INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Université de Toulouse

5th year ESPE

Low Power Software for IoT

Alexandre Boyer

alexandre.boyer@insa-toulouse.fr

www.alexandre-boyer.fr

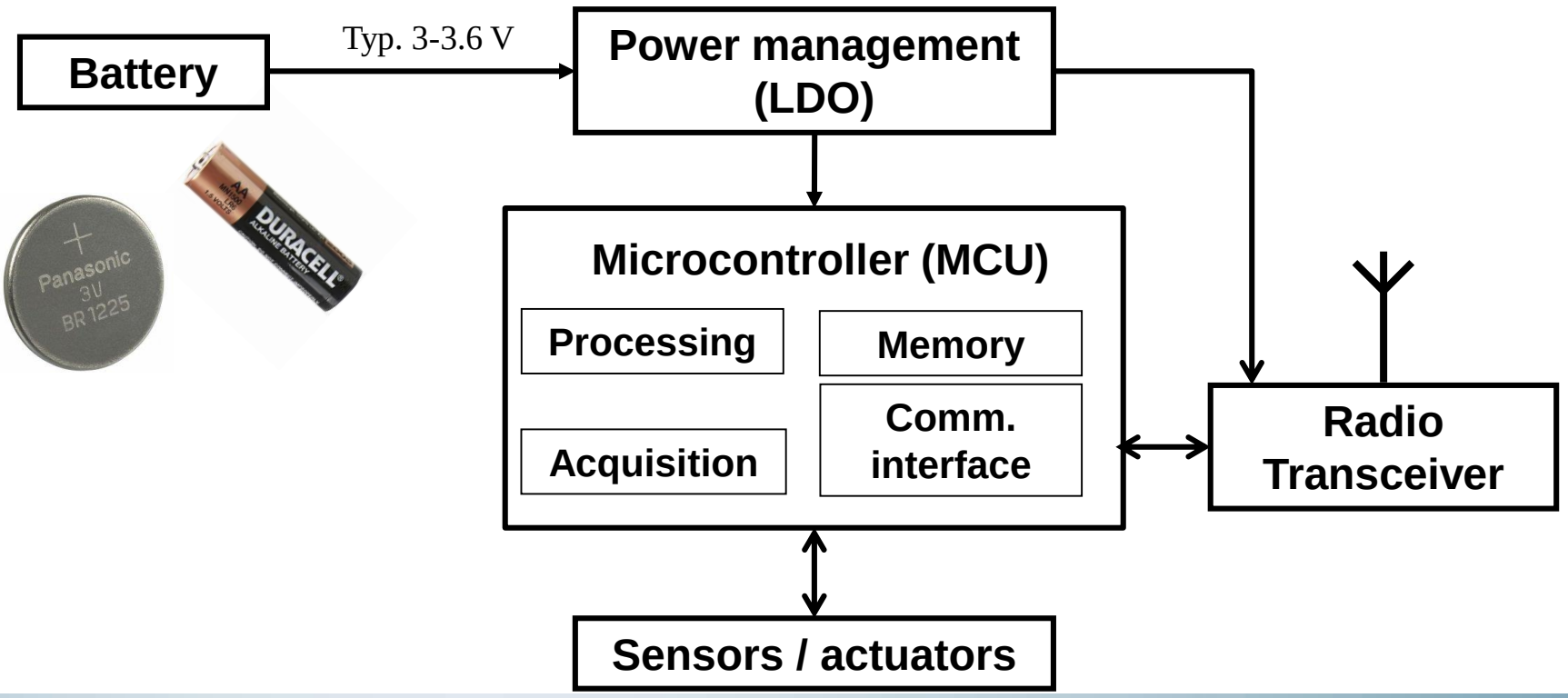
Contents

1. Context: Low energy issues for IoT node
2. Origin of energy consumption of MCU
3. Hardware solutions to reduce energy consumption of MCU
4. Software for low-power MCU

Low energy issues for IoT node

Typical IoT node

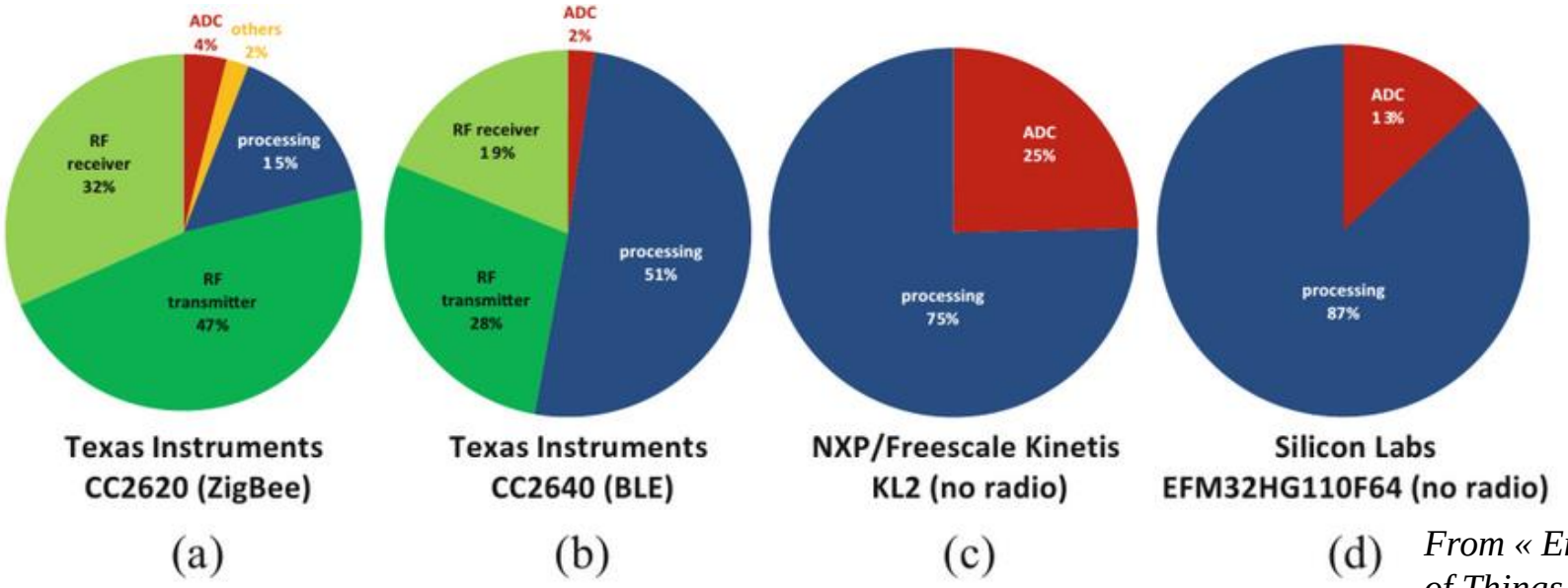
- Wireless autonomous sensing and processing node, supplied by a battery
- Typical applications: wireless sensors, wearable, water/gas flow meter, medical implants, active RFID....



Low energy issues for IoT node

IoT node – Typical energy consumption

- Dependent on radio and application constraints.
- Main contributors to energy consumption:
  - Wireless communication devices (20 – 400 mW)
  - Processing operation (MCU) (1-10 mW) **This course**
  - Analog-to-digital conversion (embedded in MCU or in sensor) (1-2 mW
  - Sensors (< 0.1 - 1 mW)

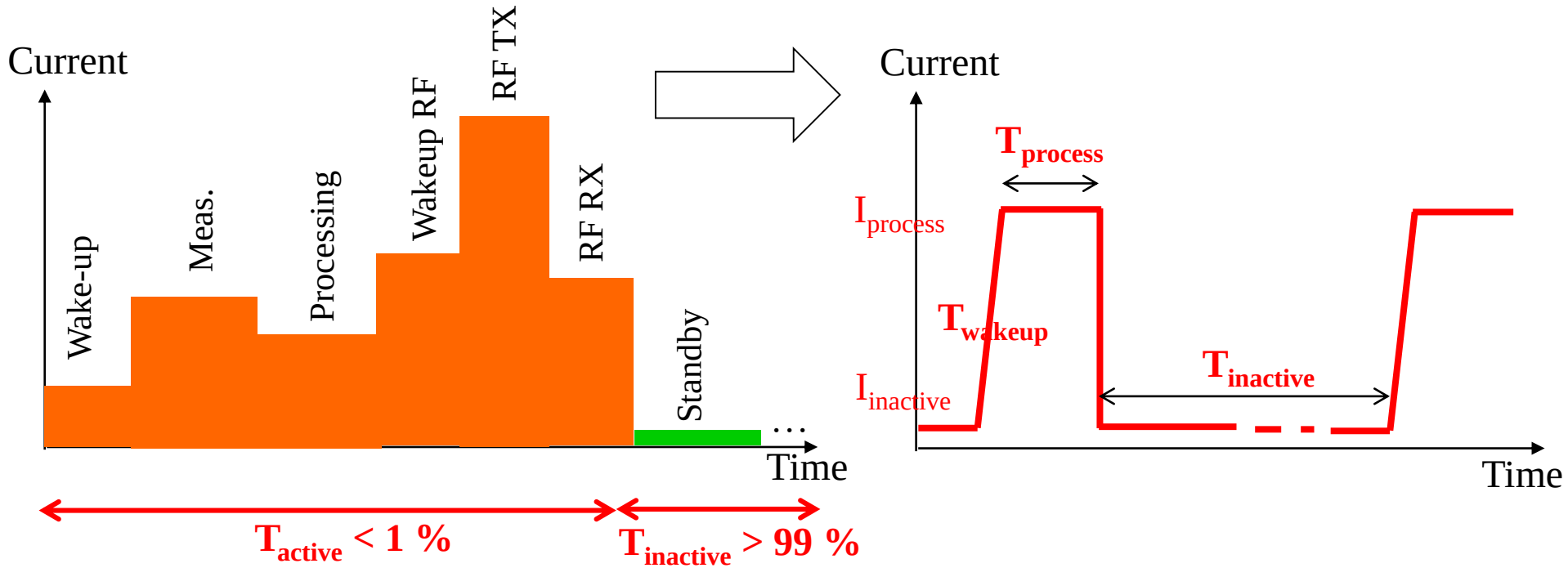


From « Enabling the Internet of Things », Springer

Low energy issues for IoT node

IoT node – Typical current consumption profile

- Discontinuous consumption profile, with low duty cycle
- Require MCU with run/standby modes and wakeup mechanisms (periodic or on external events)



Low energy issues for IoT node

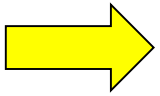
Ultra Low Power MCU …

- For MCU dedicated for IoT node.
- More a commercial than technical concepts: No official definition !
- Various performance/security requirements but high integration and low power constraints !
- The right MCU selection is a complex process (performance vs. Consumption compromise), which can be only done once all the application constraints are known.
- Forget the most ULP MCU. A large choice of MCU is necessary to be sure to find the best MCU for a given application.

Origin of energy consumption of MCU

Which measures ?

- Energy: $E\ (J) = \int V_{dd} \times i(t)\, dt = V_{dd} \times I_{avg} \times \Delta t$
- Instantaneous power : $P_{inst}(W) = V_{dd} \times i(t)$
- Average power: $P_{avg}(W) = \frac{1}{\Delta t}\int_0^{\Delta t} V_{dd} \times i(t)dt = V_{dd} \times I_{avg}$
- Average current: $I_{avg} = \frac{1}{\Delta t}\int_0^{\Delta t} i(t)dt$
- Transfered charge: $Q_{tr}(C) = \int_0^{\Delta t} i(t)dt = I_{avg} \times \Delta t$
- Transferred charge (synchronous circuit): $Q_{tr}(C) = C_L \times V_{dd} \times NOC = \int_0^{\Delta t} i(t)dt$, NOC is the number of clock cycles, C_L is the equivalent capacitive load, $\Delta t = NOC \times T_{clk}$



- As power supply is constant, all these measures depend on current.
- The duration of battery depends on consumed energy → average current
- Battery must be dimensioned also to withstand peak current
- Power is a major requirements for thermal management.

Origin of energy consumption of MCU

Dynamic vs. Static current consumption

- For CMOS digital circuits:

$$I_{tot}(A) = I_{dyn} + I_{stat}$$

Link to switching activity (expressed in A or A/Hz)

Appears when the circuit is biased, independent of switching activity (A)

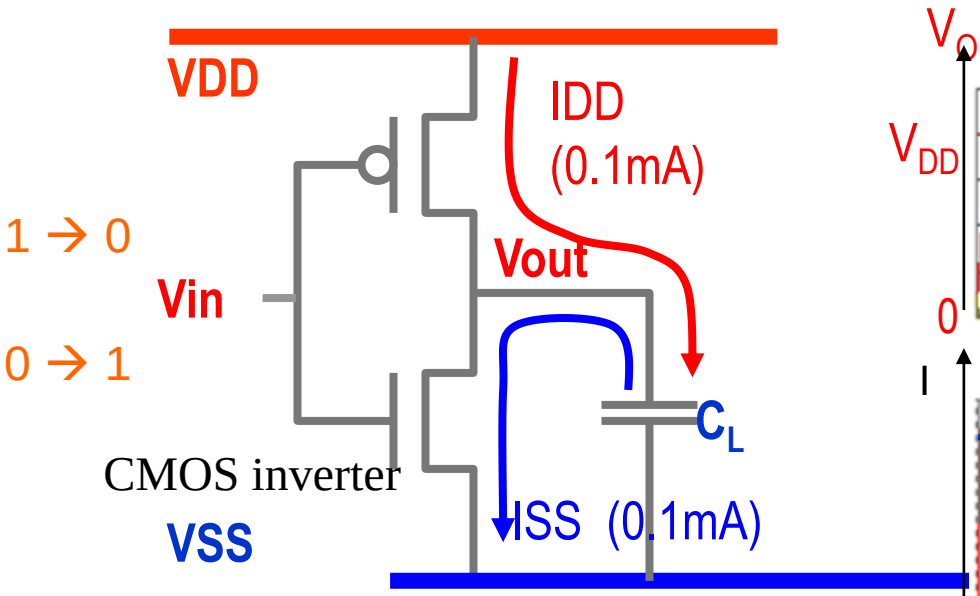
⇩

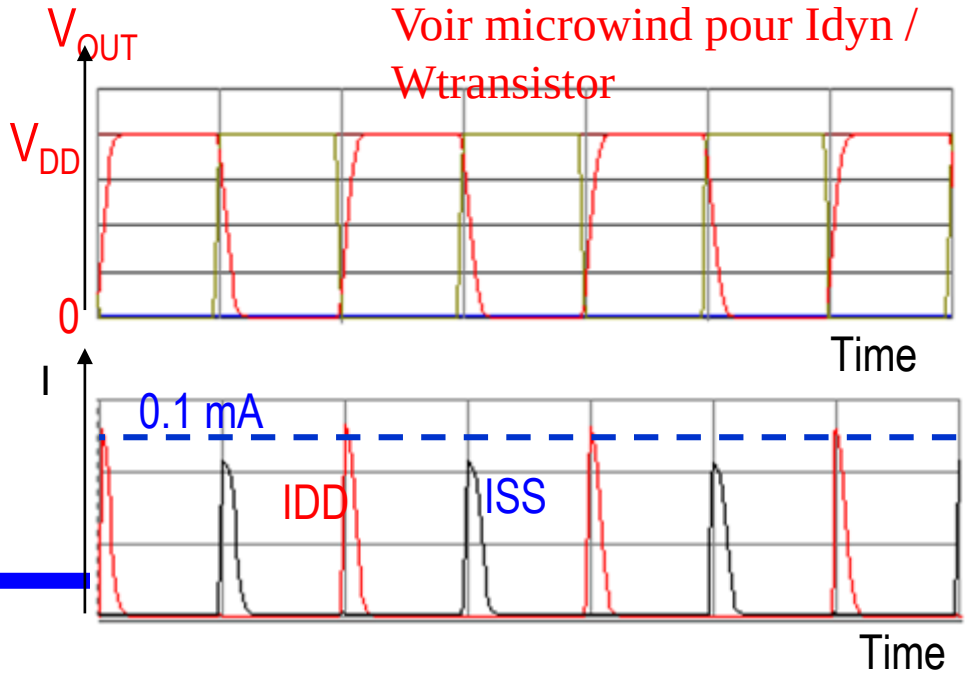
Leakage current

- Ideally, leakage should not exist. Due to the imperfection of CMOS transistors and the presence of analog functions in MCU, static current exists.
- In terms of peak amplitude, dynamic current dominates. However, even if clocks are slow down or stopped, static current is still here → must not be neglected !

Origin of energy consumption of MCU

Dynamic current in CMOS digital circuit





- For a CMOS circuit with continuous switching activity (frequency F_{clk}):

$$I_{dyn} = \frac{1}{T_{clk}} \int_0^{T_{clk}} N_{sw} C_L V_{dd} dt = N_{sw} C_L V_{dd} F_{clk} \quad \Rightarrow \quad P_{dyn} = N_{sw} C_L V_{dd}^2 F_{clk}$$

« Power efficiency » (A/Hz)

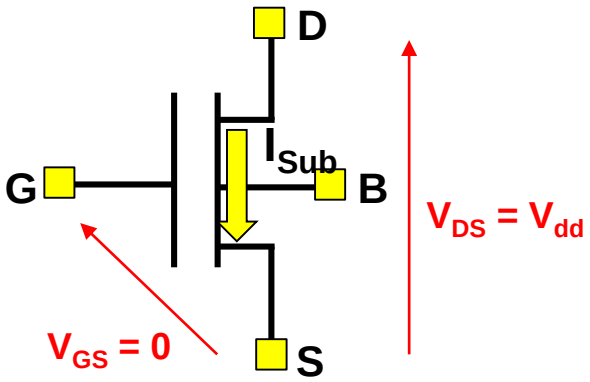
- For CMOS output buffer with periodic switching (α is the switching factor):

$$I_{dyn} = \alpha C_{Load} V_{dd} F_{SW}$$

Origin of energy consumption of MCU

Leakage current in MOS transistor

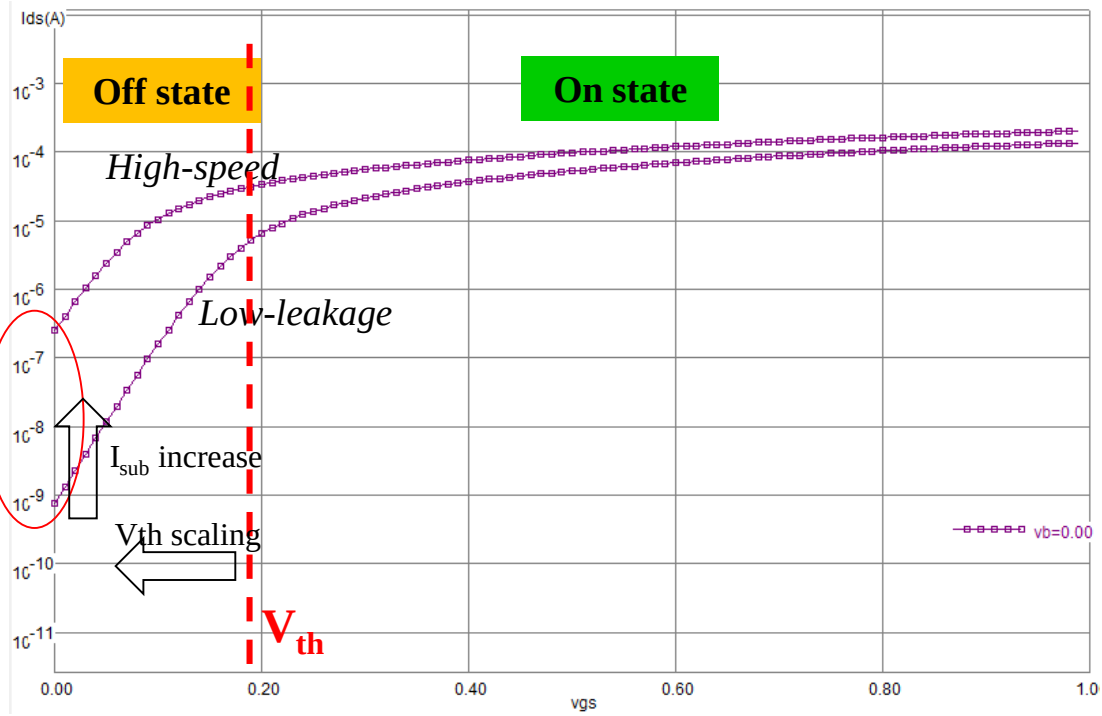
- Since the beginning of 2000's (90 nm node), one of the main issues for CMOS digital IC design is the control of leakage current while maintaining the transistor size shrinking (Moore law).
- The main contributor : Subthreshold current (about 1 to 100 nA/µm)



$$I_{Sub} = I_S \frac{W}{L} e^{-\frac{V_{GS}-V_{th}}{nV_T}} \left(1 - e^{-\frac{V_{DS}}{V_T}}\right)$$

$$\approx 10^{-7} \frac{W}{L} e^{-\frac{V_{GS}-V_{th}}{nV_T}}$$

- Is and n: experimental parameters
- Vth: threshold voltage
- $V_T = \frac{kT}{q}$

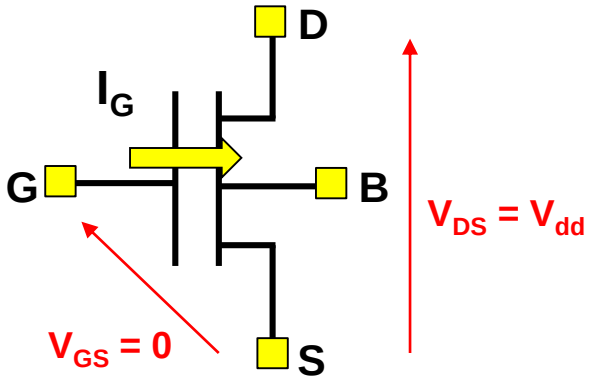


Microwind (32 nm process, NMOS, W=0.18µm, L = 0.036 µm, Vds = 1.0 V, 27° c)

Origin of energy consumption of MCU

Leakage current in MOS transistor

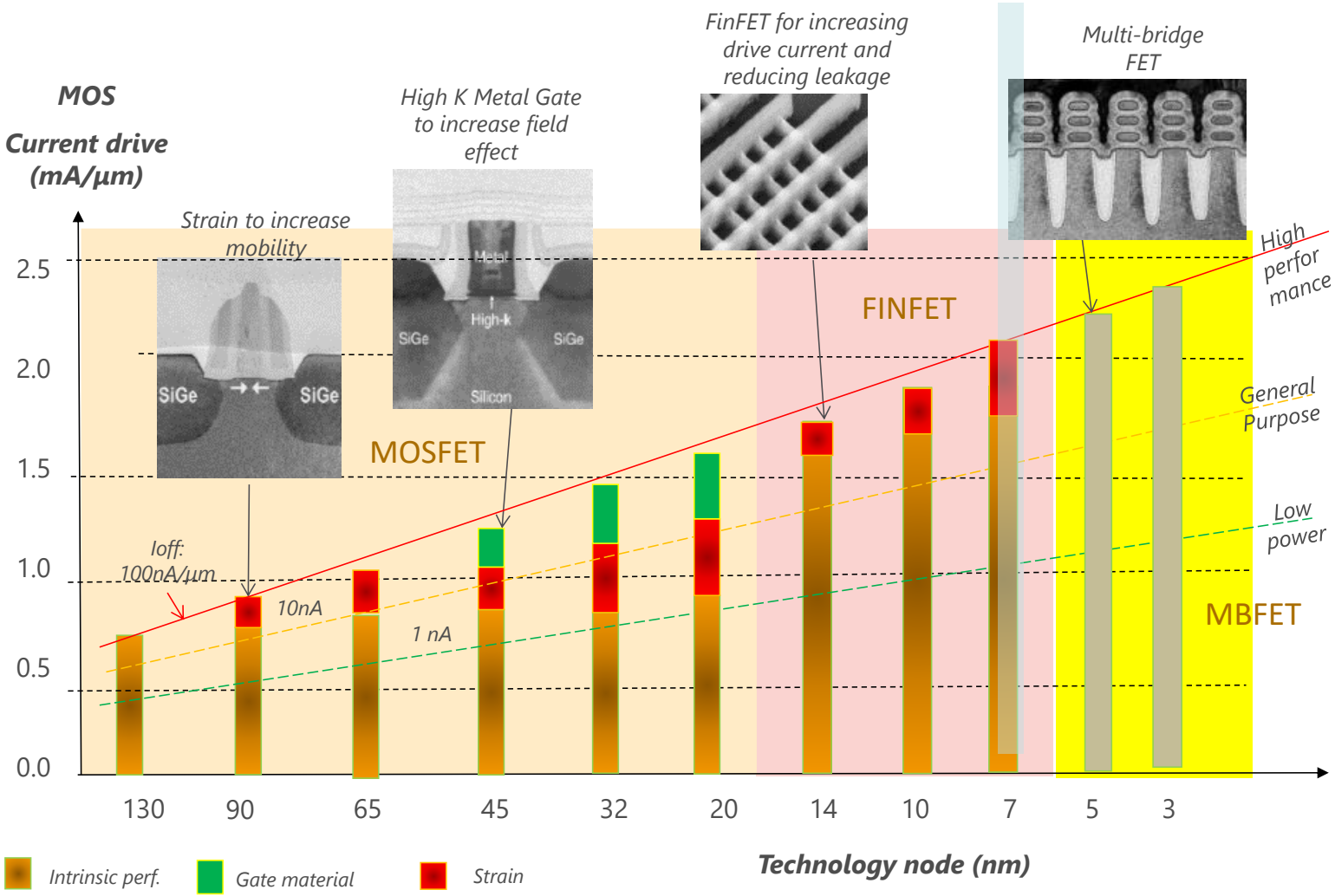
- Gate direct-tunneling leakage : get worse wth gate oxide thinning.



$$I_G = K_2 W \left(\frac{V_{dd}}{T_{ox}}\right)^2 e^{-\frac{\alpha T_{ox}}{V_{dd}}}$$

Origin of energy consumption of MCU

Evolution of Ion/Ioff of MOS transistors

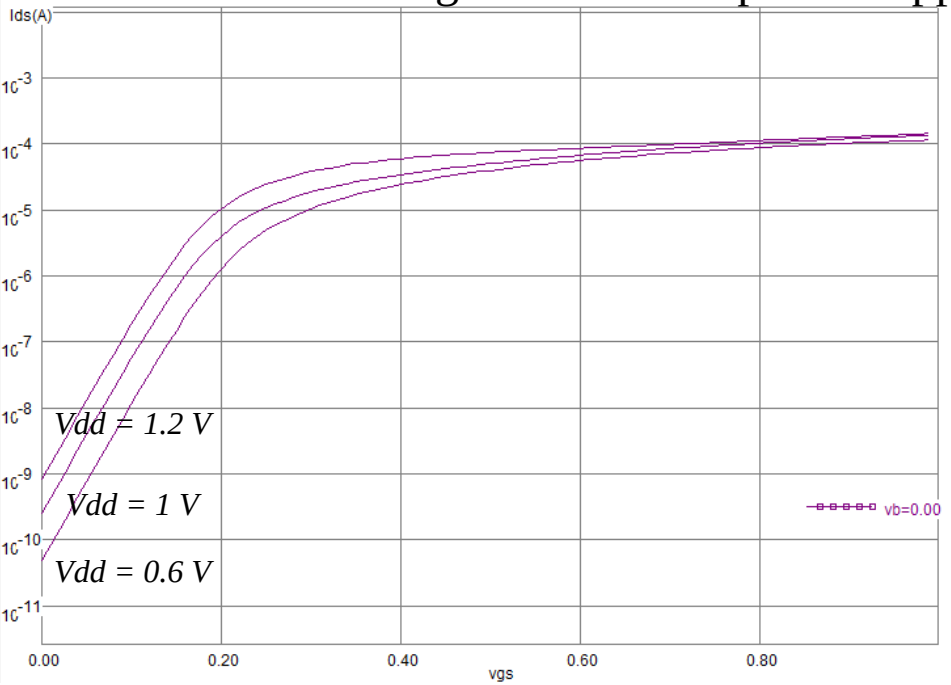


E. Sicard, Introducing 7-nm FinFET technology in Microwind

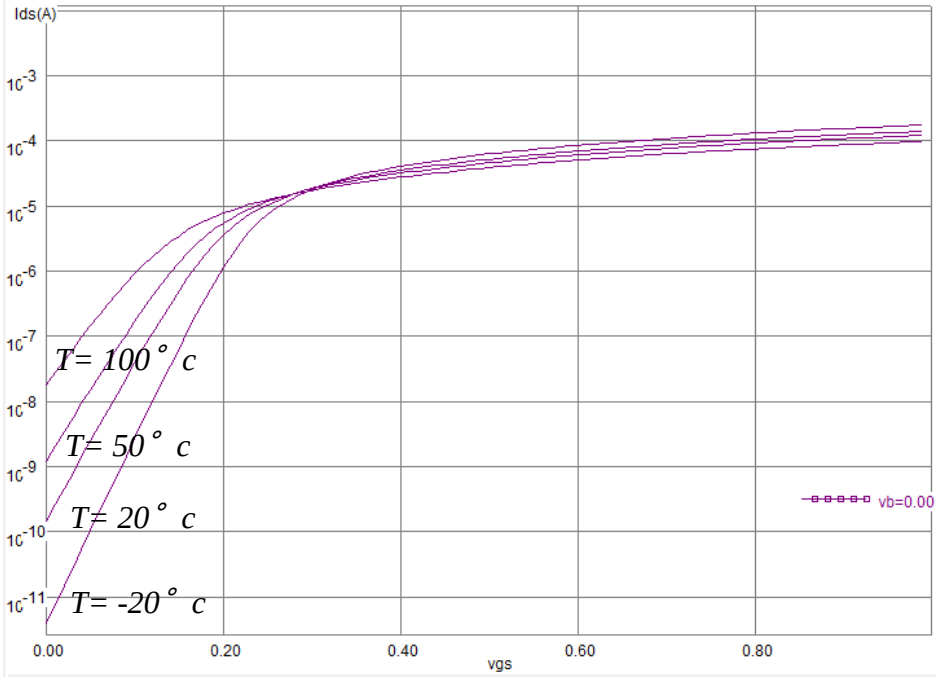
Origin of energy consumption of MCU

Influence of temperature and voltage on leakage current

- Increase of leakage current with power supply voltage and temperature



Effect of power supply voltage (T = 27° c)



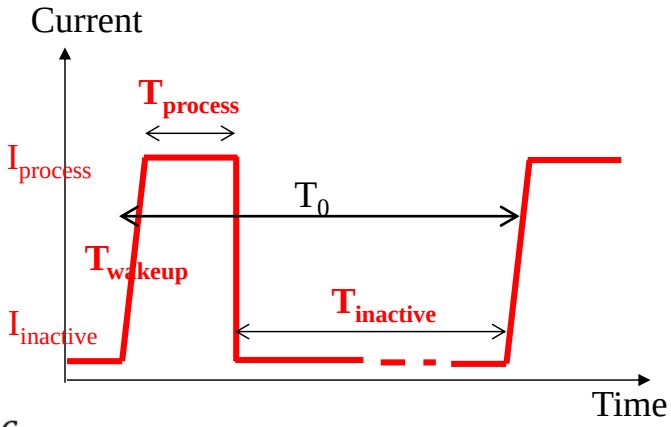
Effect of temperature (Vdd = 1 V)

Microwind (32 nm process, NMOS, W=0.18µm, L = 0.036 µm)

Origin of energy consumption of MCU

Average current consumption of an IOT MCU

- Discontinuous activity: short activity durations with long inactive periods.
- Should we reduce the frequency to decrease the average current and thus the energy consumption?



$$T_{process} = NOC \times T_{clk}$$

$$I_{process} = \frac{1}{T_{process}} \sum_{k=1}^{NOC} C_L V_{dd} = \frac{NOC \times C_L V_{dd}}{T_{process}}$$

$$I_{avg} = I_{process} \frac{T_{process}}{T_0} + I_{inactive} \frac{T_{inactive}}{T_0}$$

$$= I_{inactive} + (I_{process} - I_{inactive}) \frac{NOC}{F_{clk} T_0}$$

Duty cycle

In practice, it is better to increase MCU clock frequency to increase the inactive (standby) duration (reduction of the duty cycle).

Origin of energy consumption of MCU

Consumption vs. MCU blocks

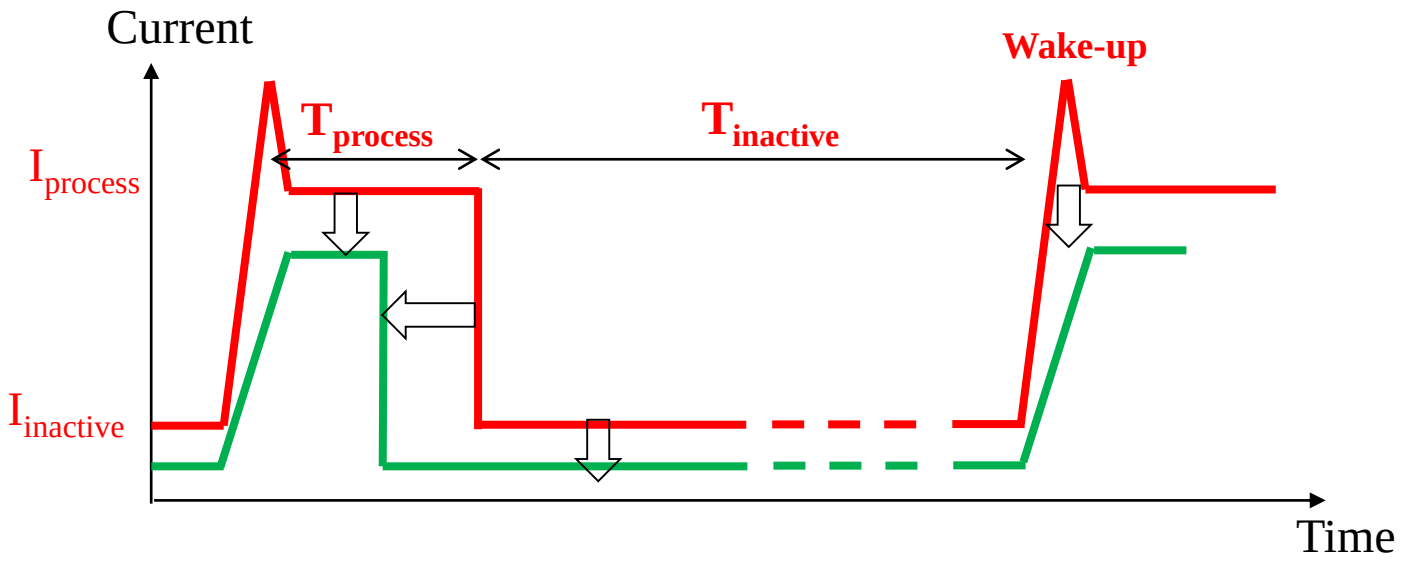
- Example: STML476RG (Vdd = 3 V (1.2 V internal) – 25° c)

Block	Typical current consumption
CPU (from Flash + HSE + 80 MHz PLL, range 1, run mode)	10 – 11 mA (124 – 137 µA/MHz)
CPU (from Flash + HSE + 26 MHz PLL, range 2, run mode)	2.8 – 3.1 mA (108 – 111 µA/MHz)
8 MHz quartz oscillator (HSE)	0.45 mA
PLL (VCO frequency = 64 – 344 MHz)	0.2 – 0.5 mA
32 kHz quartz oscillator, medium drive (LSE)	5.1 µA
Internal RC oscillator (100 kHz – 48 MHz) – PLL mode	0.6- 155 µA
Flash memory (write – erase mode)	3.4 mA
Flash	6.2 µA/MHz
SRAM1 and 2	0.9 – 1.6 µA/MHz
ADC (1 Msps, fclk = 80 MHz)	0.66 mA
I2C (I/O not included)	0.4 mA
SPI (I/O not included)	0.16 mA
GPIO (10 pF load 15 MHz)	0.45 mA
GPIO (50 pF load, 1 MHz)	0.15 mA
RTC (external 32 kHz quartz)	0.5 µA

Hardware solutions to reduce energy consumption of MCU

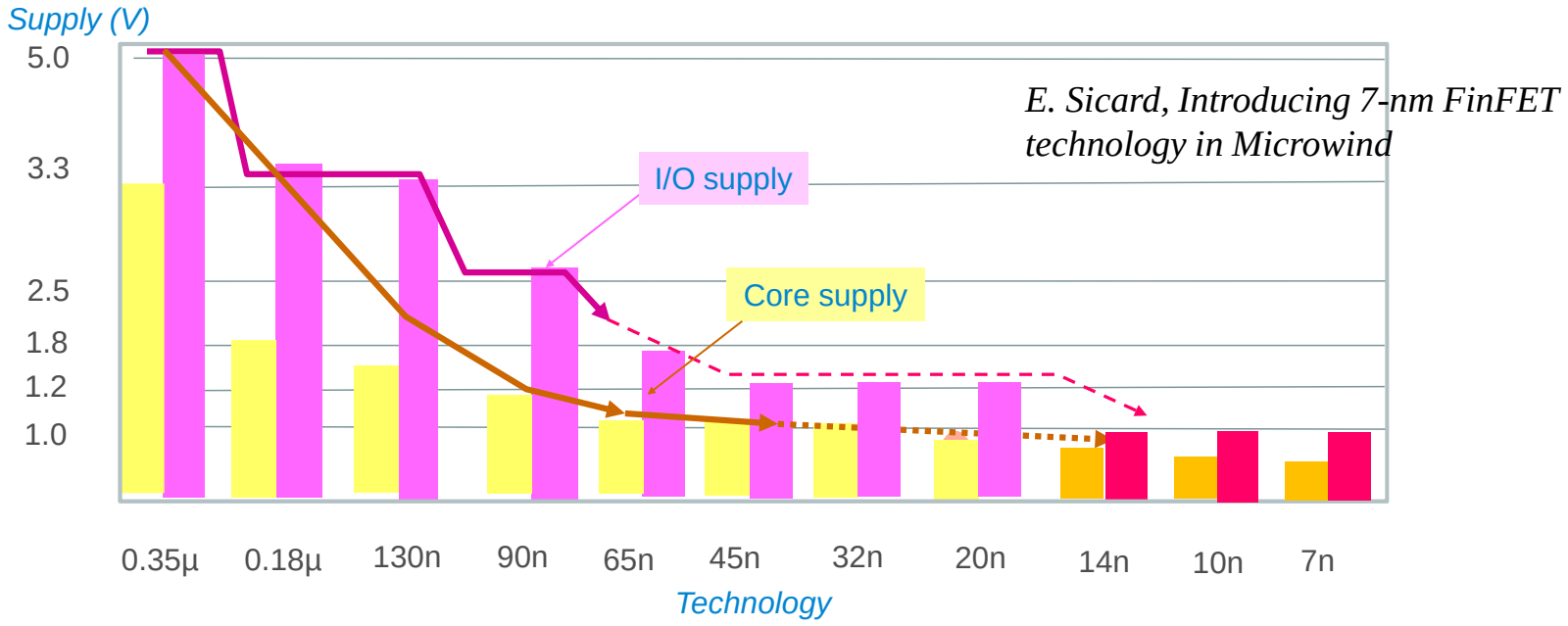
General strategies

- Reduce duty cycle (existence of low power modes and wake-up mechanisms)
- Reduce current in active mode
- Reduce current in standby mode
- Reduce peak current during wake-up
- Reduce current consumption of CPU, memories and peripherals



Dynamic Voltage and Frequency Scaling

- Adjust power supply voltage and operating frequency to optimize dynamic current consumption.
- Power supply voltage scaling vs. CMOS process node:

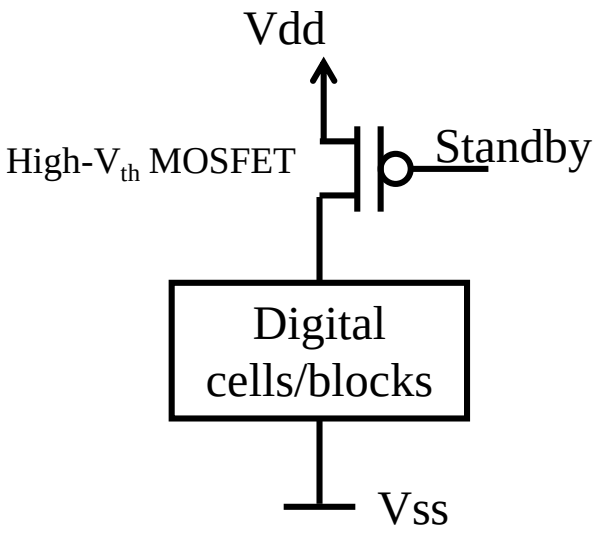


- Example: STM32L476: two selectable power supply voltage ranges (1.05 and 1.32 V).

Hardware solutions to reduce energy consumption of MCU

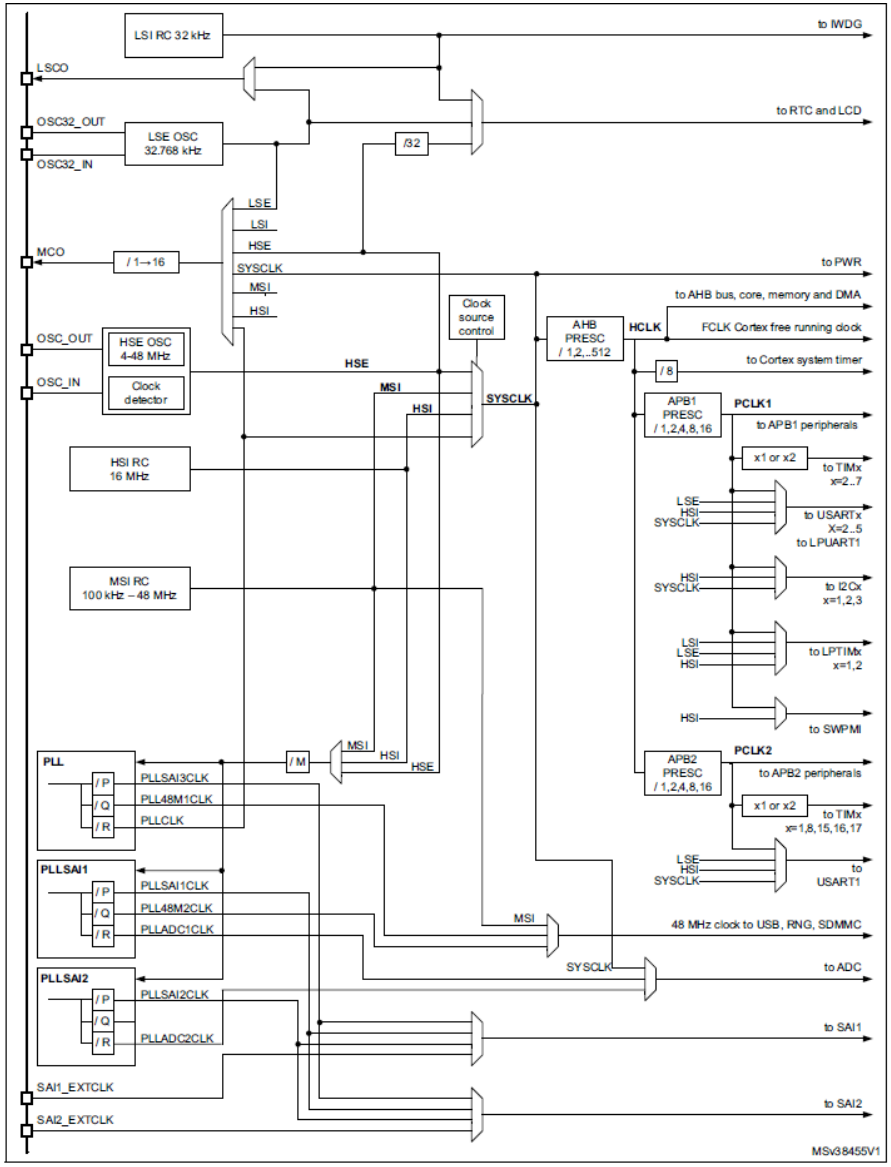
Power gating

- Disconnect unused internal blocks from power supply rails to reduce leakage currents.
- Central strategies for low-power modes (standby, shutdown modes)
- Add delays for disconnection/reconnection (long wake-up)



Clock source - management

- Depending on the clock generator nature, the current consumption changes. Trade off between consumption, max. frequency and stability.
- Using external quartz or oscillator improves stability at the cost of a higher consumption (e.g. due to I/O terminals)
- Increasing the number of internal clock buses with local prescaler to adjust peripheral operating frequency optimum
- Consequence: complex clock management



STM32L476 clock tree

Clock gating

- Disconnect unused digital blocks and peripherals from clock tree, in order to reduce dynamic current consumption.

$$P_{dyn} = N_{sw} C_L V_{dd}^2 F_{clk}$$

Clock gating reduces the equivalent capacitive load connected to the clock tree

Hardware solutions to reduce energy consumption of MCU

Memories

- Current MCU embeds SRAM and Flash/EEPROM memories.
- Flash is non-volatile but consumes more than SRAM, especially during erase/write phase.
- Flash is mainly dedicated to Code storage or data-logging
- Small area of SRAM for backup recordings supplied by a special power supply domain (back-up domain), usually supplied by a battery
- Data storing in back up registers or memories requires extra time for volatile memory recovering.
- Development of MCU with FRAM (Ferroelectric RAM) → more rapid and less consumption !

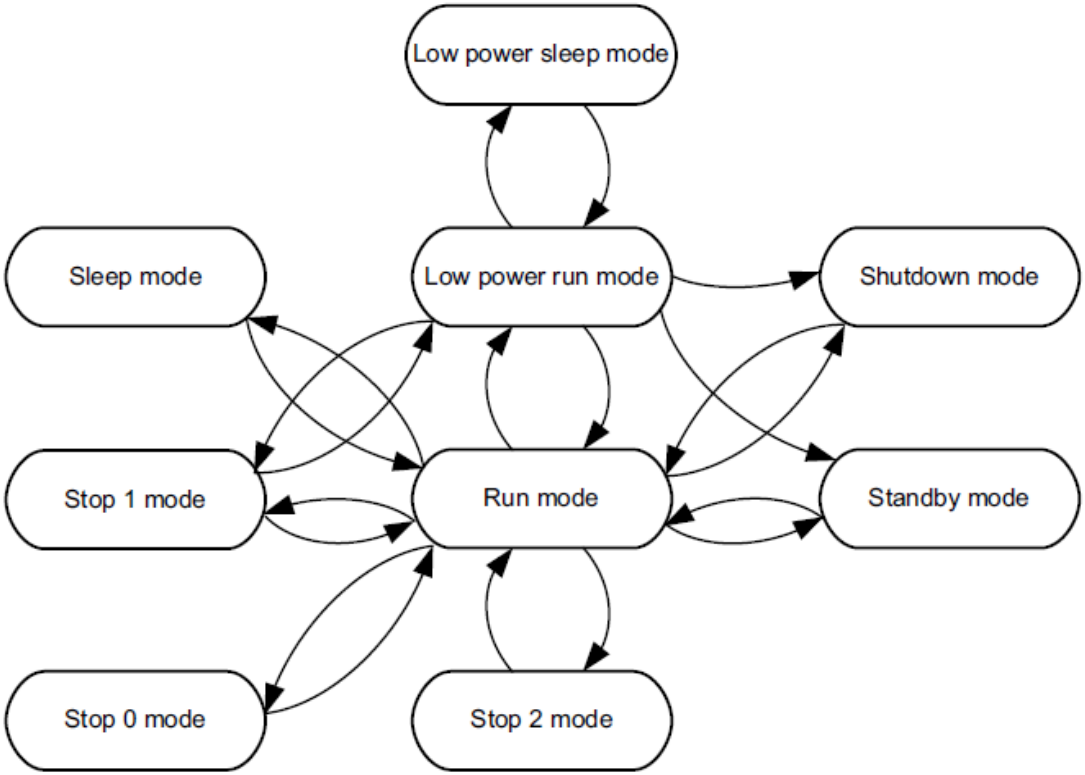
Wake-up mechanisms

- Require to exit Low-power modes
- Based on:
  - Autmatic wakeup based on RTC (periodical wake-up) → low-power counter and 32 kHz external quartz oscillator
  - Watchdog
  - External interrupt lines based on low power peripherals (UART, USB, I/O, analog comparators).
- All these mechanisms must be supplied by a specific power supply domain: backup power supply domain.

Hardware solutions to reduce energy consumption of MCU

Low-power modes

- Modern MCU (especially ultra-low power MCU) provide many run and low-power modes with different trade-offs between speed performance/current consumption/wake up times.
- A large choice to respond to the multiple requirements of end-users
- Configuration at global and local (peripheral) levels.
- Example: STM32L476



Hardware solutions to reduce energy consumption of MCU

Low-power modes

- Example: STM32L476 :

Mode	Active blocks (if activated)	Typical consumption (voltage ranges 1 & 2)	Wake-up time
Run	Everything	107 – 92 µA/MHz	
Low-power run	Low-power regulator, everything except PLL, system clock < 2 MHz	102 µA/MHz	
Sleep	CPU Off	28 – 26 µA/MHz	6 cycles
Low-power sleep	CPU Off, Low-power regulator, system clock < 2 MHz	36 µA/MHz	6 cycles
Stop0	CPU Off, Flash Off, low-speed clock, main regulator, DAC, OpAmp, Comp, UART, LPTimer … on	100 µA	0.7 µs (SRAM) – 4.5 µs (Flash)
Stop1	CPU Off, Flash Off, low-speed clock, LP regulator, DAC, OpAmp, Comp, UART, LPTimer … on	4.6 µA	4 µs (SRAM) – 6 µs (Flash)

Hardware solutions to reduce energy consumption of MCU

Low-power modes

- Example: STM32L476 :

Mode	Active blocks (if activated)	Typical consumption (voltage ranges 1 & 2)	Wake-up time
Stop2	CPU Off, Flash Off, low-speed clock, LP regulator, Comp, LPUART, LPTimer on	1.3 µA	5 µs (SRAM) – 7 µs (Flash)
Standby	CPU power Off, Flash Off, low-speed clock, LP regulator on or off, only RTC, BOR and Watchdog	0.28 – 0.45 µA	14 µs
Shutdown	CPU and SRAM power off, Flash off, low speed ext. clock, only RTC	260 nA (with RTC) – 8 nA (without RTC)	256 µs

Software for low-power MCU

Constraints for the choice of low-power MCU and software

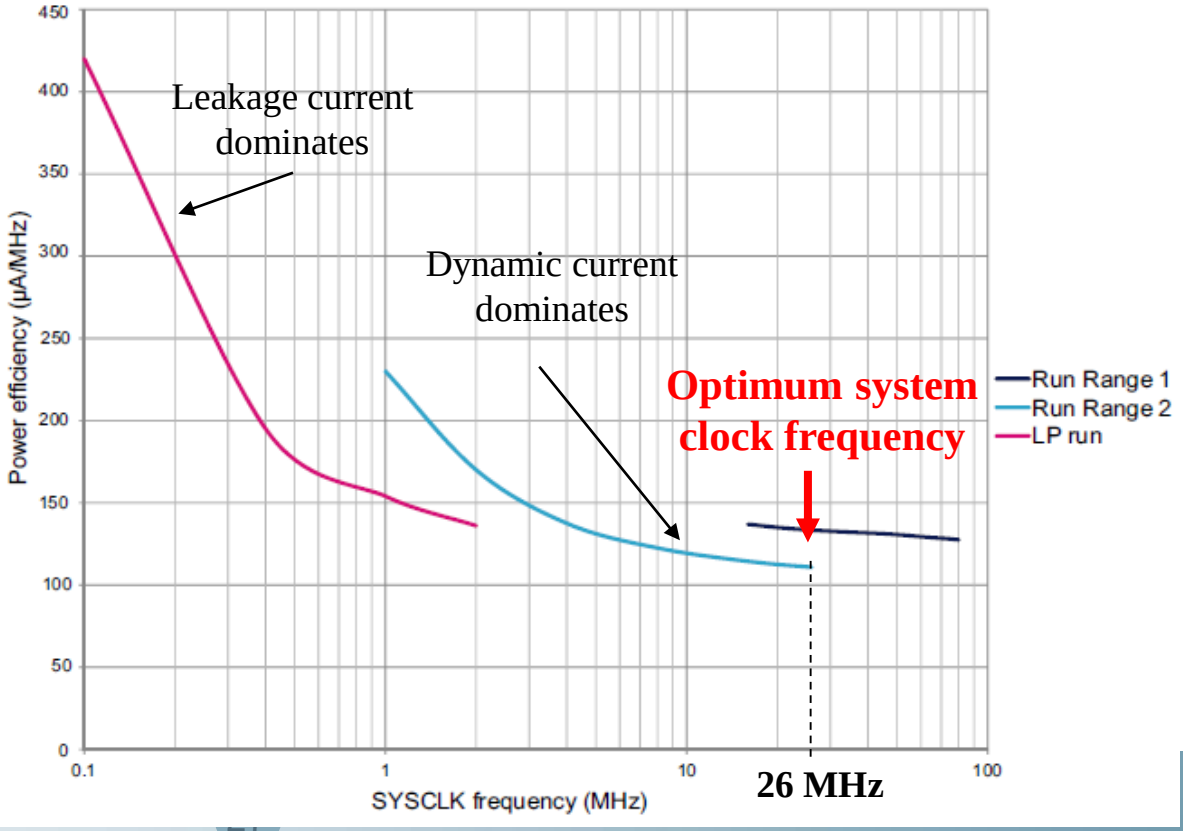
- Necessary peripherals for the application (during active and standby phases)
- Real-time constraints
- Wake-up sources
- Wake-up time requirements
- Safety / security requirements (low-voltage detection, brown-out reset, CRC…)
- Retention of memory content
- I/O state and pull-up/down maintained

Adequate use of low-power modes

- Depend on application constraints (required performance, real-time constraints, wake-up sources and periods…) and environment (temperature)
- Selection of Run modes: the optimal operating frequency should be selected according to the power efficiency and the required performances
- Example: STM32L476

AN4746 – « Optimizing power and performance with STM32L4 Series microcontrollers", ST Microelectronics

Available simulator: STM32CubeMX

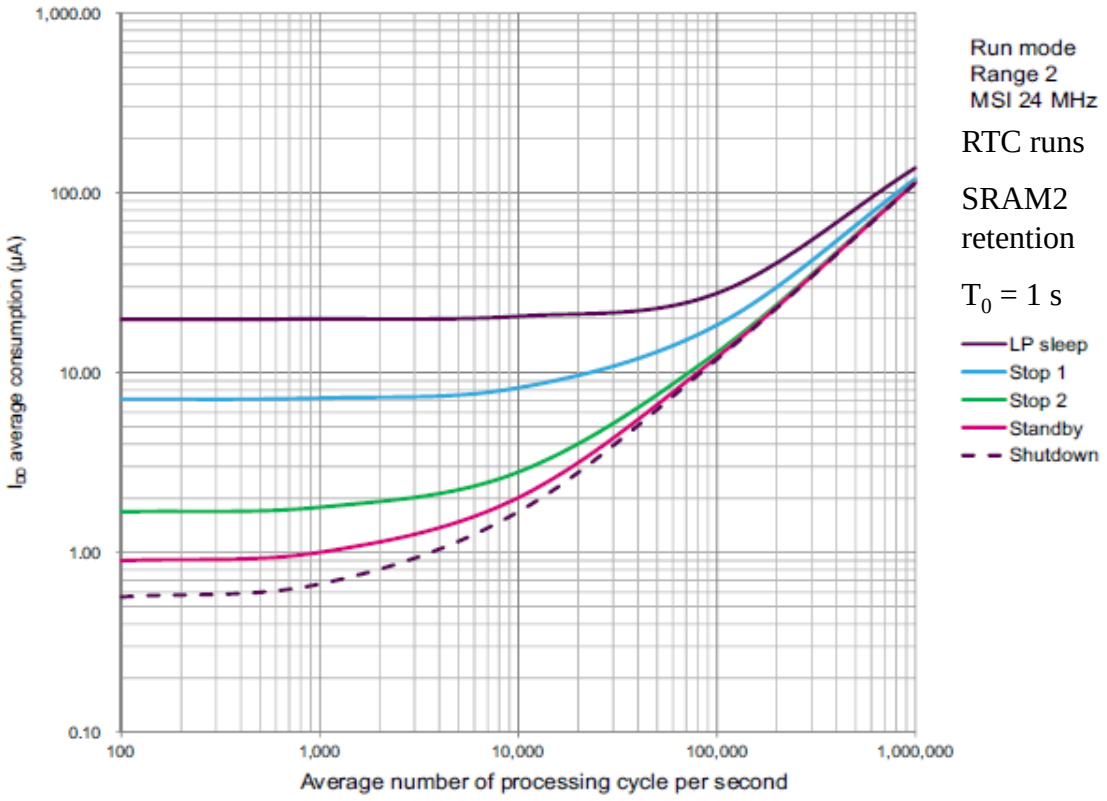


Adequate use of low-power modes

- Selection of Low-Power modes to minimize the average current: depends on Run mode frequency, number of operation cycles, wake-up period, wake-up mechanisms and max. wake-up time.

$$I_{avg} = I_{inactive} + (I_{process} - I_{inactive}) \frac{NOC}{F_{clk}.T_0}$$

- Example: STM32L476:
  - ✓ SRAM retention → Shutdown discarded
  - ✓ Standby mode is optimal choice
  - ✓ If more than 100000 NOC, Stop2 leads to the same consumption than Standby

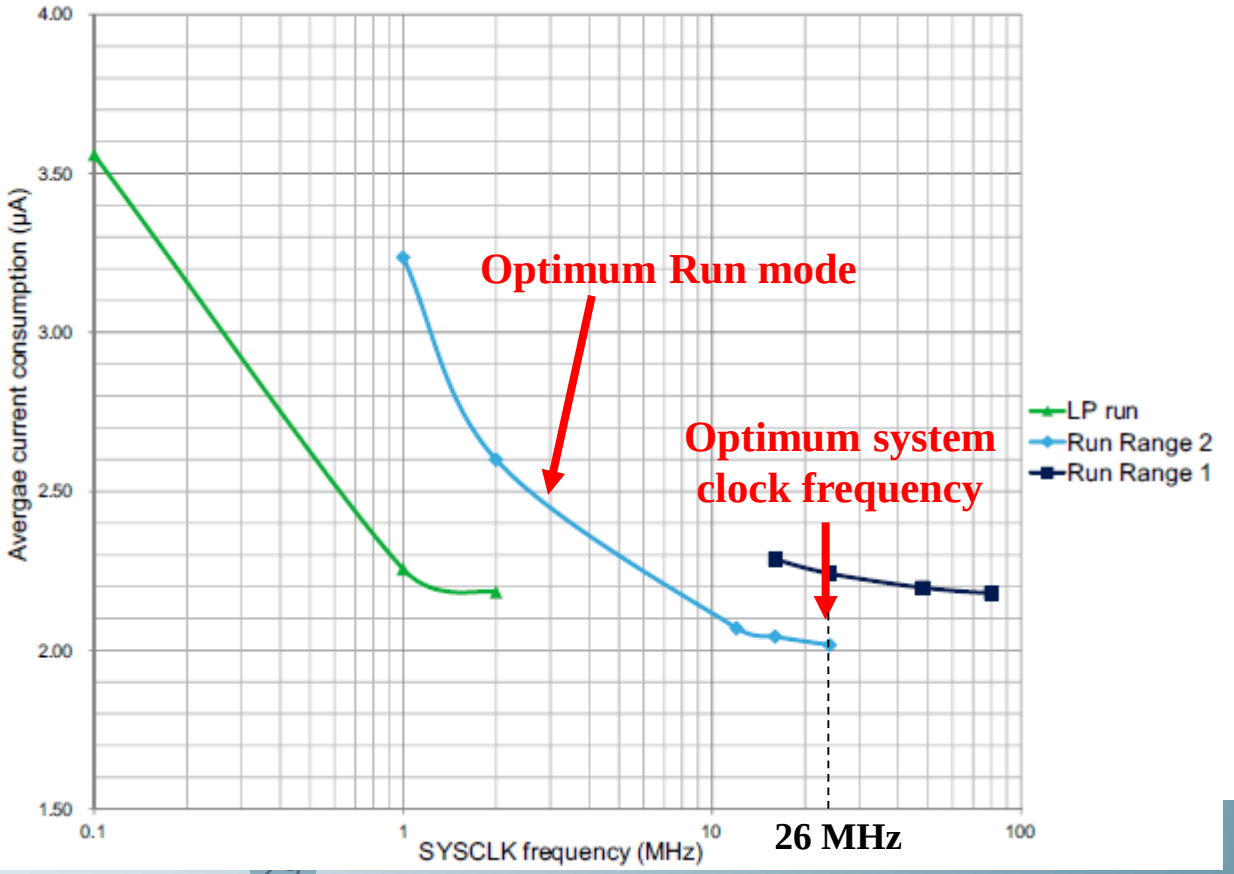


AN4746 – « Optimizing power and performance with STM32L4 Series microcontrollers", ST Microelectronics

Adequate use of low-power modes

- Example: STM32L476
- Impact of Run mode selection if Standby mode is used (same parameters than previous simulation, NOC = 10000 cycles):

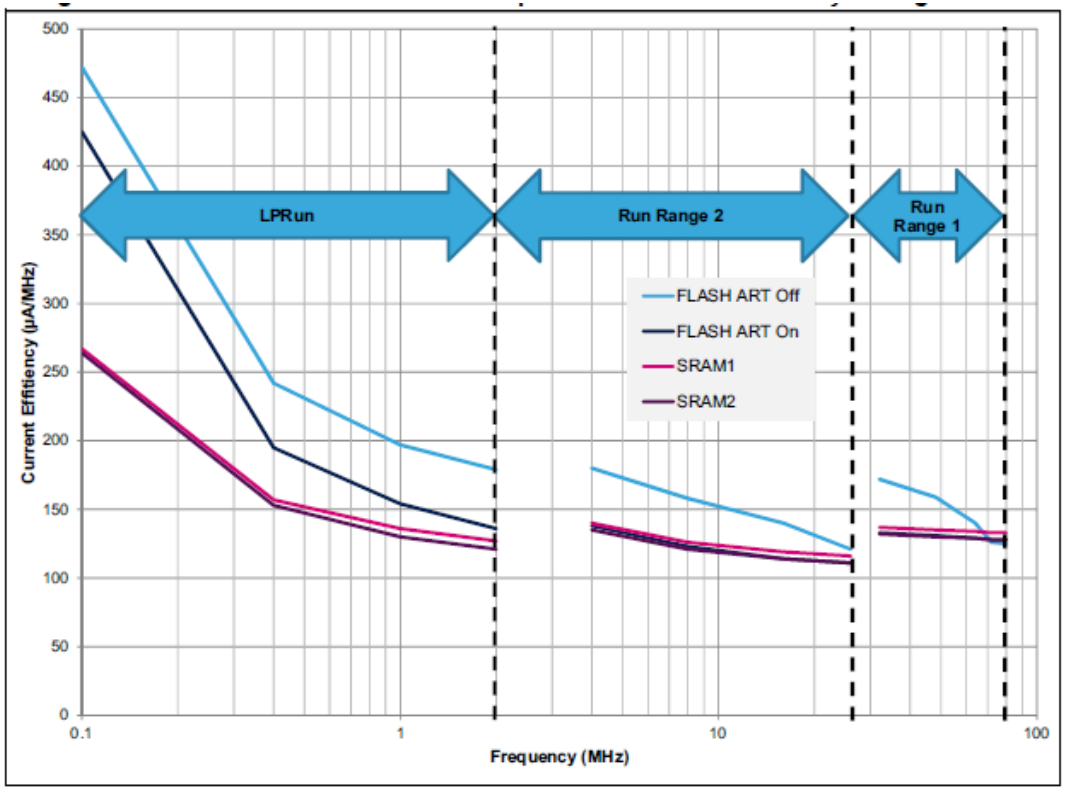
AN4746 – « Optimizing power and performance with STM32L4 Series microcontrollers", ST Microelectronics



Code execution

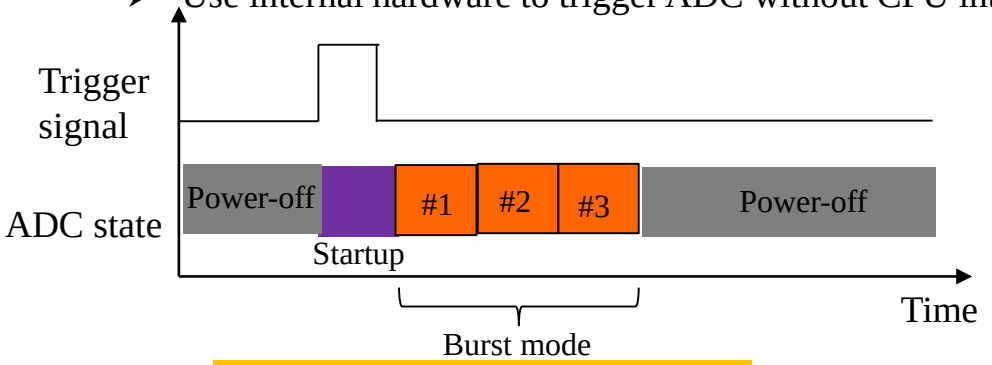
- Executing program from RAM consumes less energy and is more rapid (at a price of a copy of the program into RAM after hard Reset)
- Optimizing the code to reduce the duty cycle.
- Reduce the use of CPU, use all the peripheral mechanisms to limit CPU intervention
- Avoid polling, prefer interrupt triggering to wake-up CPU only when necessary

AN4621 – « STM32L4 and STM32L4+ ultra-low-power features overview", ST Microelectronics

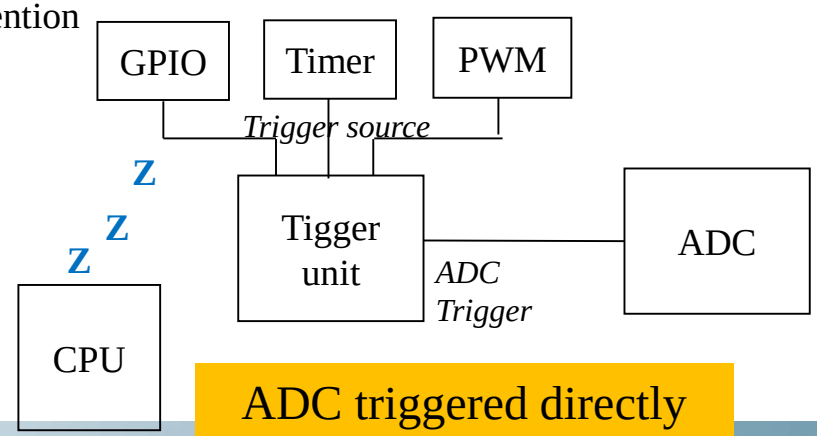


Avoid continuous operation of peripherals

- CPU is the main contributor of current consumption → low-power mode if no processing required (at least sleep mode → WFI instruction in RAM, wake-up by any interrupt)
- But some peripherals can have a dramatic consumption if they operate continuously. Examples: ADC to capture sensor voltage (up to 0.66 mA with STM32L476)
- For ADC, if available hardware options exist:
  - Operates at max. frequency and enter in low power mode as fast as possible.
  - Burst mode: several conversion from one trigger source, then enter in low-power mode
  - Auto shutdown: periodic or triggered wakeup to ensure conversion followed by an entry in low-power mode, without CPU intervention
  - Use internal hardware to trigger ADC without CPU intervention



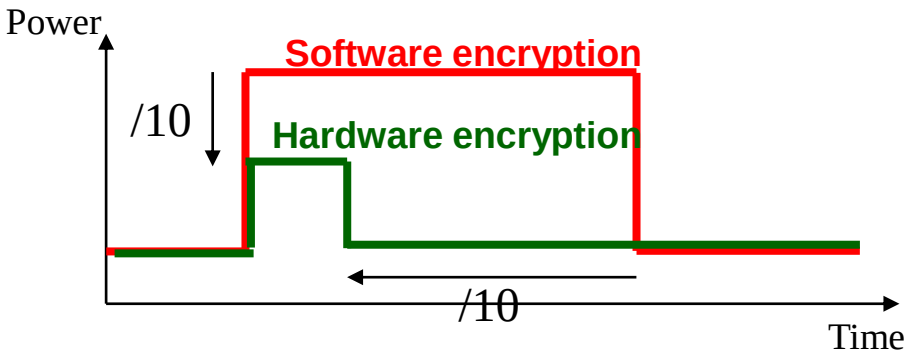
Automatic power-on/off + burst mode



ADC triggered directly by peripherals

Avoi CPU intervention

- Use DMA for data transfer from memory ← → peripherals (ADC, communication interface)
- Autonomous communication interface (e.g. Batch acquisition mode in STM32L476 for I2C and LPUART)
- Use hardware acceleration: e.g. for AES encryption



Software vs. hardware AES encryption

GPIO configuration

- Avoid floating unused input pin: intempestive switching of internal Schmitt trigger increases energy consumption.
- Configure to:
  - Digital input with internal Pull-down or Pull-Up resistor
  - Digital output in Push-Pull mode and tie to ground
  - Analog input (no Schmitt trigger)
- Switch-off the clock of unused GPIO banks
- GPIO states and pull-up/pull-down are kept during low power state. Be careful with low power mode exit. Example – STM32L476: after shutdown exit, GPIO are reconfigured with default states.