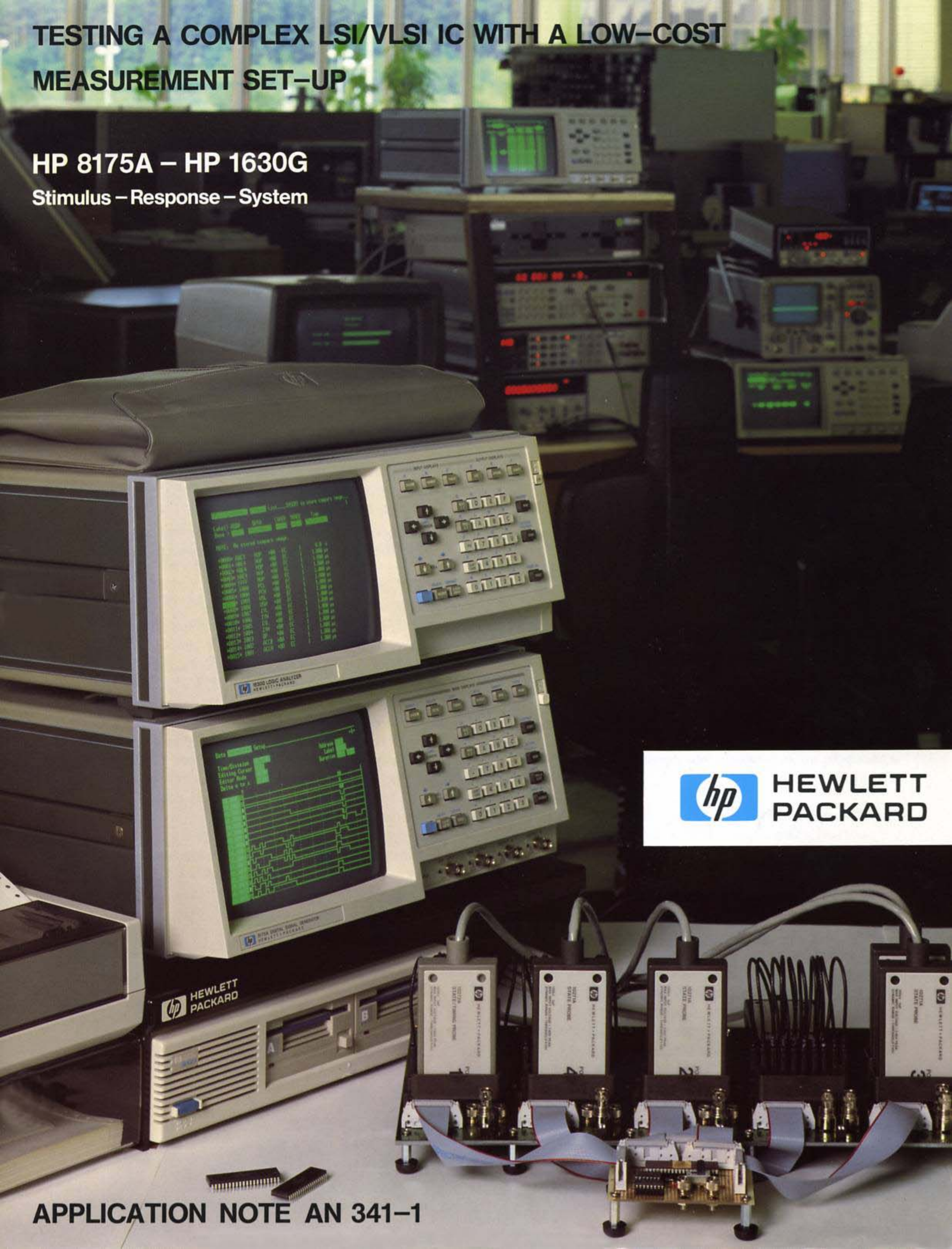
TESTING A COMPLEX LSI/VLSI IC WITH A LOW-COST MEASUREMENT SET-UP

HP 8175A – HP 1630G

Stimulus – Response – System



HEWLETT PACKARD

APPLICATION NOTE AN 341–1

INTRODUCTION

Testing today's complex digital devices requires comprehensive test systems capable of

a) generating a great variety of complex test signals in terms of appropriate data patterns, timings, logic levels and data rates,

b) emulating circuits from which the Device Under Test (DUT) is isolated during test,

c) analyzing the response of the DUT with regards to proper logic operation and expected timing.

A microprocessing unit (MPU) is one example of such a complex DUT.

MPU TEST APPROACHES

There are various philosophies for testing MPU's. However, most employ one of two basic approaches.

1. The MPU is used in its " natural environment " (with memory, I/O device etc. i.e., real execution conditions).
   If the execution of a program (stored in the program memory) provides the expected result, the MPU is assumed to operate correctly.
2. The MPU is isolated from its " natural environment ".
   It is activated and controlled by a test system that stores data patterns and control signal sequences and bursts them to the MPU. Responses of the MPU are monitored, sampled and stored for comparison with expected data.

The simplicity of the " natural environment " approach, unfortunately, has considerable drawbacks:

* The results after the test routine may be correct even if the MPU is defective (one failure negates another).
* The causes of a failure may be undetectable.
* Dynamic and parametric tolerances are not tested.

The " tester " approach eliminates these drawbacks, when succesfully implemented.
However implementation is more difficult and it requires higher investment. Computer power is necessary for logic modelling and fault simulation of the MPU, i.e. to provide the necessary test patterns and parameters. Tester hardware is necessary to physically apply test data to the MPU. Also, test software is required to link the tester to the computer, control the test equipment and supply an easy user interface for test program generation and test analysis.

MPU TEST SYSTEMS

There are big LSI/VLSI test systems on the market which combine the above features thus providing an almost complete solution. However, they are expensive in terms of initial investment and maintenance cost. They get their return from high throughput e.g in production.
As far as low and medium-volumes are concerned high investments can hardly be justified, particularly when testing occurs occasionally. Typical examples of low volume testing are found in engineering environments, such as prototype testing and quality assurance, and in incoming inspection. Although, low-cost alternatives do not compromize performance, they lack " turn key " comfort. Consequently, much more user involvement is needed regarding generation of test patterns, measurement software and system set-up.

PURPOSE OF THIS APPLICATION NOTE

This literature gives some suggestions for testing a MPU (6809E) with a low-cost measurement setup.

It focuses on the HP 8175A DIGITAL SIGNAL GENERATOR (DSG), which applies digital stimuli to the DUT, and the HP 1630G LOGIC ANALYZER (LA), which monitors, samples and analyzes the response of the DUT.
The reader will learn how easily the versatile test equipment adapts to diverse measurement problems that are encountered, when testing not only MPU's but also many other complex LSI/VLSI circuits. The following tests are covered:

- functional test of instruction codes
- reaction to control signals
- verification of timing

Measurement and analysis software and test pattern generation are not considered. Suggestions are covered in the chapter " Outlook ".

MODE OF OPERATION OF THE 6809E MPU

The hardware:
Data bus: Bidirectional 8 bit bus for data transfer between memory or peripherals and MPU. The MPU reads data from the data bus (direction from memory or peripherals to MPU) or writes data onto the data bus (direction MPU to memory or peripherals).
Address bus: Unidirectional 16 bit bus for address transfer from MPU to memory or peripherals.
Control lines: Different Input lines (e.g. interrupt and reset) to force the MPU to execute a certain operation or adopt a certain status.
Clock inputs E,Q: E and Q are clock signals required by the MPU for operation.

Normal operation:
Every instruction is defined with a one- or two-byte opcode (operation code). The opcode is transfered via the data bus from the memory to the MPU. The MPU reads the opcode, decodes and executes the instruction. Some instructions require one or two operands. These operands are contained in the memory, the internal registers of the MPU or in the peripherals.
Read cycle: During a read cycle the MPU reads data from the memory. The MPU puts an address onto the address bus, the memory puts the data, stored in this address onto the data bus. This data is then loaded into a MPU register for processing.
Write cycle: The write cycle is the inverse of the read cycle. The MPU puts an address onto the address bus and data onto the data bus. The address determines where the data is to be stored (e.g. into memory).
Execution of an instruction: Execution begins with an instruction fetch (read cycle). After that the MPU loads, if necessary, the operands (read cycle) and manipulates data as required (e.g. logic or arithmetic operations). Then it stores the result into memory (write cycle) or shows another reaction on the instruction, for example a conditional branch.
Instruction execution time: The MPU needs a definite time to execute an instruction. For example, there are three byte instructions for which the MPU needs three cycles and there are three byte instructions for which the MPU needs five cycles. The duration for an execution varies from instruction to instruction.
The RESET function: The reset input is one of the control lines that forces the MPU into a defined state.

GENERAL TEST CONSIDERATIONS

The vast amount of test data

- A large set of instructions
Testing a microprocessor requires a large amount of random data since the tester must be able to generate all possible instructions of the MPU. In the case of the 6809E, there are 59 instructions and 10 addressing modes resulting in 1464 different instructions, and about 8000 clock cycles for execution. In fact, this would be the data memory depth required in the tester.

- Sequential processing
However, even this would test isolated instructions only. Actually, an MPU processes data sequentially. The result of a command may depend on a previous operation which appeared thousands of cycles before. Take an "increment command" which increments an internal register: Executing just one increment, for instance, would not verify proper (internal) Carry Bit function, because this occurs only after several increments. Thus testing for instruction order sensitivity further increases test data. Obviously, all possible sequences cannot be tested without time consuming re-loading of test data from a controller. Even focusing on critical issues would surpass the memory capacity of a tester.

Emulation of the "natural environment"

- Synchronized interaction
Emulation of the "natural environment" requires true interaction between the test system and the DUT. One major concern is avoiding data collisions on the bidirectional data bus, since both the MPU and the test data generator have access to this bus. Collision would occur if both write data simultaneously.
Another problem is synchronizing the test system to the number of MPU execution cycles, which differ from instruction to instruction. If the data generator would generate a new instruction while the MPU is busy with the previous one, the MPU would miss data.

- Decision making
Emulation of the "natural environment" also includes real-time response to the MPU's decisions. Take a "conditional branch" instruction, for instance. Upon a successful match on a condition, the MPU calculates an effective branch address and puts it onto the address bus. The test system's data generator, emulating the program memory, must recognize this and jump to the appropriate data routine. The same is true for initialization routines, when legitimate testing can begin only when a particular output pattern appears.

HOW DOES THE TEST SYSTEM COPE WITH THIS ?

The vast amount of data

As would be expected, the major emphasis is on the re-use of data already in local memory. The data generator described in this note, the HP 8175A DIGITAL SIGNAL GENERATOR (DSG), features high economic use of data memory by storing data transitions only, i.e. consuming memory space only when data changes state. Steady-state conditions don't use up memory space. Also, different cycle times are easily simulated with a minimum of memory space. The pattern durations are stored in registers which complement each test pattern (Variable Pattern Duration). In our example, this reduces the required memory by approximately 50% ! Last not least, memory sequencing also reduces memory depth by programming patterns once and then calling up any sequence desired (Virtual Memory Expansion). Thus a loop can be executed several times, yet only has to be stored once. This enables detection of "hidden" failures (like the "Carry Bit" example mentioned before) without wasting memory.
Sampling and analyzing the response of the MPU also requires data reduction. With the HP 1630G LOGIC ANALYZER (LA) used in this measurement set-up, data of interest can be isolated by setting a variety of trace conditions. Sampled data is easily compared against expected data (Compare Image) which was previously stored, and any discrepancy is displayed (Full Compare Mode).

Emulation of the "natural environment"

Avoiding data collisions is achieved by switching the data generator's output lines into high impedance (TRI-STATE) when the MPU accesses the data bus. However, only the data bus drivers must be disabled, while the control lines have to remain active. This is done by connecting the $R/\overline{W}$ line of the MPU to the data output pod of the generator.
The Variable Pattern Duration feature mentioned above does not only save considerable memory space. What's more, it also enables correct synchronization to varying execution times of the MPU. Since the duration can be set individually for each test vector, the DSG can hold the last test vector until the instruction is executed.
Emulation of the program memory in order to respond correctly to the MPU's decision-making capabilities is achieved by a programmable 8-bit trigger input at the DSG. Connected to the address bus of the MPU, the generator recognizes a branch address, thus jumping to new data sequences before the next instruction is generated. A programmable Trigger Word Duration determines how long trigger words have to be stable, in order to avoid triggering on uncertain patterns and glitches due to race conditions.

TEST SYSTEM SET-UP

Figure 1 shows the test system set-up. The DSG and the LA are connected via pods with the DUT. Connections between pods and DSG or pods and LA are not shown.

Clocks: The MPU requires two 90 degree phase shifted clocks (E and Q clock, figure 4). Clock generation is performed in two different ways depending on the type of measurement:

1. For the measurement of the write data hold time the E,Q clock is generated using two data lines of the DSG (switch in position c). This assures high timing accuracy which is necessary, since the clock is used as a timing reference. Every clock cycle requires four memory locations of the DSG . Two lines of pod 0 are provided for generating the clock. Another line of pod 0 is connected with the LA clock input. This line then generates a strobe signal i.e. the sampling point for the LA.
2. For all other measurements the E,Q clock is generated with the external E,Q generator (switch in position a). The E,Q generator receives its input clock from the clock output of the DSG. The clock is divided into E and Q clock with the required phase shifting. External clock generation saves memory space in the DSG. One memory location is only consumed per clock cycle.

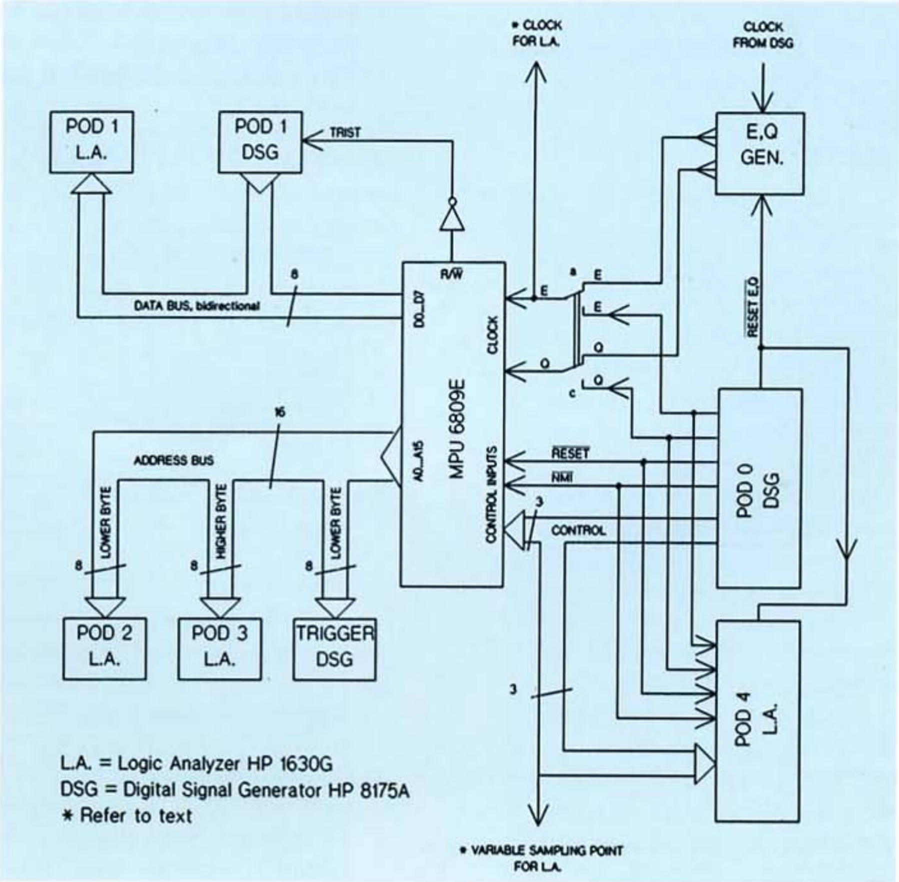
Data bus: Data generator and logic analyzer are connected with the data bus (pod 1). The logic analyzer samples data written onto the data bus either by the DSG or the MPU.

Tristate control: The tristate input of the DSG data pod 1 is connected to the R/W output of the MPU. If the MPU executes a read cycle, the pod is enabled and the DSG puts data onto the bus. If the MPU executes a write cycle the R/W output turns to low level (high level at pod 1) which forces pod 1 into tristate. This prevents MPU and DSG from writing data to the bus at the same time.

Address bus: With its trigger input pod the DSG watches the lower byte of the address bus in order to recognize branch conditions. All 16 bits of the address are sampled with the LA (pods 2 and 3) to check address data.

Control lines: Control lines are connected from pod 0 of the DSG with the control inputs of the MPU. The two important ones for this application are RESET and NMI, the others named in figure 1 with " Control " are FIRQ, IRQ and HALT. Another control line is the RESET E,Q for the E,Q generator. Pod 4 of the LA allows monitoring control signals.

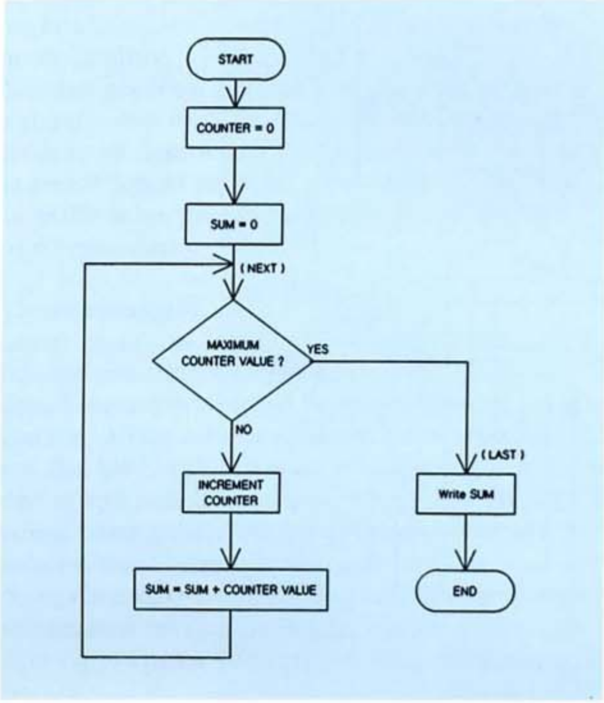
Figure 1 TEST SYSTEM SET-UP



TEST OF NORMAL OPERATION

Execution of one instruction: The instruction TFR X,D (Transfer X-Register to D-Register) has the Hex-Code 1F (1st instruction byte) and 10 (2nd instruction byte). The execution time is 6 clock cycles. The two bytes generated from the DSG are read and executed by the MPU.
The DSG must wait until the MPU has executed the operation. This is performed by setting the pattern duration to the same time the MPU requires for execution. In this case the first instruction byte is generated with a pattern duration of one cycle, the second with a pattern duration of 5 cycles (together 6 cycles).
Execution of an entire program allows a compressed test of the MPU without checking the instructions individually, because the correct result of the program permits the deduction that all instructions have been executed correctly from the MPU.This reduces test time and increases throughput. Another advantage is that the MPU is checked under real execution conditions i.e. the various instructions are not isolated from each other so that critical issues due to the MPU's sequential processing nature are also covered.In the following example the MPU must calculate the sum of natural numbers starting with 1 and incrementing by one. Incrementing is done in a counter-loop. Calculation should terminate if maximum counter value is achieved (figure 2).
MPU operations like read cycles, write cycles, arithmetic calculations, register transfers, conditional and unconditional branchings can be tested with this program.

Figure 2 FLOW CHART



Example: Sum of natural numbers to 6
(6 = maximum counter value):
Sum = 1+2+3+4+5+6 = 21

The program is stored in Hex-code in the DSG. The DSG generates the instructions byte by byte with a pattern duration corresponding with the number of cycles the MPU needs for execution (Last two columns of Chart 1). The MPU reads these instructions from the data bus and executes the program.
The labels " NEXT "and " LAST " are symbolic branch addresses. The instruction BEQ LAST is a conditional branch instruction. Branching depends on the comparison with the maximum counter value. The BRA NEXT instruction is an unconditional branch instruction. Branching is always performed . The effective branch address is put by the MPU onto the data bus and recognized by the DSG via the trigger input pod (lower byte of the address).
Label " NEXT " (chart 1 and figure 2) has the effective address (lower byte) 00000100 (binary value), Label " LAST " has the effective address 00001100. Figure 3 shows the trigger word assignment of the DSG. The DSG continues generating the program with program cycle JMP A (corresponding with label " NEXT ") or JMP B (corresponding with label " LAST ") depending on the address the MPU puts onto the address bus. The program cycle JMP A starts again with generating the instruction CMPB #06 (chart 1), the program cycle JMP B continues generating data with the instruction TFR X,D. With the last program instruction STB AABB the MPU writes the Sum onto the data bus.
The result and the branch addresses are checked with the LA.

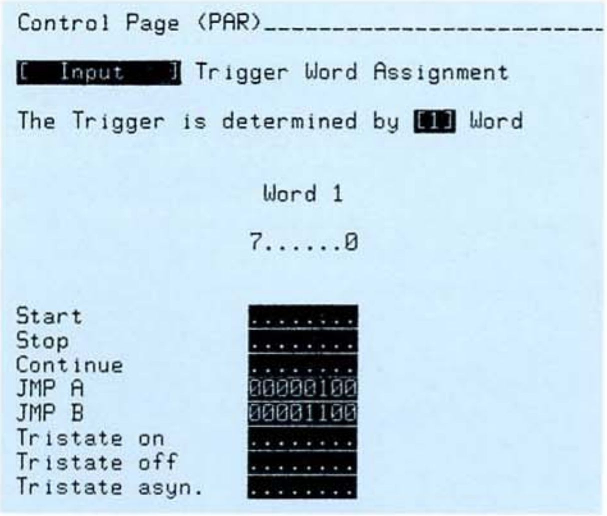
Chart 1 MPU PROGRAM

Accu B = Counter register
X-Register = Register for SUM

Description	Label	Mnemonics	Hex-code	Number of cycles
Clear Accu B (Counter = 0)		CLR B	5F	2
Load X-Register immediate with 0 (Sum = 0)		LDX #0000	8E 00 00	1 1 1
Compare Accu B with maximum counter value	NEXT:	CMPB #06	C1 06	1 1
Branch on equal to " LAST ", if not		BEQ LAST	27 04	1 2
Increment B by 1		INC B	5C	2
Add accu B into X-Reg.		ABX	3A	3
Branch always to " NEXT "		BRA NEXT	20 F8	1 2
Transfer X-Reg. to D-Register (*)	LAST:	TFR X,D	1F 10	1 5
Store Register B into memory location AABB (MPU write cycle)		STB AABB	F7 AA BB	1 1 3

(*) = Transfer of X-Reg. to Accu B

Figure 3 TRIGGER WORD ASSIGNMENT



```
Control Page (PAR)________________________

[  Input   ] Trigger Word Assignment

The Trigger is determined by [1] Word

                 Word 1

                 7.....0

Start            ........
Stop             ........
Continue         ........
JMP A            00000100
JMP B            00001100
Tristate on      ........
Tristate off     ........
Tristate asyn.   ........
```

TEST OF INTERRUPT TIMING

An interrupt on the control line $\overline{NMI}$ forces the MPU, running in normal operation, to execute an interrupt service routine. When the MPU has finished this routine it continues executing the interrupted program. Since the interrupt service routine changes data of the internal registers of the MPU, these data have to be saved before the MPU executes the service routine. Saving occurs during the 18 cycles after the interrupt signal has been detected. While the MPU writes the contents of the internal registers onto the data bus the DSG executes a wait cycle. This is achieved by programming the DSG to a NOP (no operation) instruction for the exact duration of 18 cycles. In fact any other instruction would do, since the DSG pod is disabled during the MPU's write cycles.

During the 19th and 20th cycles the MPU enables the DSG again and reads the new address which the DSG puts onto the data bus. The new address specifies the start of the interrupt service routine. Before an interrupt signal is given from the DSG to the MPU the DSG loads initial data into the internal registers of the MPU. The LA monitors address and data bus and verifies that the MPU saves exactly these data.

MEASUREMENT OF WRITE DATA HOLD TIME

Measurement problem:

The write data hold time is specified in the data sheet of the manufacturer. The time can be measured during a MPU write cycle. Write data hold time is the time write data is stable after the falling edge of the E clock.

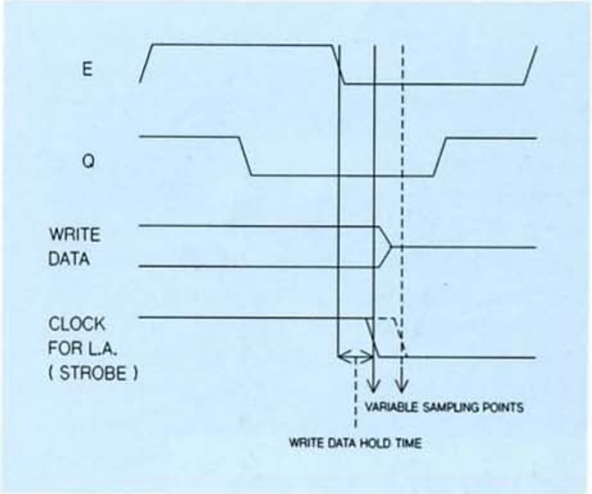
Figure 4 shows the E and Q clock as well as the data written onto the data bus by the MPU (write data).

Measurement principle:

During the write cycle of the MPU the DSG generates a strobe signal which determines the sampling point for the LA. This sampling point is variable . Therefore, the Pattern Duration setting is split into 20 ns steps for rough positioning. High resolution positioning of this edge is possible with the Fine Timing feature of the DSG. It allows to delay four channels with respect to the other channels in a range of 20 ns in 100 ps steps. One of these channels generates the variable sampling point. The LA samples at the falling edge of this strobe signal.

In order to eliminate timing uncertainties of the E,Q generator, these clocks are generated using data channels of the DSG (figure 1).

Figure 4 MEASUREMENT OF
WRITE DATA HOLD TIME



Performing the measurement: Before starting the first measurement the falling edge for the LA should be positioned to a save value i.e. a write data hold time which is smaller than the time specified from the manufacturer. The LA will then sample the correct write data (solid line of the variable sampling points, figure 4). Next measurement should be performed setting a greater value and so on until wrong data is sampled (rough positioning, dashed lines in figure 4 symbolize such a sampling point). Then rough positioning is turned back one step into the last error free setting. Fine Timing allows a high resolution sampling point placement " on the fly "; i.e., while the instrument is running . This is a convenient way of finding the critical point where errors occur. The time difference between the falling edge of E and the variable sampling point is the actual write data hold time.

CONCLUSION

A Stimulus - Response System configured with an HP 8175A DIGITAL SIGNAL GENERATOR and an HP 1630 LOGIC ANALYZER can be applied to testing complex digital LSI/VLSI IC's such as microprocessors. The versatile feature set of the test instruments enables easy adaption to the numerous test signal requirements encountered in such applications. Thus, the test system copes with such critical issues as:

- the vast amount of test data via highly economic use of the generator's test pattern memory. Also, sampled data is reduced by the analyzer's trace conditions.

- emulation of the "natural environment" of the MPU. This includes controlling the bidirectional data bus to avoid data collisions, synchronizing the test system to the MPU's varying execution times, and making real-time decisions, thus testing the MPU in exactly the same manner as it is used.

This eliminates extra circuit design which is a potential source of measurement uncertainties and engineering inefficiency, thus allowing a test engineer to focus on the real task.

Operating convenience is provided by a user interface which includes a large, menu-driven CRT. In addition, fast reconfigurations of complete instrument set-ups are possible from internal, non-volatile storage locations, and from an external disc drive which can be accessed without using a computer. Also, test documentation on an external printer is achieved by one keystroke.

OUTLOOK

Connecting an HP-IB (IEEE 488) controller to the test instruments enables automatic test pattern generation, data analysis of sampled MPU responses, documentation of results and control of the test flow.

Test patterns can be supplied from a computer running a fault simulation program. Results are then translated into the tester language. However, this requires software modelling of the DUT which may be unavailable or may lack sufficient fault coverage. Also down-link software from the mainframe computer to the instrument controller is necessary.

This can be avoided with "stored-response" testing, where the test system "learns" the response of a known good reference device placed in the test socket. In the "learn" phase, a program permits the test engineer to enter an instruction in the MPU's mnemonic language. This program then translates the mnemonics into functional stimuli sequences, making use of a previously written "look-up" table. This relates the input code to the tester language.

This heuristic technique of test pattern generation, however, fulfills it's purpose only if the test pattern generator truly emulates the MPU's actual environment, as described in the previous pages.

LITERATURE

HP 8175A Technical Data Sheet,
Pub.No. 5952-9560

HP 1630A/D/G Technical Data Sheet,
Pub.No. 5953-3939

Microprocessors Data Manual 1982, Data sheet of the MC6809E, MC68A09E, MC68B09E, MOTOROLA Inc.

Automatic Testing and Evaluation of Digital Integrated Circuits, James T. Healy, Reston Publishing Company Inc., 1981.

HEWLETT
PACKARD

For more information, call your local HP sales office listed in the telephone directory white pages. Ask for the electronic Instruments Department or write to Hewlett-Packard: **U.S.A.** - P.O. Box 10301, Palo Alto, CA 94303-0890. **Europe** - P.O. Box 999, 1080 AZ Amstelveen, The Netherlands. **Canada** - 6877 Goreway Drive, Mississauga, L4V 1M8, Ontario. **Japan** - Yokogawa-Hewlett-Packard Ltd., 3-29-21, Takaido-Higashi, Suginami-ku, Tokyo 168. **Elsewhere** in the world, write to Hewlett-Packard Intercontinental, 3495 Deer Creek Road, Palo Alto, CA 94304.

B 100
5952-9572

Printed in the Federal Republic of Germany.