CS4101 Introduction to Embedded Systems

# Lab 6: Low-Power Optimization

Prof. Chung-Ta King

Department of Computer Science

National Tsing Hua University, Taiwan

# Introduction

- In this lab, we will learn power optimization of MSP430 LanuchPad
  - Basic concepts of power optimization
  - Configuring the low-power modes of MSP430

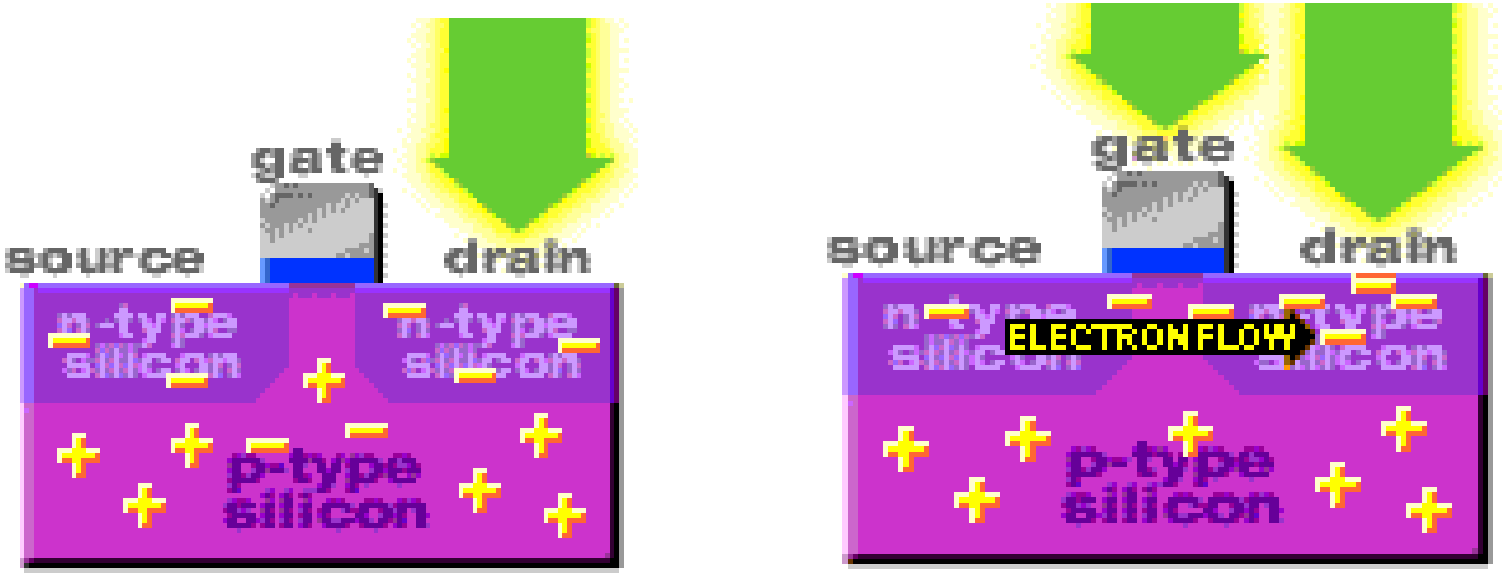
# Introduction

- Why low power?
  - Portable and mobile devices are getting popular, which have limited power sources, e.g., battery
  - Energy conservation for our planet
  - Power generates heat → low carbon
- Power optimization becomes a new dimension in system design, besides performance and cost
- MSP430 provides many features for low-power operations, which will be discussed next

# Energy and Power

- Energy: ability to do work
  - Most important in battery-powered systems
- Power: energy per unit time
  - Important even in wall-plug systems---power becomes heat
- Power draw increases with…
  - Vcc
  - Clock speed
  - Temperature

# Power Consumption: Transistor Level




- Switching consumes power → dynamic power
  - Switching slower, consume less power
  - Smaller sizes reduce power to operate
- Leakage → static power

# General Strategies for Low Power

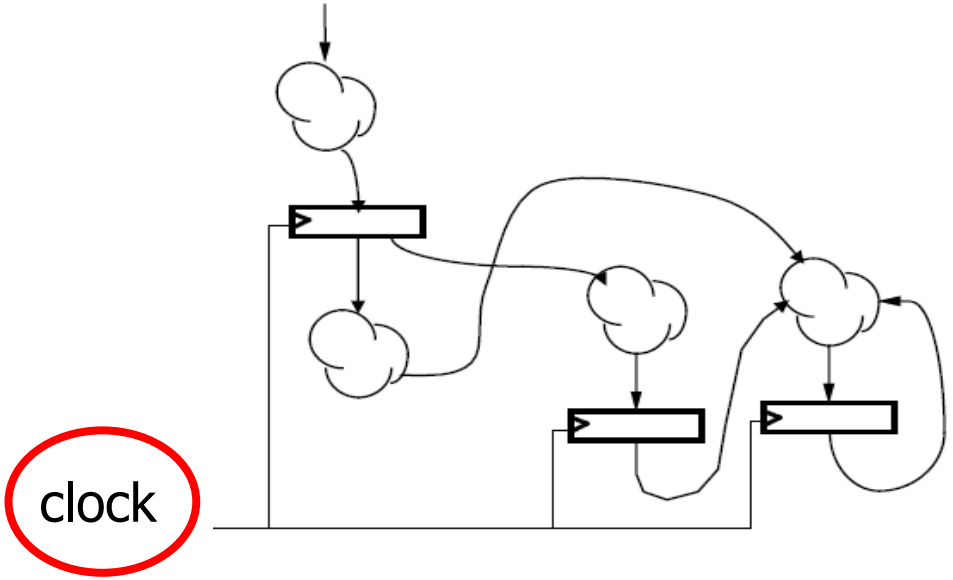
- Put the system in low-power modes and/or use low-power modules as much as possible
- How?
  - Provide clocks of diff. frequencies → frequency scaling
  - Lower supplied voltage → voltage scaling
  - Turn off clocks when no work to do → clock gating
  - Use interrupts to wake up the CPU, return to sleep when done (another reason to use interrupts)
  - Switched on peripherals only when needed
  - Use low-power integrated peripheral modules in place of software, e.g., low-power DSP

# Clock Gating

- Clock gating for synchronous sequential logic:
  - Disable the clock so that flip-flops will hold their states forever and the whole circuit will not switch

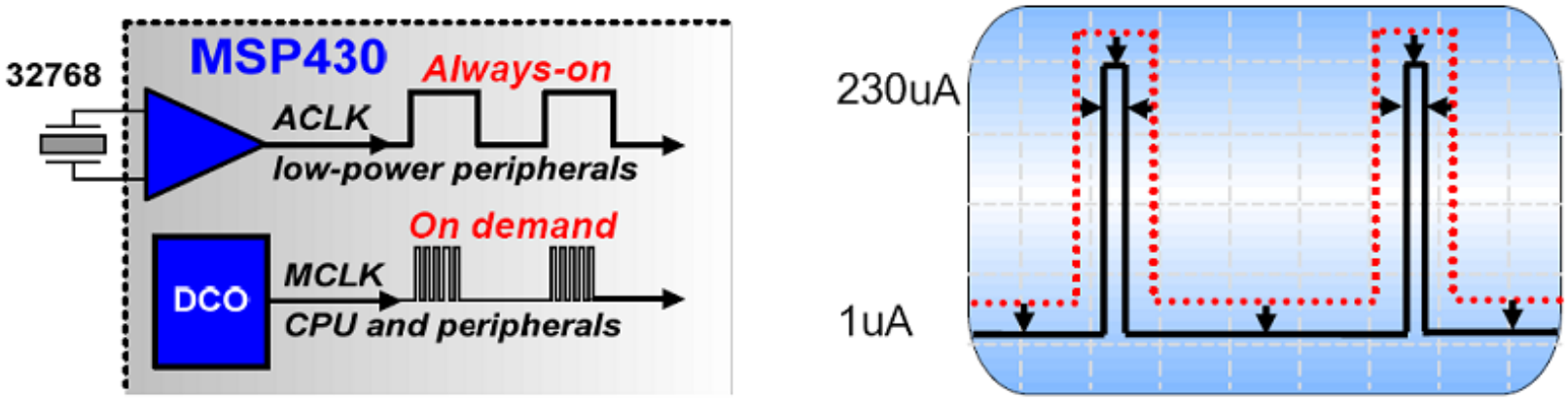
    → no dynamic power consumed

Still consume static power to hold the states



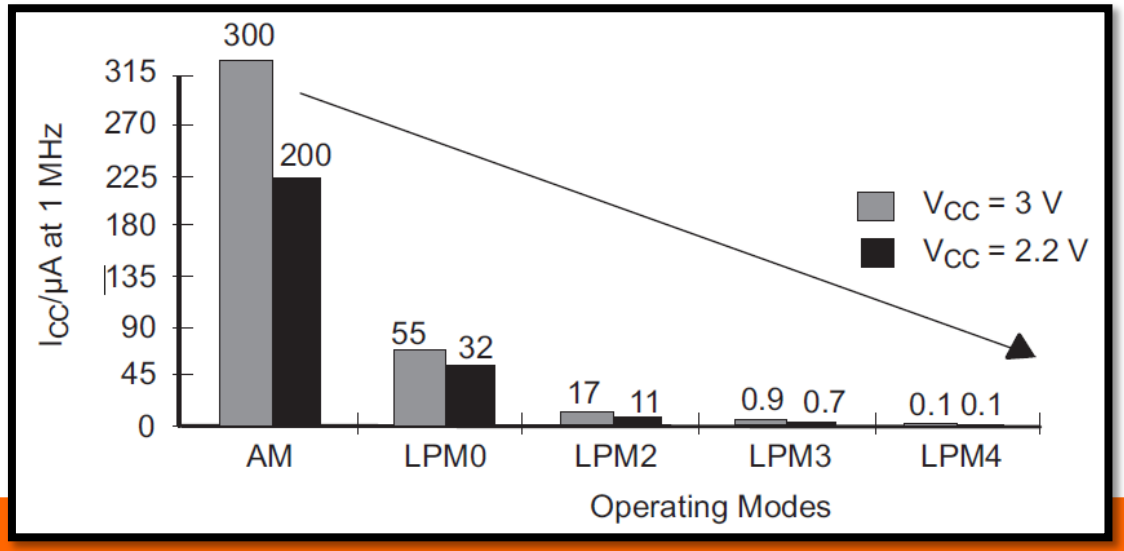
# Power Saving in MSP430

- The most important factor for reducing power consumption is using the MSP430 clock system to maximize the time in LPM3




"Instant on" clock

# MSP430 Low-Power Modes




| Mode | CPU and Clocks |
|---|---|
| Active | CPU active; all enabled clocks active |
| LPM0 | CPU, MCLK disabled; SMCLK, ACLK active |
| LPM1 | CPU, MCLK disabled; DCO disabled if not for SMCLK; ACLK active |
| LPM2 | CPU, MCLK, SMCLK, DCO disabled; ACLK active |
| LPM3 | CPU, MCLK, SMCLK, DCO disabled; ACLK active |
| LPM4 | CPU and all clocks disabled |

# MSP430 Low Power Modes

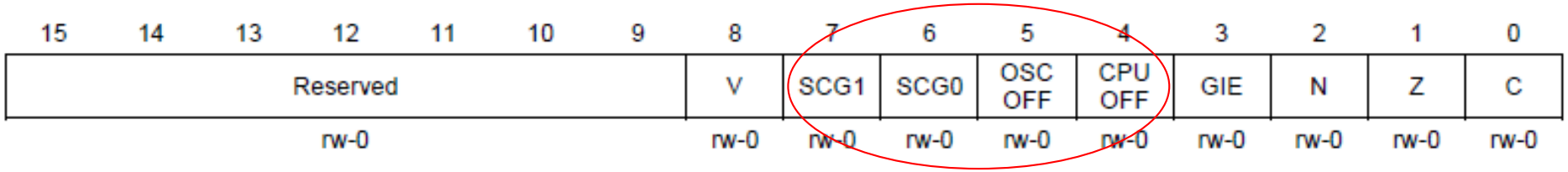
- Active mode:
  - MSP430 starts up in this mode, which must be used when the CPU is required, i.e., to run code
  - An interrupt automatically switches MSP430 to active
  - Current can be reduced by running at lowest supply voltage consistent with the frequency of MCLK, e.g. $V_{CC}$ to 1.8V for $f_{DCO}$ = 1MHz
- LPM0:
  - CPU and MCLK are disabled
  - Used when CPU is not required but some modules require a fast clock from SMCLK and DCO

# MSP430 Low Power Modes

- LPM3:
  - Only ACLK remains active
  - Standard low-power mode when MSP430 must wake itself at regular intervals and needs a (slow) clock
  - Also required if MSP430 must maintain a real-time clock
- LPM4:
  - CPU and all clocks are disabled
  - MSP430 can be wakened only by an external signal, e.g., RST/NMI, also called *RAM retention mode*

# Controlling Low Power Modes

- Through four bits in *Status Register* (SR) in CPU
  - SCG0 (System clock generator 0): when set, turns off DCO, if DCOCLK is not used for MCLK or SMCLK
  - SCG1 (System clock generator 1): when set, turns off the SMCLK
  - OSCOFF (Oscillator off): when set, turns off LFXT1 crystal oscillator, when LFXT1CLK is not use for MCLK or SMCLK
  - CPUOFF (CPU off): when set, turns off the CPU
  - All are clear in active mode



# Controlling Low Power Modes

- Status bits and low-power modes

**Table 2-2. Operating Modes For Basic Clock System**

| SCG1 | SCG0 | OSCOFF | CPUOFF | Mode | CPU and Clocks Status |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Active | CPU is active, all enabled clocks are active |
| 0 | 0 | 0 | 1 | LPM0 | CPU, MCLK are disabled, SMCLK, ACLK are active |
| 0 | 1 | 0 | 1 | LPM1 | CPU, MCLK are disabled. DCO and DC generator are disabled if the DCO is not used for SMCLK. ACLK is active. |
| 1 | 0 | 0 | 1 | LPM2 | CPU, MCLK, SMCLK, DCO are disabled. DC generator remains enabled. ACLK is active. |
| 1 | 1 | 0 | 1 | LPM3 | CPU, MCLK, SMCLK, DCO are disabled. DC generator disabled. ACLK is active. |
| 1 | 1 | 1 | 1 | LPM4 | CPU and all clocks disabled |

# Entering/Exiting Low-Power Modes

Interrupt wakes MSP430 from low-power modes:

- Enter ISR:
  - PC and SR are stored on the stack
  - CPUOFF, SCG1, OSCOFF bits are automatically reset → entering active mode
  - MCLK must be started so CPU can handle interrupt
- Options for returning from ISR:
  - Original SR is popped from the stack, restoring the previous operating mode
  - SR bits stored on stack can be modified within ISR to return to a different mode when RETI is executed

All done in hardware

# Sample Code (MSP430G2xx1 _ta_01)

```
void main(void) { //Toggle P1.0 every 50000 cycles
  WDTCTL = WDTPW + WDTHOLD; // Stop WDT
  P1DIR |= 0x01;           // P1.0 output
  CCTL0 = CCIE;            // CCR0 interrupt enabled
  CCR0 = 50000;
  TACTL = TASSEL_2 + MC_2;  // SMCLK, contmode
  _BIS_SR(LPM0_bits + GIE); // LPM0 w/ interrupt
}
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void) {
  P1OUT ^= 0x01;           // Toggle P1.0
  CCR0 += 50000;           // Add Offset to CCR0
}
```

Use _BIC_SR_IRQ(LPM0_bits) to exit LPM0

# Lab 6

- Flash green LED at 0.5 Hz using interrupt from Timer_A, driven by ACLK sourced by VLO in LPM3.
- While pushing the button:
  - Change from LPM3 to LPM0
  - Wake up every 1 sec using interrupt from Timer_A driven by SMCLK sourced by VLO.
  - On wake up, measure the temperature. If the temperature is higher than 737, flash the red LED at 5 Hz; otherwise flash the green LED at 1 Hz.
- When the button is released, returns the system to LPM3 and flash green LED at 0.5 Hz again.

# P1IES

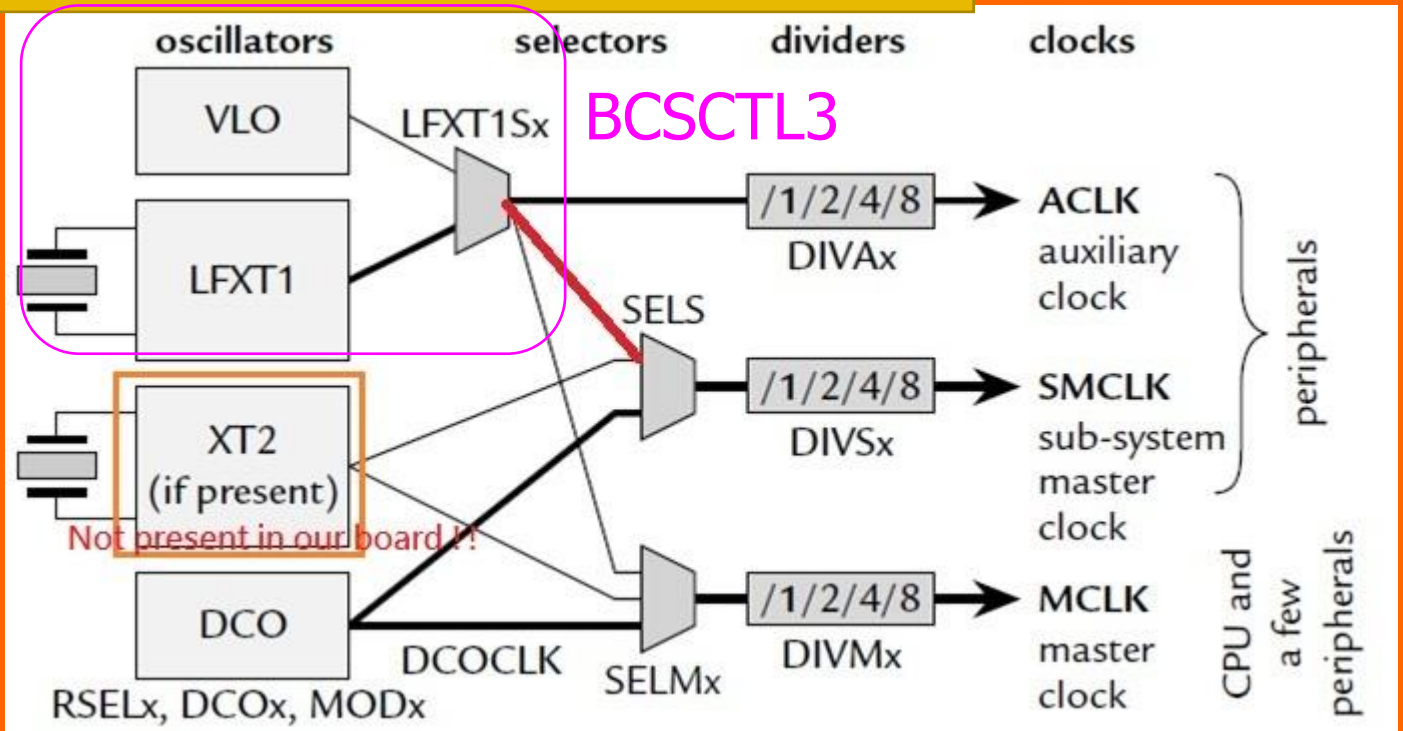
**Interrupt Edge Select Registers P1IES, P2IES**

Each PxIES bit selects the interrupt edge for the corresponding I/O pin.

Bit = 0: The PxIFGx flag is set with a low-to-high transition

Bit = 1: The PxIFGx flag is set with a high-to-low transition

## Timer_A, driven by SMCLK sourced by VLO



# Bonus

- Put launchPad to LPM4 and flash the red LED at 2Hz using interrupt from Timer_A, driven by ACLK sourced by VLO. Observe whether the LED is flashing or not.
- While pushing the button:
  - Change from LPM4 to LPM3
  - Observe whether the light is flashing or not
- When the button is released, returns the system to LPM4