The Bus Link—A microprogrammed development tool for the CMOS/SOS processor system*

by AVNER BEN-DOR
Telesensory Systems, Inc.
Palo Alto, California

and

PAUL BAKER and JON SELDEN
Hewlett-Packard Company
Cupertino, California

INTRODUCTION

The widespread use of microprocessor-based systems has made the problems of development time and development cost most urgent. With the increasing complexity of recent systems, there has come a great need for powerful and adaptive development tools.

The Bus Link is a development tool for the MC² processor system, which was developed along with the microprocessor chip. The Bus-Link can be used throughout the development cycle of any MC²-controlled system to solve either hardware or software problems. This tool is not restricted to any particular system configuration and can operate with the maximum allowable processor speed (see Figure 1).

SYSTEM DESCRIPTION

The Bus Link hardware can be partitioned into two parts—the controller and the Unit Under Test (UUT) interface (see Figure 2).

The controller consists of a microprocessor and 8K x 16 bit words of memory. In addition, the controller contains a serial data interface port (UART) and an IEEE standard 488 interface port. Since the only front panel controls are the POWER ON switch and a RESET button, the user interacts exclusively from the CRT terminal connected to the serial data port. The terminal also contains a dual cartridge tape unit which can be used to load programs to or from the UUT memory. Since most of the hardware (including the dual comparators) is software controlled, the user can add and modify the entire system by loading new routines from the cartridge tape unit.

The UUT interface consists of the following parts:

1. The dual comparator unit with an ALU.
2. The trace buffer unit.
3. The UUT bus interface.
4. The bus synchronizer unit.

Each one of the above units is a separate entity, and the connection between them is done under the user's supervision.

MODES OF OPERATION

The Bus Link can operate in two modes:

1. Management Mode.

Management Mode

In the Management Mode of operation the user enters commands from the CRT keyboard. The commands can be entered at any time even while the UUT is executing programs. The following capabilities are provided:

a. Load and Store Programs—User programs are loaded from UUT memory to the cartridge tape unit or vice versa.
b. Display and Modify UUT memory, registers and I/O.
c. Display Trace—The specified number of entries in the Trace Buffer is displayed.
d. Run, Halt and Single Step user programs.
e. Interrupt UUT—A forced hardware interrupt.
f. Reset the UUT system.
g. Force Handshake—The Bus Link records initiation and completion of handshaking activities (either Memory or I/O) on the UUT Bus. When this command is executed, the Bus Link senses the incompletely handshaking activity and simulates its completion so that UUT may resume execution.

* This work was compiled while employed by Hewlett-Packard Co. Data System Division, Cupertino, California.
h. Add Command—The user can enhance the capabilities of the Bus Link with additional commands which can be loaded into the Bus Link's memory from the system's cartridge tape unit.

Monitor Mode

Once the user issues the RUN command, control is passed to the UUT and the Bus Link enters the Monitor Mode. In this mode the Bus Link is monitoring a continuous process on the UUT. All 43 UUT bus lines (16 address, 16 data, 11 control) are continuously sampled.

A Bus Event occurs whenever a preprogrammed set of specifications describing conditions on the MC² bus has occurred. The occurrence of an event is a result of two individual comparisons being processed by an ALU unit and a delay counter (see Figure 3). Each comparator samples an actual condition on the UUT bus and compares it with an expected condition. Each one of the two comparators consists of three independent field comparators for the data field, address field and control field with a choice of: >, <, \( \geq \), \( \leq \), =, \( \neq \), and bit mask on each field along with AND operation of all three fields (refer to Figure 3). Altogether we should have six comparator circuits (2 x data, address, control) and their associated masking registers. However, no physical comparators can be found in the Bus Link since the entire task is performed by software, a technique that will be discussed later. Each of the two comparators is connected to the ALU unit and a delay counter to provide the desired event pulse. The event pulse is used to start or stop the trace buffer or to halt the execution of a program. While in Monitor Mode, the static condition on the UUT may be loaded into a local memory (64 words x 43 bits) called the Trace Buffer. The Trace Buffer may be started with an immediate command or following an occurrence of an event. Similarly, the Trace Buffer may be stopped by a command or following an event. The Trace Buffer may be loaded continuously, may be examined at any time, and may be stopped once the Buffer has been filled up.

The UUT may be programmed to halt under a certain set of conditions. When the condition occurs, the UUT is halted and control is passed to the user at the terminal. The UUT internal register values are updated on the screen and the instruction register is automatically disassembled. A UUT HALT condition can be set to follow an event condition or when the Trace Buffer is full (see Figure 4).

THEORY OF OPERATION

The main building block of any microprocessor development system is the comparator circuits. There are two popular ways of implementing this block: the first one is by using SSI gates (see Figure 5a) and second one is by using MSI circuits (see Figure 5b). When using only SSI circuits, the expected response is stored in the data register and the DON'T CARE (X) bits are stored in the mask register. The sampled data is stored into the input register and compared with the expected data stored in the data register. The result is then ANDED with the bit pattern stored in the mask register, before it is being ORed to provide a compare signal (see Figure 5a). Another common way of implementing the same block is by using an MSI comparator which can provide not only the compare signal (\(-\)), but also greater than (>) or less than (<) signals.
Figure 2—Bus link functional block diagram.
From the collection of the Computer History Museum (www.computerhistory.org)
The conventional techniques just described have three major disadvantages:

a. Expensive—Both circuits require a large amount of integrated circuits and also require a large P.C. Board since many traces must be routed between the components.

b. Incomplete—Comparing the sampled data with an expected response to determine a greater than (>), or a less than (<) equality while some of the bits are masked can’t be implemented with the above circuits.

c. Inflexible—Changes of existing design are hard to implement.

The comparator circuit can also be implemented with Random Access Memories (RAMs), and such implementation contains none of the disadvantages described earlier (see Figure 5c). The Bus Link is only one of many development tools in which the comparators are designed by utilizing RAMs. This concept will be explained with examples in the next paragraphs.

a. Comparing single breakpoint with single RAM—Let’s assume that we want to compare an n-bit word with another n-bit word (expected response vs. actual response). If an n-word x 1 bit RAM is available, and it is possible to store the data (1-bit word) in each location (n-bits address), the comparison process can be prepared by software and can be exercised by the RAM.

The software routine stores a “1” in the Kth word where the address of K is equal to the bit pattern we want to compare with (see Figure 6a).

Kth word address = expected response of n-bit word. The software also stores a “0” anywhere else in the memory. Now the RAM is ready to compare any actual data sampled on the UUT bus. The sampled data word is connected to the address field of the RAM. The RAM is read continuously by the processor which can now determine the result of the comparison. If the data word read (one bit) is found to be a “0,” it implies that the expected n-bit pattern is not equal to the actual one. If the data word is found to be a “1,” a match between the expected data and the actual one exists.

b. Comparing multiple breakpoints with single RAM—Multiple breakpoints comparison can be performed with the same hardware, only the software routine must be modified. Let’s keep the same definition of the Kth word. Again—Kth word address = expected response of n-bits word; we can deduct the following:

1b. To compare ≠ not equal, store in the Kth word a “0” and store a “1” anywhere else in the memory space. (See Figure 6b.)

2b. To compare <, less than (or ≤) store a “1” in the memory space from location 0 to (inclusive) the Kth address and a “1” from the Kth address and on. (See Figure 6c.)

3b. To compare >, greater than (or ≥), use the 2b algorithm with a complemented data word. (See Figure 6d.)

c. Handling masked (Don’t Care-X) bits with single RAM—Masked bits (X) can be easily processed by the same algorithm, with minor modification. A masked bit is essentially a multiple compare situation; each masked bit doubles the number of words to be compared with since X = 1 and also X = 0 (see Figure 6e). No mask registers or AND gates are needed.

d. Comparing multiple breakpoints with more than a single RAM—The Bus Link was designed as a development tool for the MC² microprocessor. As mentioned earlier the MC² has 16 bits of data word, 16 bits of address word and 11 bits of control word. If a RAM is to be used as a comparator, it must contain at least $2^{16} \times 1$ bit words (64K words). With an additional data bit, every single RAM can be divided down into smaller RAM units which may be cascaded to any arbitrary length. The additional bit is a dependency bit (carry bit) which enables the comparison of lower-order bits. In the example described in Figure 7, the 16-bit word comparator is implemented with two 256 x 2 bit words RAM. A comparison is always started with the high-order bits (byte) and carried to the lower-order byte only if it is needed. The software processes the expected data word format and determines whether a carry bit should be entered. This example can be carried further if either a larger than 16-bit word is to be compared or if the RAMs are to be partitioned to smaller units (for economic reasons).
e. **Multiple comparison of multiple breakpoints with more than a single RAM**—Most development tools available today offer only a single comparator, which is adequate for most applications. But a single comparator isn’t effective if complicated software routines or complicated I/O ports are to be developed. For example, an address space can be bounded to be greater than a minimum value AND less than another maximum value—upper bound ≥ address ≥ lower bound. If a hardware comparator (with SSI or MSI devices) is used, a second comparator almost doubles the amount of hardware. In using RAMs, only the RAM size (and some logic) is doubled which yields much greater price/performance ratio. As shown in Figure 7, the additional comparator function is added with almost no additional cost since the standard 256 word RAMs are four bits wide anyway. More comparators can be added to the circuit without much difficulty. The software prepares the data for each comparator separately and then concatenates the corresponding data words to form a single block before writing the entire block into each RAM.

f. **Do without multiplexers**—The address for the RAMs is provided from two sources—the UUT and the controller. When the controller writes the data into the RAMs it must provide the address field, but when the actual comparison is performed, the address field must be connected to the UUT bus. Multiplexing 43 lines between the controller and the UUT bus to the RAMs address field requires many multiplexers and consumes much of the PC Board area due to the large number of traces. Since the software prepares the data in blocks, multiplexing is not required, synchronous counters can be used instead. As shown in Figure 8, when the controller writes the information into the RAMs, the counters are automatically incremented and the processor only provides the data word to be written. When the actual comparison is to be made, the counters are used as registers where the count pulse is disabled and the load pulse is connected to the UUT clock.

g. **Single multitask sequential control circuit—the Hardware Subroutine**—In the management mode of operation, the user can enter one of many optional functions. The common method of designing the control logic is to implement each sequential logic (in a minimized...
purpose subroutine, it is called The Hardware Subroutine. A minimum of 3:1 reduction in components count was achieved utilizing this approach.

SOFTWARE DESCRIPTION

The operation of the Bus Link is controlled by the software, which resides in 8K words of RAM. The software can be partitioned into three main blocks:

a. Terminal handler.
b. Controller.
c. Comparator processor.

Terminal Handler

The terminal screen is partitioned into two areas (see Figure 4)—the dynamic and the static area. The static area is used to display the status of the processor and the status of the Bus Link. Commands are entered by the user in a form-fill-out fashion and interpreted only when the user exits this area. The dynamic area can be used for all control

form) of every function separately, and to attach a selector unit to activate each one of the functions separately (as shown in Figure 9). Further study of all the control functions revealed three important facts:

1. The functions are similar to each other.
2. Only one function can be active at any single time.
3. The functions are not time-sensitive.

Taking the previous facts into consideration and deviating from the standard approach, a new design method has resulted:

1. Find the largest common denominator among the functions. Simple functions can be made to act like complicated ones by adding redundant states.
2. Use a selector device to demultiplex the selected set of inputs (qualifiers) into the general sequential circuit block.
3. Use a demultiplexer to select one set of outputs from the sequential circuit block.

Since this method resembles the activity of writing a general-purpose subroutine, it is called The Hardware Subroutine. A minimum of 3:1 reduction in components count was achieved utilizing this approach.
Controller

The controller part is the supervisor of the entire system. It is composed of interrupt-driven routines which answer demands from either the terminal or the various hardware blocks in their assigned priorities. The controller is also capable of self-diagnosing the system once a failure is suspected.

Comparator Processor

As described earlier, the preparation of the data to be stored in the comparator RAMs is done by the software. Since setting a breakpoint could be done while the UUT processor is running, a powerful algorithm was developed to minimize the required time to process the breakpoint data. The algorithm can be described as follows:

1. Retrieve the breakpoint word.
2. Form a mask word which is a copy of the breakpoint word after all the don't care bits (x) are marked as "1"s, and all the actual bits ("1" or "0") are marked as "0"s.
3. Form a compare word which is a copy of the breakpoint word when all the don't care bits are modified to "0"s and the actual bits are left unchanged.
4. Prepare a table in which the address of the table corresponds to an identical breakpoint word (256 entries for eight-bit word) and the content of each address carries the following data—0: =(equal), 1: >(greater than), 2: <(less than).
5. Each address of the table must be ANDed with the mask word and compared with the compare word. The data of each address reflects the result of the comparison 1, 0 or 2 (stands for =, >, <).
6. For the high byte RAM refer to Table I. Select the desired row for the condition specified ( =, ≠, <, >, ≤, ≥) and change each entry of the table prepared in (5) to the corresponding bit format described in Table I (word-by-word table lookup).

<table>
<thead>
<tr>
<th>Condition</th>
<th>Hi Byte</th>
<th>Low Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>=, 0</td>
<td>=, 0</td>
</tr>
<tr>
<td>≠</td>
<td>1, 2</td>
<td>0, 2</td>
</tr>
<tr>
<td>&lt;</td>
<td>1, 2</td>
<td>0, 2</td>
</tr>
<tr>
<td>&gt;</td>
<td>1, 0</td>
<td>2, 0</td>
</tr>
<tr>
<td>≤</td>
<td>1, 2</td>
<td>0, 2</td>
</tr>
<tr>
<td>≥</td>
<td>1, 0</td>
<td>2, 0</td>
</tr>
</tbody>
</table>

7. Store the new table in the high byte comparator RAM.
8. Repeat Steps 2 through 6 for the low byte RAM and use Table I for the same task.
9. Store the new table in the low byte comparator RAM. The following is an example of using the previous algorithm for preparing the data for a four-bit compare word (see also Table I).

<table>
<thead>
<tr>
<th>Address</th>
<th>Masked Address</th>
<th>Compare Address</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1101 0000</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0001</td>
<td>1101 0001</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0010</td>
<td>1101 0000</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0011</td>
<td>1101 0001</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0100</td>
<td>1101 0100</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0101</td>
<td>1101 0100</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0110</td>
<td>1101 0101</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>0111</td>
<td>1101 0101</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>1000</td>
<td>1101 1000</td>
<td>1000</td>
<td>0 2</td>
</tr>
<tr>
<td>1001</td>
<td>1101 1001</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>1010</td>
<td>1101 1000</td>
<td>1000</td>
<td>0 2</td>
</tr>
<tr>
<td>1011</td>
<td>1101 1001</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>1100</td>
<td>1101 1100</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>1101</td>
<td>1101 1101</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>1110</td>
<td>1101 1100</td>
<td>1000</td>
<td>2 0</td>
</tr>
<tr>
<td>1111</td>
<td>1101 1101</td>
<td>1000</td>
<td>2 0</td>
</tr>
</tbody>
</table>

CONCLUSIONS

Four unique design contributions in the Bus Link have been presented in detail—RAM comparator unit, counter multiplexing logic, multitask sequential control circuit and a powerful algorithm to compute breakpoints. The above features when used together enable the designers to construct a very powerful development tool which is useful throughout the development cycle of any microprocessor-based system. The Bus Link is adaptive and can be easily modified to accommodate different systems. Each one of the previous features can also be used separately for different design applications to improve the cost-performance ratio.

The Bus Link and the associated products are currently being used in many applications with great success.

ACKNOWLEDGMENTS

The authors would like to thank Janelle Bedke for her help in defining the project and Larry Lopp for his support throughout the development phase. Special thanks are due to Kim Leeper who wrote many of the software routines and Helen Azadkhanian for typing the paper.
REFERENCES


2. Bell, C. G., and A. Newell, "The PMS and ISP descriptive systems."

