Chapter five

Low Power Computing

Introduction

- Lower power embedded systems might have a range of power management strategies implemented at the design level.
- There are also sophisticated algorithms that enable low power consumption and power management in embedded systems, and any low power system might need a combination of strategies to prevent excess battery power from being used.
- If you’re designing an embedded system that must be highly power efficient while also providing the required level of computing power, there are multiple approaches you can take.
- It all depends on the design requirements you need to meet, followed by opting for the right low-power components where possible.

Low Power Embedded Systems Design Requirements

- The first stage in any successful project execution is requirements gathering.
- For a low-power embedded system, extending battery life or ensuring overall power efficiency are likely top priorities, you'll need to develop requirements around your need for power efficiency.
- You'll need to make decisions around some of the following points:
  - If your system is battery powered, how long the system will need to be deployed before requiring charging?
  - How much compute resources does your system need to do its job?
  - Are there power-hungry peripherals or circuit blocks that need intermittent power?
  - What type of power management techniques or features can your components support?
- If you have an idea of the answers to these questions, you can come up with a strategy to design low power embedded systems.
- The design strategy starts with selecting critical components and peripherals, which may need to be paired up with a power management algorithm.

Component Selection

- Power consumption in an embedded system is heavily dependent on the main processor, analog front-end, and any peripherals like displays.
- Many processor units (i.e MCU)and other components are specifically marketed as low-power devices.
- When selecting components, here are some tips you can use to design a low-power system architecture:

1. Start with must-haves. Start here and try to find the lowest power option that satisfies the minimum functional requirements.
2. Peripheral architecture. Think about how peripheral blocks will interact with the host processor and the environment, and work these into your system architecture.
3. Interfaces and receivers. Various low-speed digital protocols (I2C, SPI, etc.) can have different power outputs. Also, receivers and converters like ADCs could be run at lower sampling rate to reduce total power consumption.

4. Components with sleep/hibernate modes. Some processors and other IC have sleep modes, where current is only supplied to critical functional blocks in the IC.

- Current can drop to well below 0.01 mA in these modes.

5. Power regulation. Once all the important components have been selected, it’s time to think about power regulation.

- Aim for the highest efficiency power regulation strategy you can.
- Careful design of regulation stages can ensure power conversion efficiency stays well above 95%.

Power Management Techniques in Embedded Systems

- ❖ Component-level power consumption is easy to address by choosing the right parts.
- ❖ However, there are times where a specific component that runs at high power is a must-have.
- ❖ At this point, the system should be designed so that certain peripherals can be toggled on and off, battery charge management is implemented, and algorithm optimization.

Peripherals

- ❖ A strategy where peripherals are switched on/off as needed by the host controller is one way to ensure power is only being consumed when it is needed.
- ❖ Some components may implement this on- chip, where groups of interfaces are switched off and the core voltage is reduced to when the component is not actively processing data.
- ❖ These could be switched off by simply cutting power to the peripheral block, or by putting a slave processor into sleep mode.

Battery Management

- This strategy is useful for multi-cell battery packs in series, but not all battery packs will support this.
- Implementing a battery management algorithm with a balancing system is one way to prevent excessive power from being drawn from a single cell and ensuring charge is evenly distributed across cells.

Algorithm Optimization

- Embedded systems designers should ensure any specialty algorithms in their firmware are optimized so that compute operations are minimized.
- Reducing algorithmic complexity leads to less power consumption in the host processor.
- By eliminating background processes and services, or by eliminating unnecessary computational operations, the processor will be performing fewer tasks while idling and will consume less power.

Importance of Low Power Design

Low power design is essential in

- High performance systems (power dissipation reduces reliability and increases the cost imposed by cooling systems and packaging)
- Portable systems (battery technology cannot keep the pace with large demands for devices with light batteries and long time between recharges)

- Minimizing the power consumption is important for
  - The design of the power supply
  - The design of voltage regulators
- Minimizing the energy consumption is important due to
  - Limited battery capacities
  - Very high cost of energy
  - High costs
  - Limited space
- Long lifetimes, low temperature

Low Power Embedded System Design

- Power consumption is a very important issue in embedded system design.
- Many of the embedded applications are built around batteries, and thus, battery life is a critical concern in judging their acceptance.
- Some of the important issues to consider include power consumption limits, size restriction, I/O requirements, operational **duty cycle**, etc.
- There are several practical considerations that are involved in the process

1. Recharging facility: it is not always possible to recharge/replace the batteries of embedded systems.
   - This is in particular true for systems employed at remote and difficult-to-reach places, for example, a data collection center for wildlife.
2. Device size: unfortunately, battery technology has not scaled down as the same rate as IC technology. Thus, the weight and size of the device is often guided by the required battery capacity and its associated size. This is a very practical problem for mobile and portable devices.

3. Duration of operation: the duration of which a device needs to be active, is another major issue determining the total energy consumption.

   If a device is idle for majority of the time, a power-down mode may be incorporated into the design to save power.

4. Power required by the device: this is the most important issue and needs to be estimated even at the stage of initial design.

   A better estimation helps in the design process to try out alternatives.

5. I/O device types: the I/O interface, such as optically isolated I/O and electro-mechanical relays consume high power.

   Thus, the system designer needs to avoid them altogether, or minimize their active periods.

6. Speed of operation: power consumption is directly proportional to the frequency of operation.

The system designer needs to identify the optimum speed for each component, so that the performance target is met,.

Power Reduction Techniques

Power reduction can be attempted at all levels of design hierarchy

- Algorithm
- Architecture
- Logic, and
- Device levels.

Algorithmic Power Minimization

- It mainly focuses on reducing the number of operations requiring larger power at implementation.
- For example, in many processors, an addition/subtraction operation may be different from a logical operation.
- "Whether x is equal to y", one may first perform a subtraction operation followed by checking the status register for zero-bit.
- But, the logical operation takes lesser power, x may be directly compared with y using a comparison instruction.

Algorithmic Power Minimization

The important issues to be judged for selecting a particular algorithm from alternatives:

1. Memory Reference
   - A large number of accesses to the memory mean good amount of activity in the address/data bus lines.
   - If the access pattern is sequential, only the last significant bits of address bus change, whereas for random access through the memory, most of the address bits will switch, thus creating higher power dissipation.
2. Presence of cache memory
   - The presence and structure of cache memory plays an important role.
   - Cache can be fruitful utilized to reduce both execution time and power of an implementation if the underlying algorithm has got locality in its behavior.
3. Compiler optimization technique

- The typical techniques used by an optimizing compiler can be used to reduce power consumption of a piece of code.

4. Number representation: This is another area for algorithmic power trade-off.
   - integer vs. floating point representation: Fixed point operation are much simpler than floating point. Thus, it normally leads to power saving, though accuracy may suffer.

Architectural Power Minimization

- The architectural level transformations can be used to introduce power saving in a design.
- The system is capable of running at frequency f and the speed of a system is proportional to the supply voltage

Logic and Circuit Level Power Minimization

- The power reduction approaches at logic and circuit level to reduce the effective switched capacitance.
- For example, for a two-input NAND gate with uniform probability distribution for the inputs,