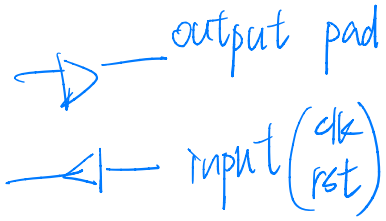
Bridge of Life Education

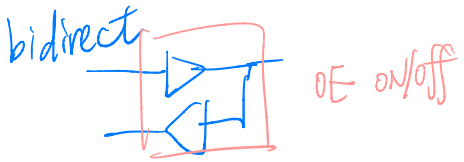
SOC Design
Peripheral - GPIO

Jiin Lai

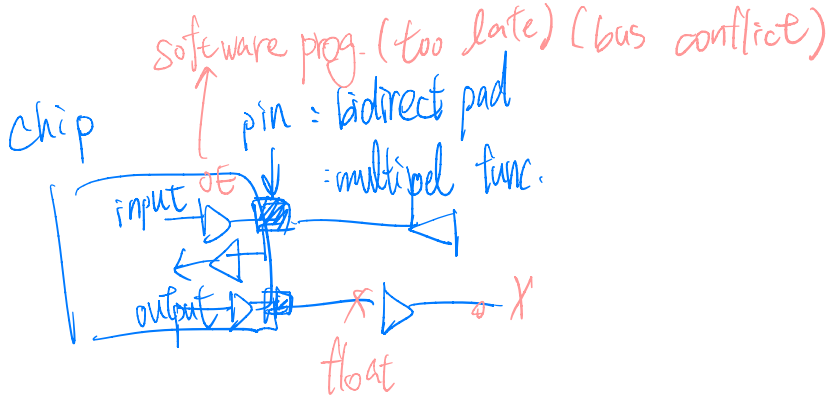
Topics



General IO pad

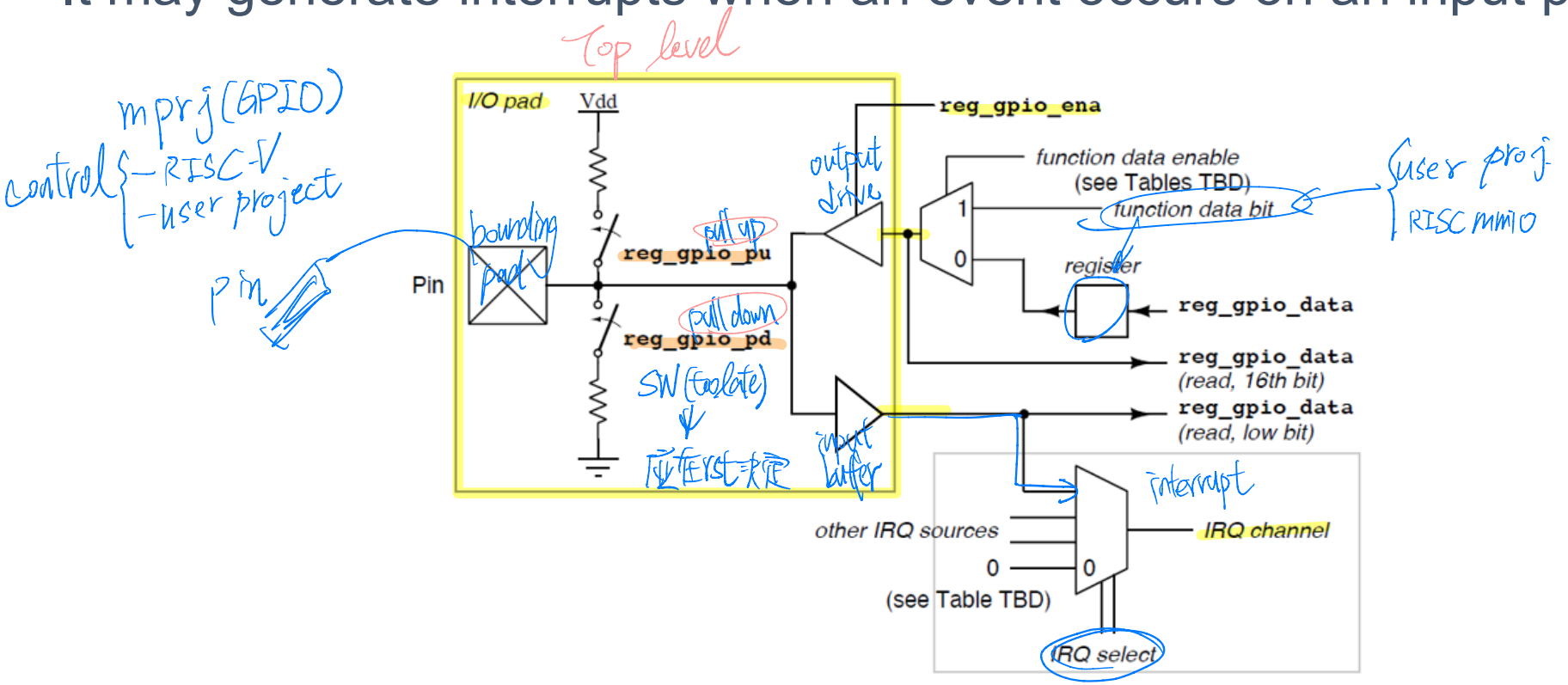


- GPIO Introduction
- GPIO During RESET
- Bit banging - Software to transmit/receive data

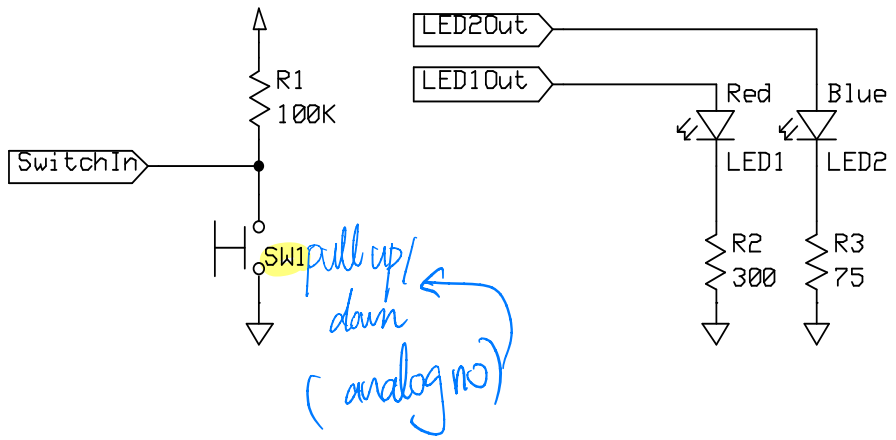


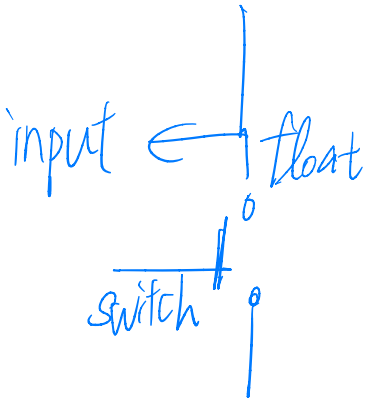
GPIO Overview

- CPU use **memory-mapped IO register** to read or write digital signals, and set the direction of the pin
- The GPIO pins can be shared with one or more special-purpose peripherals
- It may generate interrupts when an event occurs on an input pin

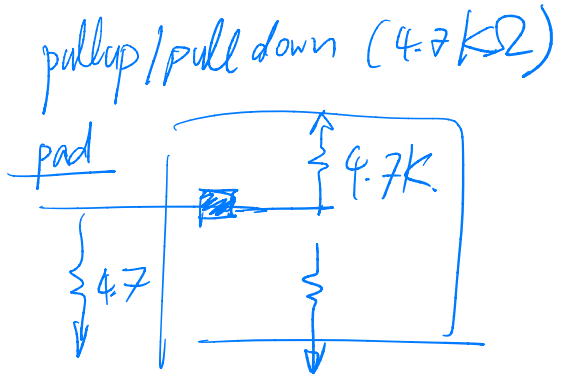


Example Usage : Switch input / LED output



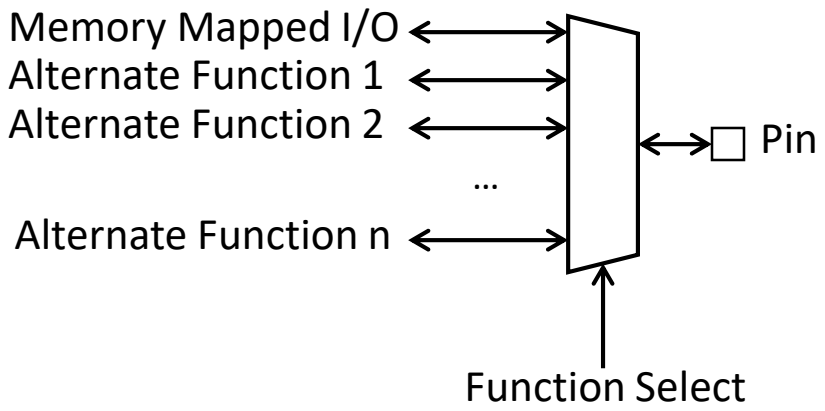


- Goal: light either LED1 or LED2 based on switch SW1 position
- GPIO
  - Input: program can determine if input signal is a 1 or a 0
  - Output: program can set output to 1 or 0
- Can use this to interface with external devices
  - Input: switch
  - Output: LEDs

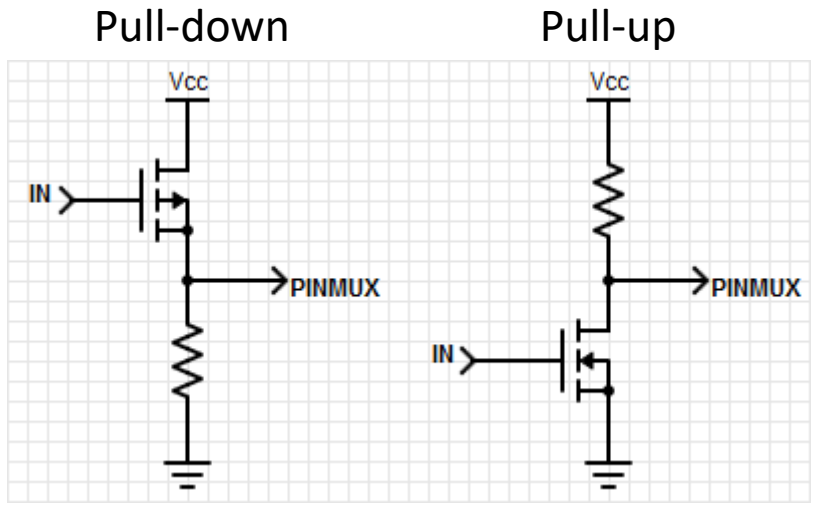


GPIO Alternative Functions

- Pins may have different features
- To enable an alternative function, set up the appropriate register
- May also have analogue paths for ADC / DAC etc.
- Advantages:
  - Saves space on the package
  - Improves flexibility



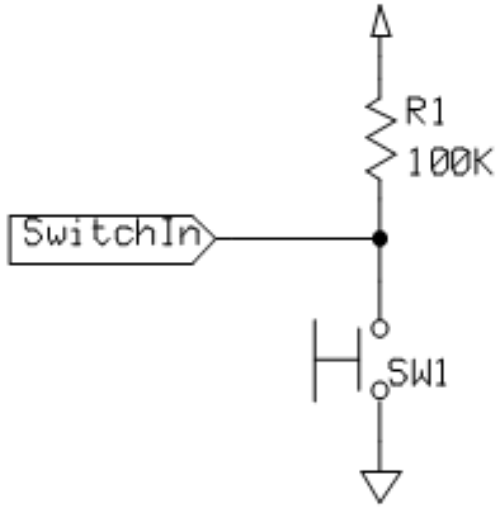
Pull-Up & Pull-Down Resistors



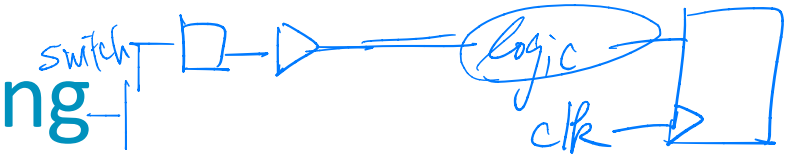
- Ensure a known value on the output if a pin is left floating

no

- In this example, we want the switch SW1 to pull the pin to ground, so we enable the pull-up
- The pin value is:
  - High when SW1 is not pressed
  - Low when SW1 is pressed

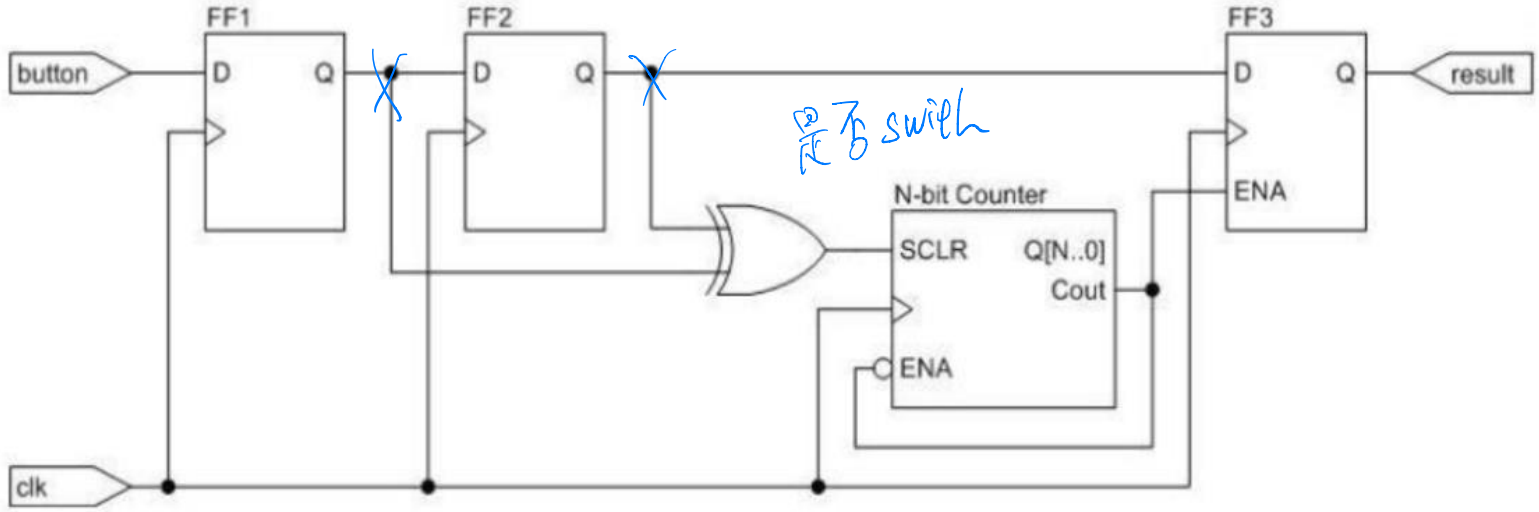


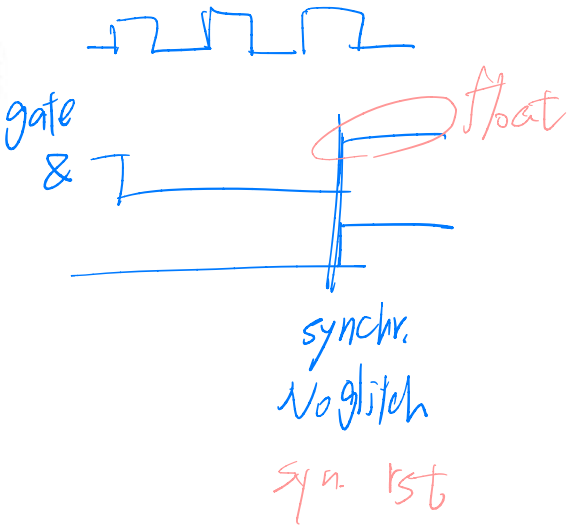
Input Synchronization & Debouncing



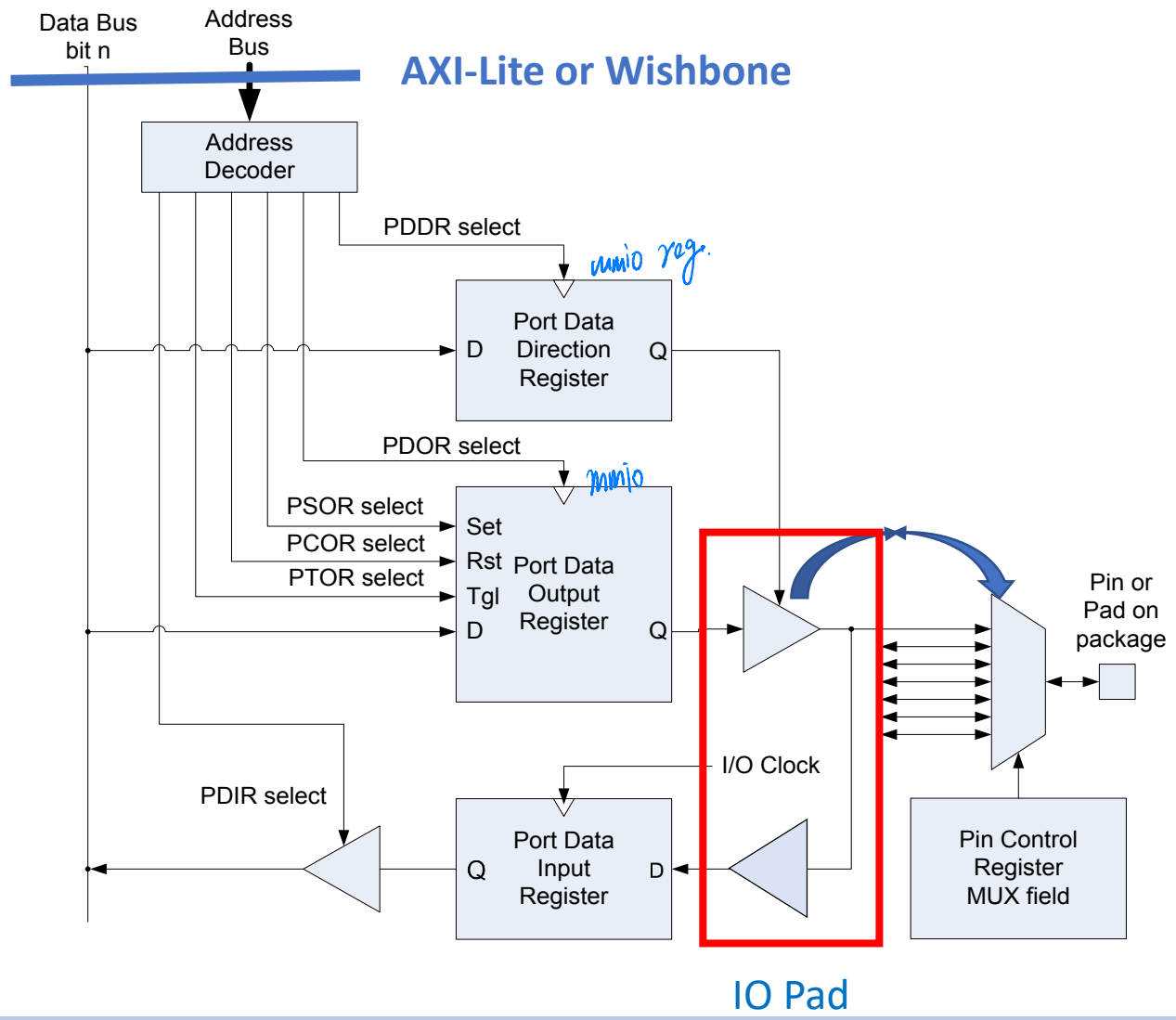
- External signals are asynchronous to internal clock – flip-flop can enter metastability
- Mechanical Switch generates bouncing signal
- It needs input synchronization and debouncing circuit

① IO pin Asynch.
② IO pin bouncing





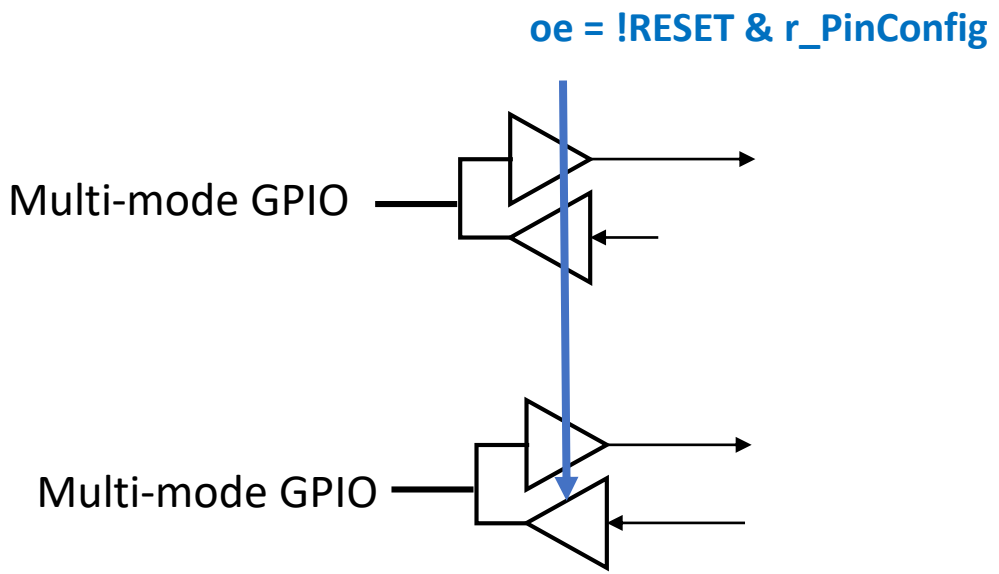
CPU Using Memory-IO to Access GPIO

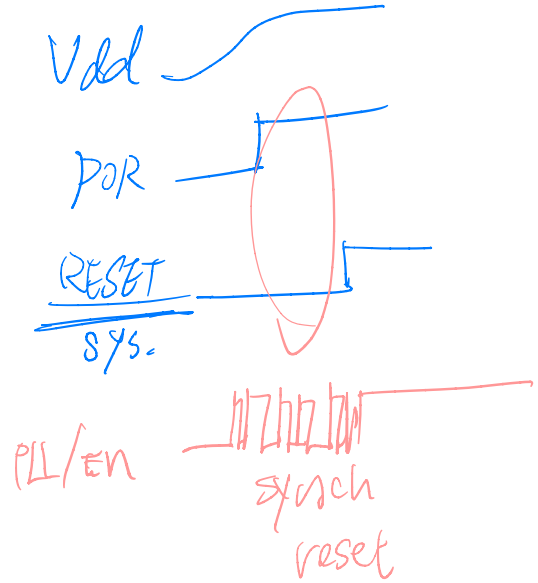


GPIO during RESET

GPIO pin state during RESET

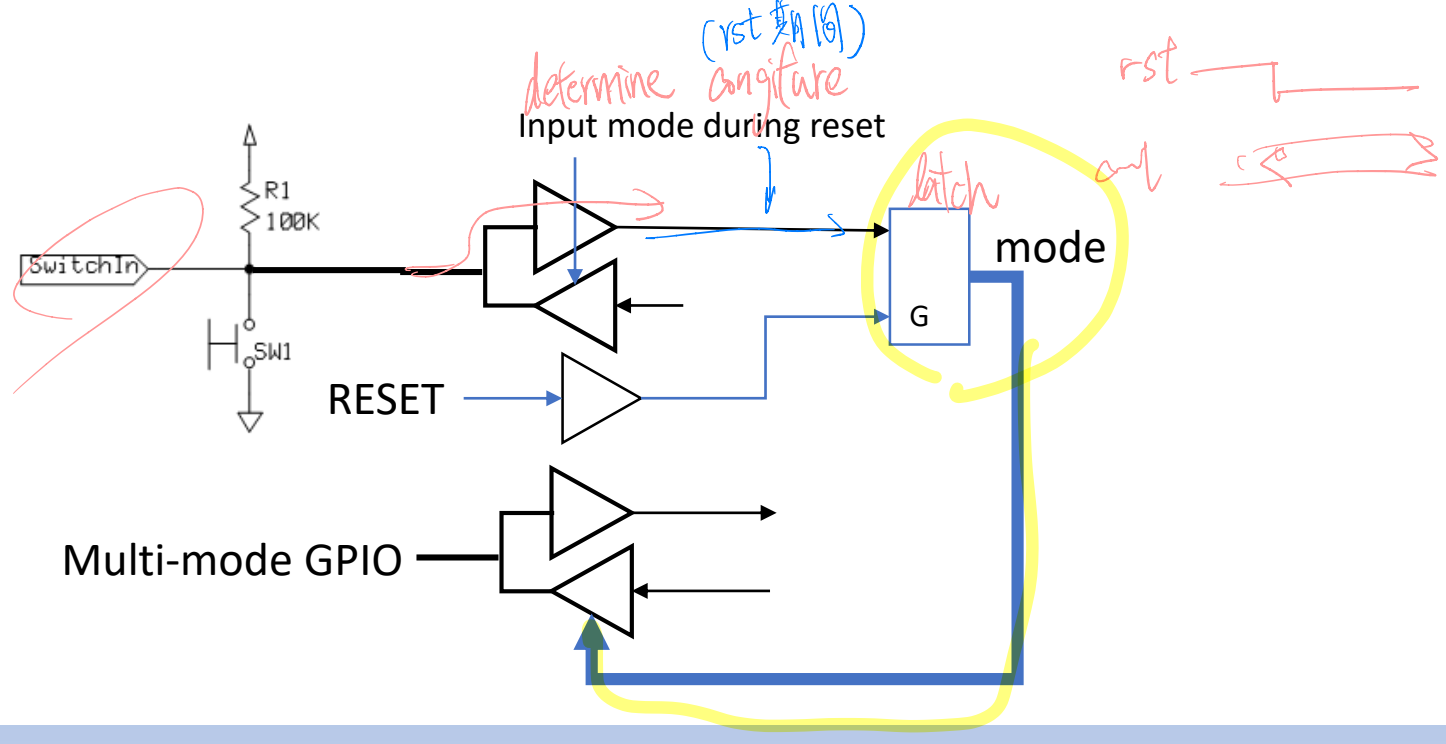
- If a GPIO pin can be used for input and output depends on operation mode. Reset to input mode (avoid bus conflict), until programmed to output





GPIO pin state during RESET

- If a GPIO pin can be used for input and output depends on operation mode. Reset to input mode (avoid bus conflict), until programmed to output
- The (dip)switch pin is used a mode configuration during reset, and as a GPIO during system operation.
- The dipswitch pad has weak pullup or pulldown as a default value, and can be over-ridden by a resistors. The resistor value does not affect the normal operation.
- If the input/output mode has to be determined before system starts, use a dip-switch latched by reset

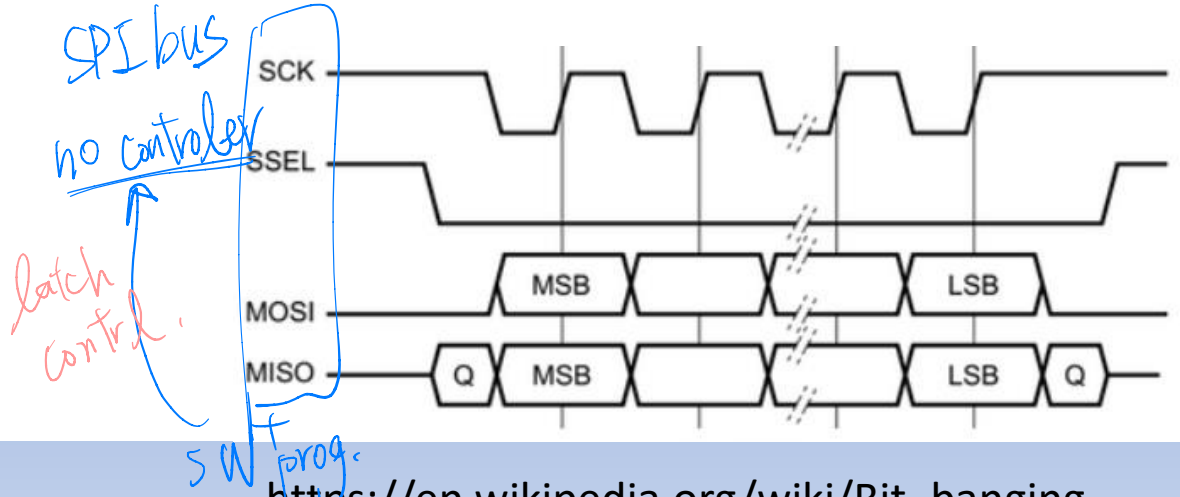


Software Emulation – Bit banging

Bit Banging

Software programming

- Software generates transmitted signals or process received signals (substitute for dedicated hardware)
- Directly set and samples the state of GPIOs
- Data buffers provided to relax software timing requirements
- Use for simple protocol (UART, SPI, I2C)



https://en.wikipedia.org/wiki/Bit_banging

Transmits a byte of data on an SPI BUS.

```
// transmit byte serially, MSB first
void send_8bit_serial_data(unsigned char data)
{
  int i;

  // select device (active low)
  output_low(SD_CS);

  // send bits 7..0
  for (i = 0; i < 8; i++)
  {
     // consider leftmost bit
     // set line high if bit is 1, low if bit is 0
     if (data & 0x80)
        output_high(SD_DI);
     else
        output_low(SD_DI);

     // pulse the clock state to indicate that bit value should be read
     output_low(SD_CLK);
     delay();
     output_high(SD_CLK);

     // shift byte left so next bit will be leftmost
     data <<= 1;
  }

  // deselect device
  output_high(SD_CS);
}
```

Supplement

User Project GPIO (MPRJ)

GPIO memory address map:

C header name	*address*	*description*
reg_gpio_data	**0x21000000**	GPIO input/output (low bit) GPIO output readback (16th bit)
reg_gpio_ena	**0x21000004**	GPIO output enable (0 = output, 1 = input)
reg_gpio_pu	**0x21000008**	GPIO pullup enable (1 = pullup, 0 = none)
reg_gpio_pd	**0x2100000c**	GPIO pulldown enable (1 = pulldown, 0 = none)
reg_pll_out_dest	**0x2f000000**	PLL clock output destination (low bit)
reg_trap_out_dest	**0x2f000004**	Trap output destination (low bit)
reg_irq7_source	**0x2f000008**	IRQ 7 input source (low bit)

Table 1 **reg_gpio_data**

0x21000003	0x21000002	0x21000001	0x21000000	address
GPIO output readback		GPIO input/output		value
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		bit

Table 2 **reg_gpio_ena**

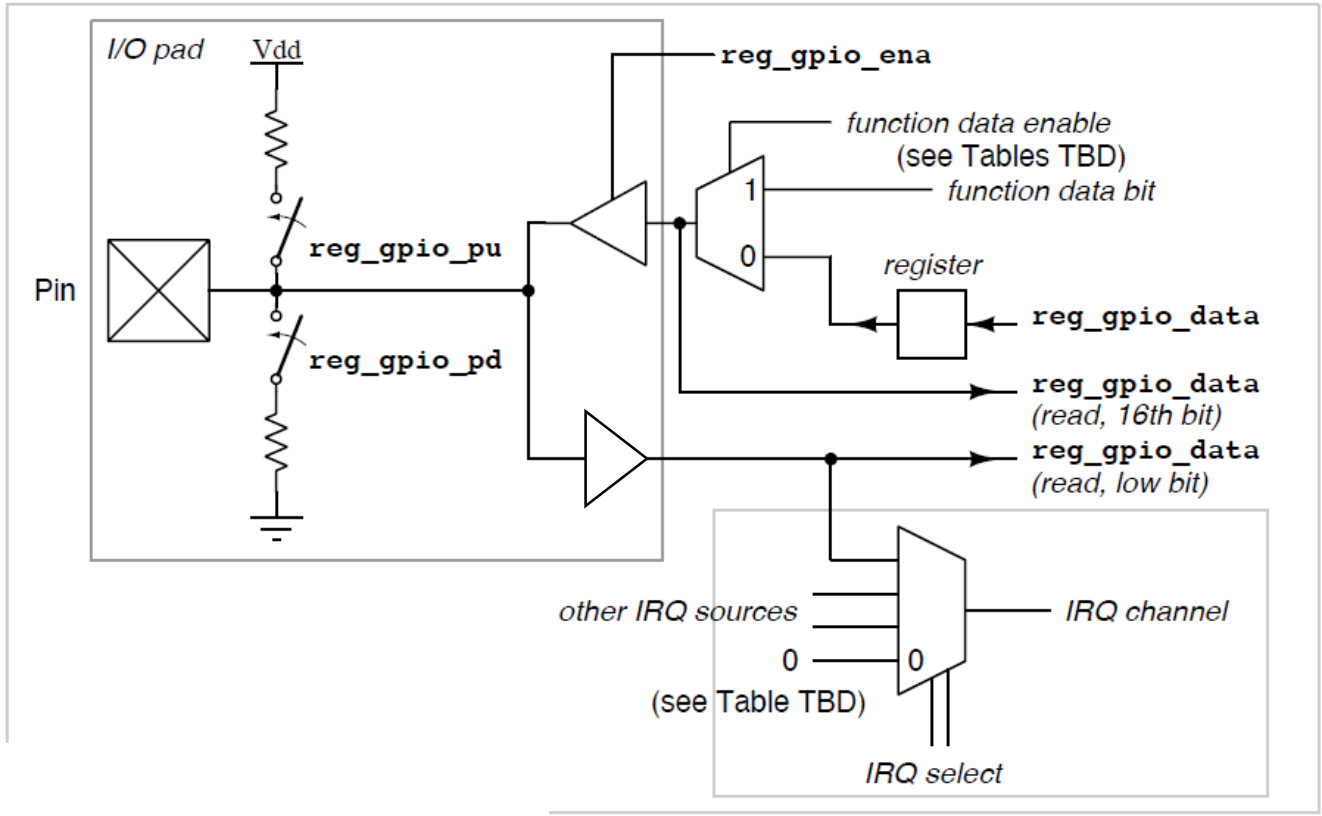
0x21000007	0x21000006	0x21000005	0x21000004	address
(undefined, reads zero)		GPIO output enable		value
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		bit

Table 3 **reg_gpio_pu**

0x2100000b	0x2100000a	0x21000009	0x21000008	address
(undefined, reads zero)		GPIO pin pull-up		value
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		bit

Table 4 **reg_gpio_pd**

0x2100000f	0x2100000e	0x2100000d	0x2100000c	address
(undefined, reads zero)		GPIO pin pull-down (inverted)		value
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		bit



https://github.com/efabless/caravel/blob/main/docs/pdf/gpio_function.pdf