ORGANISATION OF COMPUTER SYSTEMS

Z. G. Vranesic

Rapporteur : Mr. E. P. Farrell

Abstract

The main purpose of the talk given by Professor Vranesic is to consider briefly the question of how much exposure to hardware details should a computer science (or electrical engineering) student have, assuming that digital hardware is not a major part of his program.

Introduction

In the area of digital system design the relevant topics are usually taught within the framework of a single hardware oriented course, often called "Computer Organisation" or some related title. Let us assume that the course in question involves 2 to 3 lecture hours per week (with or without a laboratory) during one semester of approximately 15 weeks.

In a course of this kind, as in many others, one often finds a tendency to place far too much emphasis on topics that offer well defined and accepted concepts, that tend to be easy to teach and where a concensus exists whether or not a particular technique is better than others. This includes such favourite topics as arithmetic, logic design and switching theory. On the other side, too little time is spent on subject matter where many reasonable choices exist, for example, interfacing, bus structures, communications schemes, etc. Yet when a graduate student enters the commercial world of industry, most of his efforts in digital hardware systems are likely to involve just such "poorly defined" areas.

The rest of this discussion will concentrate on what I consider to be the minimum acceptable level of detail that may be appropriate in one topic, namely bus organisation, interfacing and standards.

1. Bus Structures

Figure 1 shows three commonly found bus structures. The single bus is conceptually simple, flexible for attaching peripheral devices, and usually found in small machines. It restricts the operating speed of the system, since it can be used for only one transfer at a time. Moreover, all the bus lines are brought to all devices, whether or not a particular device requires the potentially complex control mechanism imposed by the bus.

The simplest form of a two-bus structure is shown in Figure 1b. The CPU interacts with the memory via the memory bus. Input and output transfers are handled by the CPU, known as programmed I/O. A different version of a two-bus structure is given in part c of the figure. Here, the I/O transfers are made directly from the memory, usually involving a separate I/O processor.

1.1.1. Dedicated Bus

A bus which has a fixed assignment to a given function, or only two devices connected to it, is referred to as a dedicated bus. Such a bus involves a simple bus controller, simple addressing and simple synchronisation of devices. It allows high throughput, since there is little contention for its use. However, it also tends to be expensive and it may make the expansion of the system (in modular fashion) an awkward task.

1.1.2. Nondedicated Bus

This is a bus shared by a number of devices, used to perform a variety of transfers. A good example is the single bus of Figure 1a. Nondedicated buses are characterised by lower cost, flexibility in attaching new devices, and suitability for extension to reliable systems where duplication is needed. The offsetting disadvantages are lower throughput and the need for a relatively complex bus controller.

1.2. Bus Control

Control of a bus can be either centralised in a single hardware controller unit, or distributed over the devices connected to the bus. In either case there are essentially three accepted ways of organising the control mechanism, namely

daisy chain
independent request, and
polling

Let us consider some typical features of each of these schemes, restricting the discussion to centralised control.

1.2.1. Daisy Chain

Figure 2 illustrates the daisy chain arrangement. Devices request the use of the bus via the common BR line. The bus controller accommodates the request when the bus is available by issuing a BG signal which ripples through the devices. When the BG signal is received by the device that requested the bus it blocks further propogation of the BG signal, seizes the bus and indicates this busy status via the BB line.

Daisy chain is a simple structure. Few control lines are needed, and they are not dependent on the number of devices connected to the bus. However, there are some disadvantages. The priority structure is fixed, with the device nearest the controller having the highest priority. This raises the possibility of a remote device being locked out. Rippling of the BG signal through

the chain slows down the operation. Also the structure susceptible to failure, since a failure in a single device can affect other devices.

1.2.2. Independent Requests

This scheme is shown in Figure 3. Each device has its own pair of BR and BG lines. The most significant benefits of this structure are fast operation, flexible priority structure (determined by the controller), and ease of isolating malfunctioning devices. The offsetting factors are higher cost and the need for a more complex controller.

A compromise control mechanism is often attractive, which uses daisy chain groups where the groups are organised in the independent requests arrangement, as indicated in Figure 4.

1.2.3. Polling

The polling structure is given in Figure 5. A poll count is used to identify the device that is requesting the use of the bus. The poll count connection can be either a set of lines allowing parallel transmission of the count, or a single line requiring a more complex bit-serial transmission method. The priority is determined by the method of counting:

- if a count is started from zero each time a BR signal is received, then fixed priority results,
- if a count is continued from where it stopped during the previous request, then round-robin polling occurs.

It should be noted that the BR line can be eliminated if the polling counter runs continuously when the bus is not busy. The counter is stopped by any device using the bus.

Polling is a simple and relatively inexpensive structure. It is more reliable and flexible than the daisy chain. But, it is also slow.

It is more important to recognise that the techniques for implementing bus control are in general very similar, and often identical, to the techniques used to deal with interrupt requests.

2. Bus Communications

Having chosen a suitable bus structure, it is necessary to define the protocol for implementing data transfers on the bus. A variety of schemes for timing the transfers are possible, and they can be broadly classified as either synchronous or asynchronous.

In the case of a synchronous bus, all devices derive the timing information from a common clock line. Fixed time slots of equal width are generated (or synchronised) by a central circuit. The clock pulses must be of greater duration than the maximum propagation delay on the bus. Thus, the speed of operation is governed by the longest delay and the slowest device. However, it should be noted that a very slow device could be assigned more than one time slot. Also, the clock rate can be selected to match the

fastest device in the system, but then buffers must be included to accommodate slower devices.

Another difficulty with the synchronous bus is that the acknowledgement that the destination device received the correct data is not easily available. Verification by a reply at the end of each transmitted data unit (byte, word) is wasteful of bandwidth, as the time slots are defined by the slowest device. A dangerous alternative is verification by default, whereby if an error is detected by the destination it will return a signal to the source a few time slots later.

The most positive features of a synchronous bus are that it requires simple interfaces, and it is usually easy to implement.

The asynchronous bus provides an attractive and frequently used alternative. Here the signalling is based on a "handshake" protocol, between the sending and receiving devices. Instead of the common clock, the timing is controlled by signals on "Ready" and "Accept" lines. The Ready signal verifies the validity of the data on the bus, while the Accept signal is generated as an acknowledgement for the received information. An example of the signalling needed for an input transfer on an asynchronous bus is shown in Figure 6. The time t -t allows for the skew on the bus, as well as the time needed by the device interface to decode the address and mode information. The time t -t allows for the propagation delay of the Ready signal, and for any extra delays introduced by the interface in placing the data on the bus. At t the Accept signal has arrived and the data lines can be strobed. The time t -t also allows for bus skew. Erroneous addressing may take place if an address starts to change while Ready is still true (that is, equal to one).

The asynchronous bus has some highly beneficial features. The speed of operation is not governed by the slow devices. Verification of data transfers is naturally provided. A high degree of flexibility and reliability may be achieved. The most significant drawback is the complex interfaces that are required.

3. Interfaces

The choice of the interfacing circuits depends upon the particular application. The interface must be able to perform the following functions:

- provide a storage buffer for one word (or byte) of data,
- contain status flags that can be accessed by the computer to determine whether the buffer is full or empty, whether or not the device needs servicing, etc.,
- decode addresses to determine when it is being addressed,
- generate the appropriate timing signals, as required by the defined bus protocol, and
- perform any format conversion that may be necessary to transfer data between the bus and the I/O device (for example, parallel-serial).

Most frequently an interface is depicted as shown in Figure 7. This simple block diagram and the above discussed functional

characteristics are often deemed to provide sufficient exposure for a student. The assumption made is that interfacing is a reasonably straightforward, and usually tedious, task that therefore has little scope for classroom discussion.

A more appropriate level of detail that a student should appreciate is given in an example of a parallel I/O interface shown in Figure 8. This 16-bit interface contains data buffers for input and output (DIN and DOUT), and status registers corresponding to each buffer (SIN and SOUT). The buffers and the status registers are assigned four consecutive word addresses. Inputs to the DIN buffer and outputs from the DOUT buffer, as well as the inputs to the status registers, are not shown in the diagram, since this circuitry is internal to the device and not a part of the interface. A connection to an asynchronous bus is assumed.

Another example of a useful interface is shown in Figure 9. This serial-parallel interface is based upon the use of the UART (Universal Asynchronous Receiver Transmitter) chip, which contains the circuitry for the serial-parallel conversion. Awareness of the existence of such components and the ease of their utilisation is certain to prove beneficial to the students. One should note that the interface of Figure 9 requires the same addressing and control circuitry as the parallel interface of Figure 8. Such interfaces are suitable for connection to a variety of devices, for example, teletypewriter (as in Figure 9), modem, etc. Although the student should not be expected to remember circuits such as in Figures 8 and 9 for examinations it is important that he should be capable of looking at such a level instead of believing that a $300 interface consists of simply 3 wires linked together.

4. Standardisation

Standardisation is a touchy issue. It seems that whenever the question of defining "standards" is raised, it starts a heated debate. In my opinion, some standards are badly needed. Existence of adequate standards makes the life of an average system designer considerably easier. It should lead to better defined systems and more readily available components at competitive prices. Unfortunately, the progress of defining standards has been painfully slow. In many cases inadequate specifications have resulted, either from the equipment or functional point of view. However, mistakes of the past need not hinder developments in the future.

Among many attempts at defining standards for interfacing, there are at least two that are gaining some degree of acceptance. One of them is the EIA Standard RS-232C (also known as the CCITT recommendation V24), which specifies a 25-pin interface between data communication devices and data terminal equipment. The second one is the IEEE Standard 488-1975, which specifies a digital interface for programmable instrumentation. It is intended for connection of laboratory instrumentation that can be digitally controlled.

Awareness of standards, their benefits and pitfalls, is highly useful to a system designer. It is a serious mistake to ignore them solely on account of imperfect or incomplete specifications.

5. Conclusions

The preceding discussion is an attempt to suggest a minimum level of exposure to the subject of bussing, interfaces and standardisation that is deemed appropriate for computer science students who are not specialising in hardware. As another issue, it is hoped to impress the teachers of computer organisation that there is a very real need to deal adequately with the topics where design decisions are not black and white, but where many alternatives exist, even if they are not properly defined. To this effect, illustrative examples from the commercial environment often prove to be considerably useful.

It is regrettable that the voluminous literature on digital systems contains relatively few papers that deal with the problems of bus structures and interfacing. Occasionally one finds a paper discussing these topics in an interesting way, as for example the survey paper by Thurber [1].

Textbooks on computer organisation also leave much to be desired. Indeed, this was one of the primary reasons why two of my colleagues and I have attempted to write a book that concentrates more heavily on such topics [2]. The book contains a considerable amount of the type of material that was used in the preceding discussion.

6. Discussion

The discussion section was totally involved by a heated debate about whether standards were useful in the design of digital systems.

Professor Whitfield disputed the speaker's statement that the RS-232 standard was clearly defined. He continued by comparing it to the X25 communications protocol where you are told about the various bit pattern tables but no indication of their functionality is given, the result of this being that somebody has to write a skeleton program which attempts to define the functionality. Professor Vranesic replied that in the case of RS-232C the functionality of the pins was indeed defined, although one could perhaps argue whether or not the functions chosen were correct. Professor Hoare declared that it was not possible for two people to design an interface which would allow any data terminal to be connected to any modem, the result being that you had to connect "everything up to everything". He went on to say that no standards at all were better than standards which do not work in practice and at present we should leave the problem with manufacturers. Professor Vranesic replied that it would be ludicrous to have manufacturers specifications, for example the "Intel bus", being accepted as standards, the inevitable outcome of this attitude would be numerous standards.

Dr. Glaser said that another reason for bad standards is that they are inevitably the lowest common denominator of agreement that can be obtained from all the manufacturers and other interested parties. He said that the X25 standard was an example of such an "electropolitical" compromise which decided upon a standard that could be agreed this year instead of waiting for a properly defined

standard which would not emerge until 1985.

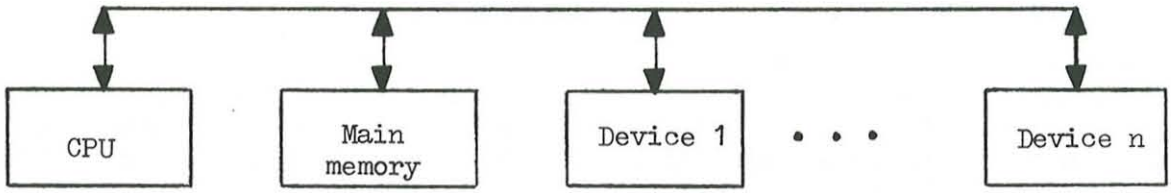
Professor Hoare said it should be our duty as professionals to have sufficient maturity to admit that certain standards are not standards at all and do not serve the purpose they were intended for and it is because we are not prepared to give these answers often enough that we get into such a fearful mess over standards. Professor Randall asked Professor Hoare whether the half-way poor was so entirely the envy of the unattainable best. Professor Hoare replied that in the case of standards the answer was 'yes' and that one should not standardise before one fully understands what is involved.

Professor Michaelson wound up the discussion by supporting Dr. Glaser's statement that Professor Hoare's answer was not true since he was ignoring the fact that the main problems in standardisation were mainly political which resulted in bad standards being defined on the basis of inadequate compromises.

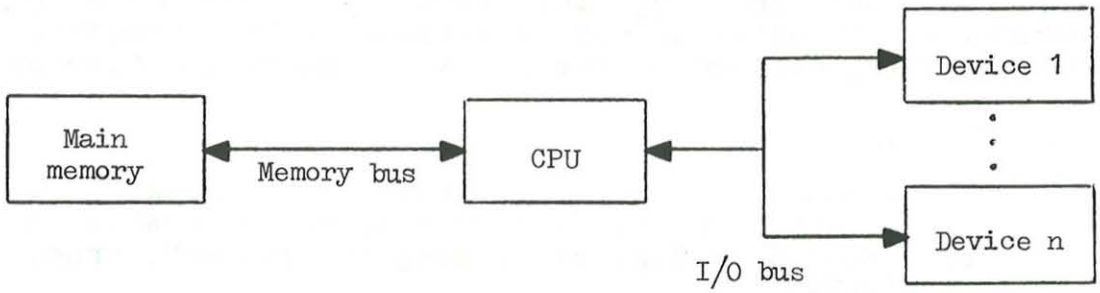
7. References

1. K. J. Thurber, E. D. Jensen, L. A. Jack, L. L. Kinney, P. C. Patton and L. C. Anderson, "A systematic approach to the design of digital bussing structures", Proc. FJCC 1972, pp 719-740.

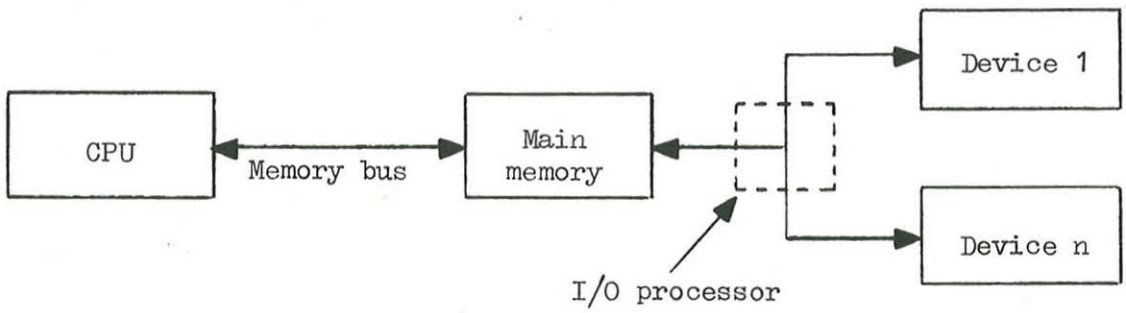
2. V. C. Hamacher, Z. G. Vranesic and S. G. Zaky, "Computer Organisation", McGraw-Hill, 1978.



a) Single bus



b) Two-bus structure - programmed I/O.



c) Two-bus structure - I/O processor.

Figure 1 : Most common bus structures

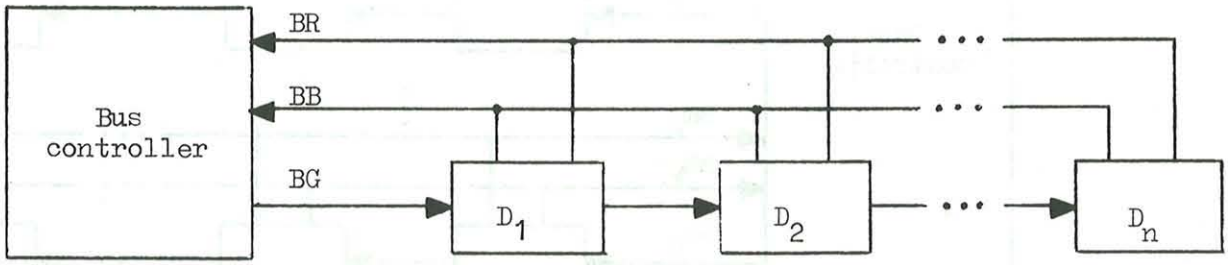


Figure 2 : Daisy chain

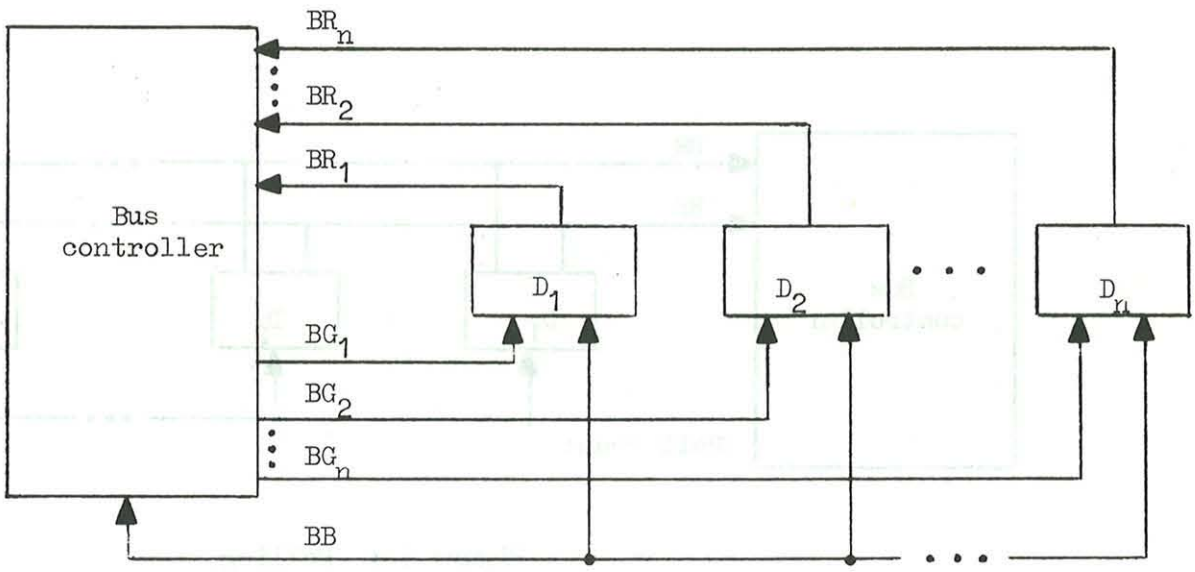


Figure 3 : Independent requests

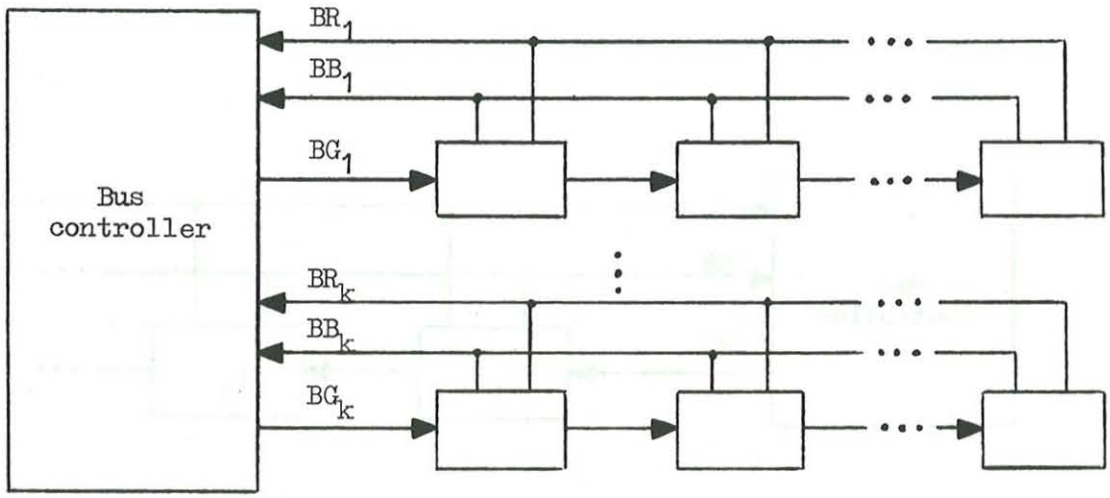


Figure 4 : Daisy chain groups

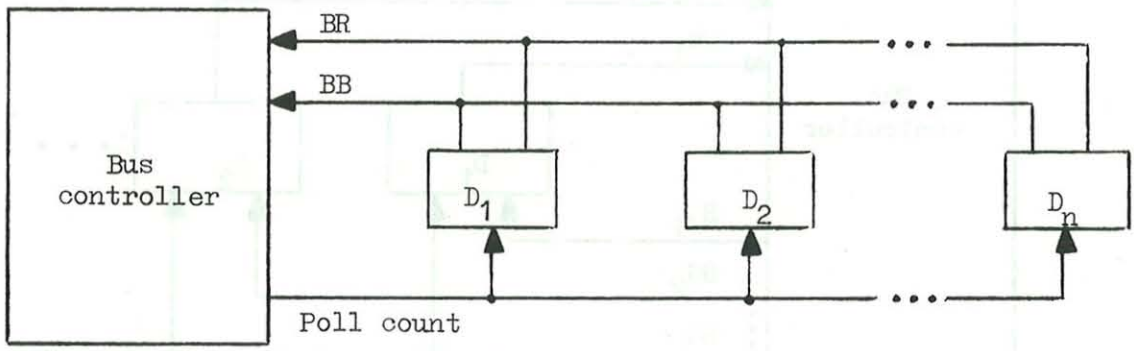


Figure 5 : Polling

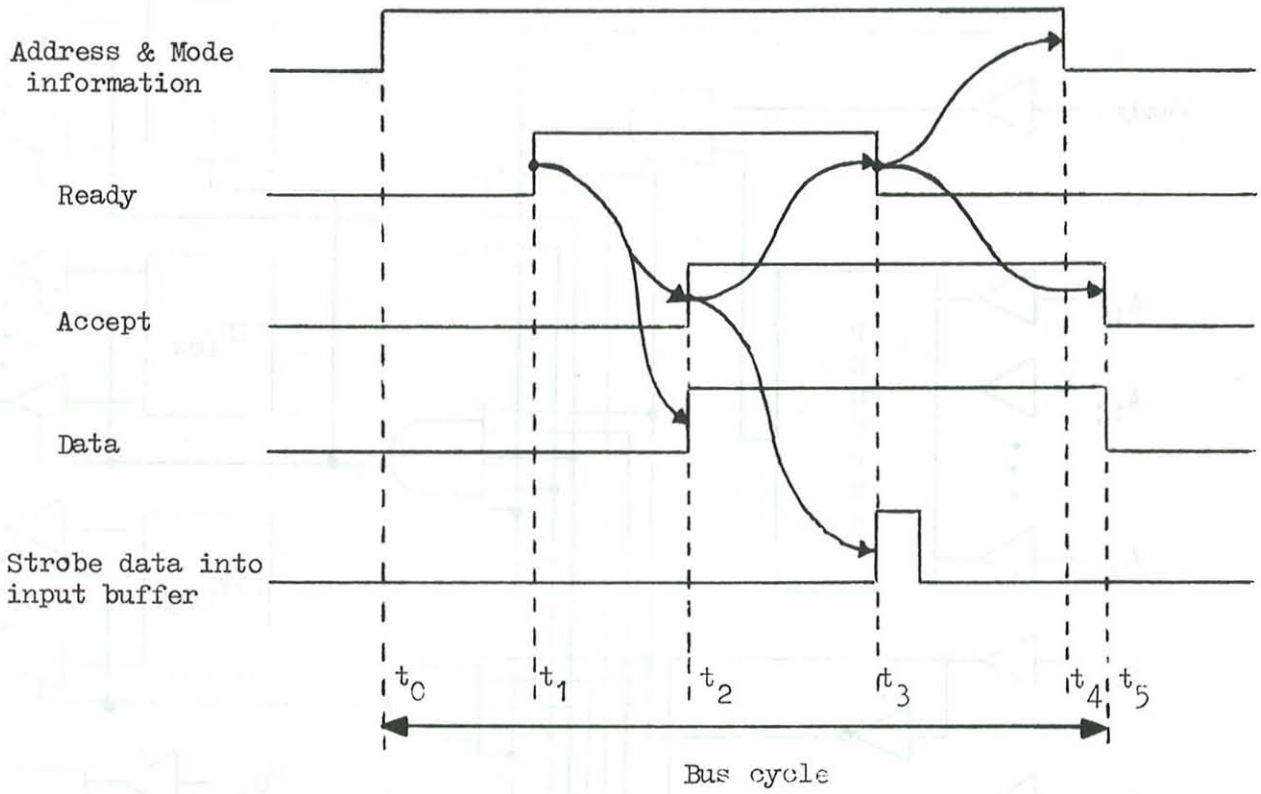


Figure 6 : Signals for input transfer on an asynchronous bus

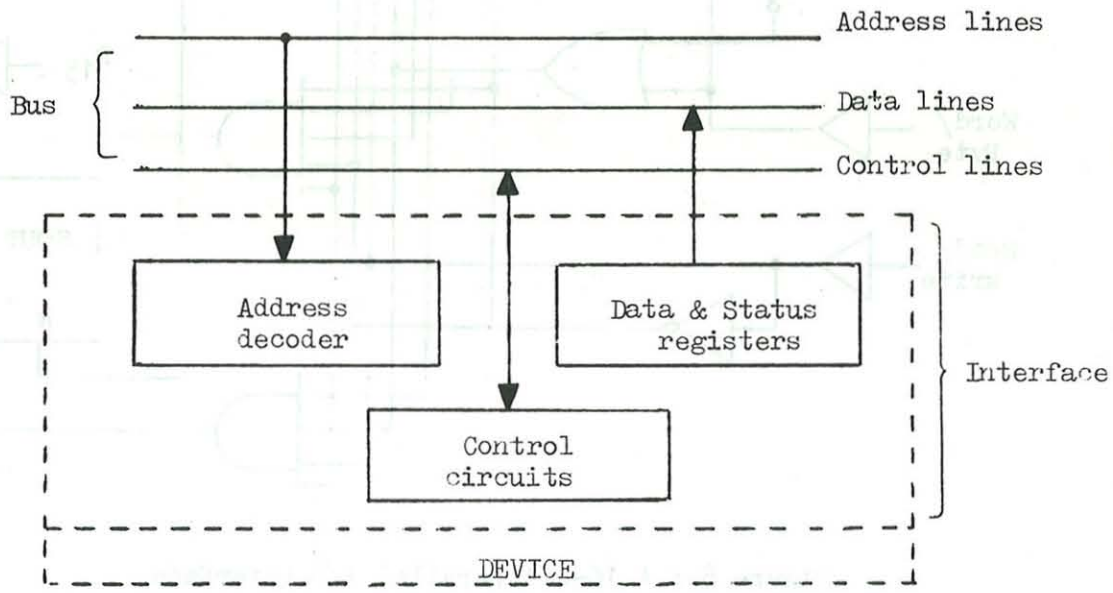


Figure 7 : Interface block diagram

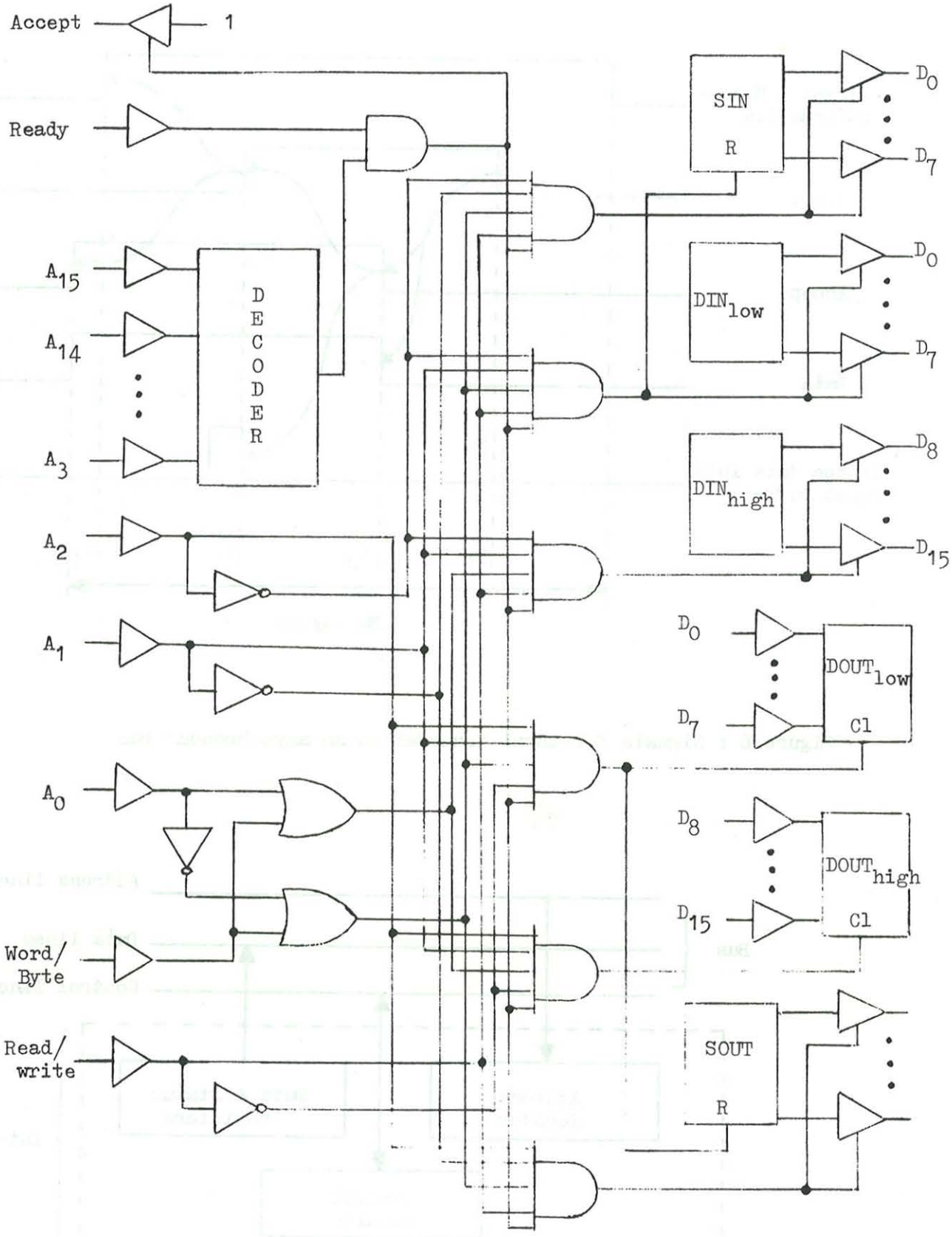


Figure 8 : A 16-bit parallel I/O interface

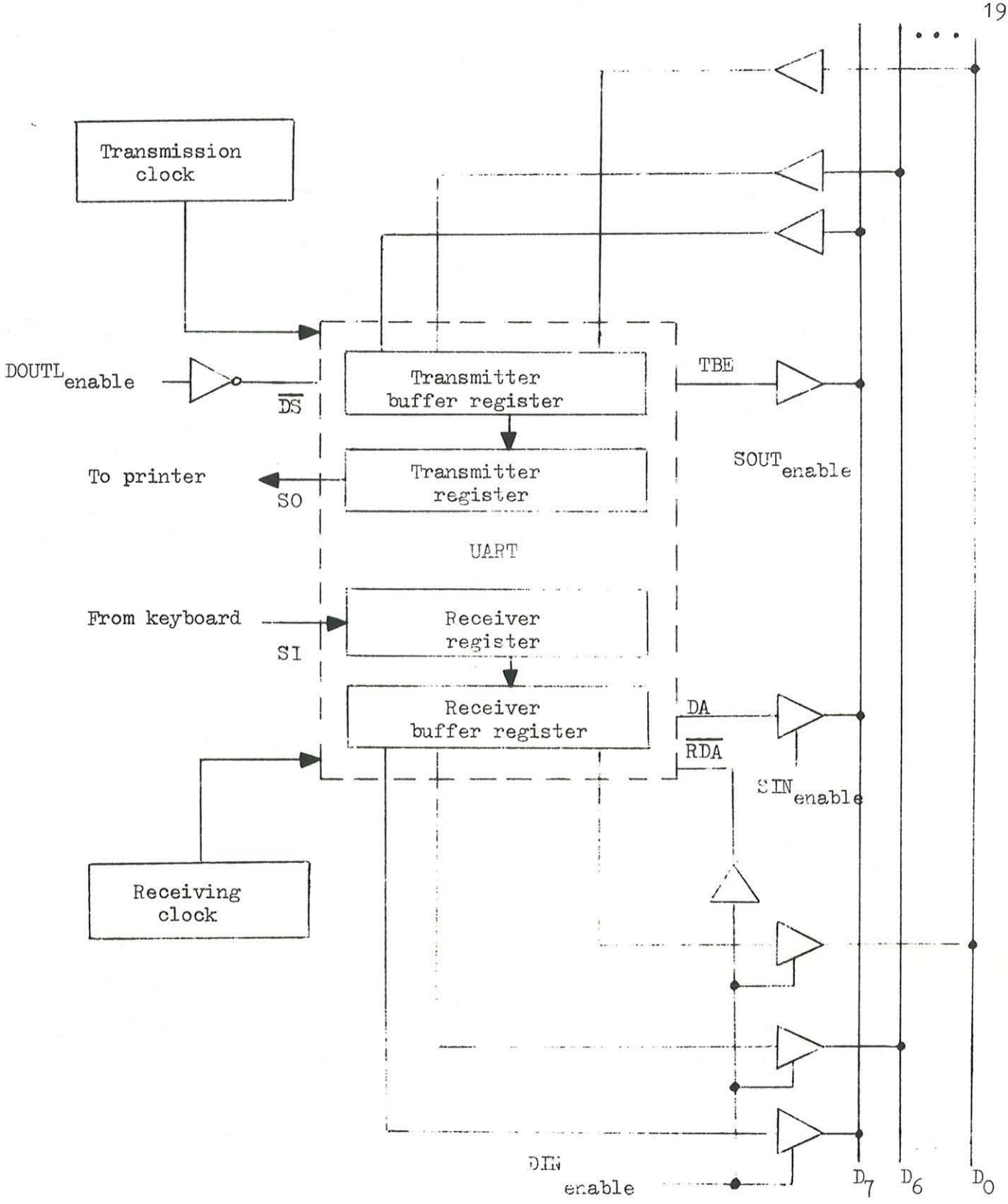


Figure 9 : A serial-parallel interface using the UART chip