# Design of Reconfigurable Multiprocessor Architecture for Embedded System

Archana Gomkar
Ph.D. Scholar,
Singhania University,
Pacheri Bari

## ABSTRACT
Embedded systems are the brains of today's most digital and industrial control systems. In systems where more than one processor is incorporated, the need for multiprocessor communication often arises. It fully utilizes microcontroller features & embedded technology concepts to minimize the complications of digital gates, size and cost too.

## Keywords
Multiprocessor Communication, Protocols, SPI, CAN, Interfacing Peripherals, Embedded System, Distributed Computing.

## 1. INTRODUCTION
### 1.1 Multiprocessing
Multiprocessing, as generally defined, is the use of two or more central processing units (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. There are many variations on this basic theme, and the definition of multiprocessing can vary with context, mostly as a function of how CPUs are defined. In present project Multi processing refers to use of multiple peripheral devices.

### 1.2 Communication Protocol
A communications protocol is the set of standard rules for data representation, signalling, authentication and error detection required to send information over a communications channel. An example of a simple communications protocol adapted to voice communication is the case of a radio dispatcher talking to mobile stations. The communication protocols for digital computer network communication have many features intended to ensure reliable interchange of data over an imperfect communication channel. Communication protocol is basically following certain rules so that the system works properly.

**Problem statement:**
Microcontrollers are versatile integrated circuits typically incorporating a microprocessor, memory, and I/O ports on a single chip. These self-contained units are central to embedded systems design where low-cost, dedicated processors are often preferred over general-purpose processors. Some embedded designs may incorporate several microcontrollers, each with a specific task. Efficient communication is central to such systems.

Exchanging data in a system comprising multiple microcontrollers. Since each microcontroller contains a separate memory, a message passing system is designed. The hardware consists of low-cost microcontrollers serving as computing nodes and a network controller to direct messages among them.

## 2. BACKGROUND OVERVIEW
We are aware of the fact that 40 pin microcontroller provide only 4 ports for external world. But for our complex requirement we need to interface a large number of peripheral devices which is not possible with a single microcontroller. The limitation of using single Controller is

- Limited code memory.
- A very few peripheral devices can be interfaced.
- Complexity in design.
- Time consuming.
- Error probability.
- Software complexity more.
- Debugging is difficult.

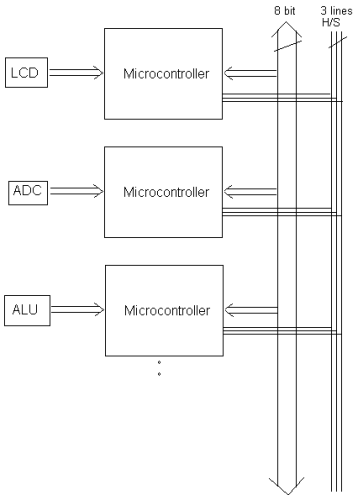
### 2.1 Existing System
The solution of above defined Problem is to use more than one controllers or multi controllers in a single project. Now communication in different controllers can be done mainly by two ways i.e. by using I2C protocol and SPI protocol.

### 2.2 Drawbacks of Existing System
- Synchronization of the clock is required.
- Data is transferred serially hence the system becomes slow.
- Master slave configuration is strictly maintained.
- Slaves cannot transfer data directly among each other
- As data transfer has to take place through master itself.

### 2.3 Proposed System
- We had used 8-bit parallel data lines to transfer the data and 3 bit parallel line for handshaking signal.
- There will be no master Slave configuration.
- Each controller can communicate or transfer data independently.
- Priorities can be set by the controllers IDs.
- We had developed that algorithm by which Up to 255 microcontrollers can transfer their data to desire destination microcontroller independently.
- Data transfer here means that any microcontroller can get data from any controller but with its permission. Similarly any microcontroller can send data from itself to any destination controller.

- The clock frequencies of these controllers can may or may not be same but the protocol will work efficiently.
- The same program will be burn in all the controllers without any modification.
- Addition and removal of any controller will not affect the protocol.

# 3. THE PROPOSED SYSTEM




Microcontrollers are versatile integrated circuits typically incorporating a microprocessor, memory, and I/O ports on a single chip. These self-contained units are central to embedded systems design where low-cost, dedicated processors are often preferred over general-purpose processors. Some embedded designs may incorporate several microcontrollers, each with a specific task. Efficient communication is central to such systems.

Now days, embedded systems are everywhere. Today they have crossed the boundary of only industrial use. They are becoming more and more complex to simplify our lifestyle. Many features are being embedded in a single tiny system to fulfill our everyday requirements. Thus more & more peripheral needs to be integrated in the single embedded system. This increases the design time of the system and also the processing load is increased for a single microprocessor.

This arises the need of such an arrangement of peripherals which can be interfaced in a network instead of directly interfacing with the processing unit. This type of arrangement of peripherals will create a distributed interface of multiple peripherals. Distributed systems require protocols for communication between microcontrollers. Currently **Serial Peripheral Interfaces (SPI)** and **Inter-Integrated Circuit (I2C)** are two of the most commonly used such protocols. Both of them work as serial communication link between the microcontrollers.

Here, in this thesis, a new method for communication is proposed for an embedded system having multiple peripherals on the board. The adapted arrangement is parallel and hence is more faster way to communicate.

The focus of this thesis is the design of a communication layer and application programming interface for exchanging data in a system comprising multiple microcontrollers. Since each microcontroller contains a separate memory, a message passing system is designed. The hardware consists of low-cost microcontrollers serving as computing nodes and a network controller to direct messages among them.

## 3.1 Embedded System

An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).

An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today.

Modern embedded systems are often based on microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces) but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also still common, especially in more complex systems.

The key characteristic, however, is being dedicated to handle a particular task. Since the embedded system is dedicated to specific tasks.

A general-purpose definition of embedded systems is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. "Embedded" reflects the fact that they are an integral part of the system.

# 4. NEED FOR PROPOSED SYSTEM

Microcontrollers are versatile integrated circuits typically incorporating a microprocessor, memory, and I/O ports on a single chip. These self-contained units are central to embedded systems design where low-cost, dedicated processors are often preferred over general-purpose processors. Some embedded designs may incorporate several microcontrollers, each with a specific task. Efficient communication is central to such systems.

The focus of this thesis is the design of a communication layer and application programming interface for exchanging data in a system comprising multiple microcontrollers. Since each microcontroller contains a separate memory, a message passing system is designed. The hardware consists of low-cost microcontrollers serving as computing nodes and a network controller to direct messages among them.

Considering a case where the embedded system requirements are as follows…

- The system will be used for SCADA purpose
- It must have multiple input for various sensors
- It must display all the sensor information on LCD
- It must sent the sensor data to remote PC
- It must have a keypad for some user input
- It must have relays for activating-deactivating actuators
- Also it must have provision for adding more features to it without disturbing the original design like
- It can be upgraded for storage device to store the record
- It can be upgraded for DAC – to be used for analog output

- It can be upgraded for DC as well as stepper motor controlling also.
- Etc.

All the above are a general overview that an embedded system may have. Our aim is to design such a flexible, reconfigurable, upgradable, distributed processing system; that can adapt any changes whenever required

# 5. EVALUATION OF COMMUNICATION OPTIONS

The Atmel microcontrollers come with a small selection of built in communication options. In order to select the best interconnect, each of these options was evaluated against implementing an alternative communication protocol. The criteria for evaluation were scalability, number of pins required, and throughput.

The first option available on the Atmel chips is the serial peripheral interface (SPI). This interface requires at least four conductors, provides full duplex operation and is synchronous. SPI defines one node as a master and the other nodes as slaves. Two wires provide the send and receive lines. The master is responsible for generating the clock signal on a dedicated line and sending a signal to a slave on its select line, thus initiating communication. The clock frequency is limited to the main oscillator frequency divided by 4, or 16 MHz / 4 = 4 MHz. One bit can be sampled per SPI clock period, thus allowing a maximum throughput of 4 Mbits/sec / (8 bits/byte) = 500 KB/sec. For connecting two microcontrollers, SPI seems to be a good choice. It provides high throughput and uses few port pins. Since this is a byte-oriented protocol, though, there would need to be at least one byte of overhead for logical addressing. This would effectively halve the transfer rate. Also, the scheme does not scale well. Since there can be only one master, microcontrollers must take turns initiating communication. This can be implemented using a token passing scheme, but this adds an extra line to inform the next processor it can be the master. Also, as the number of microcontrollers grows, the number of slave select lines increases per chip. An arbiter might be an alternative, but the bandwidth is still divided among all microcontrollers, making it a poor choice for large networks.

The next option is the universal synchronous and asynchronous serial receiver and transmitter (USART). This is a full duplex device with allows both asynchronous and synchronous communication. Data can be sized from 5 to 9 bits in length. The transmission rate can be adjusted from 2400 bits/sec up to 2 Mbits/sec. Since frames require a start and stop bit, the maximum throughput is 200 KB/sec. Additionally, in multiprocessor communication mode, an address frame must be sent at the start of a transmission. As was the case with SPI, this requirement halves the transmission rate. If variable messages were allowed at the link layer, this would not have as much of a negative impact, however. Still, the scalability problems that preclude the use of SPI apply to the USART a well.

The final hardware option provided by the microcontrollers is the two-wire serial interface (TWI), which is compatible with the industry standard I2C interface. TWI provides built in addressing and arbitration for systems with multiple masters. The clock signal is generated by the master initiating communication and can reach frequencies up to 400 kHz. Address packets are part of the TWI protocol, as are variable length messages. However, in order to support arbitration, all masters must use packets of the same length. In a byte-oriented network, the overhead of 9-bit address and data packets effectively gives a maximum throughput of 400 kbits/sec / (18 bits/byte) = 22.2 KB/sec. Due to the arbitration built into the protocol, a separate arbiter is unnecessary. However, with many masters operating on the same bus, contention would become prohibitive and the throughput would be reduced considerably.

Since the above methods all have significant drawbacks, alternative network designs were implemented and evaluate in all three shared-bus interfaces discussed above, serial transmission reduces the throughput considerably. Thus, the first attempt at new network architecture uses a parallel bus to address this problem. Also, since cooperative arbitration can be slow and may indefinitely prevent a processor from gaining the medium, a hardware arbitration unit with a fairness guarantee is used when transmission privileges over a link are contested.

The first version of the network architecture utilizes a 4-wire shared bus to transfer a nibble (4-bit chunk) of data at a time. Four additional control wires are used per node to communicate transmission requests and acknowledgements between each node and the bus controller.

For the software side of the project:

- The interface must allow variable length messages to be transmitted between microcontrollers.
- The system must provide a logical addressing scheme.
- The interface should be easy to utilize in applications.
- The implementation should fit into the limited amount of on-chip memory and allow enough free memory to implement client applications.
- The implementation must provide a good balance between I/O and computation.

Additionally, the number of wires needed for all communication links is constrained by the number of pins on a single microcontroller. Ideally, port pin usage should be as low as possible (preferably, 1 I/O port) to allow the microcontrollers to have enough I/O resources to interface with external system components. These constraints will in part determine the network topology that will be implemented using the microcontrollers.

The physical layer supports the electrical or mechanical interface to the physical medium.

The physical layer is aimed at consolidating the hardware requirements of a network to enable the successful transmission of data. Network engineers can define different bit-transmission mechanisms for the physical layer level, including the shapes and types of connectors, cables, and frequencies for each physical medium.

The physical layer sometimes plays an important role in the effective sharing of available communication resources, and helps avoid contention among multiple users. It also handles the transmission rate to improve the flow of data between a sender and receiver.

First, distributed-memory machines exhibit high performance when a task is divided so that each processor can fit its working set into the local subset of main memory. Shared-memory machines do not suffer from this since all main

memory accesses are equal. However, since distributed memory machines do not need to share the bus when working with local memory, the potential performance improvement over SMP systems is proportional to the number of separate memories. Second, without a shared memory address space, nodes in a cluster must use interconnect to explicitly send and receive messages instead of reading and writing a shared variable. This message-passing latency can be relatively long compared to the communication latency in tightly integrated systems. Thus it is important to optimize both hardware communication efficiency and software communication overhead.

The physical layer supports the electrical or mechanical interface to the physical medium.

The physical layer is aimed at consolidating the hardware requirements of a network to enable the successful transmission of data. Network engineers can define different bit-transmission mechanisms for the physical layer level, including the shapes and types of connectors, cables, and frequencies for each physical medium.

The physical layer sometimes plays an important role in the effective sharing of available communication resources, and helps avoid contention among multiple users. It also handles the transmission rate to improve the flow of data between a sender and receiver.

The physical layer provides the following services:

- Modulates the process of converting a signal from one form to another so that it can be physically transmitted over a communication channel
- Bit-by-bit delivery
- Line coding, which allows data to be sent by hardware devices that are optimized for digital communications that may have discreet timing on the transmission link
- Bit synchronization for synchronous serial communications
- Start-Stop signalling and flow control in asynchronous serial communication
- Circuit switching and multiplexing hardware control of multiplexed digital signals
- Carrier sensing and collision detection, whereby the physical layer detects carrier availability and avoids the congestion problems caused by undeliverable packets
- Signal equalization to ensure reliable connections and facilitate multiplexing
- Forward error correction/channel coding such as error correction code
- Bit interleaving to improve error correction
- Auto-negotiation
- Transmission mode control

Examples of protocols that use physical layers include:

- Digital Subscriber Line
- Integrated Services Digital Network
- Infrared Data Association
- Universal Serial Bus
- Bluetooth
- Controller Area Network
- Ethernet

# 6. CONCLUSIONS

- We had used 8-bit parallel data lines to transfer the data and 3 bit parallel line for handshaking signal.
- There will be no master Slave configuration.
- Each controller can communicate or transfer data independently.
- Priorities can be set by the controllers IDs. . .
- We had developed that algorithm by which Up to 255 microcontrollers can transfer their data to desire destination microcontroller independently.
- Data transfer here means that any microcontroller can get data from any controller but with its permission. Similarly any microcontroller can send data from itself to any destination controller.
- The clock frequencies of these controllers can may or may not be same but the protocol will work efficiently.
- The same program will be burn in all the controllers without any modification.
- Addition and removal of any controller will not affect the protocol.

# 7. REFERENCES

[1] Atmel ATmega32 and ATmega16 Datasheets. Rev. 2503H-03/05. Accessed March 2006. http://www.atmel.com/products/avr/

[2] aOS – a RTOS for AVR. Accessed March 2006. http://www.tietomyrsky.fi/projektit/aos/

[3] The Message Passing Interface (MPI) Standard. Accessed March 2006.http://www-unix.mcs.anl.gov/mpi

[4] MPI-2: Extensions to the Message-Passing Interface. Accessed March 2006.<http://www.mpi-forum.org>

[5] Hennessy, John L. and Patterson, David A. Computer Architecture: A Quantitative Approach, 3rd ed. Morgan Kaufmann 2003.

[6] Peterson, Larry L. and Davie, Bruce S. Computer Networks: A Systems Approach, 3rd ed. Morgan Kaufmann 2003.

[7] A preemptive multitasking, OS for Atmel Mega32 microcontrollers. Accessed March 2006. http://instruct1.cit.cornell.edu/courses/ee476/RTOS/index.html

[8] Benny Akesson. Predictable and Composable System-on-Chip Memory Controllers. PhD thesis, Eindhoven University of Technology, February 2010. ISBN: 978-90-386-2169-2.

[9] Benny Akesson, Kees Goossens, and Markus Ringhofer. Predator: A Predictable SDRAM Memory Controller. In

Int'l Conference on Hardware/Software Codesign and System Syn-thesis (CODES+ISSS), pages 251{256. ACM Press New York, NY, USA, September 2007.

[10] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: video coding at low bit rates.Circuits and Systems for Video Technology, IEEE Transactions on, 8(7):849 {866, nov 1998.

[11] Bernd Girod. Digital images and human vision. chapter What's wrong with mean-squared error?, pages 207{220. MIT Press, Cambridge, MA, USA, 1993.

[12] K. Goossens, J. Dielissen, and A. Radulescu. Aethereal network on chip: concepts, archi-tectures, and implementations. Design Test of Computers, IEEE, 22(5):414-421, sept.-oct.2005.

[13] K. Goossens and A. Hansson. The ethereal network on chip after ten years: Goals, evolution, lessons, and future. In Design Automation Conference (DAC), 2010 47th ACM/IEEE, pages306 -311, june 2010.