Software & Hardware for Design of Embedded Systems-Case Study

Shobharani1, Sheevaleela S2, P. Malathi3, Anitha J.4, Shailaja L.K.5, Kavyashree N.6, Thimmaraju K.N.7 and A.J.Prasad8

1 Asst.Professor; 7 Professor&HOD, 8 Professor,
2,3,4,5,6 Lecturer
Department of M.C.A.
Dr. Ambedkar Institute of Technology, Bangalore 560056
shobhananir@rediffmail.com1,shivaleela.jotappa@gmail.com2,bfgcpm@rediff.com3,alochan@rediffmail.com4,m4,shailajalk@gmail.com5,gamil.com6 and pjaajp@gmail.com8

ABSTRACT
A computer system which has been specifically designed to perform different functions in real time is known as an embedded system. The applications include television, GPS, microwaves, thermostats, network routers, game consoles, ATM machines, kiosks, aircraft, satellites, sensors, consumer electronics, smart phones, industrial automation, I-Phone, avionics, medical and IT hardware, and automobiles. The recent development in this field can be seen in telecommunications. Embedded systems deal with the complex GUI, and use touch screen devices. The hardware on which specific purpose software is fixed or stamped is called an embedded system. Embedded systems are used in homes, offices, cars, factories, hospitals, plants and consumer electronics. Embedded systems can be controlled by digital signal processing and micro controllers. Their huge numbers and new complexity call for a new design approach. The distinct fields of software and hardware design are combined in a unified approach design. The use of single-purpose processors ("hardware") and general-purpose processors ("software"), and the variety of setups used in embedded systems, memories and buses are unified. The most common example of hardware and software co-design is embedded systems. The co-design is an area of system specification, hardware software portioning, architectural design, and the iteration in between the software and hardware. This task can be completed by the hardware and software integration. The integrated circuits are finding their place in the new designs in the form of entrenched cores in a mixed fashion. Hardware and software co design makes it possible to make handy devices which can be carried by individuals. The objective of the co-design is to combine CPU memory and programs to control physical operations. The requirements of advanced computation models, control systems, chip technologies, and modern design tools are relevant and can be configured for personalized uses. An MP3 player can be considered for describing an embedded system and it has a large memory which is capable of storing many songs. Songs are stored in digital compressed form. The CPU scampers the program in main memory. The audio is generated in the form of digital signals. This information is then displayed on the screen with the help of software working in the memory. The hardware and software co-design method is used for the implementation of the MP3 audio decoder which is helpful in real time specification of MP3 player. The synthesis technology has advanced to the point that synthesis tools have become commonplace in the design of digital hardware. The integrated circuit (IC) capacities have increased to the point that both software processors and custom hardware processors now commonly coexist on a single IC. Designing an embedded application has become equivalent to port a generic SW on a, possibly heterogeneous, embedded platform. The development of complex applications involving signal, image, and control processing is related to emerging submicron technologies for integrated circuit manufacturing.

KEYWORDS

1.INTRODUCTION
The design of a system-on-chip implementing the vision system of a mobile robot requires embedded systems. In order to deal with all of the dynamic aspects of these applications, it appears necessary to embed a dedicated real-time operating system on the chip. Embedded systems designers have to face the conversion of the generic sequential C/C++ reference software into a form that begins to include architectural features of the selected embedded platform. Embedded systems can be defined as a collection of programmable parts surrounded by ASICs and other standard components that interact continuously with an environment through sensors and actuators. The programmable parts include micro-controllers and Digital Signal Processors (DSPs). Designers generally implement embedded controllers for reactive real-time applications as mixed software-hardware systems. In the formal
methodology for specifying, modeling, automatically synthesizing, and verifying such systems, design takes place within a unified framework for hardware and software implementation. Widespread use of embedded systems is occurring due to the increase in complexity of digital devices and systems and hardware/software co-design techniques. These systems are currently being implemented by software and hardware components in order to benefit from the strengths of each technology. The embedded system development process involves the integration of hardware and software. Hardware/software interfaces include Device Driver, Driver API, and Device Controller.

2. CASE STUDY
A Control Unit for digital audio system is a consumer electronic appliance. The songs are recorded and they are stored in a digital memory. The user can quickly play back any song at any time. It should be small, light, easy to use, and battery operated.

Use Case diagram

Use cases describe the functionality of the system from the user’s point of view. The user selects a song and presses the ‘record’ button. The user selects a song and then presses the ‘play’ button. The user selects a song and then presses the ‘delete’ button. The interaction between the user, the battery and the time with the system is important. What happens when the battery goes low while the system is playing a song? The solution for this is to warn the system when the battery energy is low. Then the system should switch off all the peripherals and enter the stand-by mode. When the user charges the battery, the system will leave the stand-by mode. While the system is in stand-by mode, the song is still kept in the memory. One should analyze the requirements and present a class diagram as a general solution for the problem.

Class Diagram:
1. Song Player & Recorder: A “digital tape”.
2. Buttons, Screen: The user can press the buttons. The screen shows a menu and indications.
3. Battery level sensor: Measures the remaining battery power level.
4. Screen controller: Generates the image on the LCD screen.

5. Collection of recorded songs: The contents of the digital tape.
6. Play, Record, Stop, Yes, No, Up, Down, Left, Right Buttons: The user interacts with the system by pressing the buttons.

The user interface:
The interaction with the user is menu-driven and the display shows the menu.

The Audio Class Diagram:
Each song is composed of several audio blocks and each audio block if composed of many sound samples. The Audio subsystem always records or plays a complete audio block. The Audio Input and Audio Output classes have real-time requirements. The Timer class provides accurate timing for the Audio Input and Output classes. The timer class is a wrapper for a hardware timer. The Microphone class is a hardware wrapper for the physical microphone. A Microphone class can record one sound sample. The Speaker class is able to play back sound sample through the hardware speaker. There is a need for three different classes to play a song. Playing or recording a song is a complex task that requires a precise timing and interaction with the hardware. By splitting the representation of the song, the flexibility of the design can be provided.

The Song Memory Class Diagram
The song memory class manages the storage space of the sound recorder, it keeps a directory of recorded songs and it allocates space for new songs.

The User Interface uses the Song Memory for obtaining the list of recorded songs, but it does not modify it. The Audio Controller is the only class that uses the modifiers of the Song Memory. If the User Interface wants to delete a song, instead of accessing directly to the Song Memory object, it uses the delete Song method of the Audio Controller. This mechanism prevents the User Interface from deleting a song while the Audio Controller is playing or recording it.

The User Interface Class Diagram
The User Interface class manages the interaction with the user. It receives the input from the user trough a keyboard and gives feedback to him or her through a display.

The Display Class Diagram
The Display Class is the interface to the hardware display. It can be switched on or off. The Graphic Context abstraction is used to draw on a Display. It provides some basic drawing primitives. Each Graphic Context represents a rectangular area of a Display. The Graphic Context manages the geometry transformation from its local coordinate system to the global one.

Audio Controller object is the referee for the sound channels. A sound channel can be used to record a song, or to play a song.

Audio Output object controls an output sound channel.
Microphone object is a wrapper for a real microphone.
Speaker object is a wrapper for the real speaker.

Menu User Mode
The menu user mode is the main user mode for the user interface. The buttons allow the user to navigate through the menus and invoke the desired menu option.

The concurrency model is used to specify the different execution threads of the software and the communication mechanism between them. The audio system thread is activated whenever the processor acknowledges an interrupt request. The system thread has priority and it can pre-empt the user thread.

A hardware wrapper is a software object representing a hardware device. It is an interface between the application objects and the physical devices. The constructor method of a hardware wrapper initializes the hardware device. After the creation of the wrapper, the device is ready to be used. The methods of the wrapper configure the device, but also can start or stop some activity.

The Song Memory object is the responsible of allocating and freeing the memory space used to store the songs.

The final software product for an embedded system is not program image but a non-volatile memory containing the program.

3. DESIGN ASPECTS

The factors that have to be considered are design cycle time to reach shorter time-to-market, and to reduce development and production costs, the strict performance constraints such as application execution time, integrated circuit area, overall system power dissipation. An alternative approach to a traditional design flow, called algorithm-architecture matching, aims to leverage the design flow by a simultaneous study of both algorithmic and architectural issues. Introducing such design methodology is also necessary when facing the new emerging applications such as high performance, low-power, low-cost mobile communication systems and/or smart sensors-based systems. The acceleration of the reconstruction algorithm is of great interest for various applications such as synthetic aperture radar (SAR), contactless control, and industrial X-ray applications. Due to the large amount of the acquired data and the complexity of the algorithms, reconstruction is a very time-consuming process. From a computing point of view, reconstruction methods can be classified into two main techniques: analytic (direct) reconstruction and iterative reconstruction. They both include a back-projection (BP) step that accounts for 50% to 70% of the processing time. Although CPUs have gained sufficient computing power for 2D reconstruction, with 3D reconstruction, the increase of the amount of data for high-quality images leads to higher computing times. Iterative reconstruction algorithms may reach several hours of processing. The algorithmic optimizations of reconstruction have reached some limits and it is becoming mandatory to reduce the computing time through architecture solutions. General purpose parallel computers benefit from recent competing technologies: the system on programmable chip (SoPC) and the general purpose graphical processing unit (GP-GPU). Digital cameras are being used for multimedia applications. Improvements in the growing digital imaging continue to be made with two main image sensor technologies: charge-coupled devices (CCDs) and CMOS sensors. Historically, CCDs have been the dominant image sensor technology. However, the continuous advances in CMOS technology for processors and DRAMs have made CMOS sensor arrays a viable alternative to the popular CCD sensors. This led to the adoption of CMOS image sensors in several high-volume products, such as webcams, mobile phones, PDAs, for example. Furthermore, new recent technologies provide the ability to integrate complete CMOS imaging systems at focal plane, with analog-to-digital conversion, memory and processing. The main advantage of CMOS image sensors is the flexibility to integrate processing down to the pixel level. As CMOS image sensors technologies scale to 0.18 µm-processes and under, processing units can be realized at chip level (system-on-chip approach). Smart cameras are more and more applied for their specific performances and their processing capabilities in different application fields. Signal and image processing applications require a lot of computing resources. Despite the increasing speed of PC processors and bus frequencies, the implementation of embedded coprocessor architectures expressly conceived for image sensors and inserted in the acquisition loop presents several advantages. Very high processing speed and reduced image data bandwidth are achievable. The architecture also provides high degree of flexibility in the preprocessing stage for the different acquisition modes specific of CMOS imaging.

CONCLUSION

A large portion of the embedded system development process involves the integration of hardware and software. The communication across the hardware/software boundary is tedious and error-prone to create. The requirements for an embedded system are to be identified, enumerated, analyzed, designed and implemented. Embedded software designers can easily use hardware just like system libraries. One can use UML as a graphical notation for design. One can use required software patterns for real-time embedded systems. UML helps the designer to express the ideas, and also helps the programmer to understand them. There is not a systematic approach for converting UML diagrams into a complete implementation. Every embedded system is a compact composition of hardware and software. A large portion of the embedded system development process involves the integration of hardware and software. Embedded software programmers do not need to know the low level characteristics of hardware such as memory mapped I/O or kernel functions. One of the tedious aspects of hardware synthesis is assigning every hardware input/output port a specific hardware pin number.

FUTURE SCOPE

Embedded controllers for real-time applications requiring split second control for fusion energy development and space exploration may have to be addressed.
REFERENCES


[2] Readings in hardware/software co-design, Giovanni De Micheli, Rolf Ernst, Wayne Hendrix Wolf - 2002 - 697 pages
