SELECTING THE RIGHT MICROCONTROLLER UNIT

Takawira F., Dawoud D.S.
University of Natal University of Natal
Durban, South Africa
email ftakaw@nu.ac.za , dawoudd@nu.ac.za

Abstract:

Selecting the proper microcontroller unit (MCU) for a given application is one of the critical decisions which control the success or failure of the task. There are numerous criteria to consider when choosing an MCU, or in general the processor technology. This paper is numerating and explaining most of the metrics to consider during the phase of selecting the right MCU. It presents an outline of the thought guiding this decision.

Keywords: Microcontroller Unit (MCU), process technology

1. Introduction –

1.1 Technologies Involved in Design Process

We can define technology as a manner of accomplishing a task, especially using technical processes, methods, or knowledge. Three types of technologies are central to embedded system design:
i. ***Design technologies***: Design technology involves the manner in which we convert our concept of desired system functionality into an implementation.
ii. ***IC technologies***: Every processor must be eventually be implemented on an integrated circuit (IC). IC technology involves the manner in which we map a digital (gate-level) implementation onto an IC. IC technologies differ by how customized the IC is for particular design. In other words it differs from one another by who is responsible to connect the three groups of layers of any IC; the bottom layers or the transistor layers, the middle layers or the logic components layer and the top layers that connect these components with wires. Three IC technologies can be identified.
a. Full-custom/VLSI: All the layers are optimized for a particular embedded system's implementation. Normally it has very high cost, but gives excellent performance with small size and power. Such technology is usually used only in high-volume or extremely performance critical applications.
b. Semicustome Application-specific IC (ASIC): In ASIC technology, the bottom layers are fully built, leaving us to finish the upper layers. The gate array and standard cell are examples of this IC technology.
c. Programmable Logic Device (PLD): In this technology, all the layers already exist. The layers
implement a programmable circuit. The programming that takes place may consist of creating or
destroying connections between wires that connect gates, either by blowing a fuse, or setting a bit
in a programmable switch. PLA, PAL and FPGA are the most popular types of PLD technology.
The designer by using some equipment and IC technology is independent from processor technology; any type o processor can be mapped to any type of IC technology.
iii. ***Processor technologies***: relates to the architecture of the computation engine used to implement a system's desired functionality. Three processor technologies can be identified; use of general-purpose processor (software solution), use of single-purpose processor (hardware solution), and use of application specific processor (e.g. use of microcontroller).

Any of the above mentioned three processor technologies can be used to implement any task. It is the job of the designer to select the processor that optimizes some design metrics.

The subject of this paper is the selection of the right processor technology that can be used to implement a given task. For this reason, the next three subsections are used to highlight the three processor technologies and the design metric benefits/drawbacks of each technology. In section-2 we present the design metrics that can be used to quantify the suitability of the selected processor technology. In section-3 we present an outline of thought process guiding the designer to select the optimum microcontroller unit (MCU)

1.2 Processor Technologies

Three technologies can be identified [1-2]:

1.2.1 General-Purpose Processors – Software Solution

This is the case of designing a microprocessor-based embedded system. The microprocessor, by definition, is a programmable device that is suitable for a variety of applications. It has a general datapath. The datapath is general enough to handle a variety of computations, so such a datapath typically has a large register file and one or more general-purpose arithmetic-logic units (ALUs). An embedded system designer, however, need not be concerned about the design of a general-purpose processor. An embedded system designer simply uses a general-purpose processor, by programming the processor’s memory to carry out the required functionality.

Many people refer to this part of an implementation as the "software" portion. Using a general-purpose processor in an embedded system may result in several design metric benefits. Time-to-market and design costs are low because the designer must only write a program but not do any digital design. Flexibility is high because changing functionality requires changing only the program.

Unit cost may be low. Performance may be fast for computation-intensive applications, if using a fast processor, due to advanced architecture features and leading-edge IC technology.

However, there are also some design-metric drawbacks. Performance may be slow for certain applications, especially in case of real-time systems. Size and power may be large due to unnecessary processor hardware.

1.2.2. Single-Purpose Processors – Hardware Solution

A *single-purpose processor* is a digital circuit designed to execute exactly one program. An embedded system designer may create a single-purpose processor by designing a custom digital circuit. Alternatively, the designer may purchase a predesigned single-purpose processor. Many people refer to this part of the implementation simply as the "hardware" portion, although even software requires a hardware processor on which to run. Other common terms include coprocessor, accelerator, and peripheral. The datapath of the single-purpose processor contains only the essential components to fulfil the task. Since the processor only executes this one task, it is possible to hardwire the program’s instructions directly into the control logic and use a state register to step through those instructions, so no program memory is necessary.

The use of a single-purpose processor in an embedded system results in several design-metric benefits and drawbacks, which are essentially the inverse of those for general-purpose processors. Performance may be fast, size and power may be small, and unit cost may be low, while design time and design costs may be high, flexibility low, unit cost high for small quantities, and performance may not match general-purpose processors for some applications.

1.2.3.Application-Specific Processors

An *application-specific instruction-set processor* (ASIP) can serve as a compromise between the other processor options. An ASIP is a programmable processor optimized for a particular class of applications having common characteristics, such as embedded control,

digital-signal processing, or telecommunications.
The datapath of such a processor is optimized for the application class, perhaps adding special functional units for common operations and eliminating other infrequently used units.
Using an ASIP in an embedded system can provide the benefit of flexibility while still achieving good performance, power, and size.

Microcontrollers (MCU) and **digital signal processors** (DSP) are two well-known types of ASIPs that have been used for several decades. A microcontroller is a microprocessor that has been optimized for embedded control applications. Such applications typically monitor and set numerous single-bit control signals but do not perform large amounts of data computations. Thus, microcontrollers tend to have simple datapaths that excel at bit-level operations and at reading and writing external bits. Furthermore, they tend to incorporate on the microprocessor chip several peripheral components common in control applications, like serial communication peripherals, timers, counters, pulse-width modulators, and analogue-digital converters, all of
which will be covered in a later chapter. Such incorporation of peripherals enables single-chip implementations and hence smaller and lower-cost products.

1.3 Selection Process

The process of selecting the right MCU goes through the following steps.
Step 1: ***Choose the right processor technology***: The goal of this step is to select one of the three processor technologies. The selection is based on the system requirements and the strengths and weaknesses of each technology.
Step 2: ***Define the System Requirements***: If you decided to use MCU, the designer must answer the following question "What does the MCU need to do in my system?" The answer to this simple question dictates the required MCU features for the system and, thus, is the controlling agency in the selection process.
Step 3: ***Prepare a List of all MCUs candidate***: Conduct a search for MCUs which meet all of the system requirements. This usually involves searching the literature and also consultations. This step considered to be successful, if the search results in more than one MCU fulfil all the system requirements. If this is so, the designer goes to the next step. If this step failed to find any MCU that fulfill the requirement, the designer has to go back to step 1 and selects another processor technology.
Step 4: Finalize the Selection: Attempt to reduce the list of the acceptable MCUs to a single choice.

In the following sections we are going to discuss in details the four steps. The case if step 3 failed to find single MCU that fulfill all the system requirements will be discussed at the last section.

2. Choosing the Right Processor Technology

The system analysis phase of the project will result in identifying the subsystems required to implement the given task and also the function and the specifications of each subsystem. As a matter of fact any of the above mentioned three processor technologies can be used to implement any of the subsystems. With each processor technology has its own strength and weakness, it is important to quantify the suitability of each technology to implement the subsystems. In most applications the following metrics can be used to quantify the suitability of processor technology (accordingly, they are needed to be considered before selecting the processor technology) [3-5]:

Cost: In the case of using off the shelf products for the implementation of the system, this metric represents the cost of the used ICs, e.g., the cost of the selected MCU together with the supporting ICs that may be needed to fulfil the system requirements. In the case of designing special hardware, the cost will consist of two parts: the nonrecurring engineering cost (NRE) (the "first silicon" cost including the research and development cost), and the cost of manufacturing the unit.

Performance: The execution time of the system or the processing power. It is usually taken to mean the time required to complete a task (*latency or response time*), or as the number of tasks that can be processed per unit time (*throughput*). Factors which influence throughput and latency include the clock speed, the word length, the number of general purpose registers, the instruction variety, memory speed, programming language used, and the availability of suitable peripherals. *Size*: The physical space required by the system, often measured in bytes for software, and number of gates or transistors for hardware. In case of using off the shelf products (as in case of using MCU or MPU), the size is the physical area required to accommodate the sockets of the used ICs (the footprint).

Power: The amount of power consumed by the system, which may determine the lifetime of a battery, or the cooling requirements of the IC, since more power means more heat.

1. Heat generation is a primary enemy in achieving increased performance. Newer processors are
larger and faster, and keeping them cool can be a major concern.
2. Reducing power usage will be the primary objective in case of designing a project that needs the components to be crammed into small space. Such applications are very sensitive to heat
problems.
3. With millions of PCs in use, and sometimes thousands located in the same company, the desire to conserve energy has grown from a non-issue to a real issue in the last five years.
4. Power consumption has an impact on everything from cooling method selection to overall system reliability.

Flexibility: The ability to change the functionality of the system without incurring heavy additional cost.

Software is typically considered very flexible. Generally speaking, single chip MCU are not very flexible in use compared with single chip microprocessor. The latter, if based on a standardised bus, can be reconfigured by swapping peripherals and altering input-output routine. The ultimate in flexibility is probably achieved through the use of general-purpose processor.

Reliability: Reliability is an attribute of any computer-related component (software, hardware, or a network, for example) that consistently performs according to its specifications. It has long been
considered one of three related attributes that must be considered when making, buying, or using a computer product or component. Reliability, availability, and serviceability - RAS, for short – are considered to be important aspects to design into any system. In theory, a reliable product is totally free of technical errors; in practice, however, vendors frequently express a product’s reliability quotient as a percentage. Evolutionary products (those that have evolved through numerous versions over a significant period of time) are usually considered to become increasingly reliable, since it is assumed that bugs have been eliminated in earlier releases.

Software bugs, instructions sensitivity, and problems that may arise due to durability of the EEPROM and Flash memories (The nature of the EEPROM architecture, limits the number of updates that may be reliably performed on a single location – this is called the *durability of* the memory. At least 10,000 updates are typically possible for EEPROM and 100 updates for flash memory), are some of the possible reasons of the failure of embedded systems.

Reliability of a system depends on the number of devices used to build the system. As the number of units used increases, the probability of error (or failure) increases which means lower reliability. This is why the microcontroller-based systems are, generally, more reliable than microprocessor-based systems.

. *Availability* - *Second source suppliers*: Most of the major microcontrollers and microprocessors are now made by more than one manufacturer (Siemens and Philips are the biggest manufacturers for Intel 8051 family).

. *Serviceability - Manufacturer’s support*: Manufacturer’s support covers the provision of a range of services from the development system and its associated software through the documentation,
maintenance of the development system and providing answers to technical queries.

Maintainability: The ability to modify the system after its initial release, especially by designers who did not originally design the system.

Range of complementary hardware: For some applications the existence of a good range of compatible ICs to support the microcontroller/microprocessor may be important.

Special environmental constraints: The existence of special requirements, such as military specifications or minimum physical size and weight, may well be overriding factors for certain tasks. In such cases the decision is often an easy one.

Ease of use: This will affect the time required to develop, to implement, to test it, and to start using the system. These three factors - design time, manufacturing time, and testing time- are the main factors defining the time-to-market merit which is very important if the system is designed for commercial use.

In commercial applications, introducing an embedded system to the marketplace early can make a big difference in the system' profitability, since market windows for product are becoming quit short. This is very important factor because the time-to-market factor defines the profitability.

Software Support: Newer, faster processors enable the use of the latest software. In addition, new processors such as the Pentium with MMX Technology, enable the use of specialized software not usable on earlier machines. Easier language means shorter time to learn and better maintainability

Motherboard Support: The processor you decide to use in your system will be a major determining factor in what sort of chipset you must use, and hence what motherboard you buy. The motherboard in turn dictates many facets of your system's capabilities and performance.

Correctness: Our confidence that we have implemented the system's functionality correctly. We can check the functionality throughout the process of designing the system, and we can insert test circuitry to check that manufacturing was correct.

Safety: The probability that the system will not cause harm.

Metrics typically compete with one another; Improving one often leads to worsening of another. For example, if we reduce an implementation’s size, the implementation’s performance, power consumption and NRE may suffer. Keeping this in mind and also the fact that each processor technology has its own metrics strength and weakness, the key embedded system design challenge is the simultaneous optimization of competing design metrics.

To optimize the system under design, it is the job of the designer to study the rule of each subsystem within the complete frame of the system, and from here he can identify the most critical metrics that optimize the subsystem. Based on that, the designer can define the most suitable technology that he is going to use to implement each subsystem. Accordingly, this phase will determine the subsystems that the designer is recommending the use of MCU for implementation and also the requirements from each MCU. The complete list of requirements must cover all (or the maximum possible number) the metrics mentioned above. For example, it must cover the requirements from hardware and software resources, performance, interfacing with other subsystems, power supply specifications, etc.

3. What does the MCU need to do in the system?

In this section we present an outline of the thought process guiding the designer to prepare a complete list of the system requirements. This represents the most important step towards finalizing the selection process.

The proposed way takes the form of questions to be answered by the designer. Some of the questions that the designer has to ask and to find for them answer are:

3.1 Questions related to the system hardware requirements:

All MCUs have on-chip resources to achieve a higher level of integration and reliability at a lower cost. An on-chip resource is a block of circuitry built into the MCU which performs some useful function under control of the MCU. Built-in resources increase reliability because they do not require any external circuitry to be working for the resource to function. They are pre-tested by the manufacturer and conserve board space by integrating the circuitry into the MCU. The on-chip resources defer from one MCU to another. Then it is very important in the analysis phase to define exactly the system hardware requirements to be able to select the proper MCU. Failing to get an MCU that fulfill all the system hardware requirements means one of two possibilities:

1. To study whether the budget and the available space constraints are going to allow you to select the nearest suitable MCU and support it with additional ICs, to fulfill the system requirements, or not?
2. The possibility of going to another processor technology (e.g. use of microprocessor or use of single-purpose processor).

Some of the questions to be asked and answer are:

What peripheral devices are required?

How many devices/bits (I/O pins) need to be controlled? Among the many possible types of I/O devices to be controlled/monitored are RS-232C terminals, switches, relays, keypads, sensors (temperature, pressure, light, voltage, etc.), audible alarms, visual indicators(LCD displays, LEDs), analog-to-digital (A/D), digital-to-analog (D/A), liquid crystal display drivers(LCD), and vacuum fluorescent display drivers (VFD).

What is the expected capacity of the ROM and RAM that are required to store the programs and the data?

. How many timers the application needs? Timers include both real-time clocks and periodic interrupt timers.

What is the range and resolution of the timer? Also consider if your system needs any subfunctions, such as timer compare and/or input capture lines.

Is your application needs some special resources as internal/external bus capability, computer operating properly (COP) watchdog system, clock operating properly detection, selectable memory configurations, and system integration module (SIM).The SIM replaces the external "glue" logic usually required to interface to external devices via chips elect pins.

Is your application needs special arithmetic hardware resources such as multiply, divide, and table lookup/interpolate?

What is the word length? MCUs generally can be classified in to 8-bit,16-bit, and 32-bit groups based upon the size of their arithmetic and index register(s). The following questions must be answered before deciding about the word length:

a. Is a lower-cost 8 -bit MCU able to handle the requirements of the system, or is a higher-cost 16- bit or 32-bit MCU required?

b. Can 8-bit software simulation of features found on the 16-bit or 32-bit MCUs permit using the
lower-cost 8-bit MCU by sacrificing some code size and speed? For example, can an 8 -bit MCU be used with software macros to implement 16-bit accumulator and indexing operations?

These questions are directly related to the choice of implementation language (high-level versus assembler).

3.2 Questions related to the software (instruction set) requirements:

The instruction set, registers, addressing modes, etc., of each MCU should be considered carefully, as they play critical roles in the capability of the system. Some of the questions to be answered are:

Is the application to be bit manipulating or number crunching? Remember that bit manipulation
instructions (bit set, bit clear, bit test, bit change, branch on bit set, branch on bit clear) allow easier implementation of controller applications.

Once data is received, how much manipulation is required? Is the system to be driven by interrupt? Polled, or human responses?

Which implementation language are you going to use?

As mentioned before, the choice of implementation language (high-level versus assembler) can greatly affect system throughput, which can then dictate the choice of 8-,16-, and 32- bit architectures. System cost restraints may override this.

Do you need any special instructions to be available which could be used in your system, such as
multiply, divide, and table lookup/interpolate?
Do you need the instruction set to include instructions to handle low-power modes for battery
conservation, such as stop low-power stop, and/or wait?
How about big field instructions? In other words, do you prefer MCU with Long Instruction Word
(LIW)? Remember that the real measure of performance is how many clock cycles the system takes to complete the task in hand, not how many instructions were executed.
What are the recommended addressing modes?

3.3 Questions related to the performance:

As mentioned before, the performance can be measured by the time required to complete a task. One of the factors which influence throughput is the clock speed. Clock speed, or more accurately bus speed, determines how much processing can be accomplished in a given amount of time by the MCU.
Some MCUs have a narrow clock speed range, whereas others can operate down to zero. Sometimes a specific clock frequency is chosen to generate another clock required in the system, for example, for serial baud rates. In general, computational power, power consumption, and system cost increase with higher clock frequencies. System costs increase with frequency because not only does the MCU cost more, but so do all the support chips required, such as RAMs, ROMs, PLDs (programmable logic device), and bus drivers.
Concerning the performance, some of the questioned that must be answered are:
Is it a real-time application, and if so, are you going to build or purchase a real-time kernel program or may be a public domain version will suffice?
What is the expected CPU core throughput?
What is the CMU bus speed? Or what is the clock speed?
For serial ports, what is the data rate?

3.4 Questions related to MCU interrupts

Examining the interrupt structure is a necessity when constructing a real-time system. The designer should look at:
How many interrupt lines or levels are there versus how many does your system require?
Is there an interrupt level mask?
Once an interrupt level is acknowledged, are there individual vectors to the interrupt handler routines or must each possible interrupt source be polled to determine the source of the interrupt
In speed critical applications, such as controlling a printer, the interrupt response time, for example, the time from the start of the interrupt (worst case phasing relative to the MCU clock) until the first instruction in the appropriate interrupt handler is executed, can be the selection criterion in determining the right MCU.

3.5 Questions related to power supply

Is a single or multiple voltage power supply required for the system?
What is the power supply tolerance?
Is the device characterized for operation at your system supply voltage?
Are the voltages to be held to a small fixed percent variation or are they to operate over a wider range?
What is the operating current?
Is the product to be a core battery operated? lf battery operated, should rechargeable be used, and if so, what is the operational time required before recharging and the required time for recharging?

3.6 Questions related to size and environment:

Are there size and weight restrictions or aesthetic considerations such as shape and/or color?
Is there anything special about the operating environment such as military specifications,

temperature, humidity, atmosphere (explosive, corrosive, particulates, etc.), pressure/altitude?

3.7 Questions related to available budget and time:

Does your schedule contain enough time and personnel to develop your own application? The importance of this question will be very clear if the designer failed to find suitable MCU and the only solution is to design single-purpose processor.
What about royalty payments and bug support?

4. Finalizing the Selection

As a final step to help in the selection process, built a table listing each MCU under consideration on one axis and the important attributes on the other axis. Then fill in the blanks from the manufacturer's data sheets to obtain a fair side-by-side comparison. Some manufacturers have pre made comparison sheets of their MCU product line which makes this task much easier, but as with all data sheets, be sure they are up-to-date with current production units.
From the discussions given at section - 2, some of the possible attributes are:
Price
On-chip RAM, ROM, EPROM, and EEPROM
Timer(s) (number of timers, range, resolution, etc.)
Watchdog timer
A/D and D /A
Serial ports and parallel ports (I/O control lines)
Bus speed (minimum/maximum)?
Special hardware (multiply/divide, table lookup/interpolate, etc.)
Special instructions (multiply, divide, etc.)
Word length.
Number of available interrupts, interrupt response time (time from start of interrupt to execution of the first interrupt handler instruction)
Package size/type (ceramic DIP(dual in-line package) or LCC, plastic 0.3-inch DIP or 0.6-inch DIP, shrink DIP (.071-inch pin spacing), PLCC (plastic-leaded chip carrier), PQFP (plastic quad flat pack), EIAJQFP, SOIC (small outline integrated circuit), some involve surface mount technology Power supply requirements
Any other items important to your system design
An example of the proposed Table is given in Fig…
After constructing this table the designer may face one of the following options:
1. There is one MCU fulfils all the system requirements.
2. You have more than one MCU on the list that fulfils the requirements. In such a case the designer has to consider other metrics as availability, manufacturer support, expandability and value. For instance, consider:
What expansions in the system requirements can you predict that will be needed in possible future iterations of this product?
And lastly, consider value, for if two MCUs cost the same but one offers a few more features which are not required today but would make future expansion easier for no additional cost, chose that MCU.
3. No single MCU is fulfilling all the system requirements. In this case the designer has two options:
Study your budget and available space on the motherboard to see if they allow the use of additional ICs to support the MCU to fulfil all the requirements.
If after all this, you still did not find the suitable MCU and the supporting units, the designer has no choice and he has to go for another processor technology, i.e., the use of general-purpose processor (use of microprocessor) or to design his own single-purpose processor.
RAM bytes
ROM EEPROM Flash Serial Channel
Timers Bus
Freq.
MHz
A/D
channels
PWM
channels
I/O Operating
voltage
Package
options
Temp.
M68HC
12
8000 3200 4096 128000 3 16 8 8 8 91

5. References

[1] Semiconductor Industry Association. *International Technology Roadmap for Semiconductors: 1999 Edition.* Austin TX: International SEMATECH, 1999.
[2] Debadelaben, J., Madisetti V.K., el, " Incorporating Cost Modeling into Embedded System Design," *IEEE Design and Test of Computers*, July 1997, pp. 24-35.
[3] Gajski, Daniel D., *Principles of Digital Design.* Englewood Cliffs, NJ: Prentice-Hall, 1997.
[4] Gonzales D.R., "Understanding the Key Architectural Features of a Microcontroller" http://www.dedicatedsystem.com
[6] Kress R. et al, "Customized Computers: a generalized survey" , Proc. Workshop on Field-programmable Logic and Applications (FPL' 96), Darmstadt, Germany, 1996.