800_0000	0x0800_1FFF	RWX A	E31 ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF		Reserved	
0x0C00_0000	0x0FFF_FFFF	RW A	PLIC	
0x1000_0000	0x1000_0FFF	RW A	AON	
0x1000_1000	0x1000_7FFF		Reserved	
0x1000_8000	0x1000_8FFF	RW A	PRCI	
0x1000_9000	0x1000_FFFF		Reserved	
0x1001_0000	0x1001_0FFF	RW A	OTP Control	
0x1001_1000	0x1001_1FFF		Reserved	
0x1001_2000	0x1001_2FFF	RW A	GPIO	
0x1001_3000	0x1001_3FFF	RW A	UART 0	
0x1001_4000	0x1001_4FFF	RW A	QSPI 0	
0x1001_5000	0x1001_5FFF	RW A	PWM 0	
0x1001_6000	0x1001_6FFF	RW A	I2C 0	
0x1001_7000	0x1002_2FFF		Reserved	
0x1002_3000	0x1002_3FFF	RW A	UART 1	
0x1002_4000	0x1002_4FFF	RW A	SPI 1	
0x1002_5000	0x1002_5FFF	RW A	PWM 1	
0x1002_6000	0x1003_3FFF		Reserved	
0x1003_4000	0x1003_4FFF	RW A	SPI 2	
0x1003_5000	0x1003_5FFF	RW A	PWM 2	
0x1003_6000	0x1FFF_FFFF		Reserved	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	

GPIO registers are mapped at 0x10012000 – 0x10012FFF

Offset	Name	Description
0x00	input_val	Pin value
0x04	input_en	Pin input enable*
0x08	output_en	Pin output enable*
0x0C	output_val	Output value
0x10	pue	Internal pull-up enable*
0x14	ds	Pin drive strength
0x18	rise_ie	Rise interrupt enable
0x1C	rise_ip	Rise interrupt pending
0x20	fall_ie	Fall interrupt enable
0x24	fall_ip	Fall interrupt pending
0x28	high_ie	High interrupt enable
0x2C	high_ip	High interrupt pending
0x30	low_ie	Low interrupt enable
0x34	low_ip	Low interrupt pending
0x40	out_xor	Output XOR (invert)

General Purpose I/O (GPIO)

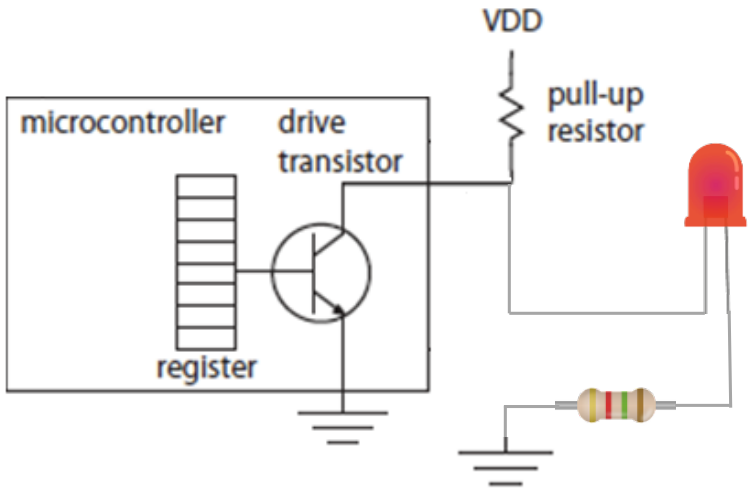
- Programmable digital input/output pins
- Use voltage levels to represent digital signals
  - 5V = logic 1 (for 5V CPUs)
  - 0V = logic 0
- Can be configured as
  - Input or output
- Useful to interact with external devices



9

Example: Turn on an LED

- Assume a GPIO pin can draw up to 18mA, and when the LED is on, it has a voltage drop of 1V
- Ohm's law: $I \times R = V$
- What resistor is needed?



Example: Turn on an LED

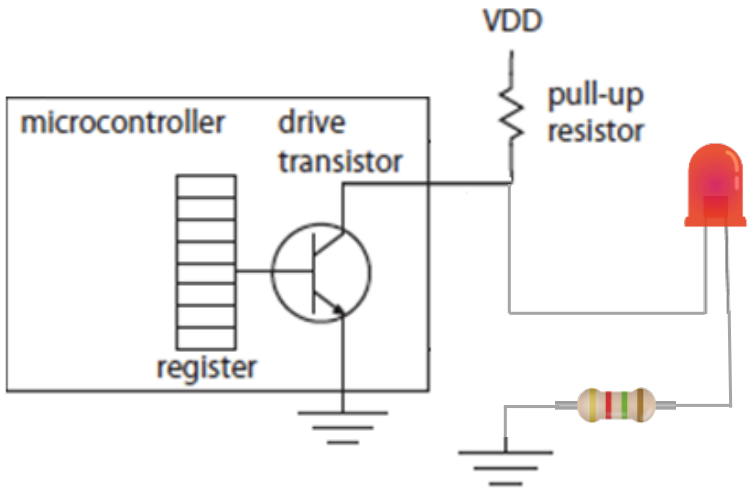
- Assume a GPIO pin can draw up to 18mA, and when the LED is on, it has a voltage drop of 1V
- Ohm's law: $I \times R = V$
- What resistor is needed?

  I = V / R = 1 / R < 18mA

  R > 1V/18mA

  = 1000mV/18mA

  = 56 ohm



11

Blinking the LED from lab 1

- We are blinking the GREEN LED by turning it ON & OFF
- But we don't see the how internally the configuration is done
- We need to look at the lib.h and lib.c to see what is gpio_mode & gpio write

```c
#include <stdint.h>

#include "eecs388_lib.h"

int main()
{
    int gpio = GREEN_LED;

    gpio_mode(gpio, OUTPUT);

    while(1)
    {
        gpio_write(gpio, ON);
        delay(1000);
        gpio_write(gpio, OFF);
        delay(300);
    }
}
```

Memory map of GPIO

0x1001_1000	0x1001_1FFF			Reserved
0x1001_2000	0x1001_2FFF	RW	A	GPIO
0x1001_3000	0x1001_3FFF	RW	A	UART 0
0x1001_4000	0x1001_4FFF	RW	A	QSPI 0
0x1001_5000	0x1001_5FFF	RW	A	PWM 0
0x1001_6000	0x1001_6FFF	RW	A	I2C 0
0x1001_7000	0x1002_2FFF			Reserved
0x1002_3000	0x1002_3FFF	RW	A	UART 1
0x1002_4000	0x1002_4FFF	RW	A	SPI 1
0x1002_5000	0x1002_5FFF	RW	A	PWM 1
0x1002_6000	0x1003_3FFF			Reserved

On-Chip Peripherals

Base Address of GPIO PINS

13

Memory MAP of GPIO

Offset from the Base address we saw in last slide

To config. GPIO as output, write 1 in this address (base+offset)

Offset	Name	Description
0x00	input_val	Pin value
0x04	input_en	Pin input enable*
0x08	output_en	Pin output enable*
0x0C	output_val	Output value
0x10	pue	Internal pull-up enable*
0x14	ds	Pin drive strength
0x18	rise_ie	Rise interrupt enable
0x1C	rise_ip	Rise interrupt pending
0x20	fall_ie	Fall interrupt enable
0x24	fall_ip	Fall interrupt pending
0x28	high_ie	High interrupt enable
0x2C	high_ip	High interrupt pending
0x30	low_ie	Low interrupt enable
0x34	low_ip	Low interrupt pending
0x40	out_xor	Output XOR (invert)

14

Memory MAP of GPIO

Offset from the Base address we saw in last slide

To config. GPIO as output, write 1 in this address (base+offset)

Offset	Name	Description
0x00	input_val	Pin value
0x04	input_en	Pin input enable*
0x08	output_en	Pin output enable*
0x0C	output_val	Output value
0x10	pue	Internal pull-up enable*
0x14	ds	Pin drive strength
0x18	rise_ie	Rise interrupt enable

```
/************************************************************************
 *    memory map
 ************************************************************************/
#define GPIO_CTRL_ADDR      0x10012000  // GPIO controller base address
#define GPIO_INPUT_VAL      0x00        // input val
#define GPIO_INPUT_EN       0x04        // input enable
#define GPIO_OUTPUT_EN      0x08        // output enable
#define GPIO_OUTPUT_VAL     0x0C        // output_val
#define GPIO_OUTPUT_XOR     0x40        // output XOR (invert)

#define CLINT_CTRL_ADDR     0x02000000  // CLINT block base address
#define CLINT_MTIME         0xbff8      // timer register
```

These addresses are Defined with alternative key words inside lib.h file

15

Inside Lib.c

2. We are first reading from output_val to val

3. Then we change 1 bit to val

4. Then we update output_val with val

1. Base + offset denotes to address of 'output_en' in mem map

```
void gpio_mode(int gpio, int mode)
{
  uint32_t val;

  if (mode == OUTPUT) {
    val = *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_EN);
    val |= (1<<gpio);
    *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_EN) = val;

    if (gpio == RED_LED || gpio == GREEN_LED || gpio == BLUE_LED) {
      // active high
      val = *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_XOR);
      val |= (1<<gpio);
      *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_XOR) = val;
    }
```

Inside lib.h

```
/*****************************************************************************
 *   hifive1 platform related definitions
 *****************************************************************************/
#define RED_LED             22 // gpio 22
#define BLUE_LED            21 // gpio 21
#define GREEN_LED           19 // gpio 19
```

In lib.h all names are assigned to different GPIO port for the ease of use

Inside lib.c

```
    if (gpio == RED_LED || gpio == GREEN_LED || gpio == BLUE_LED) {
      // active high
      val = *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_XOR);
      val |= (1<<gpio);
      *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_XOR) = val;
    }
```

Inside lib.c we can say gpio=RED_LED

17

Memory Map of SiFive FE310

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x0001_2000	0x0001_FFFF		Reserved	
0x0002_0000	0x0002_1FFF	R XC	OTP Memory Region	
0x0002_2000	0x001F_FFFF		Reserved	
0x0200_0000	0x0200_FFFF	RW A	CLINT	On-Chip Peripherals
0x0201_0000	0x07FF_FFFF		Reserved	
0x0800_0000	0x0800_1FFF	RWX A	E31 ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF		Reserved	
0x0C00_0000	0x0FFF_FFFF	RW A	PLIC	
0x1000_0000	0x1000_0FFF	RW A	AON	
0x1000_1000	0x1000_7FFF		Reserved	
0x1000_8000	0x1000_8FFF	RW A	PRCI	
0x1000_9000	0x1000_FFFF		Reserved	
0x1001_0000	0x1001_0FFF	RW A	OTP Control	
0x1001_1000	0x1001_1FFF		Reserved	
0x1001_2000	0x1001_2FFF	RW A	GPIO	
0x1001_3000	0x1001_3FFF	RW A	UART 0	
0x1001_4000	0x1001_4FFF	RW A	QSPI 0	
0x1001_5000	0x1001_5FFF	RW A	PWM 0	
0x1001_6000	0x1001_6FFF	RW A	I2C 0	
0x1001_7000	0x1002_2FFF		Reserved	
0x1002_3000	0x1002_3FFF	RW A	UART 1	
0x1002_4000	0x1002_4FFF	RW A	SPI 1	
0x1002_5000	0x1002_5FFF	RW A	PWM 1	
0x1002_6000	0x1003_3FFF		Reserved	
0x1003_4000	0x1003_4FFF	RW A	SPI 2	
0x1003_5000	0x1003_5FFF	RW A	PWM 2	
0x1003_6000	0x1FFF_FFFF		Reserved	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	

UART0 registers are mapped at 0x10013000 – 0x10013FFF

Offset	Name	Description
0x00	txdata	Transmit data register
0x04	rxdata	Receive data register
0x08	txctrl	Transmit control register
0x0C	rxctrl	Receive control register
0x10	ie	UART interrupt enable
0x14	ip	UART interrupt pending
0x18	div	Baud rate divisor

Volatile in C

- When we use volatile, compiler generates code to re-load the variable item each time it is referenced in your program.
- Without volatile, the compiler may generate code that merely re-uses the value it already loaded into a register.

```
void ser_write(char c)
{
  uint32_t regval;
  /* busy-wait if TX FIFO is full  */
  do {
    regval = *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA);
  } while (regval & 0x80000000);

  /* write the character */
  *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA) = c;
}
```

volatile uint32_t

uint32_t

```
ser_write:
        li      a4,268513280
.L17:
        lw      a5,0(a4)
        bltz    a5,.L17
        sw      a0,0(a4)
        ret
```

a4 = 0x10013000

a5 = *a4

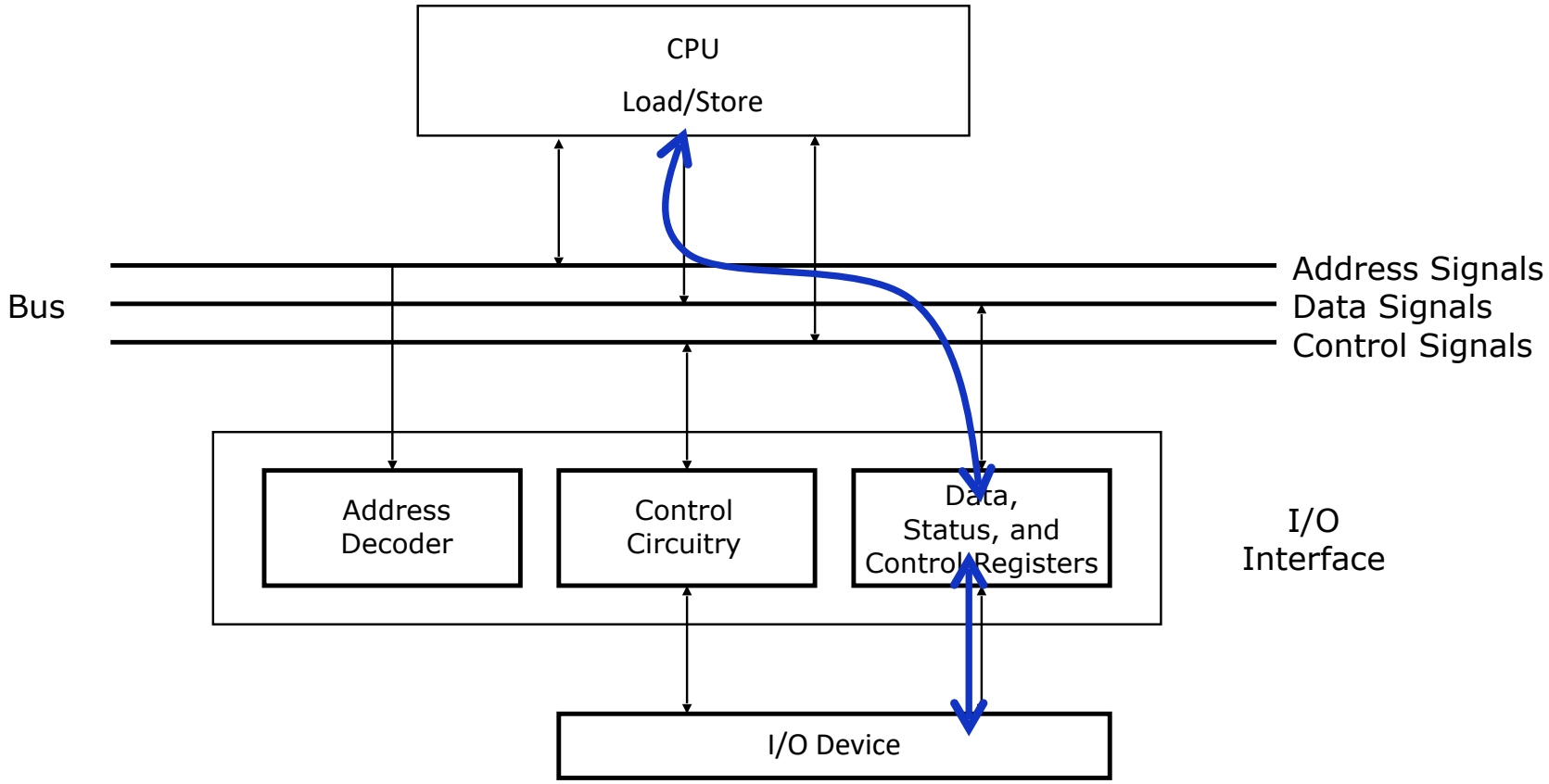
Branch to .L17 if a5 < zero

```
ser_write:
        li      a5,268513280
        lw      a5,0(a5)
.L17:
        bltz    a5,.L17
        li      a5,268513280
        sw      a0,0(a5)
```

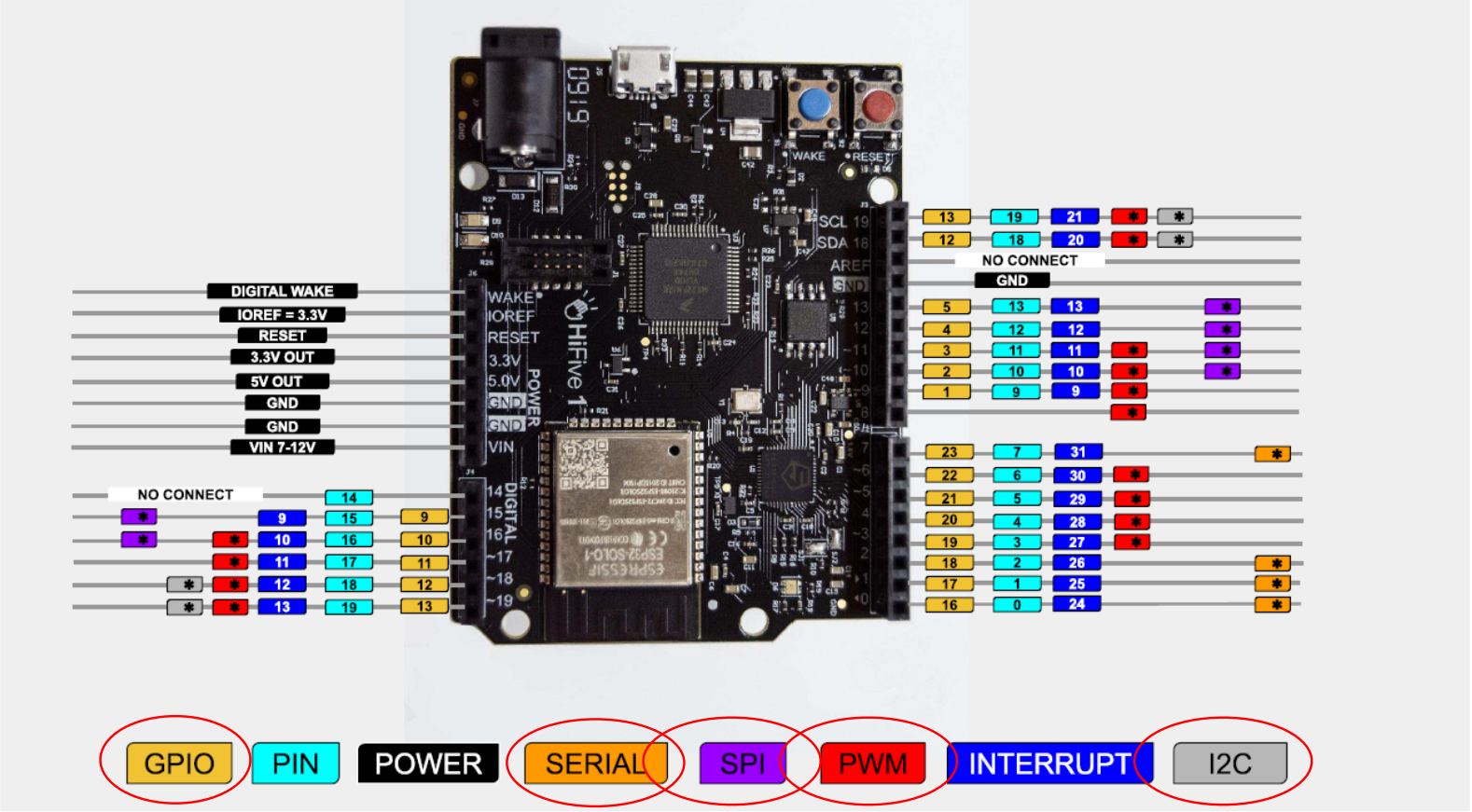
- When used volatile, the lw happens inside the loop every time the loop occurs.

19

I/O registers/memory can be updated by both CPU and I/O device



External I/O Interfaces

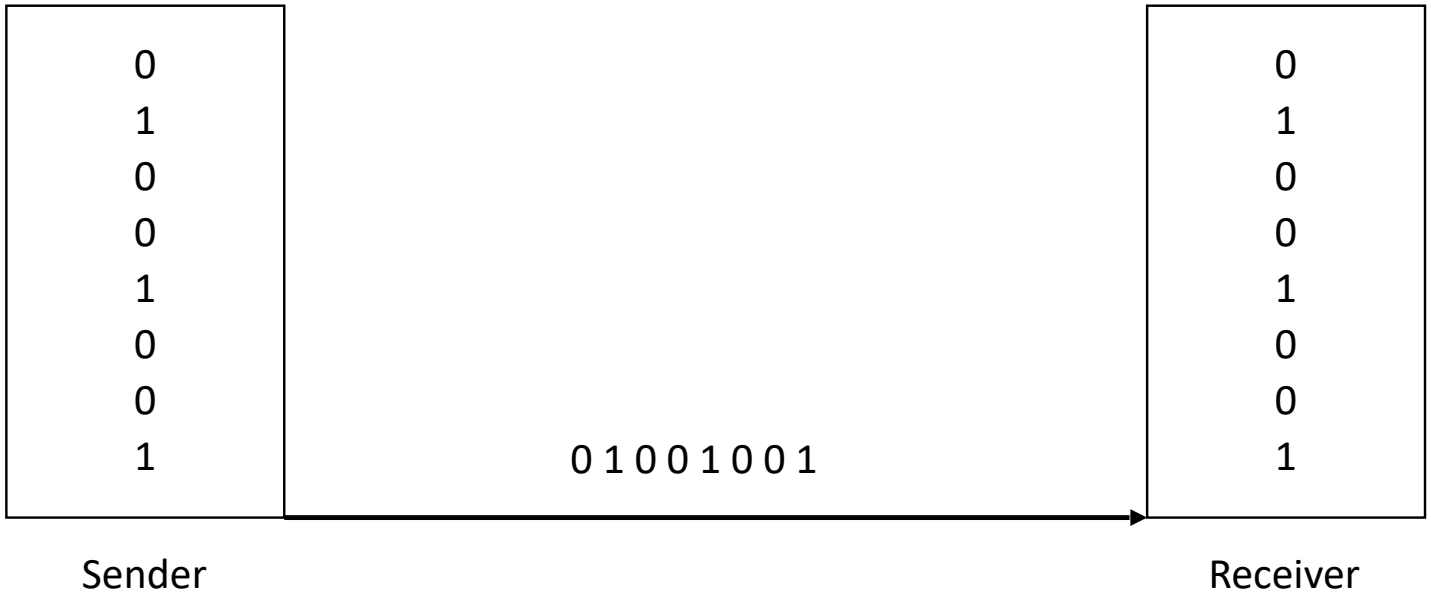


HiFive1 Rev B Pinout

21

Serial vs. Parallel

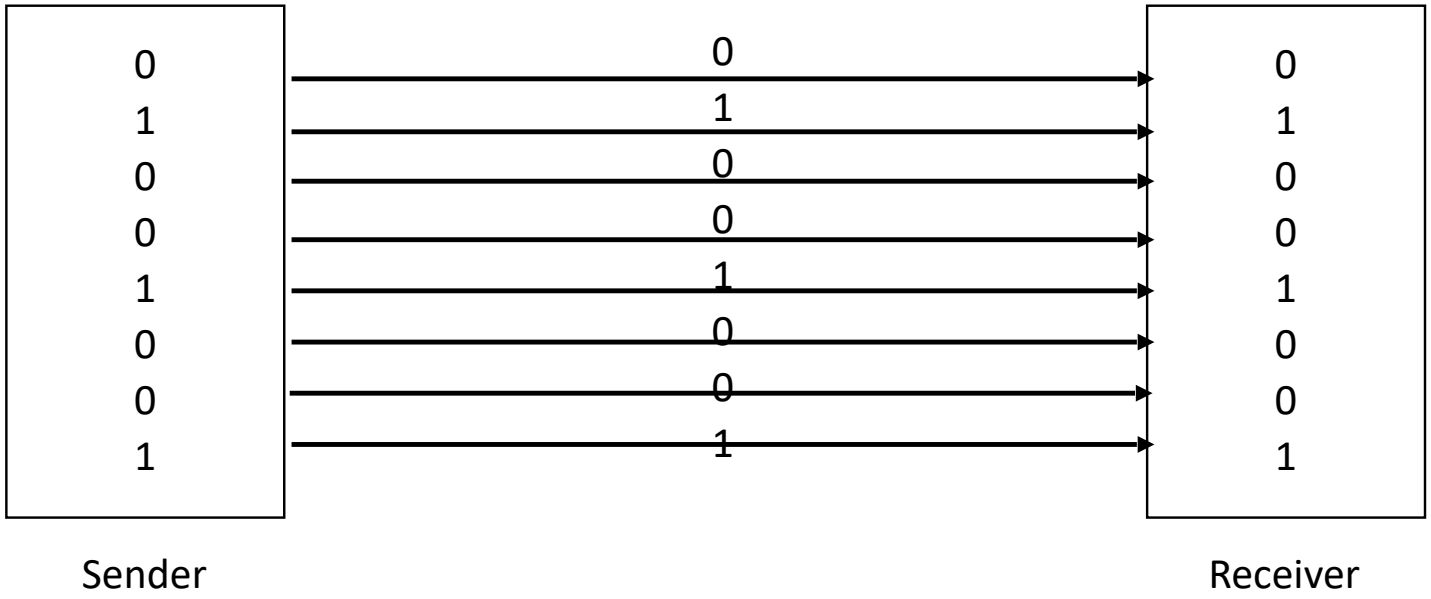
Serial interfaces use a single line



22

Serial vs. Parallel

Parallel interfaces use multiple lines

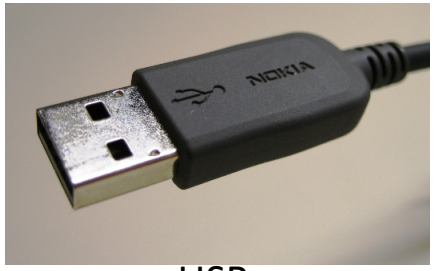


Serial vs. Parallel Interfaces

- Serial interfaces
  - RS-232: serial communication standard
  - USB: universal serial bus
  - I²C: inter-integrated circuit
  - SPI: serial peripheral interface bus
  - SATA: serial ATA
  - …

I²C

SATA



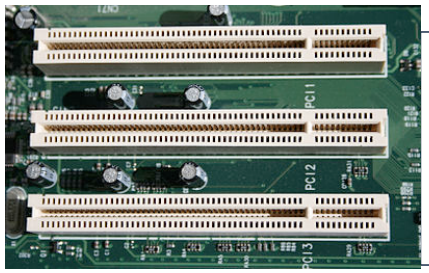
USB


RS-232

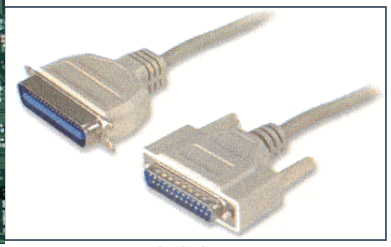
24

Serial vs. Parallel Interfaces

- Parallel interfaces
  - Parallel ATA: advanced technology attachment
  - SCSI: small computer system interface
  - PCI: peripheral component interface
  - …



PCI



SCSI

PATA

Parallel port

25

Serial Interfaces

- ✓ Use fewer pins and wires
- ✓ High scalability(more device connection)
- ✓ Low power consumption
- ✗ Lower bandwidth for the same clock speed

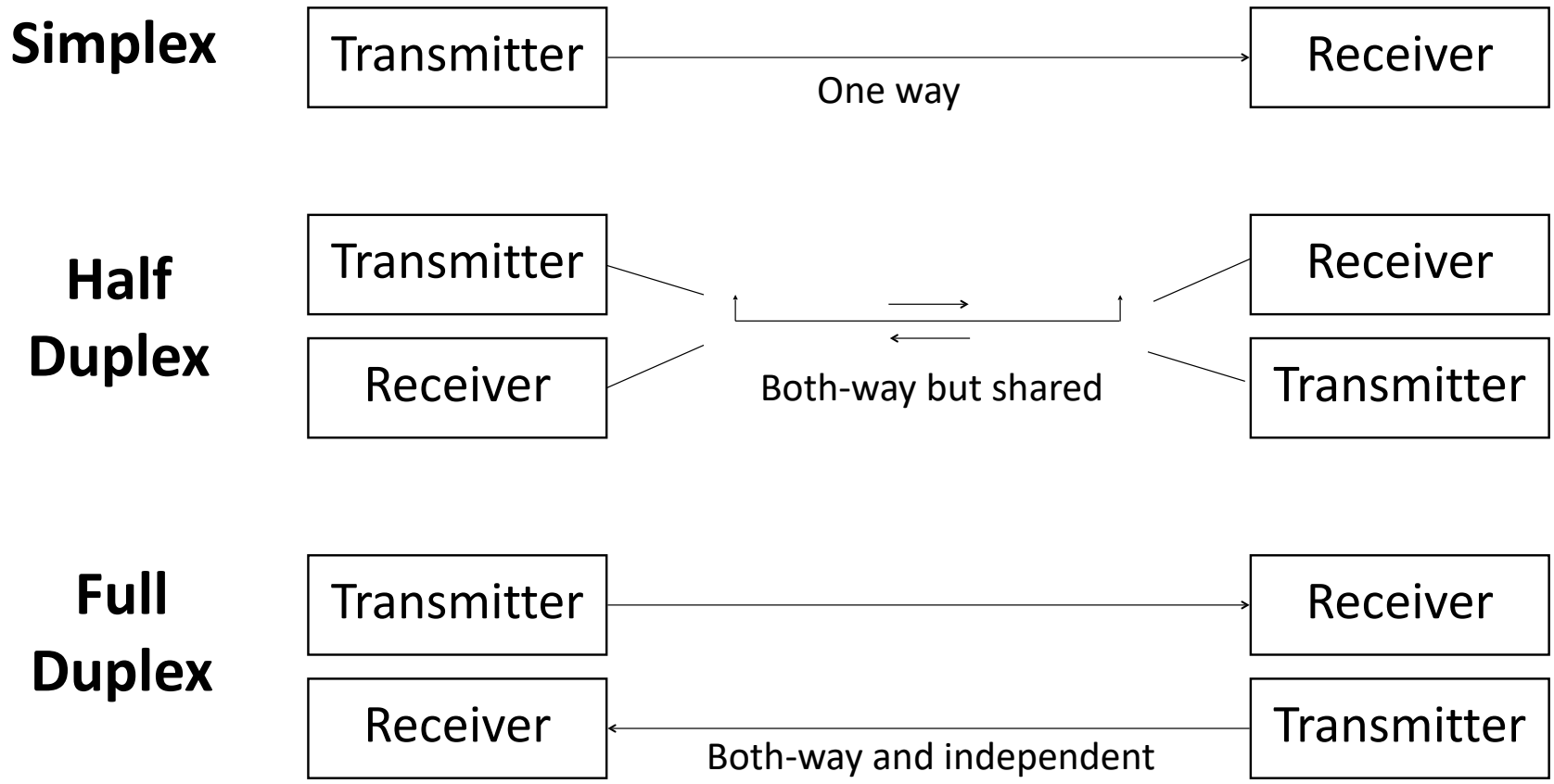
Parallel Interfaces

- ✗ More pins and wires
- ✗ Synchronization among lanes reduces scalability
- ✗ High power consumption
- ✓ Higher bandwidth

- Serial interfaces for external I/O (e.g., USB, SATA, PCIe)
- Parallel interfaces for on-chip interconnects and memory interfaces (e.g., AXI, TileLink, DDR)

26

Transfer Types

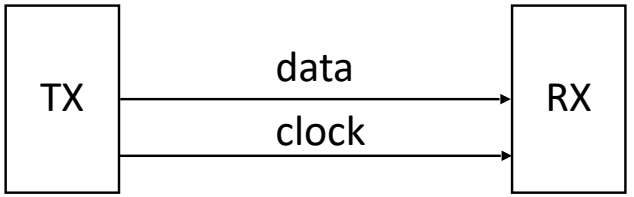


27

Synchronous vs. Asynchronous

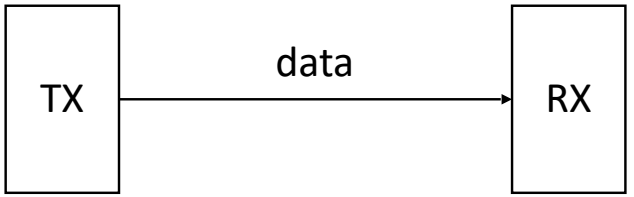
Synchronous transmission

- Requires a common shared clock
- Higher throughput communication
- Low scalability
- Parallel interfaces are usually sync



Asynchronous transmission

- No shared clock
- Asynchronous start/stop
- Self clocked, based on an agreement between the transmitter and receiver



28