Hierarchically Accessing 1149.1 Applications in a System Environment

Lee Whetsel
Texas Instruments Incorporated
Senior Member Technical Staff
Semiconductor Group

SCTA033

TEXAS INSTRUMENTS

Hierarchically Accessing 1149.1 Applications in a System Environment

Lee Whetsel
Senior Member Technical Staff
Texas Instruments Inc
Ph/Fx 214-575-2601/6198

Abstract

This paper presents a novel connection method that enables an 1149.1 test bus controller to hierarchically access and test 1149.1 circuits, independent of where the circuits exists within in an electronic system. The advantage of this approach is that it enables the 1149.1 test bus to be used hierarchically as a system level test bus, instead of only as a board level test bus.

Background

In the electronics industry, significant improvements have been made in the area of board level testing, largely due to the development of a standard test access port and boundary scan architecture for ICs, referred to as IEEE Std 1149.1-1990 [1]. The 1149.1 standard defines a test access port and associated test circuitry that can be included in ICs to simplify testing at the board level.

One of the most important features of the 1149.1 standard is a four wire test bus that enables commands and data to be serially communicated to ICs on a board for testing purposes. The presence of this standard board level test bus interface has inspired the test community to focus on a common approach to board level testing problems.

The challenge now facing the electronics test industry is how to gain access to 1149.1 testability features after the board has been embedded within a system. Most system companies who invest in 1149.1 at the board level, do so under the assumption that the board testability resources can be reused to simplify their system testing problems. However, currently there are no proposed methods of accessing 1149.1 in a system environment. There are proposals for accessing 1149.1 boards in a backplane environment, but these proposals are focused on single level access operations and do not anticipate the hierarchical access needs found within system architectures. [2,3,4,5,7,8]

If test access solutions progress the way they currently are, the industry will eventually have a unique test access standard for each different environment level within a system. For example, the industry may eventually have different standards for: accessing multi-chip modules on a board, accessing boards in a backplane, accessing backplanes in a subsystem, and accessing subsystems in a system. If system test access evolves in this fashion, the end result may well turn out to be an overly complicated, inefficient network of non-compatible test buses interfaced together using protocol translating devices.

Introduction

This paper presents an approach which anticipates the hierarchical test access needs of system architectures. Using this approach, hierarchical connections can be made in a system architecture to enable 1149.1 applications to be accessed directly via the standard 1149.1 test bus. The primary benefit of this approach is that it eliminates the need to use other environment-specific test buses to gain access to embedded 1149.1 applications within systems.

A paper presented at ITC in 1992 described how an 1149.1 test bus controller (TBC) could select and access 1149.1 applications in a single level environment, such as boards in a backplane, using a protocol referred to as a shadow protocol and a connection device referred to as an addressable shadow port (ASP) [6]. The approach described in this paper is based on an extended shadow protocol and a connection device referred to as a hierarchically addressable shadow port (HASP). The extended shadow protocol and HASP allow the connection features described in the 1992 paper to be used hierarchically in a system architecture, instead

of being limited to single level connection applications.

Throughout the remainder of this paper the words "environment" and "application" are used. The word "environment" is used to indicate a physical level within a system architecture where one or more ASPs or HASPs reside. The "root environment" is the lowest level environment, and is where the 1149.1 TBC resides. The word "application" is used to indicate an 1149.1 circuit within an environment that can be accessed by a TBC after a hierarchical connection has been made. In this paper, the hierarchical access examples are described as coming from a lower level environment to a higher level environment.

Shadow Protocols

The 1149.1 test bus has four signal wires, test clock (TCK), test mode select (TMS), test data output (TDO), and test data input (TDI). 1149.1 protocol transmitted on the TMS signal controls applications on the test bus to scan data, enter an idle state, or enter a reset state. When TMS places the test bus in an idle state (i.e. 1149.1 RT/IDLE, PAUSE-IR or PAUSE-DR states) or a reset state (i.e. 1149.1 TLRST state), all 1149.1 applications are disabled from responding to data transmitted on TDI and TDO. While the 1149.1 test bus is idle or reset, the shadow protocol can be transmitted over the TDI and TDO wires to make a connection between a TBC and a target application within a system, via an ASP or via an ASP connected to one or more HASP circuits.

The shadow protocol comprises two protocols, a select protocol and an acknowledge protocol. At the beginning of a shadow protocol, a select protocol is transmitted from the TBC to an ASP either directly or through one or more HASPs residing between the TBC and ASP to make a connection. After the select protocol has been transmitted, an acknowledge protocol is transmitted to the TBC from the selected ASP either directly or through one or more HASPs residing between the ASP and TBC to confirm the connection. The shadow protocol is transmitted on the TDI and TDO bus wires using a bit-pair signaling method. The TMS signal is not involved with the shadow protocol, therefore the TMS signal can be used to hold 1149.1 applications in a desired 1149.1 steady state while the shadow protocol is transmitted to make a connection.

The bit-pair signaling method used in the shadow protocol allows control and data to be transmitted together on a single wiring channel. The control signals are used to start and stop the select and acknowledge protocols and to frame data signals that form addresses within the select and acknowledge protocols. The bit-pair signals used in the select and acknowledge protocols are defined in the following list.

Idle Signal (I) – a control signal identified by the transfer of two successive logic one bits from a transmitter to a receiver.

Select Signal (S) – a control signal identified by the transfer of two successive logic zero bits from a transmitter to a receiver.

Data 1 Signal (D) – a logic one signal identified by the transfer of a logic zero bit followed by a logic one bit from a transmitter to a receiver.

Data 0 Signal (D) – a logic zero signal identified by the transfer of a logic one bit followed by a logic zero bit from a transmitter to a receiver.

The bit-pair signals are output from the transmitting device's (TBC's, ASP's, or HASP's) TDO on the falling edge of TCK and are input to the receiving device's (TBC's, ASP's, or HASP's) TDI on the rising edge of TCK. Since this data transfer is consistent with the way 1149.1 serial data is transferred, upgrading a TBC to support this approach is simply a matter of controlling TMS to idle or reset 1149.1 applications, while using otherwise normal 1149.1 scan operations to transmit and receive the select and acknowledge protocols.

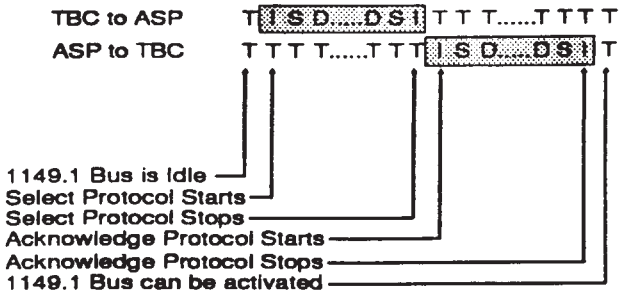

FIGURE 1 Select and Acknowledge Protocols

Select and Acknowledge Protocols

In Figure 1, a diagram is shown of a single level select and acknowledge protocol being transmitted while the 1149.1 test bus is idle to make a connection between a TBC and

ASP application. The tristate signals (T) before and after the protocol sequences indicate that the 1149.1 test bus is idle and that the TDO outputs from the TBC and ASP are disabled and pulled high. The "ISD..DSI" sequence transmitted from the TBC to the ASP is the select protocol. The "ISD..DSI" sequence transmitted from the ASP to the TBC is the acknowledge protocol. After the ASP transmits the acknowledge protocol it connects the TBC up to the application.

The I signal at the beginning of each protocol is designed to be indistinguishable from the preceding T signals. This avoids unintentional entry into a select or acknowledge protocol when the 1149.1 bus enters an idle or reset state. However, the I signal at the end of each protocol is designed to be distinguishable from the preceding S and D signals so that it can be used to terminate the protocol. Inside each protocol, first and second S signals are used to frame the address which is defined by a series of logic 1 and logic 0 D signals.

In this single level access shadow protocol, only one address frame (SD..DS) is transmitted between the first and second I signals of the select and acknowledge protocol. However, in a multi-level shadow protocol, two or more address frames (SD..DS) are transmitted between the first and second I signals of the select and acknowledge protocol to make a hierarchical connection.

Hierarchical Access Examples

The following examples illustrate how a TBC can use ASP and HASP circuits to hierarchically access 1149.1 applications in systems having from one to "m" environment levels. In the examples, analogies are made between the way this approach hierarchically links to an 1149.1 application through multiple system environments, and the way an operating system hierarchically links to a file through multiple software directories. In the analogies, "Sys"=system, "Sub"=subsystem, "Bpn"=backplane, "Brd"=board, "App" = application, "Env"=environment, and "Dir"=directory.

Single Level System Test Access

In the single level environment of Figure 2, a Backplane resident TBC is connected to Board ASPs (1-n) via a four wire 1149.1 test bus. Each ASP is further connected to ICs on a board (application) via a four wire 1149.1 test bus. The naming convention given to the ASPs in Figure 1 is "ASPn:m", where "n" indicates the ASP's address and "m" indicates the environment level the ASP resides on. The ASPs are connected to the TBC via their primary port signals (PTDI, PTMS, PTCK, PTDO) and to the application via their secondary port signals (STDI, STMS, STCK, STDO). While all four 1149.1 signal wires are shown in Figure 2, only the ASP PTDI, STDI, PTDO, and STDO, and application and TBC TDI and TDO signals are named. The environment level number of the ASP is included in the primary and secondary port signal names, i.e. PTDI1, PTDO1, STDO1, and STDO1. The above mentioned naming conventions are followed throughout the remainder of this paper.

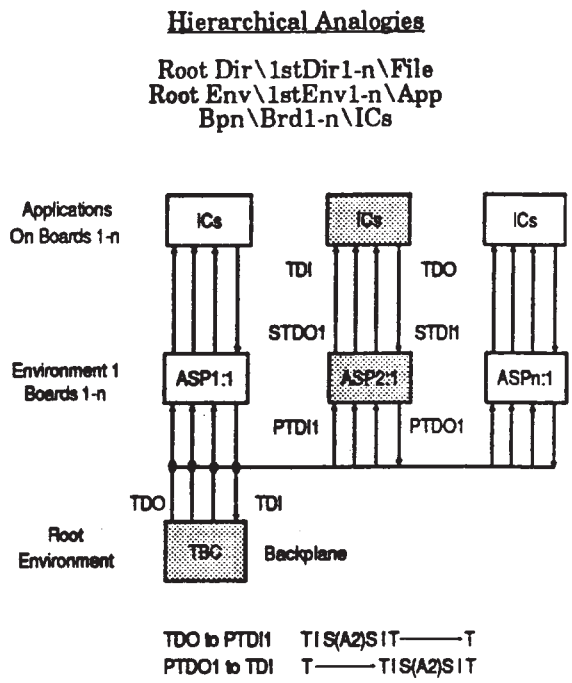

FIGURE 2 Single Level System Test Access

Before an application on one of the Boards (1-n) can be serially accessed by the TBC, a connection must be made between the TBC and application. To make a connection between the application of ASP2:1 and the TBC, the TBC outputs a single level select protocol from its TDO output to the PTDI1 input of all ASPs on Environment 1. In this example, the address sent in the select protocol is address 2 (A2). In response to receiving the select protocol with an address of 2, ASP2:1 outputs an acknowledge protocol containing its

address (A2) from its PTDO1 output to the TBC's TDI input, then connects the application to the TBC (as shown in darkened boxes). The TBC verifies the connection by inspecting the address returned in the acknowledge protocol, then accesses the application using the 1149.1 protocol. This single level connection example is identical to the one described in the 1992 paper, since only a single level connection is made, i.e. board and backplane.

Two Level System Test Access

In the two level environment of Figure 3, a Subsystem resident TBC is connected to Backplane HASPs (1-n) in Environment 1. Each Backplane HASP (1-n) is further connected to Board ASPs (1-n) in Environment 2. Each Board ASP (1-n) is further connected to an Application. While connections are only shown between HASP1:1 and its ASP/Application group, each HASP is similarly connected to an ASP/Application group.

Hierarchical Analogies

Root Dir\1stDir1-n\2ndDir1-n\File
Root Env\1stEvn1-n\2ndEvn1-n\App
Sub\Bpn1-n\Brd1-n\ICs

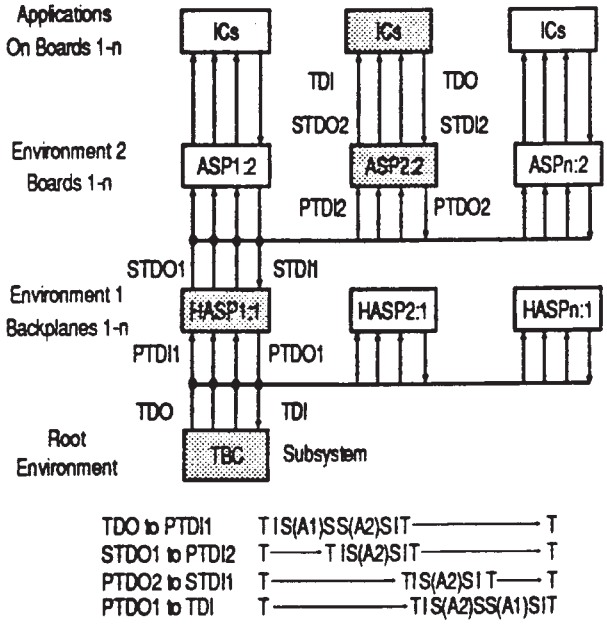

FIGURE 3 Two Level System Test Access

Before an application on one of the Boards (1-n) can be serially accessed by the TBC, a hierarchical connection must be made between the TBC and application. To make a connection between the application of ASP2:2 and the TBC, the TBC outputs a two level select protocol from its TDO output to the PTDI1 input of all HASPs on Environment 1. The two level hierarchical select protocol differs from the single level select protocol of Figure 1 in that two address frames are transmitted between the first and second I signals. The first address frame (A1) selects HASP1:1 and the second address frame (A2) selects ASP2:2.

After HASP1:1 has received its address frame (SA1S), it looks to see what signal follows the address frame. If an I signal were to follow the first address frame, HASP1:1 would recognize the protocol as a single level type and would start its acknowledge protocol. However, since an S signal follows the first address frame, HASP1:1 recognizes that the select protocol is hierarchical and that a new address frame is being transmitted. After HASP1:1 recognizes that the select protocol is hierarchical, it does not respond to any of the additional address frames it receives, thus it cannot be deselected by subsequent address frames transmitted within the current hierarchical select protocol. Also when HASP1:1 recognizes that the select protocol is hierarchical, it sets an internal flag which modifies the way it operates during the hierarchical acknowledge protocol.

In response to the start of the second address frame (SA2S) from the TBC, HASP1:1 enables its STDO1 output, sends an I signal, then relays the second address frame to ASPs of Environment 2. There is a one bit-pair signal latency between the end of the first address frame from the TBC (SA1S) and the start of the relayed second address frame from HASP1:1 (SA2S). This latency is caused by the decision step HASP1:1 performs to determine what signal (S or I) follows the first address frame (SA1S).

When the TBC completes the transmission of the hierarchical select protocol, it outputs T signals (or logic 1's) on its TDO output and monitors its TDI input for the start of an acknowledge protocol from the PTDO1 output of HASP1:1. Likewise, when HASP1:1 completes relaying the hierarchical select protocol it outputs T signals on its STDO1 output and monitors its STDI1 input for the start of an acknowledge protocol from the PTDO2 output of ASP2:2.

After ASP2:2 has received its address frame (SA2S) from HASP1:1, it starts an

acknowledge protocol output to HASP1:1. After transmitting a first I signal to initiate the acknowledge protocol, ASP2:2 outputs its address frame sequence (SA2S) from its PTDO2 output to the STDI1 input of HASP1:1. In response to the first S signal of the address frame input from ASP2:2, HASP1:1 enables its PTDO1 output and starts relaying the acknowledge protocol from ASP2:2 to the TBC's TDI input by outputting a first I signal. After ASP2:2 has transmitted its address frame to the STDI1 input of HASP1:1, it terminates its acknowledge protocol by outputting a second I signal, then makes a connection between its application and the secondary port of HASP1:1.

In response to the second I signal input from ASP2:2, HASP1:1 continues the acknowledge protocol sequence by inserting and outputting its own address frame (SA1S) to the TDI input of the TBC. After HASP1:1 has transmitted its address frame to the TBC, it terminates the hierarchical acknowledge protocol by outputting a second I signal, then makes a connection between ASP1:1 and the TBC. After the TBC receives the second I signal it determines that the hierarchical acknowledge protocol is complete and examines the addresses received to confirm the correct hierarchical connection was made. If the connection is correct, the TBC accesses the application using the 1149.1 protocol.

Three Level System Test Access

In the three level environment of Figure 4, a System resident TBC is connected to Subsystem HASPs (1-n). Each Subsystem HASP (1-n) is further connected to Backplane HASPs (1-n). Each Backplane HASP (1-n) is further connected to Board ASPs (1-n). Each Board ASP (1-n) is further connected to an Application.

The steps for the TBC to hierarchically connect and access the application of ASPn:3 is similar to the two level access example of Figure 3 and comprises; (1) outputting a three level hierarchical select protocol to select HASP1:1, HASP2:2, and ASPn:3, (2) receiving and confirming a three level hierarchical acknowledge protocol from ASPn:3, HASP2:2, and HASP1:1, and (3) accessing the application via the connections made between HASP1:1, HASP2:2, and ASPn:3 using the 1149.1 protocol.

Hierarchical Analogies

Root Dir\1stDir1-n\2ndDir1-n\3rdDir1-n\File
Root Env\1stEvn1-n\2ndEvn1-n\3rdEnv1-n\App
Sys\Sub1-n\Bpn1-n\Brd1-n\ICs

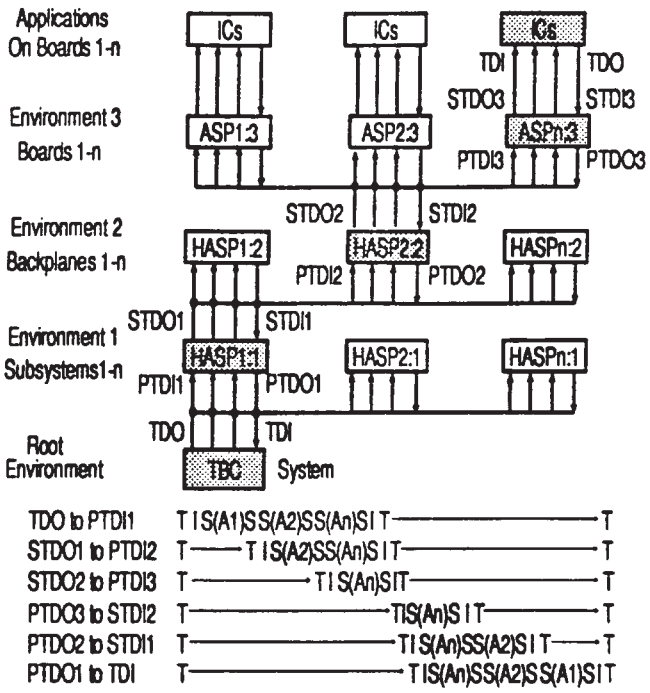

FIGURE 4 Three Level System Test Access

Mth-Level System Test Access

While the previous examples have shown how shadow protocols and ASP/HASP circuits are used to access applications existing on 1, 2, and 3 level system environments, the approach can be used to access any environment level (m) within a system.

For example, the hierarchical select and acknowledge protocols of Figure 5 illustrate connecting an application in environment level "m" to a TBC in the root environment (RE) via intermediate environment levels E1, E2, E3...Em-2, and Em-1. Each address frame in the hierarchical select and acknowledge protocols is indicated by the sequence "Sn:mS", where "n" is the address and "m" is the environment level the address is sent to or received from. The ability of the this approach to hierarchically connect a TBC to an application on any environment level within any type of system, provides an extremely simple, yet powerful method of accessing and testing 1149.1 applications.

Hierarchical Select Protocol to Environment "m"

RE to E1	T I Sn:1SSn:2SSn:3S.....Sn:m-2SSn:m-1SSn:mS I T.....T
E1 to E2	T.....T I Sn:2SSn:3S.....Sn:m-2SSn:m-1SSn:mS I T.....T
E2 to E3	T.....T I Sn:3S.....Sn:m-2SSn:m-1SSn:mS I T....T
Em-3 to Em-2	T .....T I Sn:m-2SSn:m-1SSn:mS I T...T
Em-2 to Em-1	T .....T I Sn:m-1SSn:mS I TT
Em-1 to Em	T .....T I Sn:mS I T

Hierarchical Acknowledge Protocol from Environment "m"

Em to Em-1	T I Sn:mS I T ..... T
Em-1 to Em-2	TT I Sn:mSSn:m-1S I T ..... T
Em-2 to Em-3	T...T I Sn:mSSn:m-1SSn:m-2S I T ..... T
E3 to E2	T....T I Sn:mSSn:m-1SSn:m-2SSn:m-3_Sn:3SI T..... T
E2 to E1	T.....T I Sn:mSSn:m-1SSn:m-2SSn:m-3_Sn:3SSa:2SI T..... T
E1 to RE	T.....T I Sn:mSSn:m-1SSn:m-2SSn:m-3_Sn:3SSn:2SSn:1S I T

FIGURE 5 Mth-Level System Test Access

Acknowledge Protocol Address Ordering

The ordering of the address frames in the hierarchical acknowledge protocol is key to making the scheme work in various hierarchical arrangements. By having the selected ASP of the highest accessed environment level initiate the hierarchical acknowledge protocol, HASPs in lower environment levels only have to monitor their STDI inputs to determine when the hierarchical acknowledge protocol has been started.

Since the higher level acknowledge protocols are framed by first and second I signals, it is a simple process for a lower level HASP to determine when a higher level acknowledge protocol transmission is complete so that it can insert its own address frame in the hierarchical acknowledge protocol being relayed to the TBC. By its design, the operation of the hierarchical acknowledge protocol is simple and structured, and independent of the number of environment levels it traverses.

Hierarchical Connection Example

In Figure 6, a System TBC is connected to Subsystem HASP1 of Environment 1, HASP1/E1. HASP1/E1 is connected to Backplane HASP1 of Environment 2, HASP1/E2. HASP1/E2 is connected to Board ASP1 of Environment 3, ASP1/E3. ASP1/E3 is connected to the Application on Board 1 (ICs).

The TBC has a transmitter (XMT) to output the select protocol, a receiver (RCR) to receive the acknowledge protocol, and a controller (CTL) to regulate the operation of the transmitter and receiver, and the TMS output signal. When the controller is not using the transmitter and receiver to communicate select and acknowledge protocols, the controller can use them to communicate the 1149.1 protocol by activation of the TMS output signal.

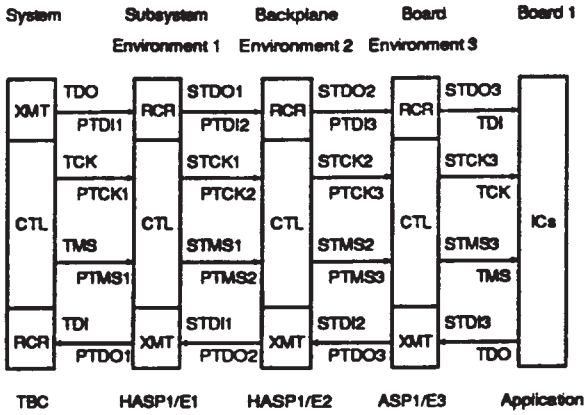

FIGURE 6 Hierarchical Connection Example

The ASPs and HASPs have a receiver (RCR) to receive the select protocol, a transmitter (XMT) to output the acknowledge protocol, and a controller (CTL) to regulate the operation of the transmitter and receiver, and the PTMS to STMS signal connection. If the receiver and transmitter are not being used to communicate select and acknowledge protocols, and if the ASP/HASP is selected, serial data and TMS control may be transferred from PTDI to STDO, from STDI to PTDO, and from PTMS to STMS during 1149.1 scan operations.

When scan access of the ICs of Board 1 is required, the TBC outputs a hierarchical select protocol to the receiver of HASP1/E1. In response, HASP1/E1's receiver strips off its address frame and relays the remaining portion of the select protocol to the receiver of HASP1/E2. In response, HASP1/E2's receiver strips off its address frame and relays the remaining portion of the select protocol to the receiver of ASP1/E3.

After the hierarchical select protocol completes, the transmitter of ASP1/E3 outputs an acknowledge protocol to the transmitter of HASP1/E2. In response, HASP1/E2's

transmitter relays the acknowledge protocol to the transmitter of HASP1/E1, inserting its own address frame before terminating the acknowledge protocol. In response, HASP1/E1's transmitter relays the acknowledge protocol to the receiver of the TBC, inserting its own address frame before terminating the acknowledge protocol.

After each HASP and ASP completes its acknowledge protocol, it connects its primary and secondary ports together. After receiving and verifying the hierarchical acknowledge protocol, the TBC's control circuit enables the transmitter, receiver, and TMS output to access the ICs of Board 1, via the connections made through HASP1/E1, HASP1/E2, and ASP1/E3, using the 1149.1 protocol.

Synchronizing 1149.1 Data Transfers

In Figure 7, an important difference is shown between the methods used by the HASPs and ASP of Figure 6 to connect their primary and secondary bus signals. In the ASP, a simple electronic switch is used to connect the primary and secondary port signals, since only a single level connection is made. However, since any number of levels may be connected using hierarchically arranged HASPs, it is important to provide a method of synchronizing the data transfer between HASP primary and secondary ports using clocked storage elements such as the D-flip flops (DFF) of Figure 7.

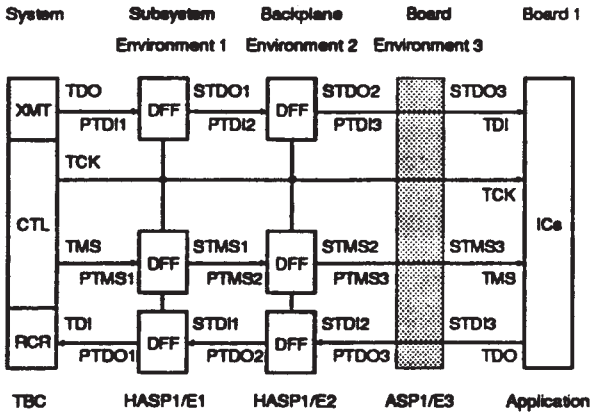

FIGURE 7 HASP Port Synchronization

If the primary and secondary bus signal connections of the HASP were made by simple electronic switches, as in the ASP, the accumulation of delays through the switches of hierarchically connected HASPs would limit the 1149.1 data transfer rate through the connection. However, with the primary and secondary HASP connections synchronized through DFFs as shown in Figure 7, delays do not accumulate as more HASPs are included in a hierarchical connection. Thus, no limitation is placed on the 1149.1 data transfer rate as more environment levels are connected between the TBC and application.

Accessing and Testing Applications

In Figure 8, a TBC is shown connected to the primary ports (PP) of ASPs 1-(n-2) via an 1149.1 test bus. Each ASP's secondary port (SP) is connected to an appropriately numbered 1149.1 application via an 1149.1 test bus. The TBC and applications illustrate the 1149.1 steady states in which shadow protocols can be transmitted to make or break connections between the TBC and applications. The darkened boxes of Figure 8, indicate reserved addresses "0", "n-1" and "n", which are not used as application addresses.

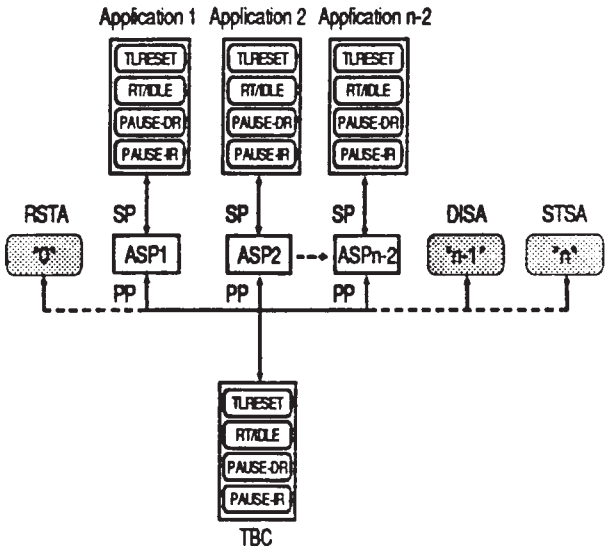

FIGURE 8 Single Level Test Access Example

The "0" address is a reset address (RSTA) recognizable by all ASPs and HASPs. ASPs and HASPs receiving the RSTA address disconnect their ports and force their applications into the TLRST state by setting their STMS output to a logic one. The "n-1" address is a disconnect address (DISA) recognizable by all ASPs and HASPs. ASPs and HASPs receiving the DISA address

disconnect their ports and remain in the 1149.1 steady state they were in when the disconnect occurs. The "n" address is a self test synchronization address (STSA) recognizable by all ASPs and HASPs. ASPs and HASPs receiving the STSA address while in the 1149.1 PAUSE-IR/DR states connect PTMS to STMS and disconnect PTDI and STDO, and STDI and PTDO.

Executing Single Application Self Tests

If Application 1 needs to be tested using a BIST based test instruction, like the 1149.1 RunBist instruction [1], and is currently in the TLRST state, the TBC moves itself to the TLRST state to synchronize states with Application 1, then executes a shadow protocol to connect to Application 1. After a connection is made between the TBC and Application 1, the TBC transitions itself and Application 1 from TLRST to the RT/IDLE state to prepare for testing.

After entering the RT/IDLE state, the TBC executes an 1149.1 instruction scan operation to load the test instruction, then returns to the RT/IDLE state. In the RT/IDLE state the test instruction starts and executes until it terminates either on its own (like RunBist) or in response to the TBC transitioning from the RT/IDLE state. While the test executes, the TBC is free to synchronize, connect, and start tests in other applications if desired. At the end of the test, the TBC synchronizes and connects to Application 1 (if it has disconnected) and accesses the test results using 1149.1 scan operations.

Executing Parallel Application Self Tests

If all applications of Figure 8 need to be tested in parallel using a BIST based test instruction, and all are currently in the TLRST state, the TBC moves itself to the TLRST state and executes the following sequence.

The TBC transmits a shadow protocol to connect Application 1 to the TBC. After the connection is made, the TBC transitions itself and Application 1 from TLRST to the RT/IDLE state. After entering the RT/IDLE state, the TBC executes an instruction scan operation to load the test instruction into Application 1 then terminates the instruction scan operation in the PAUSE-IR state. The TBC then transmits a shadow protocol containing the DISA address to disconnect Application 1 in the PAUSE-IR state. The TBC then transitions itself from PAUSE-IR to the TLRST state to synchronize 1149.1 states with the next application to be connected. These steps are repeated on Applications 2 through n-3.

After Application n-3 has been setup as previously described, the TBC transmits a shadow protocol to connect Application n-2 to the TBC. After the connection is made, the TBC transitions itself and Application n-2 from TLRST to the RT/IDLE state. After entering the RT/IDLE state, the TBC executes an instruction scan operation to load the test instruction into Application n-2 then terminates the instruction scan operation in the PAUSE-IR state. While in PAUSE-IR, the TBC transmits a shadow protocol containing the STSA address. All ASPs currently in the PAUSE-IR state respond to the STSA address to connect their PTMS and STMS signals together, enabling the TBC's TMS output to be input to each ASP application. After making this global TMS connection, the TBC transitions itself and all applications from PAUSE-IR to RT/IDLE to start the parallel test operation.

The parallel test operation continues until all tests have terminated either on their own or in response to the TBC transitioning from the RT/IDLE state. At the end of the parallel test operation, the TBC executes shadow protocols to connect to each application, one at a time, to access the test results using 1149.1 scan operations. If one or more of the ASPs in Figure 8 had not been positioned in the PAUSE-IR state when the STSA address was issued, they would have ignored the STSA address and remained in their present state. Thus, selective execution of parallel test operations is provided.

Executing Application Interconnection Tests

Testing the functional interconnection between applications is easy if each application includes 1149.1 boundary scan on its functional I/O. The test is setup by the TBC synchronizing states with and connecting to each application, one at a time. After the TBC is connected to an application it performs an 1149.1 instruction scan operation to load the 1149.1 Sample/Preload instruction. After loading the Sample/Preload instruction, the TBC performs a data scan operation to load a safe initial boundary I/O test pattern. Next,

the TBC performs an instruction scan operation to load the 1149.1 Extest instruction, terminating the instruction scan operation in the PAUSE-IR state. While in the PAUSE-IR state, the TBC executes a shadow protocol containing the DISA address to disconnect from the application so that other applications can be connected and setup as described. Since the application is disconnected in PAUSE-IR, the Extest instruction has not been updated to take effect.

After all applications have been setup as described, the TBC positions itself in the PAUSE-IR state, then transmits a shadow protocol containing the STSA address. In response to the STSA address, all applications are connected to the TBC's TMS output, via their ASPs. The TBC then transitions itself and all applications from PAUSE-IR to RT/IDLE to cause all applications to simultaneously update their Extest instructions and enter test mode. Since each application was loaded with a safe initial boundary I/O test pattern, no I/O conflicts occur between applications when the Extest mode is entered.

Following this setup procedure, interconnection testing is accomplished by the TBC selectively connecting to each application, one at a time and while in the RT/IDLE state, to set boundary outputs and read boundary inputs using 1149.1 data scan operations.

While this example used the Extest instruction to scan test the interconnections between applications, at-speed interconnection tests based on BIST (i.e. pseudorandom pattern generation/data compaction) could be used as well. If BIST interconnection testing is performed the applications are setup and enabled for testing as described in the "Executing Parallel Application Self Test" section.

Hierarchical Access and Testing

In the example of Figure 8, the TBC was shown directly connected to the ASP of each application. In actual systems however, the TBC may not always be directly connectible to the ASP of each application due to the placement of the applications in the system hierarchy. Therefore HASPs may reside between the TBC and ASPs to provide a hierarchical connection between the TBC and applications, as shown in Figure 9.

The access and testing procedures previously described in regard to the single level connection arrangement shown in Figure 8 still apply to the three level connection arrangement shown in Figure 9. The only difference in Figure 9 is that a three level connection is traversed to access and test the applications.

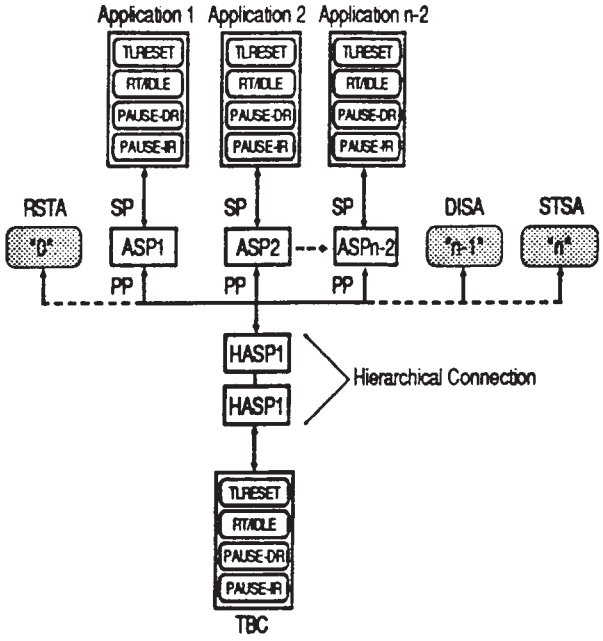

FIGURE 9 Three Level Test Access Example

Hierarchically Sending RSTA Addresses

After hierarchically accessing and testing applications within a system, numerous applications may be left in various 1149.1 steady states and test modes. To insure the stability of the system in its mission mode, all applications need to be placed in the TLRST state following test.

One way of placing system applications in the TLRST state is for the TBC to individually connect and set each application into the TLRST state. However, a faster way of placing system applications in TLRST is for the TBC to globally transmit a hierarchical select protocol consisting of multiple RSTA address frames. Alternately, the TBC can locally place applications existing at and above a selected environment level in TLRST by sending actual address frames in a hierarchical select protocol prior to sending the RSTA frames. The actual address frames direct the reset action to only the applications at and above the selected environment level.

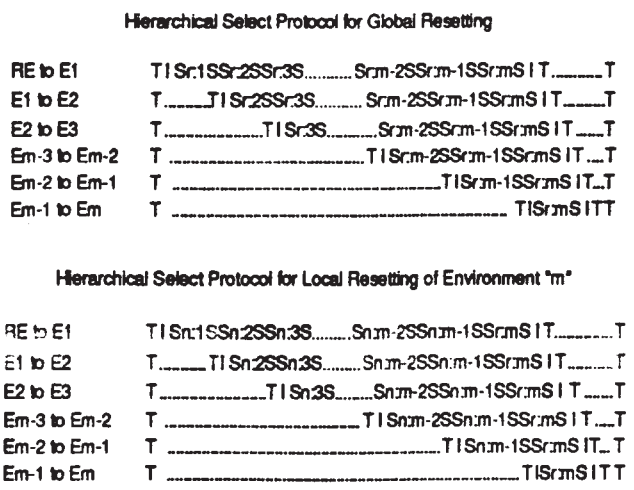

FIGURE 11 Hierarchically Resetting Applications

Examples of global and local resetting in a system containing "m" environment levels are shown in Figure 11. The "r" in the address frames 1 through m indicate the RSTA address. The "n" in the address frames 1 through m indicate an actual address. All HASPs in each intermediate environment level (1 through m-1) between the root environment (RE) and environment "m" wait to respond to the RSTA address until after the hierarchical select protocol terminates, so that all relayed RSTA address frames can be transmitted to the highest environment level (m). HASPs and ASPs receiving a RSTA address do not execute an acknowledge protocol. However, HASPs that receive an actual address do respond with an acknowledge protocol.

Hierarchically Sending DISA/STSA Addresses

While Figure 11 illustrates RSTA address frames "r" being globally and locally transmitted to connections and applications, DISA and STSA address frames can be globally or locally transmitted as well. DISA address frames are transmitted in place of the "r" addresses in the hierarchical select protocols to disconnect connections and applications. STSA frames are transmitted in place of the "r" addresses in the hierarchical select protocols to synchronize selected connections and application groups to the TBC's TMS output. As with the RSTA address frame, HASPs and ASPs receiving DISA or STSA address frames do not execute an acknowledge protocol.

Conclusion

This paper has described a hierarchical test access problem the industry will eventually face in system architectures, and a proposed solution to this problem. The solution is based on a connection circuit and protocol that can operate in any hierarchical arrangement to provide access to and test control of 1149.1 applications in a system environment. This access method provides a simple way to effectuate expanded use of the 1149.1 test bus in systems. The advantages of this approach are: (1) connection protocol is supportable by most 1149.1 TBCs/ATEs, (2) connection protocol rides on top of existing 1149.1 bus wiring, (3) does not require protocol translation in the connection path, (4) does not impact 1149.1 test bus bandwidth, (5) supports parallel 1149.1 test operations, (6) supports any type of hierarchical connection arrangement, and (7) simple, cost-effective implementation.

References

1. IEEE Std 1149.1-1990, Standard Test Access Port and Boundary Scan Architecture

2. IEEE Standard Proposal 1149.5, A Standard Module Test and Maintenance Bus.

3. L. Whetsel, "JTAG Compatible Devices Simplify Board Level Design For Testability", IEEE Wescon Convention, November 1989.

4. N. Jarwala and C. Yau, "The Boundary-Scan Master: Target Applications and Functional Requirements", IEEE International Test Conference, September 1990.

5. D. Bhavsar, "An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes", IEEE International Test Conference, October 1991.

6. L. Whetsel, "A Proposed Method of Accessing 1149.1 in a Backplane Environment", IEEE International Test Conference, September 1992.

7. D. Landis, "Applications of the IEEE 1149.5 Module Test and Maintenance Bus", IEEE International Test Conference, September 1992.

8. D. Ohnesorge, "A Position Independent Scan Architecture", IEEE European Design for Test Workshop, 1992.