MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this document
by AN1500/D

AN1500

IEEE 1149.1 Boundary Scan for H4EPlus™ Arrays

Prepared by: Roy Jones and Nick Spence
Edited by: Clarence Nakata and Tim Hunkler
Application Specific Integrated Circuits Division, Chandler AZ

Table of Contents **Page**

1. Introduction

This application note describes how IEEE standard boundary scan, commonly referred to as “JTAG,” has been implemented on Motorola’s H4EPlus family of sub-micron CMOS gate arrays. The user is assumed to have a working knowledge of JTAG boundary scan. For background information refer to the IEEE specification entitled “Standard Test Access Port and Boundary-Scan Architecture, IEEE Std. 1149.1-1990,” and to the textbook entitled “The Test Access Port and Boundary-Scan Architecture” by Maunder and Tulloss, published by the IEEE Computer Society Press.

Section 2.0 describes the macros which have been added to the H4EPlus library to facilitate designing boundary scan circuitry into an H4EPlus gate array.

Section 3.0 describes how the JTAG clock and control signals are distributed around the chip periphery to each pin’s boundary scan cell (BSC). The design constraints associated with the distribution of these signals are also described.

Section 4.0 describes how to add boundary scan to a chip whose system logic has been designed using conventional scan techniques. An ATPG (Automatic Test Pattern Generation) tool is then used to test the JTAG circuitry as well as the system scan circuitry.

Section 5.0 presents an example JTAG circuit and describes the process the designer must go through to establish the chip pin-out. The constraints described in Section 3.0 must be taken into consideration.

Section 6.0 describes the CAD design flows used when designing an H4EPlus array which incorporates JTAG boundary scan.

Appendix A lists the MARV (MAnufacturing Rules Verifier) rules that are specific to JTAG circuitry. The majority of these rules are associated with the design constraints described in Section 3.

Appendix B provides background information relevant to the “ATPG-Compatible JTAG” discussed in Section 4.

2. H4EPlus JTAG Macro Descriptions

Technical data, including logic diagrams, for all JTAG macros in the H4EPlus library can be found in the H4EPlus Series Design Reference Guide. These macros have been placed in three categories in the descriptions that follow: I/O macros, core macros and special purpose macros.

2.1 I/O Macros

I/O macros include the input, output and bidirectional boundary scan cells. The JTAG boundary scan logic associated with these macros is diffused into the peripheral I/O sites. The JTAG logic in a given I/O site is used only if a JTAG BSC macro is instantiated at the package pin bonded to that I/O site.

2.2 Core Macros

Macros residing in the core of the array include the Bypass Register (BPREG), Device Identification Register (IDREG), Instruction Register (MC_IREG4), and TAP Controller (FMC_TAPCB).

The IDREG and BPREG are hard macros in the H4EPlus library. Motorola assigns JTAG device identification codes according to the following format:

```
Bit #:   31-28 27---22  21----------12 11----------0
Value:   VVVV 000111  DDDDDDDDDD000000011101
```

where
Bits 31-28: version number assigned by Motorola ASIC
Bits 27-22: unique number assigned to Motorola ASIC
Bits 21-12: sequence number assigned by Motorola ASIC
Bits 11-0: unique number assigned to Motorola Inc.

The MC_IREG4 is a soft macro; it consists of a schematic capture symbol (or Verilog HDL module) which is comprised of individual gate and flip-flop hard macros, which are placed and routed individually. There is no fixed layout for the MC_IREG4 as an entity. A functional diagram of the MC_IREG4 is shown in Section 4.3, Figure 4-2.

The FMC_TAPCB is a firm macro; it is like a soft macro in that it is comprised of, and modeled as, individual gate and flip-flop hard macros. Unlike a soft macro, a firm macro such as the FMC_TAPCB has been placed and routed as a single entity. Consequently, both the internal metal interconnect and timing of a firm macro are fixed and do not change when the chip is laid out. A functional diagram of the FMC_TAPCB is shown in Figure B-3 in Appendix B, which discusses this macro in detail.

2.3 Special Purpose Macros

2.3.1 TAP macros: TCK, TMS, TRSTB, TDI and TDOUT.

TCK, TMS, TRSTB and TDI are simply input buffers with no BSC logic. Each must be used at the pin driven by the JTAG signal of the same name.

The TDOUT macro is a tristatable output driver and a input buffer. The output driver is controlled from the core logic and used to terminate the JTAG scan chain, while the input buffer returns the TDI/TDO boundary scan chain from the periphery into the core through the 'TDOC' port.

A diagram illustrating the use of the TDOUT macro is shown in Figure 2-1. To be compliant with the IEEE1149.1 specification the customer should add the mux and flip-flop shown. The 'EN' signal comes from the TAP controller and goes to the 'EN' port of the TDOUT macro. The 'SL' signal from the TAP controller is used to select the scan chain between either the 'IR' signal from the TDO port of the Instruction Register or the 'DR' signal from whichever JTAG data register is activated by the current JTAG instruction (The output of the Data Register Mux). The resulting signal is registered on the rising edge of TCKB (falling edge of TCK) to avoid timing problems, and then output through the 'DO' port of the TDOUT macro.

The 'TDOC' port will be connected to the TDI port of the first ENSCANI macro if there is one, otherwise to the appropriate input of the Data Register Mux (see Figure 4-4).

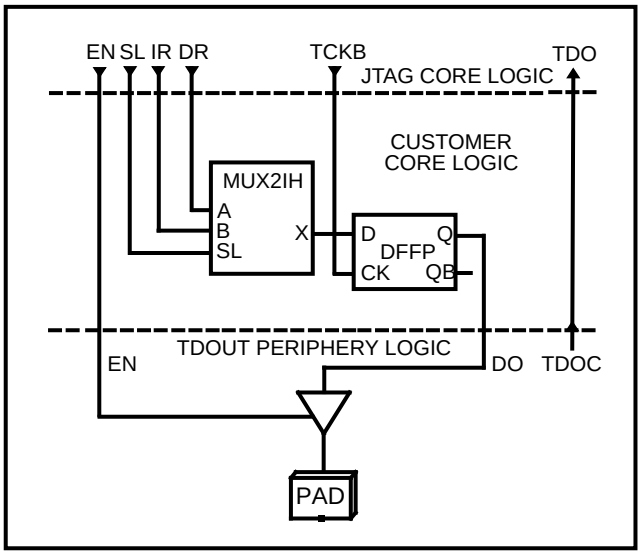

Figure 2-1 TDOUT Macro

2.3.2 ENSCANJ/I

The ENSCAN BSC's are used to drive the enable port of 3-state bidirectional and output BSC's. The ENSCANJ and ENSCANI are functionally identical. The difference is that the ENSCANJ must reside in the I/O area on an unused I/O site or a power/ ground site. The ENSCANI macro must reside in the array core.

In the example of Figure 2-2 an ENSCANJ supplies the 3-state enable to an output bus. The BSEN input is driven from the core by the system 3-state enable signal, and the OEN output feeds into the core where it can be buffered if necessary before driving the EN inputs of the 3-state output buffers.

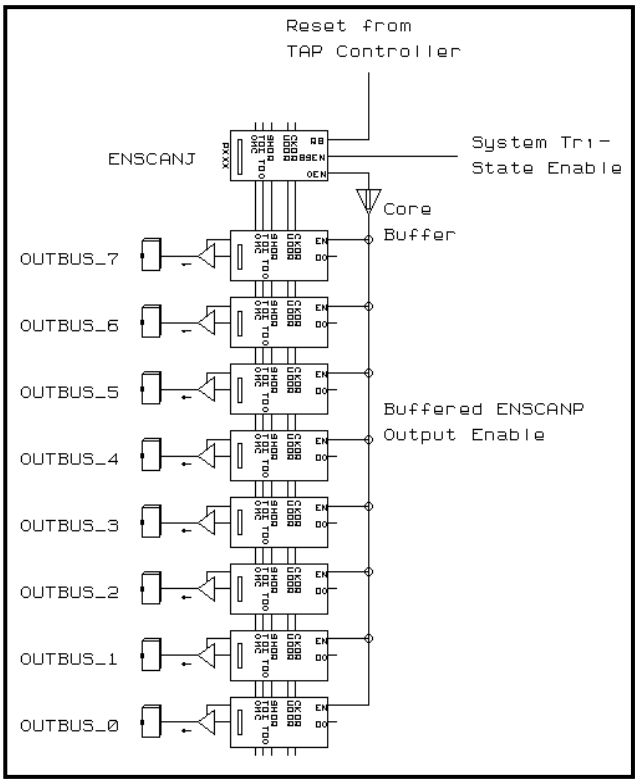

Figure 2-2 ENSCANJ Driving 3-State Enable of 8-bit Output Bus

If an ENSCANI had been used instead, it would go at the end of the BSC scan chain closest to TDO as shown in Figure 2-3. The BSC scan data path enters the core through a port on the TDOUT macro, passes through all ENSCANI's, then gets multiplexed with the scan paths from all other JTAG data registers before passing to the DR signal shown in Figure 2-1 on its way off chip. **(Note that test data must shift counterclockwise through the BSC's around the periphery of the chip.)** ENSCANI's in the core receive CKDR, SHDR, UDDR, and OMC directly from the core signals which drive the inputs to the peripheral CKDRMID, SHDR, UDDR, OMCDR, and OMCDR buffers, respectively.

It is recommended that ENSCANI's be used only if there are no power sites or unused I/O sites available on which to place ENSCANJ's.

2.3.3 TDBUF

If consecutive peripheral BSC's are separated by more than seven I/O sites, a TDBUF buffer macro must be inserted between them since a BSC's TDO output has limited drive strength. The TDBUF must not be more than seven I/O sites away from the BSC driving it.

2.3.4 I/O BSC Control Signal Buffers

These are the buffers that distribute CKDR, SHDR, UDDR, IMC and OMC to the peripheral BSC's. These buffers are described in detail in Section 3.0, "JTAG Clock & Control Signal Distribution."

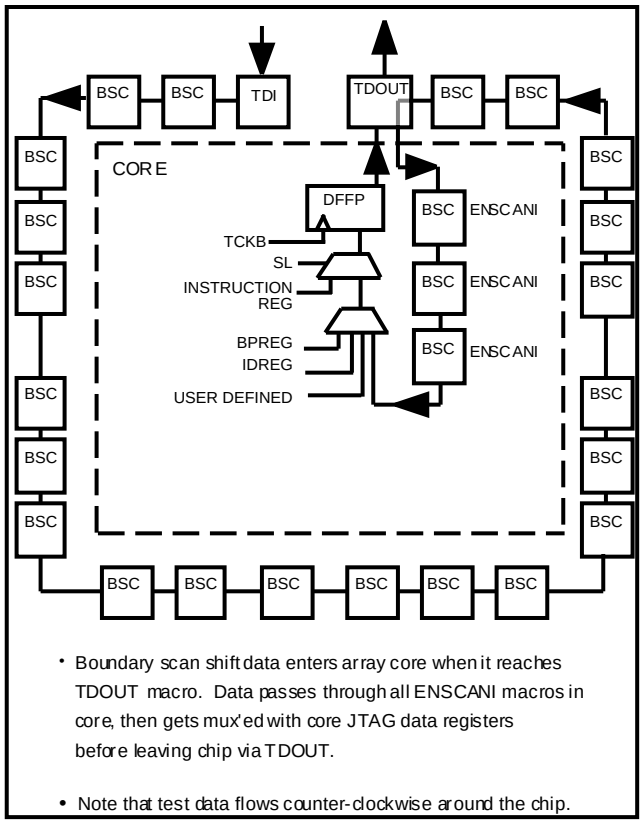

Figure 2-3 Position of ENSCANI 3-State Enable Macros within I/O Boundary Scan Register

3. JTAG Clock & Control Signal Distribution

3.1 Overview

On Motorola's sub-micron H4EPlus arrays the JTAG boundary scan cells are diffused into the periphery, or I/O area, of the chip. The advantages realized, as compared to implementing the BSC's with core macros, are as follows:

1. Area savings
   a. 100% utilization in the periphery versus 60-70% in the core (i. e., no unused gates in the periphery).
   b. transistor sizes can be optimized to their small, non-varying loads
   c. interconnect between BSC's is done by abutment, minimizing wire length. For these reasons, diffusing the BSC's into the I/O area conserves a significant amount of chip area compared to implementing the BSC's in the array core, even though the chip I/O area is ~20% larger than it would be if it did not include built-in JTAG logic.
2. There is less additional data path delay due to the mux in the input BSC's since it has been optimized in terms of transistor size and minimum wire interconnect.
3. There are no distribution "trees" for the BSC control signals to increase routing congestion in the core.
4. RAM and MPU diffused blocks don't interfere with the peripheral distribution "rings" for these control signals, preventing increased signal skew.
5. Hold time violations should not occur when shifting the boundary scan register since the clock nets to all BSC's have a fixed routing.

The five JTAG clock and control signals are CKDR, SHDR, UDDR, IMC and OMC. A method is provided for distributing these signals to the boundary scan cells located in the periphery of H4EPlus arrays.

In Figures 3-1 to 3-3, a dotted line marks the boundary between the core and periphery of the array. All of the special buffers for the JTAG clock and control signals reside in power sites or unused I/O sites in the periphery in order to:

1. maximize the drive capability of these buffers by utilizing the large transistors that normally drive off-chip, and to:
2. facilitate optimum buffer placement to achieve:
   a. minimum insertion delay for each signal,
   b. minimum skew for each signal between a buffer's nearest and farthest BSC loads, and
   c. minimum SHDR-to-CKDR skew and CKDR-to-UDDR skew at any given BSC.

For packages with highly inductive leads HSPICE simulations have shown large voltage spikes on OUTVDD and OUTVSS (the output driver power and ground buses) due to simultaneously switching outputs (SSO). These spikes can couple to the outputs of "quiet" (inactive) drivers. For this reason the JTAG buffers are powered from INPVDD/INPVSS (the core power and ground buses), since they drive BSC's which are also powered from INPVDD/INPVSS. These buffers are also slew-rate controlled in order to inject as little switching noise as possible onto INPVDD and INPVSS. The JTAG buffers are all roughly equivalent to an ON4S4 output buffer.

3.2 CKDR Distribution

Because the CKDR ring must encircle the entire chip, the resistance of the metal can be several hundred ohms. The same is true of the control lines as well. As a result, one very large buffer cannot drive the ring without suffering severe performance loss in terms of long prop delays and edge-rates at the more distant BSC's. A much better approach is to use multiple, distributed buffers to drive the ring. As shown in Figure 3-1, the TAP controller drives a CKDRMID buffer, which in turn drives one CKDRCC1 and one CKDRCC2 buffer via an "extra" ring. The CKDRCC1 and CKDRCC2 each drive roughly half of the JTAG I/O cells on the chip via the CKDR ring. Because these two buffers are placed diametrically opposite to each other, only half of the extra ring is needed to distribute CKDR to them. By detaching the unneeded half of

the extra ring, metal capacitance on this net is greatly reduced and substantial speed improvement is realized. There is a physical cut in the extra ring within the CKDRCC1 and CKDRCC2 macros such that detachment of the unneeded half of the extra ring is accomplished automatically when these macros are placed.

There is a physical cut in the CKDR ring within the TDOUT macro and another within the ISOR macro. This is shown in Figure 3-1, CKDRCC1 drives all BSC's on CKDRNET1 (macros placed between TDI and ISOR) and CKDRCC2 drives all BSC's on CKDRNET2 (macros placed between ISOR and TDOUT).

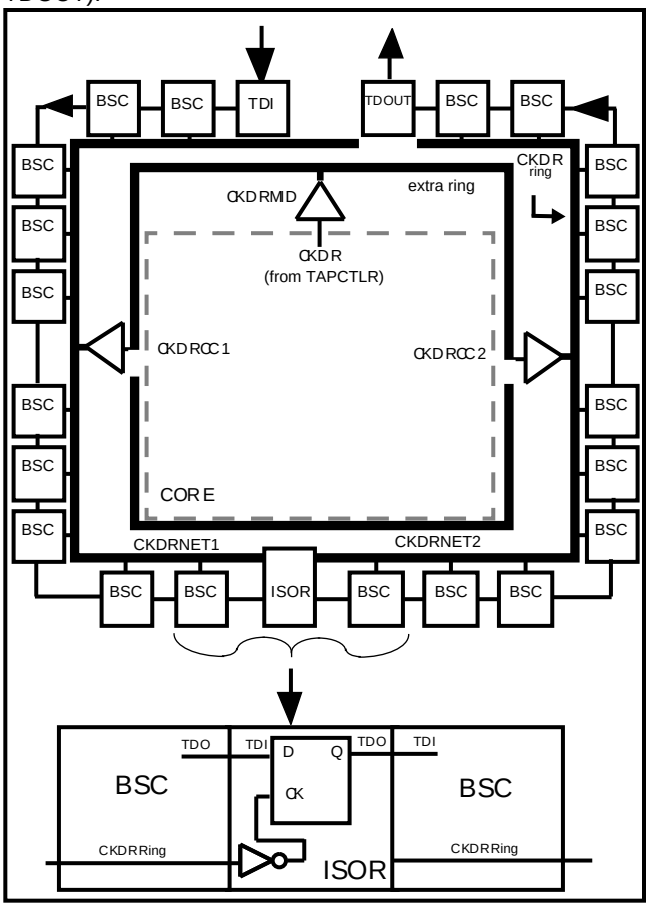

Figure 3-1 CKDR Distribution

The ISOR macro registers the TDI/TDO signal on the falling edge of the clock to prevent setup and hold time errors between the two halves of the Boundary Scan Chain.

CKDR, SHDR, UDDR and OMC are routed within the core directly from the TAP Controller to ENSCANI bidirectional enable BSC's, which reside in the core.

3.2.1 SHDR, UDDR Distribution

SHDR and UDDR use a distribution scheme which is different from the CKDR scheme, which was able to make use of the extra ring. SHDR will be used for illustration (see Figure 3-2).

The TAP controller drives two SHDR buffers, each of which drives roughly half of the JTAG I/O cells on the chip via the SHDR ring. The SHDR and UDDR rings are each split into two halves in the same way as the CKDR ring, with cuts inside the TDOUT and ISOR macro.

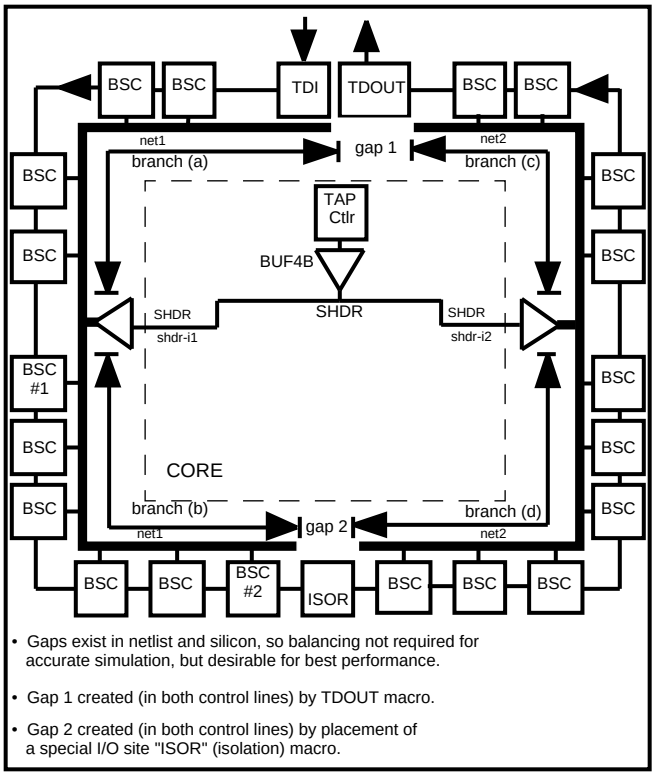

Figure 3-2 SHDR, UDDR Distribution

3.2.2 IMC, OMC Distribution

The distribution scheme for IMC and OMC is similar to the scheme for SHDR. The difference is that the SHDR line is cut by the TDOUT and ISOR macros, whereas the IMC and OMC lines are cut by the CKDRCC1 and CKDRCC2 macros. The IMCDR and OMCDR buffers can now be placed in a different area from the SHDR and UDDR buffers (see Figure 3-3), relieving buffer congestion so that as many JTAG buffers as possible can be placed on power sites, allowing more efficient use of the I/O sites.

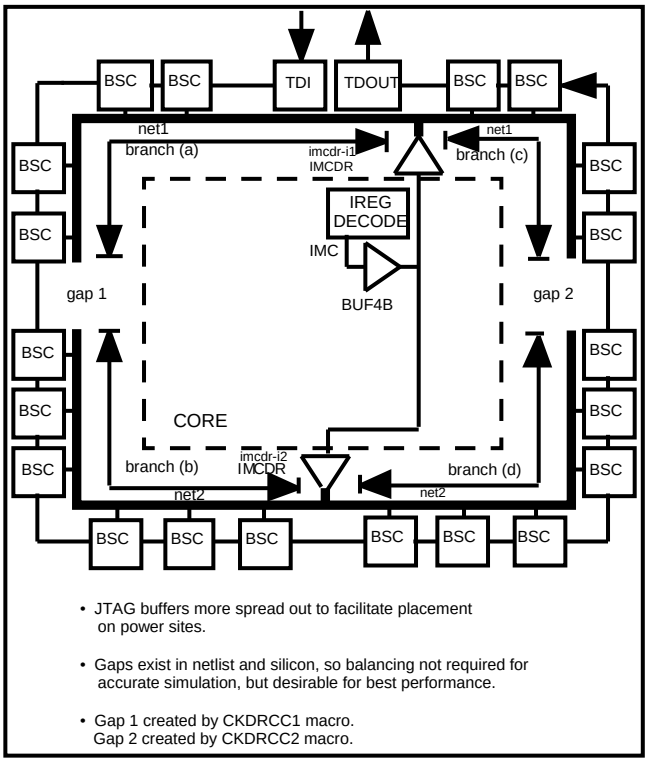

Figure 3-3 IMC, OMC Distribution

3.3 JTAG I/O signal timing

The routing of the JTAG signals is the I/O is fixed by the layout of the option, so the parasitics and timing can be determined before the design has been placed or routed.

The customer must use PREDIX to compute the actual RC's for the peripheral nets. This can be done any time after the pinout has been assigned, but should be done before serious timing analysis is performed. The JTAG parasitics are determined by PREDIX and stored in the predix.jtagrc file. DECAL merges the PREDIX peripheral RC's with either DECAL-estimated RC's for the array core (for pre-predix simulations), PREDIX-estimated RC's for the array core (for post predix simulations) or with Gate Ensemble-generated actual RC's for the array core (for post-layout simulations).

Simulation or timing analysis using PREDIX RC's now will show if enough skew exists between CKDRNET1 and CKDRNET2 to cause timing violations when shifting data through the boundary scan chain.

4. ATPG-Compatible JTAG

4.1 Introduction

On the Motorola H4EPlus family of CMOS arrays, JTAG boundary scan circuitry has been designed to be compatible with many popular ATPG tools. However, as defined by the IEEE 1149.1 specification, JTAG boundary scan violates several conventional scan design rules. Section 4.0 and Appendix B describe how the JTAG boundary scan circuitry has been implemented on H4EPlus arrays in order to allow it to be tested with patterns generated by ATPG methods.

4.2 Design Overview

Motorola has designed scan-compatibility into the JTAG circuitry by modifying the TAP Controller to include scannable flip-flops and by making other changes to satisfy some common ATPG design rules. However, the user is still responsible for:

- ensuring that no timing problems arise due to clock skews, and
- interconnecting all JTAG circuitry such that ATPG DRC rules are satisfied.

4.2.1 Handling of Clock Skew

Motorola's H4EPlus arrays use the gated-clock JTAG implementation shown in IEEE 1149.1. That is, the original clock TCK is gated within the TAP Controller and the instruction decoding logic to provide the CKIR, CKDR, UDIR and UDDR signals to the JTAG cells. This can cause skew problems in ATPG test mode, particularly due to the large clock delays to the boundary scan cells. Section 4.3 addresses the prevention of timing problems due to clock skew.

4.2.2 JTAG Circuitry Interconnection

An extra pin, called "MTST" for "Motorola Test Mode", must be added to the chip to logically reconfigure the JTAG circuitry when ATPG testing is to be done. When this input is active all circuit elements including the TAP Controller will become scan-compatible, the JTAG TMS pin will be used as the scan/shift enable control signal, and the scan chain connected between the TDI and TDO pins will contain all the flip-flops which are part of the JTAG circuitry. The JTAG logic must also be correctly controlled in ATPG scan mode to ensure that the scan paths are completed (requires that IMC and OMC both be low). Section 4.4 describes in detail how to properly hook-up all JTAG circuitry for ATPG compatibility.

4.3 Handling of Clock Skew

The JTAG design can suffer from clock skew problems during ATPG test mode because the gated clocks (CKIR, UDIR, CKDR's for each data register) must be enabled at the same time. The problems occur when the clock arrives early to one flip-flop causing its output to change before the clock arrives at the next flip-flop. This can result in hold time violations and/or the wrong data being loaded.

The JTAG logic does not suffer from this problem during normal operation because some sections are clocked on the rising edge of TCK while others are clocked on the falling edge, such that input data to each JTAG register always changes on the inactive edge of the clock to that register. For

example, a new instruction becomes active when the shadow latches in the instruction register are “clocked” by UDIR, which occurs on the falling edge of TCK. The new instruction drives decode logic which enables CKDR to the appropriate data register (e. g. the Identification Register, Bypass Register or peripheral boundary scan register). These CKDR enable signals change on the falling edge of TCK in order to be stable during the rising edge of TCK/CKDR. Likewise the SHDR and TDI signals, which feed each data register, change on the falling edge of TCK in order to be stable during the rising edge of TCK/CKDR. Also, the flip-flop within the TDO macro is clocked on the falling edge of TCK because its input data, which comes from either the “CKIR” flops in the Instruction Register or from one of the data registers, changes on the rising edge of TCK.

As described above, during normal JTAG operation input data to each JTAG register always changes on the inactive edge of the clock to that register, so that the data is stable during clocking of the state elements. However, when operating in ATPG test mode all flip-flops should clock on the same edge of the clock signal. This can cause problems both during scan operation (the shift in and shift out of scan data) and also during the pulsing of the system or TCK clocks. (An ATPG scan test consists of three parts: shift in of scan stimulus, pulsing of zero or one of the clocks or asynchronous sets/resets -- referred to as “clock pulse mode,” and shift out of the chip’s response to the scan stimulus.)

As described in Section 4.4, all JTAG registers will be included in the same scan chain during ATPG testing. Each data register is clocked by its own gated version of CKDR, and the Instruction Register’s CKIR and UDIR flops are clocked by CKIR and UDIR respectively. Consequently, the JTAG scan chain is operated by several different clocks which have different insertion delays, creating the potential for skew problems during the scan operation. Skew problems during scanning can be controlled by putting registers whose clocks have longer insertion delays at the beginning of the scan chain. Since the I/O boundary scan register has the slowest clock distribution it should be the first part of the scan chain. Also, if timing analysis shows it to be necessary, delays can be added between flip-flops driven by different clocks.

In order to prevent skew problems when ATPG pulses the system or TCK clocks, the JTAG clock gating must be altered so that the clock pulse is applied either to the flip-flops that would normally change on the rising edge of TCK or to the flip-flops that would normally change on the falling edge of TCK. This is done by adding a flop to the TAP controller which controls the clock gating only during ATPG clock pulse mode. (This flop is labeled “TCK/TCKB Select Flop” in Figure B-3 of Appendix B.) Putting this flop inside the TAP Controller allows all of the JTAG logic within the TAP controller to remain in a single scan chain using a single clock edge.

As shown in Figure 4-1 the Instruction Register UDIR latches have been changed to flops, for the following reasons:

1. to eliminate the undetectable stuck-at-one faults which are present on each latch gate input (for more details see Section B.1 of Appendix B),
2. to enable separate clocking of the UDIR flop, which must change on the falling edge of TCK.

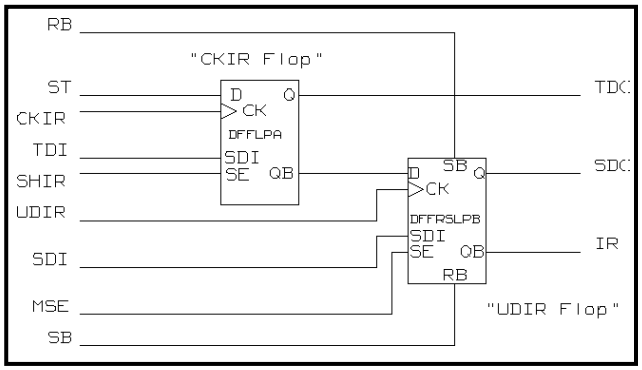

Figure 4-1 Single-Bit Instruction Register Cell (MC_IREG).

The “MC_IREG4” four-bit Instruction Register macro is shown in Figure 4-2. During scan mode CKIR and UDIR are active simultaneously, with CKIR leading UDIR by a few nanoseconds at the output of the TAP Controller. Consequently, to prevent hold-time violations during scan mode, the “UDIR flops” must precede the “CKIR flops” in the JTAG scan chain. For this purpose, SDI and SDO ports are provided on the MC_IREG and MC_IREG4 macros to serve as “Scan-Data-In” and “Scan-Data-Out” for the UDIR flops (see Figures 4-1, 4-2 and 4-3). As shown in Figure 4-3, during ATPG scan mode (MSE high) scan data from the TAP Controller’s “TDO” port enters the UDIR flops in the MC_IREG4 via the SDI port. After exiting at SDO, the scan data is fed back to the MC_IREG4 “TDI” port to pass through the CKIR flops. During normal JTAG operation (MSE low), JTAG test data from the TAP TDI pin is passed to the CKIR flops in the MC_IREG4 as required.

Extra delays are included in the scan paths within the MC_IREG4 because it is currently a soft macro. As such, the metal interconnect between its four MC_IREG cells will differ somewhat from layout to layout, potentially causing a small amount of CKIR or UDIR skew between the four cells. The DLY8 macros within the MC_IREG4 add delay in the scan path to compensate for any such skew.

Referring to the ATPG-compatible TAP Controller in Appendix B, Figure B-3, note the inclusion of the following circuitry:

1. A “TCK/TCKB Select Flop” to control the clock gating in ATPG clock pulse mode.
2. An extra delay on TDO to prevent skew problems caused by the early clocking of the flip-flops in the TAP Controller. The TDO signal would normally be passed onto the Instruction Register as shown in Section 4.4. The Instruction Register clock CKIR will arrive later than the clock to the flip-flops in the TAP Controller because of the clock gating circuitry within the TAP Controller.
3. A delay macro in the scan path between the top left flip flop and the top right flip-flop and on the “TCK/TCKB Select Flop” because these flops have separate clocks which pass through different gating logic.

Since the FMC_TAPCB ATPG-compatible TAP Controller is a firm macro, its fixed layout guarantees no timing problems will ever arise internal to the FMC_TAPCB.

It is very difficult to control clock skew between core/system flip-flops and the boundary scan cells because of the long insertion delay of clock CKDR in the periphery. In order to prevent skew problems from occurring between the JTAG logic and the system logic during ATPG test mode, either:

1. the clock TCK should be different from the system clock, or
2. circuitry similar to the “TCK/TCKB Select Flop” and “Clock Select” gates in the TAP Controller should be implemented to prevent ATPG tools from pulsing both TCK and the system clock in the same clock cycle, or
3. the ATPG tool should be setup to avoid asserting multiple clocks any test cycle, except during scan chain shifting.

It is important that timing analysis be done to verify that no setup or hold time violations occur due to the aforementioned sources of clock skew. Motive and Veritime are able to take into account the effects of variations in process/voltage/temperature across a chip.

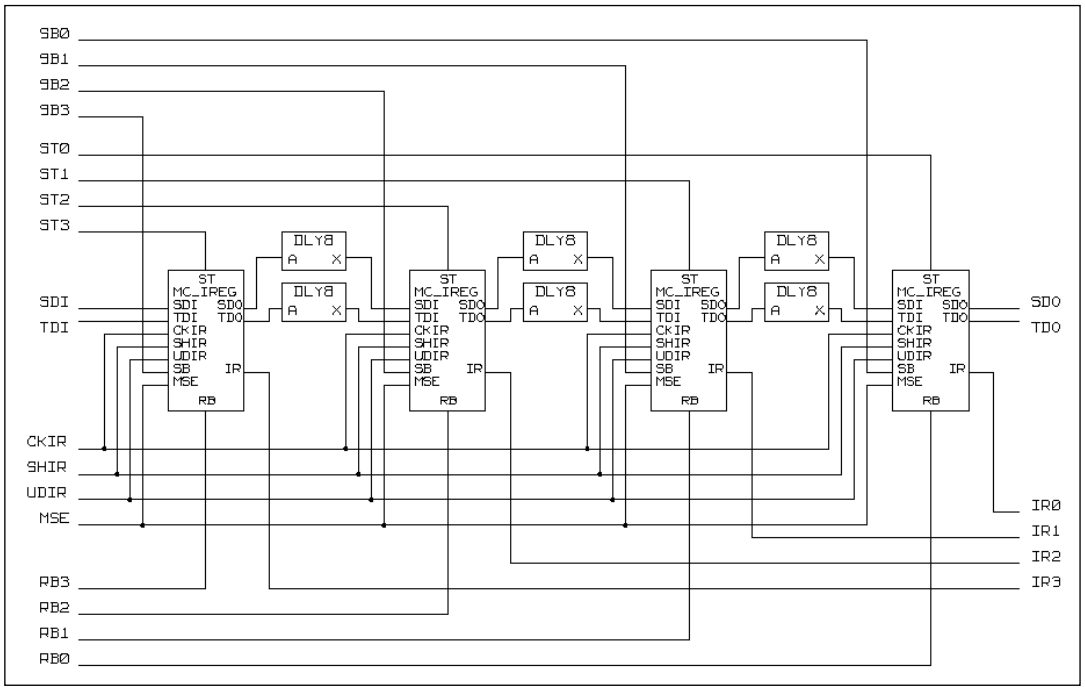

Figure 4-2 MC_IREG4 Functional Diagram

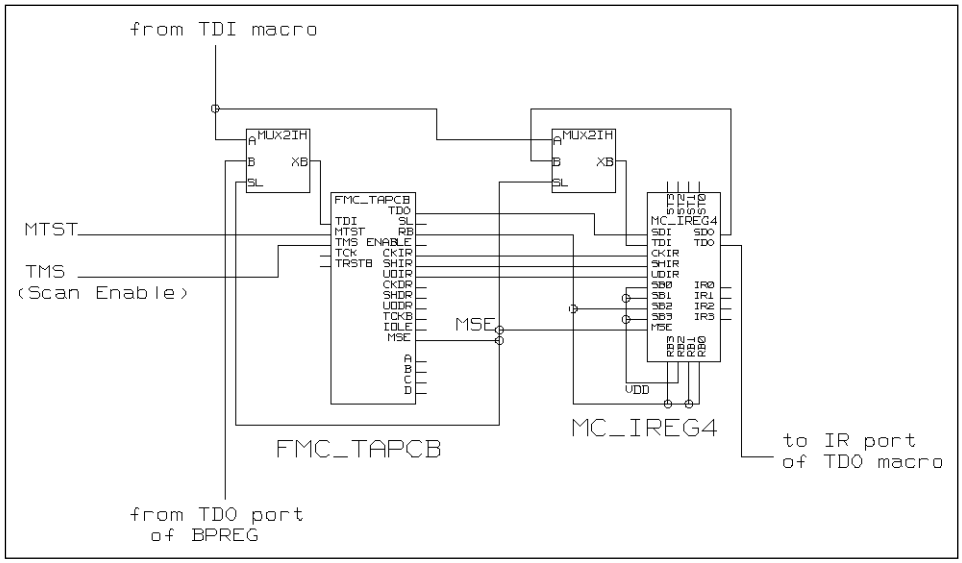

Figure 4-3 Scan Chain Hook-up of MC_IREG4

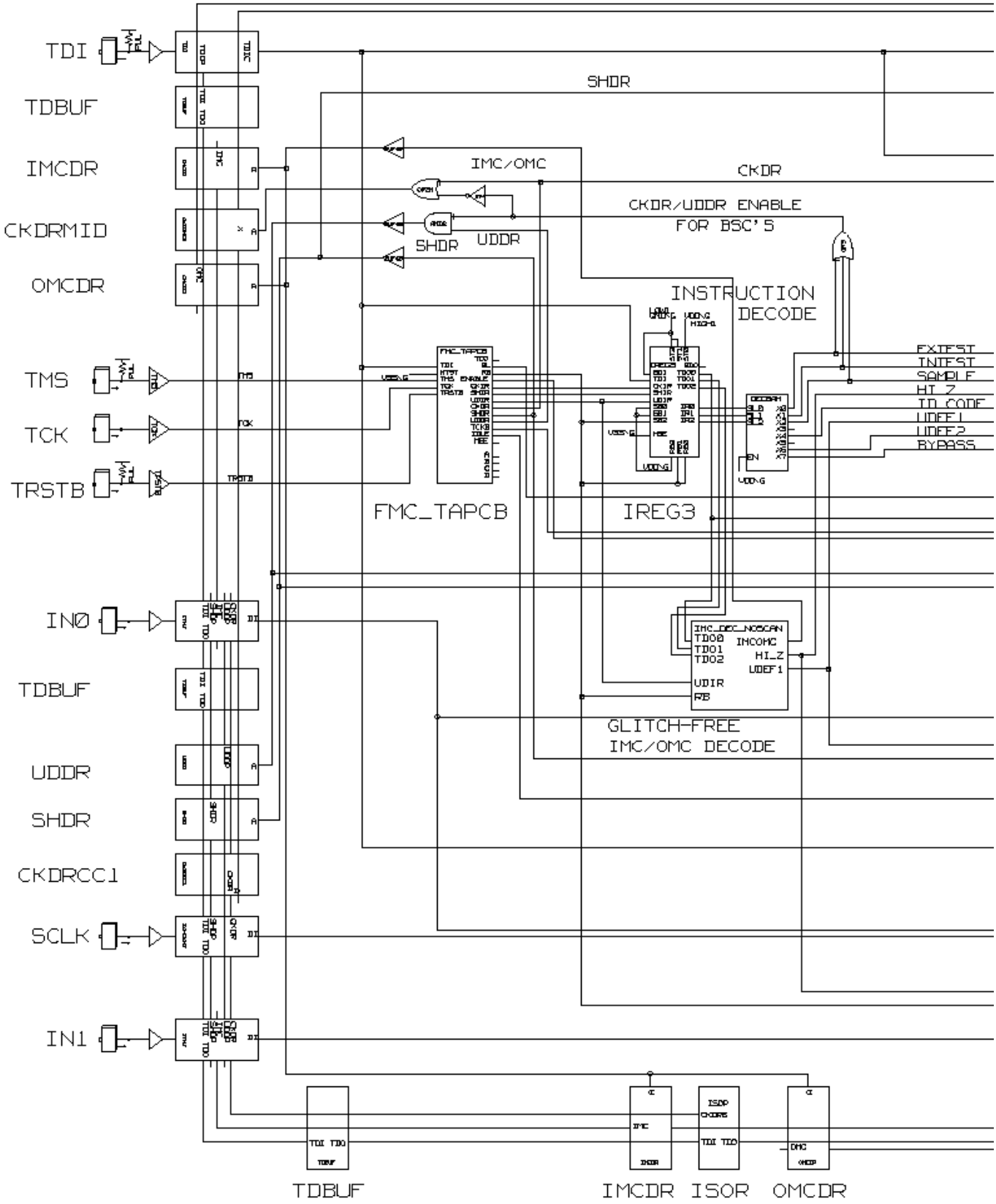

Figure 4-4 Non-Scan JTAG Example Circuit

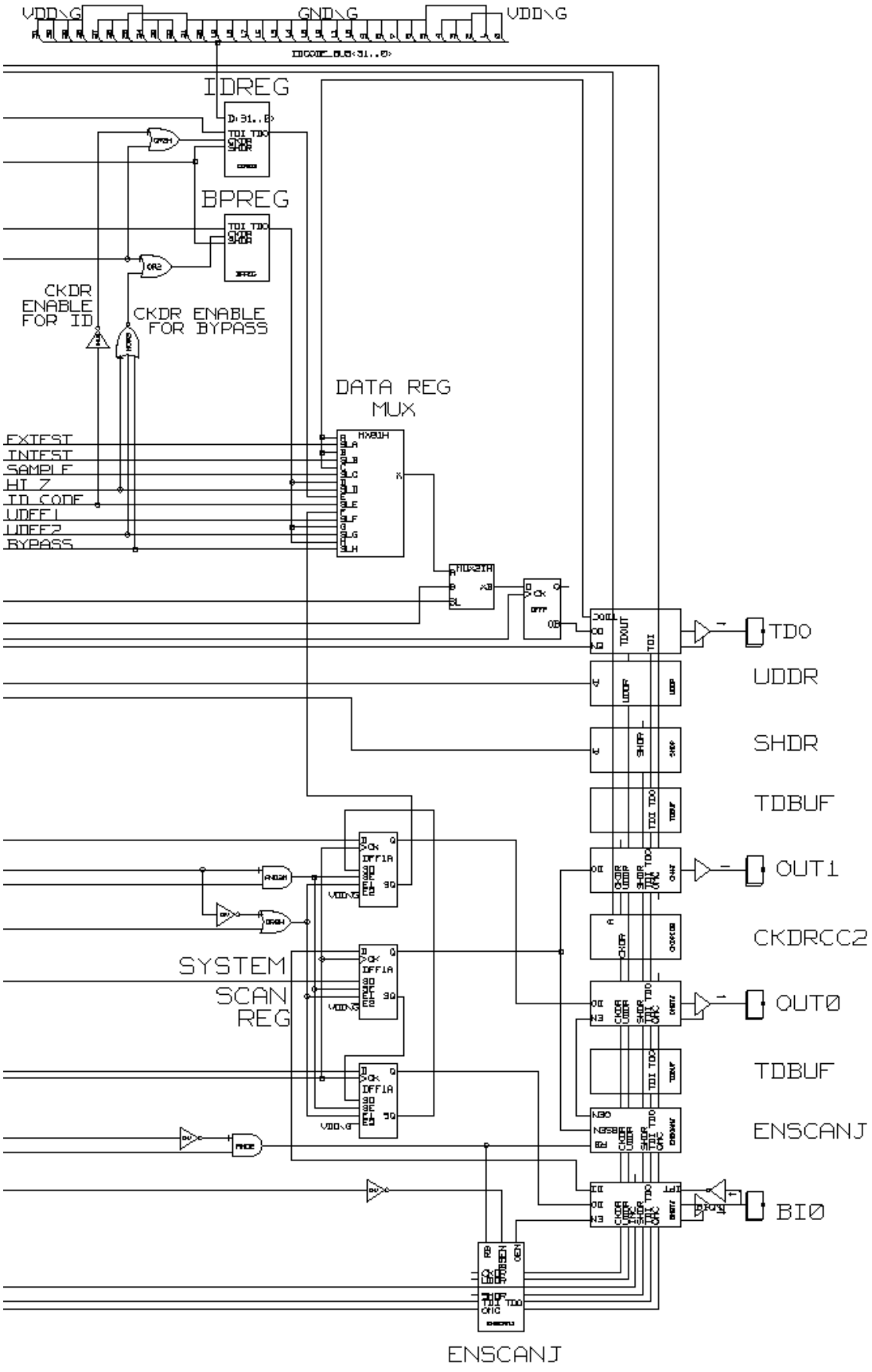

Figure 4-4 Non-Scan JTAG Example Circuit (continued)

4.4 JTAG Circuitry Interconnection

First a non-scan, JTAG design will be discussed to show the essential circuitry required to implement JTAG on H4EPlus arrays. Afterwards, the requirements for a scan-compatible JTAG design will be discussed and illustrated.

4.4.1 Non-Scan JTAG Interconnection

An example of a non-scan JTAG design is shown in Figure 4-4. Note that:

- the MTST pin is not required
- the MSE and TDO TAP Controller outputs are not used
- the TAP Controller MTST input and Instruction Register MSE input should be tied low

Except for the three flip-flops labeled “System Logic,” all of the circuitry in Figure 4-4 is part of the JTAG logic. The BSC’s and peripheral JTAG buffers are located around the periphery of the schematic. **Note that test data must shift counter-clockwise through the BSC’s around the periphery of the chip.**

On H4EPlus arrays, JTAG boundary scan uses a gated CKDR signal as described in the IEEE 1149.1 specification. That is, CKDR is gated to the appropriate data register (peripheral boundary scan register, Bypass Register, Device Identification Register, etc.) under control of the Instruction Register decode logic. For example, if the Instruction Register holds the Sample, INTEST or EXTEST instruction, then CKDR and UDDR will be gated to the peripheral boundary scan register. In addition, the Instruction Register decode logic must generate the Input Mode Control (IMC) and Output Mode Control (OMC) signals. IMC and OMC control the select lines of the data path multiplexers within the input and output BSC’s, respectively. Since users may define their own JTAG instruction sets, the Instruction Register decode logic is design specific; therefore it is not implemented as a special macro in the H4EPlus library. However, in this example a DEC8AH macro from the H4EPlus library is sufficient to implement the Instruction Register decoder.

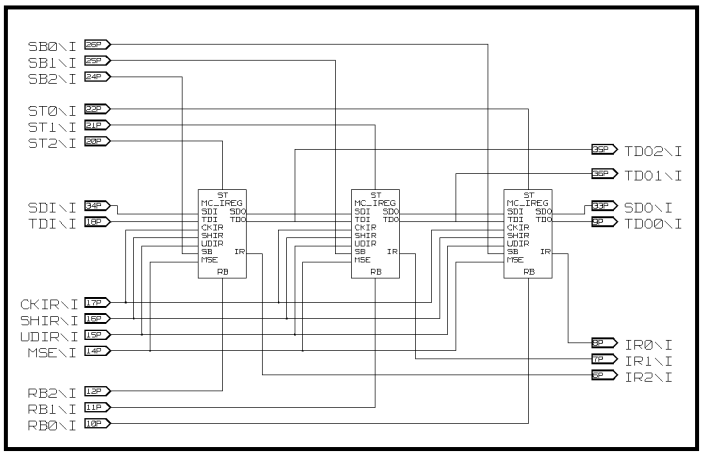

Figure 4-5 IREG3 3-Bit Instruction Register

Because instruction decoding is done by combinatorial logic, the decoded control signals may glitch temporarily when UDIR activates a new instruction. Such glitches are harmless on some control signals, but not on others. Control signals which cannot afford to be glitched should be decoded from the instruction register CKIR flops instead of the UDIR flops. The decoded signals then drive flops which are clocked by UDIR. In Figure 4-4 the “Glitch-Free IMC/OMC Decode” block uses this method to decode the IMC/OMC, HI-Z and UDEF1 signals. The IREG3 is a 3-bit instruction register built from three 1-bit MC_IREG cells in order to bring out the CKIR flop outputs at the TDO0, TDO1 and TDO2 ports (see Figure 4-5). These signals are used to decode the IMC/OMC, HI-Z and UDEF1 signals as shown in Figure 4-6. Alternatively, all instruction decoding could be performed on the instruction register CKIR flops, with each decoded signal driving its own “UDIR” flop.

Note that even though IMC and OMC are independent lines in the chip periphery, driven by separate IMCDR and OMCDR drivers, both IMC and OMC are functionally equivalent to the “Mode” control signal defined in IEEE 1149.1. Therefore they would normally have a common source in the array core. In Figure 4-6 this common source is labeled “IMC/OMC.”

The ENSCANP and ENSCANJ macros have been hooked-up such that they can be reset either by resetting the TAP Controller or by loading a “HI_Z” instruction, which puts all 3-state outputs in the hi-impedance state.

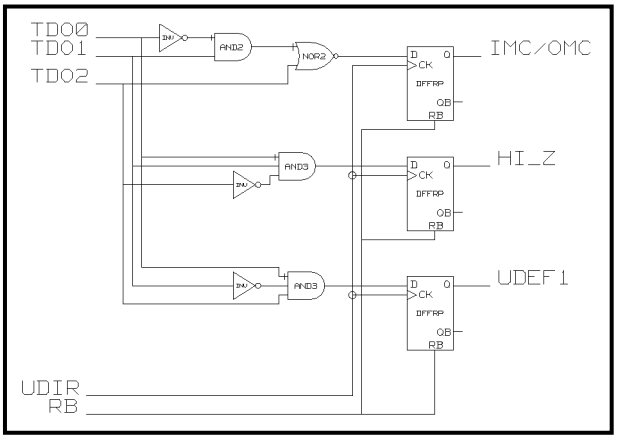

Figure 4-6 Glitch-Free IMC/OMC Decode Block

The implementation of the “System Scan Reg” shown in Figure 4-4 is just an example of what could be done. When a user-defined JTAG instruction (code = binary 101) is active, the “E1” input to the System Scan Reg becomes the IDLE output from the Tap Controller so as to allow the System Scan Reg to be clocked while in the “Run-Test/Idle” state. When the user-defined test is completed the contents of the System Scan Reg can be shifted out through TDO under control of SHDR from the TAP Controller, just like any other JTAG test data register such as the Bypass register or peripheral boundary scan register. If the user has no interest in doing such a test, E1 could be tied high (always enabled) and SE could be wired directly to MSE, for example.

4.4.2 ATPG Compatible JTAG Interconnection

In order to make a JTAG design ATPG compatible the FMC_TAPCB TAP Controller of Figure B-3 must be used, as well as an Instruction Register like the one in Figure 4-2. The designer must also add extra circuitry to link up all of the JTAG registers into one scan chain during ATPG test mode. The requirements, which are illustrated in Figure 4-8, are as follows:

1. An extra MTST input pin, which will only be active during ATPG testing, must be added to the design. A pull-up/pull-down resistor may be used to hold this pin inactive during normal operation. **Note: the input macro driven by MTST must be either a non-JTAG macro or a "sample-only" macro such as an ICNCKHJ.**
2. The ATPG test mode input pin must be connected to the MTST input of the TAP Controller and to OR gates which perform clock gating for JTAG data registers. This will cause all of the JTAG scan chains (i. e. the TAP Controller, Instruction Register, and data registers) to be clocked together in ATPG test mode, during which they are part of one scan chain between TDI and TDO.
3. The ATPG scan path through the JTAG logic must be connected starting with the boundary scan chain, then any internal data registers (e.g. Bypass register and ID code register), then the TAP Controller and finally the Instruction Register. This reduces the probability of clock skew problems occurring.
4. Two-input multiplexers must be placed on the connections between the TDI pin and each internal/core JTAG scan chain to enable all JTAG scan chains to be connected up as one long chain during ATPG test mode. The output of each multiplexer will feed the input to a JTAG scan chain. The select input on the multiplexer must be connected to the TAP Controller's Motorola Scan Enable (MSE) output (MSE is TMS logically AND'ed with MTST). The TDI signal should be connected to the A input of the multiplexers. The B input of the multiplexers should be connected to the end of the previous JTAG scan chain, where the order is that defined in item 3 above.
5. In order to use JTAG input and output BSC's as the input to or output from a scan chain they must be forced into transparent mode. If the output has an enable line then this too must be activated during scan. This is done by adding gating logic to the IMC and OMC control lines to force them low during scan mode.
6. If the TCK and system clocks are derived from the same source then they must be clocked on the same edge of the source clock. This may require placing an exclusive-or gate on the TCK input to the TAP Controller, as well as adding clock gating circuitry similar to that built into the TAP Controller, which is described in Appendix B. This circuitry would serve to prevent race conditions between the BSC's and the system/core logic.
7. ATPG design rules require all asynchronous control lines to be controlled only by an external input pin. Any internal sources which provide asynchronous set or reset must be gated with the MTST input to block their control during ATPG test mode.
8. All inputs which operate in ATPG test mode as a shift clock, a capture clock, a RAM read or write clock, or an asynchronous set or reset control must enter the chip through a non-JTAG macro or through a "sample-only" macro such as an ICNCKHJ.

Other Motorola tester induced design rules should be mentioned at this point:

9. Bidirectional I/O pins which are not used as scan chain outputs should be placed in their input mode during scan chain loading and unloading (TMS=1, MSE=1). This helps to avoid potential excessive switching noise on I/O during scan load/unloads.

4.5 ATPG-Compatible JTAG Example Circuit

In Figure 4-8 the non-scan JTAG design in Figure 4-4 has been modified, according to the requirements of Section 4.4.1, to create an ATPG-compatible JTAG design. The system logic in this example consists solely of the "System Scan Reg," which is a conventional scan design consisting of one scan chain starting at the IN0 input and exiting at the OUT0 output. The remaining circuitry implements JTAG boundary scan which, during ATPG testing, is configured as one scan chain which enters the chip at pin TDI and exits at pin TDO, as described in Section 4.4-2. (Larger designs typically have multiple scan chains for the system logic because of the long time taken to load/scan them.) The "Glitch-Free IMC/OMC Decode" block functions as described in Section 4.4-1, except that the flops inside it are now scan flops (see Figure 4-7).

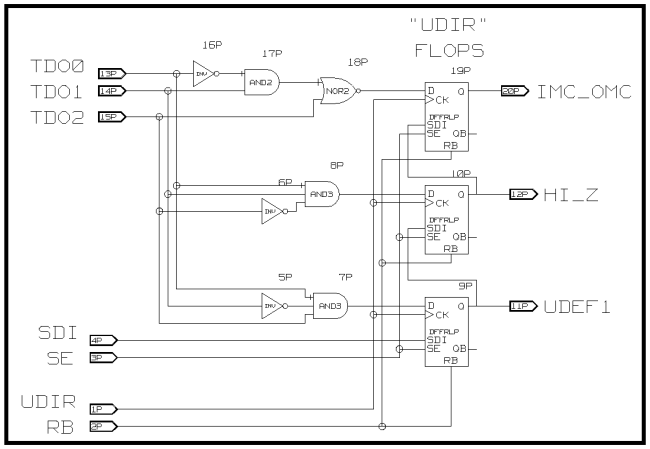

Figure 4-7 ATPG Glitch-Free IMC/OMC Decode Block

ATPG test mode is established by forcing the MTST pin high. During ATPG test mode the TMS pin controls scan mode (MTST and TMS high), during which bidirectional pin output drivers must be disabled. Using TRSTB, instead of TMS, to disable the bidirectionals during scan mode improves the fault coverage of the bidirectionals.

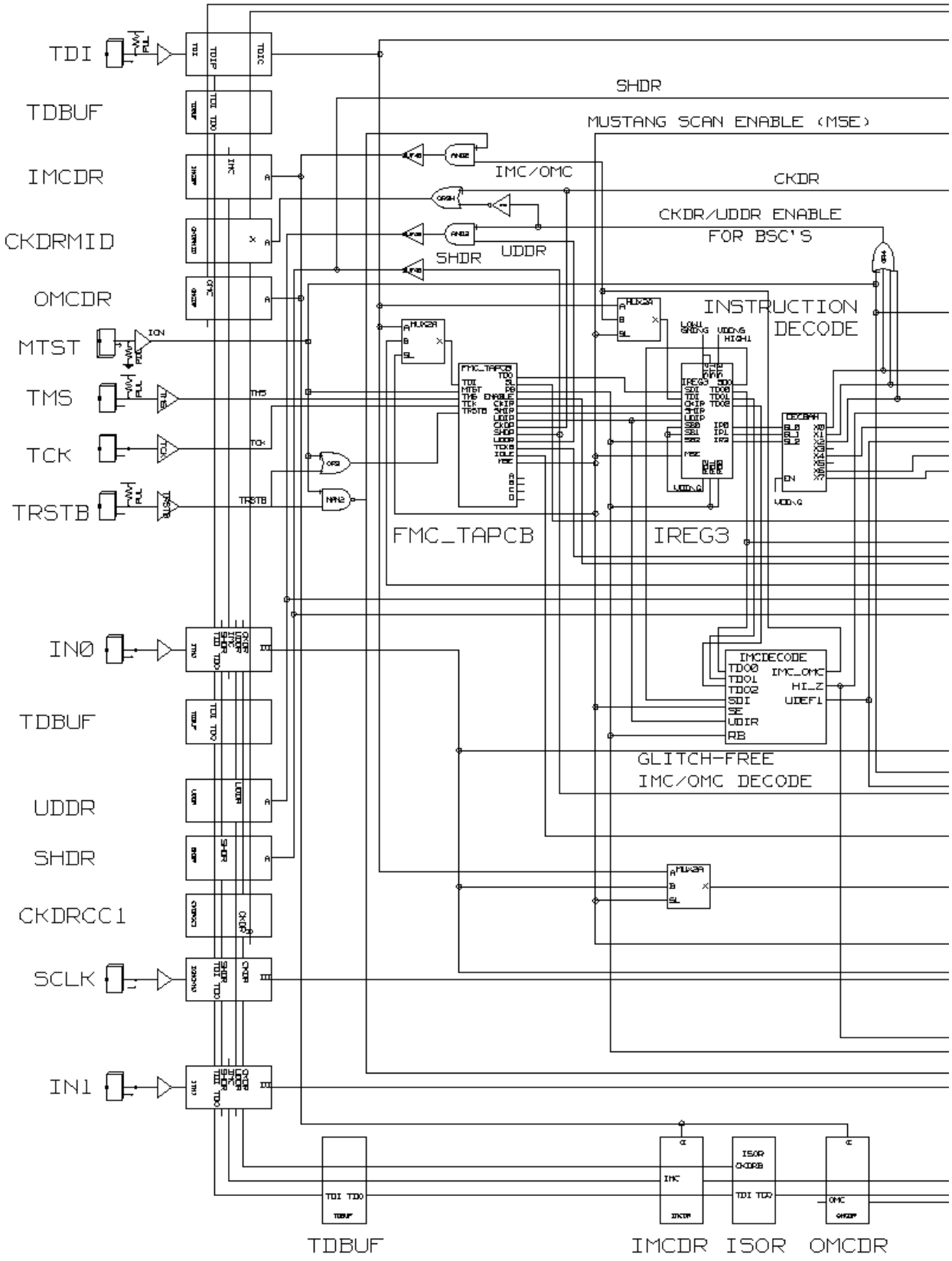

Figure 4-8 ATPG-Compatible JTAG Example Circuit

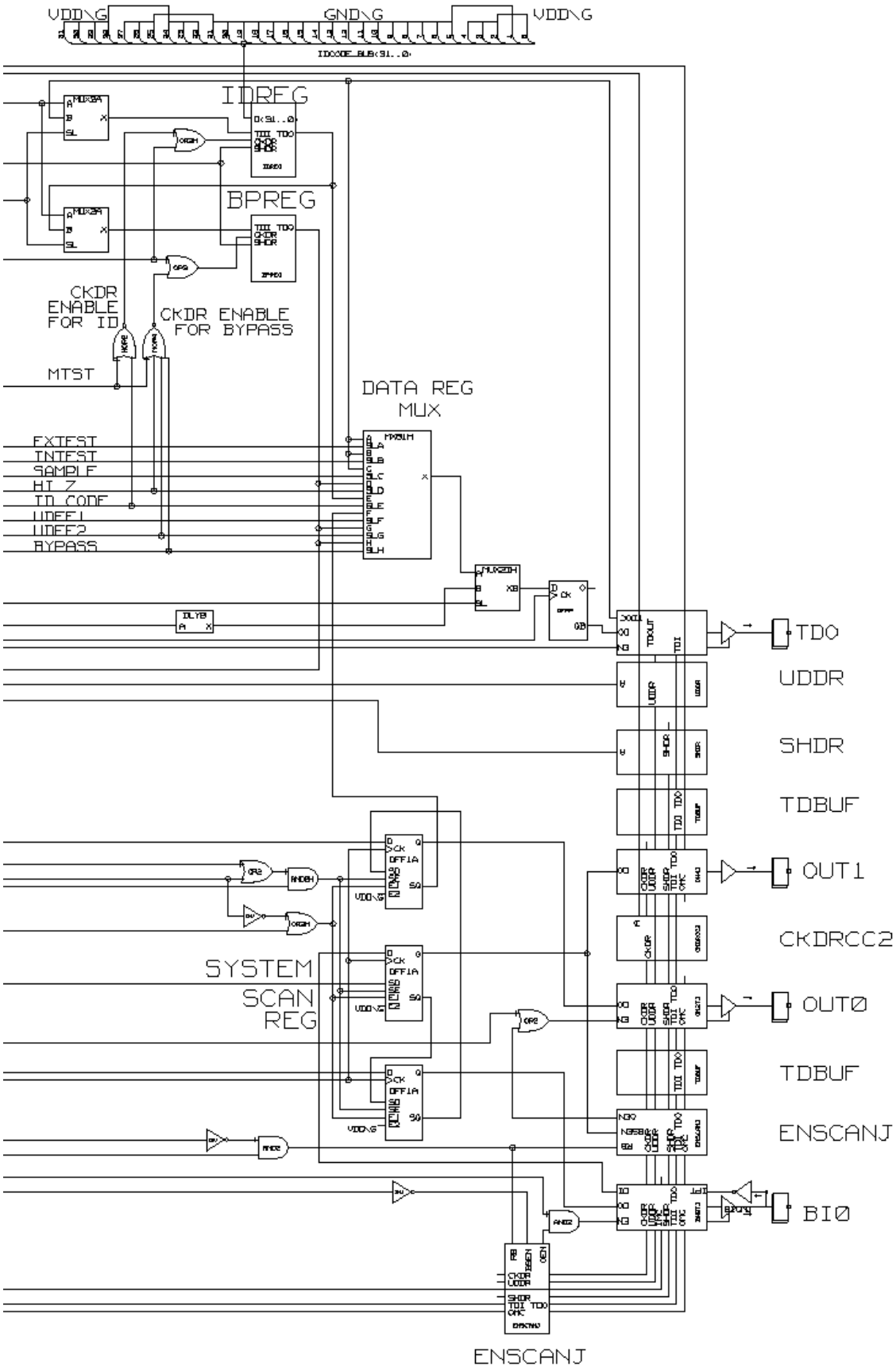

Figure 4-8 ATPG-Compatible JTAG Example Circuit (continued)

During scan mode all JTAG registers are configured into one scan chain via 2-input multiplexers at the TDI inputs of the Device Identification Register, Bypass Register, TAP Controller, and Instruction Register. Within the Instruction Register, UDIR flops as well as CKIR flops become part of this same chain as described previously in Section 4-3. The scan chain order is as follows: TDI, BSC's, IDREG, BPREG, FMC_TAPCB, MC_IREG4, IMC/OMC DECODE, TDO. Also, during ATPG scan mode IMC and OMC are forced low so that I/O BSC's will pass scan data into and out of the chip.

The Fastscan control files for this design are shown below:

```
 //        *****  Fastscan Control File  *****
 // <DESIGN>/<VECTORS>/test/fastscan.control

 // --- Place the environment into a known state  ---
set system mode setup      // make sure we are in setup mode
set sim mode comb -d 0     // ensure combinational atpg mode
del read controls -all     // remove prior read controls
del write controls -all    // remove prior write control lines
del scan chains -all       // remove prior scan chains
del scan groups -all       // remove prior scan groups
del clocks -all            // remove prior clocks/resets
del cell constraint -all   // remove prior cell constraints
del cone blocks -all       // remove prior clock cone blockages
del nofault -all           // remove prior nofault settings
del pin constraint -all    // remove prior constraints on pins
del tied signals -all      // remove prior tied nets or pins

 // --- DECLARE CLOCKS AND RESETS and their OFF states ---
add clock 0 SCLK          // CLK's off state is 0
add clock 0 TCK           // JTAG clock
add clock 1 TRSTB         // JTAG async reset

 // --- DEFINE SCAN GROUP PROCEDURE NAME and FILE ---
add scan group grp1 g1    // define label "grp1', file = "g1"

 // --- DEFINE SCAN CHAIN NAMES, INPUTS, OUTPUTS ---
add scan chain REG1 grp1 IN0 OUT0     // chain REG1
add scan chain JTAG grp1 TDI TDO      // typical JTAG scan chain

 // --- DECLARE PIN CONSTRAINTS  ---
add pin constraint MTST c1 // hold MTST at constant 1 at all times

 // --- ADJUST DESIGN RULE CHECKING ---
set drc hand C9 note nover // hide details of C9 checks
set capture hand -te X -atpg   // autocorrect C3 & C4 problems

 // --- SWITCH TO ATPG MODE and PERFORM DFT CHECKS -
set sys mode atpg          // prepare to generate patterns
report drc rules -verb     // record DRC rules into log

 // --- POPULATE FAULTS INTO THE DESIGN ---
add faults -all            // populate faults

 // --- GENERATE TEST PATTERNS ---
set dofile abort off       // from this point on, ignore problems
run                        // generate patterns
report stat                // report fault stats so far

set abort limit 500        // increase abort limit
run                     // go for any remaining undetected faults
report stat                // report fault of 2nd run

 // --- STATIC PATTERN COMPRESSION ---
comp pat 100 -reset -max 8 // try until 8 iterations w/o decrease

 // --- SAVE PATTERNS ---
save pat fastscan.patterns -binary -pad0 -rep
save pat utic2.fastscan_0_9 timeplate -utic -rep -pad0 -scan -beg
0 -end 9
save pat utic2.fastscan timeplate -utic -pad0 -rep

report env                 // log the environment

 // --------------------------------
 // --- OTHER DESIGN INFORMATION ---
 // --------------------------------
set fault mode collapsed
report faults -hier 12 -min 100 -class au
report nonscan cells       // report flops & latches not in scan
chains
rep stat                   // final fault coverage
exit
```

Figure 4-9 Fastscan Control file

```
 // ============== PROCEDURE FILE ================
 //    <DESIGN>/<VECTORS>/test/g1

// SHIFT timing suitable for S1650 tester
procedure shift =
  force_sci            0;    // force scan chain inputs
  measure_sco         50;    // measure scan chain outputs
  force SCLK      1  100;    // shift clock ON
  force TCK       1  100;    // shift clock ON
  force SCLK      0  150;    // shift clock OFF
  force TCK       0  150;    // shift clock OFF
  period             200;    // shift cycle period
end;

// - The load_unload procured is applied prior to each shift
procedure load_unload =
   force MTST          1   0;   // place in test mode
   force TMS           1   0;   // enable scan mode
   force SCLK          0   0;   // clk to off state
   force TCK           0   0;   // clk to off state
   force TRSTB         1   0;   // async rst to off state
// - put bidirects NOT used as scan outputs into hi-z for 1 cycle
   force BI0           Z   0;
   break                 500;
// - then force them to a known state
   force BI0           1 500;
// --- shift procedure applied at end of period
   apply shift       999 1000;
   period              1000;
end;

procedure test_setup =
   force MTST          1   0;   // constrained pin to 1 for ATPG
   force SCLK          0   0;   // system clk to off
   force TCK           0   0;   // JTAG clk to off
   force TRSTB         1   0;   // JTAG reset to off
   force BI0           Z   0;   // bidi pin to Z
   period 500;
end;
```

Figure 4-10 Fastscan g1 Procedure File

```
   // =============  TIMEPLATE FILE ===============
   // <DESIGN>/<VECTORS>/test/timeplate

  set time scale           1 nS;
  set split_bidi_cycle time 500;
  set strobe_window time     32;
  set procedure file;

  Timeplate "master" =
      force_pi                  0;
      bidi_force_pi           500;
      measure_po              700;
      capture_clock_on        750;
      capture_clock_off       800;
      period                 1000;
  end;
```

Figure 4-11 Fastscan Timeplate File

These control files can be edited from the template control file built by UDS_FASTSCAN in the <DESIGN>/design_data directory.

In the Fastscan Control file it is important to use the command '**set capture hand -te X -atpg**' to automatically correct C3 and C4 DRC violations, otherwise Fastscan will not handle the ISOR macro correctly and miscompares will occur in Testsim re-simulation.

The fault coverage obtained for this example circuit was >85%. This fault coverage is essentially that of the JTAG circuitry alone since the system logic in this example consists solely of one 3-stage shift register. The fault coverage for the JTAG logic is limited by the extra circuitry added for test, but can be improved by applying JTAG vectors during normal functional testing.

4.6 Conclusions

The methodology presented here allows ATPG methods to be used with JTAG by modifying the designs of the TAP Controller and the Instruction Register. The results show that it is possible to achieve high fault coverage using fully automated test pattern generation. However, ATPG methods cannot test the gate (G) input to the BSC shadow latches. Nor can ATPG methods test all of the JTAG Instruction Register decoding logic. Testing of these areas should be achieved by supplementing the ATPG test patterns with some manually written vectors which test the JTAG circuitry in its normal functional mode of operation. Merging of these functional vectors with the ATPG vectors is accomplished by TESTMERGE, which is one of the Motorola supplied CAD tools.

5. JTAG I/O Macro Placement and Pin-out Assignment

5.1 Pin & I/O Site Placement of JTAG I/O Macros

5.1.1 Placement of Test Access Port (TAP) Pins

It is recommended that TMS be adjacent to TCK, and that TDI be adjacent to TDO. TDI must be counterclockwise from TDO, and no BSC's should be placed between TDI and TDO (clockwise from TDI) because these BSC's would not be contained in the I/O boundary scan data register.

5.1.2 Placement of Non-bonded JTAG Macros

"Non-bonded" macros reside in I/O or power sites but have no off-chip connections. These macros include TDBUF, ENSCANJ, ISOR, CKDRMID, CKDRCC1, CKDRCC2, SHDR, UDDR, IMCDR and OMCDR.

It is preferable to place these macros on I/O sites that cannot be used for normal I/O, but they can be placed on any unused I/O. For example referring to the "H4EP075 100 QFP Pad-to-Pin Cross Reference" in the H4EPlus Series Design Reference Guide, the following I/O sites would be good choices:

- I/O sites which do not connect to a pad (e. g., I/O sites 26 & 31), or
- I/O sites which connect to a pad which is not bonded out to a package pin (e. g., I/O sites 5 and 6), or
- I/O sites which are used for power/ground pads (e. g., I/O sites 15 and 25).

In addition, the JTAG buffers must reside within 25 sites of the nearest INPVSS and INPVDD (or BOTHVSS and BOTHVDD). The Pad-to-Pin Cross Reference for the pertinent array and package is used to select a site for each non-bonded macro.

5.1.3 Use of Pinout for Placement of JTAG I/O

Once the design has been entered (schematically or texturally as a netlist) the I/O pinout must be created. This is done using the PINOUT tool. When PINOUT is run it will read in the netlist and design information, and create a list of the I/O cells that need to be placed.

The designer should position the I/O in the manner described in this section using the graphical entry tools. PINOUT also provides the capability to shift blocks of I/O macros. Power, ground and the ENID macro must then be added.

The pinout can then be verified using the 'Check' function within PINOUT, and the designer can make any corrections needed to make the option pass the manufacturing rules.

When PINOUT saves the I/O placement information it is stored in the <DESIGN>/design_data/attributes file. An example attribute file is shown for the design in Figure 4-8.

```
* Pinout(TM) Version 3.02 Sun Jul 28 18:33:40 1996
* Copyright 1994 - 1996, Motorola Inc.
* Design     "JTAG_EP"
* Technology "H4EP5"
* Array      "h4ep075"
* Package    "100qfp"
* Bondout    "h4ep075_100qfp"
*
* Top Level Module Scope
-MODULE JTAG_EP
* Power Cells
-POWERCELL
*   PowerCellType            InstanceName         PortName
    BOTHOVDD                 BOTHOVDD_1           PAD_BOTHOVDD_1 ;
    BOTHOVDD                 BOTHOVDD_2           PAD_BOTHOVDD_2 ;
    BOTHOVDD                 BOTHOVDD_3           PAD_BOTHOVDD_3 ;
    BOTHOVDD                 BOTHOVDD_4           PAD_BOTHOVDD_4 ;
    ENID                       ENID_1               PAD_ENID_1 ;
    OVSS                       OVSS_1               PAD_OVSS_1 ;
    OVSS                       OVSS_2               PAD_OVSS_2 ;
    OVSS                       OVSS_3               PAD_OVSS_3 ;
    OVSS                       OVSS_4               PAD_OVSS_4 ;
    VDD                        VDD_1                PAD_VDD_1 ;
    VSS                        VSS_1                PAD_VSS_1 ;
*
*
* Instance Properties
-INSTANCE
*   InstanceName             PropertyName         "PropertyValue"
    \TDBUF.120P_1              FIX                  "P185" ;
    \TDBUF.116P_1              FIX                  "P006" ;
    \TDBUF.35P_1               FIX                  "P053" ;
    \ISOR.112P_1               FIX                  "P076" ;
    \ENSCANJ.107P_1            FIX                  "P082" ;
    \BN2TJ.83P_1               OUTPUT_THRESHOLD     "TTL" ;
    \ENSCANJ.85P_1             FIX                  "P097" ;
    \TDBUF.87P_1               FIX                  "P100" ;
    \ON2TJ.86P_1               OUTPUT_THRESHOLD     "TTL" ;
    \ON4J.88P_1                OUTPUT_THRESHOLD     "TTL" ;
    \TDBUF.89P_1               FIX                  "P142" ;
    \TDOUT.110P_1              OUTPUT_THRESHOLD     "CMOS" ;
    \CKDRCC1.114P_1            FIX                  "P027" ;
    \CKDRCC2.121P_1            FIX                  "P140" ;
    \CKDRMID.118P_1            FIX                  "P190" ;
    \IMCDR.53P_1               FIX                  "P187" ;
    \IMCDR.65P_1               FIX                  "P075" ;
    \OMCDR.113P_1              FIX                  "P078" ;
    \OMCDR.117P_1              FIX                  "P195" ;
    \SHDR.38P_1                FIX                  "P026" ;
    \SHDR.91P_1                FIX                  "P143" ;
    \UDDR.109P_1               FIX                  "P144" ;
    \UDDR.115P_1               FIX                  "P025" ;
*
* Port Properties
-PORT
*   PortName                 PropertyName         "PropertyValue"
    TDI                        IO_PIN1              "91" ;
    IN0                        IO_PIN1              "4" ;
    SCLK                       IO_PIN1              "25" ;
    IN1                        IO_PIN1              "28" ;
    BI0                        IO_PIN1              "44" ;
    OUT0                       IO_PIN1              "68" ;
    OUT1                       IO_PIN1              "69" ;
    TDO                        IO_PIN1              "88" ;
    PAD_BOTHOVDD_1             IO_PIN1              "2" ;
    PAD_BOTHOVDD_2             IO_PIN1              "26" ;
    PAD_BOTHOVDD_3             IO_PIN1              "52" ;
    PAD_BOTHOVDD_4             IO_PIN1              "77" ;
    PAD_ENID_1                 IO_PIN1              "85" ;
    MTST                       IO_PIN1              "95" ;
    PAD_OVSS_1                 IO_PIN1              "3" ;
    PAD_OVSS_2                 IO_PIN1              "30" ;
    PAD_OVSS_3                 IO_PIN1              "53" ;
    PAD_OVSS_4                 IO_PIN1              "79" ;
    TCK                        IO_PIN1              "98" ;
    TMS                        IO_PIN1              "97" ;
    TRSTB                      IO_PIN1              "100" ;
    PAD_VDD_1                  IO_PIN1              "78" ;
    PAD_VSS_1                  IO_PIN1              "27" ;
*
* End Attributes
```

Figure 5-1 Example Attribute File

5.2 Guidelines for Finding a MARV-Compatible Chip Pin-out

In both procedures that follow, wherever possible the JTAG clock and control signal buffers should be placed on power sites or I/O sites that are not connected to package pins.

5.2.1 Full Boundary Scan Pin-Out Guidelines

In selecting the pin-out for a chip which uses full boundary scan (the vast majority of system signal pins use BSC's), it is recommended that the following steps be done in sequence:

1. Place TDI and TDOUT on a pair of adjacent pins. TDI must be counterclockwise from TDO. Place TCK, TMS and TRSTB on pins in the same general area as TDI and TDOUT.
2. Place remaining system pins, and any additional power/ground pins required, in conformance with the MARV rules governing the placement of output drivers relative to power pin locations. Also keep in mind the rules governing sharing of a single I/O site by two different macros.
3. Place ENSCANJ 3-state control BSC's on available I/O or power sites.
4. Between every pair of BSC's separated by >7 non-BSC I/O sites, a TDBUF must be inserted on an I/O site within 7 sites of the BSC whose TDO port drives the other's TDI port. (TDBUF is built from the input buffer portion of an I/O site, and can therefore share an I/O site with a non-JTAG output driver or with a "paralleled" output driver used to build a hi-drive.)
5. Place the CKDRCC1 and CKDRCC2 buffers such that, in Figure 3-1, nets CKDRNET1 and CKDRNET2 drive a similar number of loads.
6. Place the CKDRMID buffer equidistant between CKDRCC1 and CKDRCC2 as shown in Figure 3-1.
7. Place the ISOR macro diametrically opposite from TDOUT (approximately).
8. Place the SHDR buffers approximately in the center of nets 1 and 2 as shown in Figure 3-2. Do the same for the UDDR buffers.
9. Place the IMCDR buffers approximately halfway between the CKDRCC1 and CKDRCC2 buffers, which create gap 1 and gap 2 in Figure 3-3. Do the same for the OMCDR buffers.

5.2.2 Partial Boundary Scan Pin-Out Guidelines

"Partial boundary scan" refers to a chip on which many system signal pins use non-JTAG I/O macros. In selecting the pin-out for such a chip, it is recommended that the following steps be done in sequence:

1. Place system pins (i. e., all pins except TDI, TDOUT, TMS, TCK, and TRSTB), and any additional power/ ground pins required, in conformance with the MARV rules governing the placement of output drivers relative to power pin locations. Also keep in mind the rules governing sharing of a single I/O site by two different macros.
2. Place ENSCANJ 3-state control BSC's on available I/O or power sites.
3. Choose a pair of adjacent pins for TDI and TDOUT (with TDI counterclockwise from TDO) such that a line drawn from TDOUT through the center of the chip creates two halves which each contain an equal number of BSC's, and to the extent possible, an equal number of non-JTAG pins. Place TCK, TMS and TRSTB on pins in the same general area as TDI and TDOUT.
4. Between every pair of BSC's separated by >7 non-BSC I/O sites, a TDBUF must be inserted on an I/O site within 7 sites of the BSC whose TDO port drives the other's TDI port. (TDBUF is built from the input buffer portion of an I/O site, and can therefore share an I/O site with a non-JTAG output driver or with a "paralleled" output driver used to build a hi-drive.)
5. Place the CKDRCC1 and CKDRCC2 buffers such that between each buffer and TDOUT there is an equal number of BSC's, and to the extent possible, an equal number of non-JTAG pins. Refer to Figure 3-1.
6. Place the CKDRMID buffer equidistant between CKDRCC1 and CKDRCC2 (typically near TDOUT). Refer to Figure 3-1.
7. Place the ISOR macro diametrically opposite from TDOUT (approximately).
8. Place the SHDR buffers approximately halfway between the TDOUT and ISOR, in terms of BSC's (see Figure 3-2). Do the same for the UDDR buffers.
9. Place the IMCDR buffers approximately halfway between the CKDRCC1 and CKDRCC2 buffers, in terms of BSC's (see Figure 3-3). Do the same for the OMCDR buffers.

6. CAD Design Flows

For Verilog simulation, design entry can be done by either schematic capture or, for Synopsys users, by writing a Verilog HDL circuit description. The “Schematic Capture/Verilog Design Flow” in Section 6.1 and the “Synopsys/Verilog Design Flow” in Section 6.2 both follow a “top-down” design methodology where the chip is initially described behaviorally using Verilog HDL. In Section 6.1 the HDL is manually converted into gates using schematic capture, whereas in Section 6.2 the HDL is synthesized into gates using Synopsys.

For real world schematic capture designs, it may be more practical to capture the BSC’s in rows rather than trying to capture the I/O in the shape of a chip outline as in Figure 4-8.

During pre-Predix simulations, estimated parasitic resistance and capacitance values are used for the metal interconnect between peripheral JTAG macros, as is done for interconnect between core macros. **As a result, it is possible to get pre-layout DECAL edge-rate warnings or errors for nets in the periphery. These warnings and errors are invalid.** After Predix generation of the actual resistance and capacitance values (predix.jtagrc file) are used, at which time no edge-rate errors should occur on peripheral nets.

6.1 Schematic Capture/Verilog Design Flow

I. Behavioral-Level Design

1. As part of the behavioral verification of the entire system, create and verify a Verilog HDL behavioral description for all system logic on the H4EPlus chip.

II. Determine Chip Pin-Out and Capture JTAG I/O

2. Use UDS_CONCEPT to capture a schematic of the JTAG I/O, following the “Guidelines for Finding a MARV-Compatible Chip Pin-Out” in Section 5.2.
3. Use PINOUT to assign I/O placement and run peripheral rule checks. Repeat steps 2 and 3 until all checks pass.
4. Use MARV to verify all I/O and core rules.

III. Design Chip’s System (Non-JTAG) Logic

A. Convert Behavioral Description to an RTL (Register-Transfer Level) Description

5. For one chip sub-block “X,” convert the behavioral description to an RTL description. If X is to be converted into gates by logic synthesis as opposed to schematic capture, then the RTL description must use only those Verilog constructs supported by Synopsys.
6. Simulate X’s RTL description, by itself. Modify and re-simulate X’s RTL description until its functionality matches X’s behavioral description.
7. Repeat step 1’s simulation of the chip behavioral description, but use the RTL description for X in place of X’s behavioral description. Modify X’s RTL description as necessary until chip functionality matches that of the all-behavioral chip description in step 1. Repeat steps 5-7 for each of the chip’s sub-blocks.
8. Simulate all system logic on the chip at the RTL level. Modify the sub-blocks’ RTL descriptions as necessary until chip functionality matches that of the all-behavioral chip description in step 1.

B. Convert Sub-Block RTL Descriptions to Gate-Level Netlists

9. Use UDS_CONCEPT to capture a schematic for one system sub-block “X”. Set the Write Verilog option to ‘Y’ to create a Verilog netlist when you exit Concept.
10. (Optional) Do unit-delay simulation of X’s gate-level netlist by itself, using the same vectors used to verify X’s RTL description in step 6. If X’s functionality is incorrect return to step 9 to correct the schematic, or correct X’s RTL description and return to step 6, 7 or 8 to verify the correction.
11. (Optional) Repeat step 8’s simulation of the chip RTL description, but use the gate-level netlist for sub-block X in place of X’s RTL description. Handle bugs as prescribed in step 10.
12. Run PINOUT to assign and verify I/O placement. If errors are found correct the I/O placement or return to step 9 to correct the schematic.
13. Run MARV to verify X conforms to electrical design rules. If violations occur return to step 9 to correct the schematic.
14. If the chip is a scan design, run the FASTSCAN design rule checker to verify X conforms to scan design rules. (Also generate test patterns if Y’s fault coverage is desired.) If violations occur return to step 9 to correct the schematic.
15. Verify X via real-time simulation and timing analysis:
    a. Run DECAL to calculate real-time delays. If edge-rate violations occur, return to step 8 to correct the schematic.
    b. Repeat step 10, using DECAL delays instead of unit delays.
    c. Repeat step 11, using DECAL delays instead of unit delays.
16. Repeat steps 8-15 for each system sub-block on the chip.

C. Combine Netlists for All of Chip’s System Sub-Blocks

17. Use UDS_CONCEPT to combine the sub-blocks for all system logic on the chip. Set the Write Verilog option to ‘Y’ to create a Verilog netlist when you exit Concept.
18. Run MARV.
19. If the chip is a scan design, run the ATPG tool.
20. Run DECAL; then simulate the gate-level netlist for the chip’s system logic using the same vectors which were used in step 7 to verify the RTL description of the chip’s system logic. If violations occur in any of these tools, do one of the following for each erroneous sub-block:
    a. return to step 9 to correct the sub-block’s schematic.
    b. correct the sub-block’s RTL description and return to step 6, 7 or 8 to verify the correction.
    c. Re-verify each corrected sub-block individually to the extent desired in section III, part B, then return to step 17.

IV. Combine JTAG Circuitry with Chip's System Logic

21. Use UDS_CONCEPT to capture a schematic of the core JTAG logic (including the TAP Controller etc.).

Verify All JTAG Circuitry by Itself (Optional)

22. In UDS_CONCEPT, combine the core JTAG logic with the JTAG I/O from step 2. Set the Write Verilog option to 'Y' to create a Verilog netlist when you exit Concept.
23. Use PINOUT to assign I/O placement determined in step 2 and run peripheral rule checks. Repeat steps 22 and 23 until all checks pass.
24. Run MARV.
25. If the chip is a scan design, run the ATPG tool's design rule checker.
26. Run PREDIX to generate the JTAG RC file (predix.jtagrc) using that actual I/O placement information.
27. Run DECAL followed by Verilog real-time simulation. If violations occur in any of these tools, return to step 22 to correct the schematic.

Verify Combined System and JTAG Circuitry

28. Use UDS_CONCEPT to combine the core JTAG logic, JTAG I/O, and system logic. Set the Write Verilog option to 'Y' to create a Verilog netlist when you exit Concept
29. Run MARV to verify entire chip conforms to electrical design rules.
30. If the chip is a scan design, run the ATPG tool's design rule checker to verify entire chip conforms to scan design rules.
31. Run PREDIX to generate the JTAG RC file (predix.jtagrc) using that actual I/O placement information.
32. Verify entire chip via real-time simulation and timing analysis:
    a. Run DECAL to calculate real-time delays.
    b. Simulate the gate-level netlist for the entire chip using the same vectors which were used in step 8 to verify the RTL description of the chip's system logic. Then exercise the JTAG logic in a separate simulation. If errors occur in any of steps 29-32, return to step 22 to correct the core JTAG logic, or do one of the following for each erroneous sub-block:
        i) return to step 9 to correct the sub-block's schematic.
        ii) correct the sub-block's RTL description and return to step 6, 7 or 8 to verify the correction. Re-verify each corrected sub-block individually to the extent desired in section III, part B, then continue at step 17 or 22 as desired.
33. If the chip is a scan design, run the ATPG tool to generate scan test patterns.
34. Run TESTPAS to combine the functional and scan test patterns from steps 32 and 33, respectively.

6.2 Synopsys/Verilog Design Flow

I. Behavioral-Level Design

1. As part of the behavioral verification of the entire system, create and verify a Verilog HDL behavioral description for all system logic on the H4EPlus chip.

II. Determine Chip Pin-Out and Create JTAG I/O Netlist

2. Create a Verilog netlist for the JTAG I/O only (no core module instantiation).
3. Use PINOUT to assign I/O placement and run peripheral rule checks. Repeat steps 2 and 3 until all checks relating solely to the I/O pass.

III. Design Chip's System (Non-JTAG) Logic

A. Convert Behavioral Description to RTL (Register-Transfer Level) Description

4. For one chip sub-block "X," convert the behavioral description to an RTL description. If X is to be converted into gates by logic synthesis as opposed to schematic capture, then the RTL description must use only those Verilog constructs supported by Synopsys.
5. Simulate X's RTL description, by itself. Modify and re-simulate X's RTL description until its functionality matches X's behavioral description.
6. Repeat step 1's simulation of the chip behavioral description, but use the RTL description for sub-block X in place of X's behavioral description. Modify X's RTL description as necessary until chip functionality matches that of the all-behavioral chip description in step 1. Repeat steps 4-6 for each of the chip's sub-blocks.
7. Simulate all system logic on the chip at the RTL level. Modify the sub-blocks' RTL descriptions as necessary until chip functionality matches that of the all-behavioral chip description in step 1.

B. Synthesize Sub-Block RTL Descriptions into Gate-Level Netlists

Synopsys/Verilog Debug Loop

8. For one sub-block "X," synthesize the RTL description into a gate-level Verilog netlist using Synopsys.
9. Simulate X's gate-level netlist by itself, using the same vectors used to verify X's RTL description in step 5. Use Synopsys delays. If X's functionality or timing is incorrect:
    a. return to step 8 to modify the synthesis constraints and re-run Synopsys.
    b. correct X's RTL description and return to step 5, 6 or 7 to verify the correction.
10. Repeat step 7's simulation of the chip RTL description, but use the gate-level netlist for sub-block X in place of X's RTL description. Use Synopsys delays for X. Handle bugs as prescribed in step 9.

OACS Verification

11. Create the required netlists for X:
    a. Run Synopsys to generate a Verilog netlist.
    b. Edit the verilog.index file in the <DESIGN>/netlists directory to include the pathname of the verilog netlist.
    c. Run MARV to verify X conforms to electrical design rules. If violations occur, correct X's RTL description and return to step 5, 6 or 7 to verify the correction.
12. If the chip is a scan design, run the ATPG tool's design rule checker to verify X conforms to scan design rules. (Also generate test patterns if X's fault coverage is desired.) If violations occur, correct X's RTL description and return to step 5, 6 or 7 to verify the correction.
13. Verify X via real-time simulation and timing analysis:
    a. Run DECAL to calculate real-time delays. Handle edge-rate violations as prescribed in step 9.
    b. Repeat step 9, using DECAL delays instead of Synopsys delays.
    c. Repeat step 10, using DECAL delays instead of Synopsys delays.
14. Repeat steps 8-14 for each synthesized sub-block until each one has a correct gate-level Verilog netlist.

C. Design "Not-To-Be-Synthesized"/Schematic Capture Sub-Blocks

OACS Verification

15. Use UDS_CONCEPT to capture a schematic for one system sub-block "Y". Set the Write Verilog option to 'Y' to create a Verilog netlist when you exit Concept.
16. (Optional) Do unit-delay simulation of Y's gate-level netlist by itself, using the same vectors used to verify Y's RTL description in step 5. If Y's functionality is incorrect return to step 15 to correct the schematic. or correct Y's RTL description and return to step 5, 6 or 7 to verify the correction.
17. (Optional) Repeat step 7's simulation of the chip RTL description, but use the gate-level netlist for sub-block Y in place of Y's RTL description. Handle bugs as prescribed in step 16.
18. Run MARV to verify Y conforms to electrical design rules. If violations occur return to step 15 to correct the schematic.
19. If the chip is a scan design, run the ATPG tool's design rule checker to verify Y conforms to scan design rules. (Also generate test patterns if Y's fault coverage is desired.) If violations occur return to step 15 to correct the schematic.
20. Verify Y via real-time simulation and timing analysis:
    a. Run DECAL to calculate real-time delays. If edge-rate violations occur, return to step 8a to correct the schematic.
    b. Repeat step 16, using DECAL delays instead of unit delays.
    c. Repeat step 17, using DECAL delays instead of unit delays.
21. Repeat steps 15 - 20 for each sub-block to be entered via schematic capture until each one has a correct gate-level Verilog netlist.

D. Combine All of Chip's System Sub-Blocks

Synopsys/Verilog Debug Loop

22. Read into Synopsys the gate-level Verilog netlists for all system sub-blocks on the chip. Run Synopsys with logic optimization turned off (i. e., no "compile") to write out one Verilog netlist which contains all system logic on the chip. There should not be any busses at the top level of the design.
23. Simulate the gate-level netlist for all system logic, using Synopsys delays. Use the same vectors which were used in step 7 to verify the RTL description of the chip's system logic. If violations occur, do one of the following for each erroneous sub-block:
    a. return to step 15 to correct the sub-block's schematic, or
    b. correct the sub-block's RTL description and return to step 5, 6 or 7 to verify the correction.

Re-verify each corrected sub-block individually to the extent desired in section III, part B or C, then return to step 22.

OACS Verification

24. Edit the verilog index file to include the path to the block level Verilog netlist.
25. Run MARV. Handle violations as prescribed in step 23.
26. If the chip is a scan design, run the ATPG tool to generate patterns. Handle violations as prescribed in step 23.
27. Run DECAL and then repeat step 23, using DECAL delays instead of Synopsys delays.

IV. Combine JTAG Circuitry with Chip's System Logic

OACS Verification

28. Create a gate-level Verilog netlist for the core JTAG logic (including the TAP Controller etc.) by one of two methods:
    a. capture a schematic and write Verilog netlist.
    b. write a Verilog netlist manually.
    c. run PINOUT and add all the macros required in the design. The macros can then be placed and PINOUT will write out a top level Verilog netlist (verilog.net_pinout_top)

Verify All JTAG Circuitry by Itself (Optional)

29. Merge the core JTAG logic and the JTAG I/O into one Verilog netlist using Synopsys. (The Verilog netlists for any "soft" macros used, such as the MC_IREG or MC_IREG4, must be read into Synopsys; likewise for the Verilog netlists for any "firm" macros used, such as the FMC_TAPCB.)

30. Use PINOUT to assign I/O placement and run peripheral rule checks. Repeat steps 29 and 30 until all checks pass.
31. (Optional) Do a unit-delay simulation on all JTAG circuitry. (Synopsys delays are not usable for simulation because they only include the core<-->PAD data path through each BSC.) If violations occur return to step 28 to correct the netlist.

Verify Combined System and JTAG Circuitry

32. Run MARV to verify entire chip conforms to electrical design rules.
33. If the chip is a scan design, run the ATPG tool's design rule checker to verify entire chip conforms to scan design rules.
34. Verify entire chip using estimated delay simulation and timing analysis:
    a. Run PREDIX to generate the JTAG RC file (predix.jtagrc) using that actual I/O placement information.
    b. Run DECAL to calculate estimated delays.
    c. Simulate the gate-level netlist for the entire chip using the same vectors which were used in step 7 to verify the RTL description of the chip's system logic. Then exercise the JTAG logic in a separate simulation.

If errors occur in any of steps 32-34, return to step 28 to correct the core JTAG logic, or do one of the following for each erroneous sub-block:

i) return to step 15 to correct the sub-block's schematic, or
ii) correct the sub-block's RTL description and return to step 5, 6 or 7 to verify the correction.

Re-verify each corrected sub-block individually to the extent desired in section III, part B or C, then continue at step 22, 28 or 32 as desired.

35. If the chip is a scan design, run the ATPG tool to generate scan test patterns.
36. Run TESTPAS to combine the functional and scan test patterns from steps 34 and 35, respectively.

Appendix A: Manufacturing Rules Verifier (MARV) Rules for JTAG

Each rule presented in this appendix has been classified as either a warning (W) or an error (E) based upon the severity of the violation. Warnings are used to indicate a possible violation of JTAG specification requirements which will not cause any failure in the design methodology or manufacture. As such, a warning may be ignored if the condition that it flags is truly what the designer intended to implement. On the other hand, errors must be corrected.

Appendix A.1: General Rules for H4EPlus

1. (E, W)All the MARV rules that apply to non-JTAG input macros apply to input BSC macros and to TCK, TMS, TDI, TRSTB macros.
2. (E, W)All the MARV rules that apply to non-JTAG output macros apply to output BSC macros and to TDOUT macros.
3. (E, W)All the MARV rules that apply to non-JTAG bidirectional instances apply to bidirectional BSC instances. A bidirectional BSC instance is constructed from a bidirectional output BSC macro and a bidirectional input BSC macro.
4. (E, W)All the MARV rules that apply to non-JTAG bidirectional output macros apply to bidirectional output BSC macros.
5. (E, W)All the MARV rules that apply to non-JTAG bidirectional input macros apply to bidirectional input BSC macros.
6. (E, W)All the MARV rules that apply to non-JTAG oscillator macros apply to oscillator BSC macros.
7. (E) For peripheral JTAG macros, an I/O site can be shared only in the following ways.
   a. Any normal drive input BSC macro and any normal drive non-JTAG input macro can share its I/O site with a parallel/slave buffer used in JTAG or non-JTAG high-drive output and high-drive bidirectional macros.
   b. A TDBUF macro can share its I/O site with a parallel/slave buffer used in JTAG or non-JTAG high-drive output and high-drive bidirectional macros
   c. A bidirectional BSC instance is constructed from a bidirectional output BSC macro and a bidirectional input BSC macro. A bidirectional input BSC macro shares a site with the bidirectional output BSC macro to which it gets connected.
8. (E) The special JTAG I/O buffer cells which do not have pads can only be placed in the following ways:
   a. On unused I/O sites.
   b. On output power and ground sites.
   c. All buffer macros except CKDRCC1, CKDRCC2, TDBUF, ISOR and ENSCANJ can also be placed on input power and ground sites.
9. The paralleled output buffer portion of a hi-drive, such as an ON32, can reside on an output-ground I/O site if that ground is an OVSSP macro.
10. (E) When placing I/O macros and JTAG buffers in the periphery, the user must leave room for hi-drive parallel/slave buffers. (Paralleled buffers cannot be placed on power sites.) There also must be enough empty I/O sites for VERILOG2X to place all BUFX and INVX macros used in the design.
11. (E) Only peripheral macros can be located in the periphery of a chip. Peripheral macros must be located in the periphery of a chip.

Appendix A.2: Test Access Port Connections

The rules in this section ensure that the Test Access Port is implemented correctly.

1. (W)There must be one and only one macro from the set {TCK, TCKT, TCKH, TCKHT, TCKX, TCKTX, TCKHX, TCKHTX} in the design.
2. (E) There must be one and only one macro from the set {TMS, TMST, TMSX, TMSTX} in the design.
3. (E) There must be one and only one macro from the set {TDI, TDIT, TDIX, TDITX} in the design.
4. (E) There must be one and only one TDOUT or TDOUTX macro in the design.
5. (E) There must not be more than one macro from the set {TRSTB, TRSTBT, TRSTBX, TRSTBTX} in the design.
6. (E) There must be a pullup resistor connected to the IC ports of {TMS, TMST, TMSX, TMSTX, TDI, TDIT, TDIX, TDITX, TRSTB, TRSTBT, TRSTBX, TRSTBTX} macros.

Appendix A.3: JTAG Conformance

The rules in this section ensure that the design conforms to the internal requirements of the JTAG specification. Since most of these are not required in order to have a fully functional device they are warnings.

1. (W)The number of BPREG macros in the design must be greater than zero.
2. (E) There must not be more than one BPREG macro in the design.
3. (E) There must not be more than one IDREG macro in the design.
4. (E) If the device I. D. code is specified in the design information then there must be one IDREG macro in the design.
5. (E) If there is one IDREG macro in the design then the value set on the D31 to D0 pins must match the I. D. code specified in the design information. D31 is the most significant bit. If a bit is 1(0), the corresponding D-port must be connected to VDD (VSS).
6. (W)If any 3-state BSC's are used in the design then there must be one or more instances of a macro from the set {ENSCANI, ENSCANJ}.

Appendix A.4: JTAG I/O Scan Ring

The rules in this section ensure that the connection and placement of JTAG I/O macros in the periphery conform to the H4EPlus array implementation of JTAG.

1. (E) No BSC JTAG macros may be placed between a TDI macro and a TDOUT macro in the direction clockwise from the TDI macro.
2. (E) The fanout of the TDIP port of the TDI macro must be one. The TDIP port of the TDI macro must be connected to the TDI port of a peripheral BSC or the TDI port of a TDBUF macro or the TDOUT macro.
3. (E) The fanout of the TDO port of every peripheral BSC or TDBUF macro must be one. The TDO port of such a macro must be connected to the TDI port of another peripheral BSC or the TDI port of a TDBUF macro or the TDOUT macro.
4. (E) The fan-in of the TDI port of every peripheral BSC and TDBUF macro must be one. The TDI port of such a macro must be connected to the TDO port of another peripheral BSC, TDBUF, or the TDIP port of the TDI macro.
5. (E) The fanin of the TDI port of the TDOUT macro must be one, and must be connected to the TDO port of a peripheral BSC, TDBUF, or the TDIP port of the TDI macro.
6. (E) Starting from the TDI macro, the order of the peripheral BSC's obtained by tracing fanouts of their TDO ports must be the same as the order obtained by traversing I/O sites in the counter-clockwise direction from the TDI macro.
7. (W)If there are any unused I/O sites then there should be zero ENSCANI macros.
8. (E) The number of I/O sites between a peripheral BSC and the fanout instance of its TDO port must be <= 7 (not including the I/O sites of the driver and the receiver).

Appendix A.5: JTAG Clock & Control Signal Distribution

The rules in this section ensure correct placement and connection of the JTAG I/O control signals CKDR, SHDR, UDDR, IMC and OMC. An “E” or “W” in parenthesis classifies each rule as either an error or a warning.

The following rule applies to all six of these signals:

1. (E) All JTAG buffers must reside within 25 I/O sites of the nearest INPVSS or BOTHVSS macro, and within 25 I/O sites of the nearest INPVDD or BOTHVDD macro.

Appendix A.5.1: CKDR Distribution

The rules in this section verify proper distribution of the CKDR signal in the periphery, as shown in Figure 3-1.

1. (E) There must be one and only one occurrence of each of the following macros: CKDRCC1, CKDRCC2 and CKDRMID.
2. (E) The CKDRMID must be driven by a core macro.
3. (E) CKDRMID can only drive CKDRCC1 and CKDRCC2.
4. (E) CKDRCC1 and CKDRCC2 can only be driven by a CKDRMID.
5. (E) CKDRCC1 and CKDRCC2 cannot drive same net.
6. (E) CKDRCC1 and CKDRCC2 can only drive the CKDR port of peripheral BSC's. Conversely, the CKDR port of peripheral BSC's can only be driven by either CKDRCC1 or CKDRCC2.
7. (E) CKDRCC1 must drive only the CKDRNET1 net.
8. (E) CKDRCC2 must drive only the CKDRNET2 net.
9. (E) The CKDRMID must reside in an I/O site between TDI and CKDRCC1, or between CKDRCC2 and TDOUT, or between TDOUT and TDI.
10. (E) There must not be any common IO sites among IO sites covered by physical CKDRNET1 and physical CKDRNET2.
11. (E) CKDRCC1 must reside at an IO/power/ground site covered by physical CKDRNET1. CKDRCC2 must reside at an IO/power/ground site covered by physical CKDRNET2.

Appendix A.5.2: SHDR, UDDR Distribution

The rules in this section verify proper distribution of the SHDR and UDDR signals in the array periphery, as shown in Figure 3-2. The rules are given for SHDR explicitly. These need to be repeated for UDDR by substituting “UDDR” wherever “SHDR” appears.

1. (E) There must be two and only two occurrences of the SHDR macro.
2. (E) Both SHDR's cannot drive the same net.
3. (E) Each SHDR must be driven by a core macro.
4. (E) There must be one and only one occurrence of ISOR macro.
5. (E) Each SHDR can only drive the SHDR port of peripheral BSC's. Conversely, the SHDR port of peripheral BSC's can only be driven by a SHDR.

In Figure 3-2, the SHDR macro driving net1 is called ‘shdr-i1’ and the SHDR macro driving net2 is called ‘shdr-i2’, for the purpose of explanation.

6. (E) shdr-i1 must reside in an I/O site between TDOUT and ISOR, as IO sites are traversed counter-clockwise from TDOUT. shdr-i2 must reside in an I/O site between TDOUT and ISOR, as IO sites are traversed clockwise from TDOUT.
7. (E) All the peripheral BSC's on IO sites between TDOUT and ISOR, as IO sites are traversed counter-clockwise from TDOUT, must have their SHDR ports driven by shdr-i1. All the peripheral BSC's on IO sites between TDOUT and ISOR, as IO sites are traversed clockwise from TDOUT, must have their SHDR ports driven by buffer shdr-i2.

Appendix A.5.3: IMC, OMC Distribution

The rules in this section verify the distribution of the IMC and OMC signals in the periphery, as shown in Figure 3-3. The rules are given for IMC explicitly. These need to be repeated for OMC by substituting OMC in place of IMC.

1. (E) There must be two and only two occurrences of IMCDR.
2. (E) Both IMCDR's cannot drive the same net.
3. (E) An IMCDR must be driven by a core macro.
4. (E) Each IMCDR can only drive the IMC port of peripheral BSC's. Conversely, the IMC port of peripheral BSC's can only be driven by an IMCDR.

In Figure 3-3, the IMCDR driving net1 is called ‘imcdr-i1’ and the IMCDR driving net2 is called ‘imcdr-i2’, for the purpose of explanation.

5. (E) imcdr-i1 must reside in an IO/power/ground site between CKDRCC1 and CKDRCC2, as I/O sites are traversed clockwise from CKDRCC1. imcdr-i2 must reside in an IO/power/ground site between CKDRCC1 and CKDRCC2, as I/O sites are traversed counter-clockwise from CKDRCC1.
6. (E) All the peripheral BSC's on IO sites between CKDRCC1 and CKDRCC2, as I/O sites are traversed clockwise from CKDRCC1, must have their IMC ports driven by imcdr-i1. All the peripheral BSC's on IO sites between CKDRCC1 and CKDRCC2, as I/O sites are traversed counter-clockwise from CKDRCC1, must have their IMC ports driven by imcdr-i2.

Appendix B: TAP Controller Design for ATPG Compatibility

The TAP Controller implementation given in the IEEE 1149.1 JTAG specification is shown in Figure B-2. This TAP Controller design is not compatible with scan design rules because:

a. it contains static elements that are not scannable
b. it contains signals which are gated by the TCK clock
c. both the rising and falling edges of TCK are used as active edges.

In order to correct these problems the following steps were taken:

1. Two extra inputs and one extra output have been added to the TAP Controller. The MTST input is active high whenever the design is to be used in ATPG-compatible mode (such as during production test at Motorola).Scan data enters the TAP Controller via the TDI input and leaves via the TDO output. The TDI and TDO mentioned here are ports on the TAP Controller macro and should not be confused with the TDI and TDO pins.
2. All of the flip flops were changed to scannable devices.
3. The inverter in the path generating TCKB was replaced with an exclusive-or gate to ensure that all of the static elements will be clocked on the rising edge of the TCK clock. The paths to the CKIR and CKDR outputs are unaltered since these already clock on the correct edge.
4. NAND gates were added to the following outputs to put them into the specified state during ATPG scan mode:
   a. SL = 1: TDO used as scan output for Instruction Register scan chain.
   b. ENABLE = 1: Enables TDO 3-state output.
   c. RB = 1: Prevents reset of JTAG logic while shifting scan chains.
   d. SHIR = 1: Puts Instruction register into scan mode.
   e. SHDR = 1: Puts Data registers into scan mode.
   f. UDDR = 1: Holds data register shadow latches transparent.

In addition, the TMS input is AND'ed with the MTST input to form the Motorola Scan Enable (MSE) signal. MSE is passed to the scan enable of all flip-flops in the TAP Controller. When high, MSE places these flops in scan/shift mode.

Suppose a core/system flop gets its data from an input BSC having the standard shadow register structure. The latch within this BSC is driven by UDDR, which is derived from clock TCK in Figure B-2. Some ATPG tools cannot handle the condition when a clock derivative (UDDR) propagates through a latch to drive the data input of the system flop. The problem can be overcome in the TAP Controller by replacing the NAND which gates TCK to UDDR with a flip-flop which is clocked on the falling edge of TCK. This has the additional effect of extending the update pulse from half a cycle to a complete cycle, which satisfies the ATPG requirement that the data input to a flop be an NRZ (non-pulsed) waveform.

Since the latch portion of the Instruction Register has been changed to a flop (see Figure 4-1) this problem does not exist on UDIR, which therefore need not be generated by a flop. In fact, UDIR must not be generated by a flop now that it drives a flop clock port instead of the shadow latch. The reason is that ATPG tools require flip-flops to have pulsed clocks.

The new functionality for UDDR is shown in the waveform diagram of Figure B-1, which demonstrates the loading of a data register.

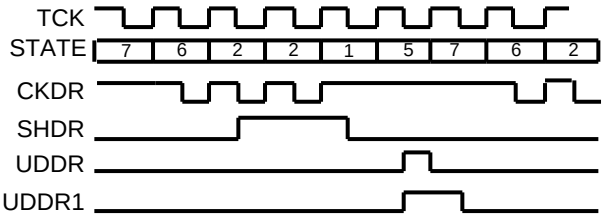

Figure B-1 JTAG Control Signals Waveform Diagram

The waveform diagram shows the operation of the TAP Controller while performing the fastest cycle of loading and updating the data register. This shows the shortest time possible, during JTAG operation, between UDDR going inactive and the next occurrence of a CKDR pulse. UDDR is the current update signal while UDDR1 is the signal that is generated by the new TAP Controller design in Figure B-3. This diagram shows that the addition of a flip-flop on the UDDR signal causes no functional change in the operation of the JTAG boundary scan circuitry.

Some additional circuitry has been added to the TAP Controller to improve its fault coverage:

1. A large number of faults on the NAND gates on the left are not detectable because the TMS line which feeds into them also puts the flip flops which observe them into scan mode. This is resolved by adding gating to allow the TDI input to control the NAND gates when the device is in ATPG test mode.
2. Faults on gates feeding the CKDR, CKIR, and UDIR outputs are undetected because they are unobservable. (MTST overrides the TAP Controller state for control of these signals during ATPG test mode because ATPG tools requires control of all clocks from a pin, in this case the TCK pin.) The fault coverage is improved by monitoring all three signals with an exclusive-or gate, which is observed by a flop that is added to the scan chain.

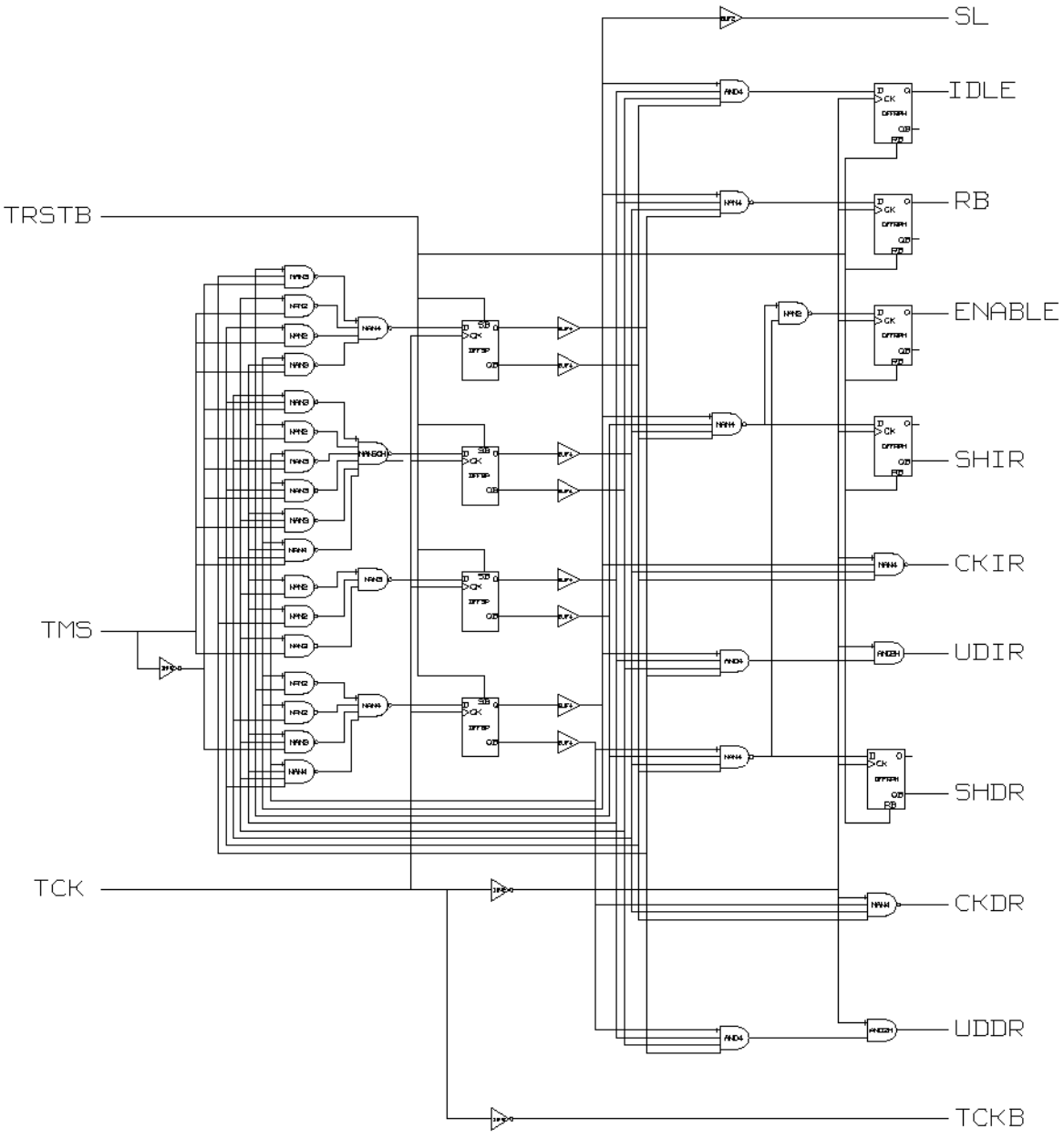

Figure B-2 TAP Controller Shown in IEEE 1149.1 JTAG Specification

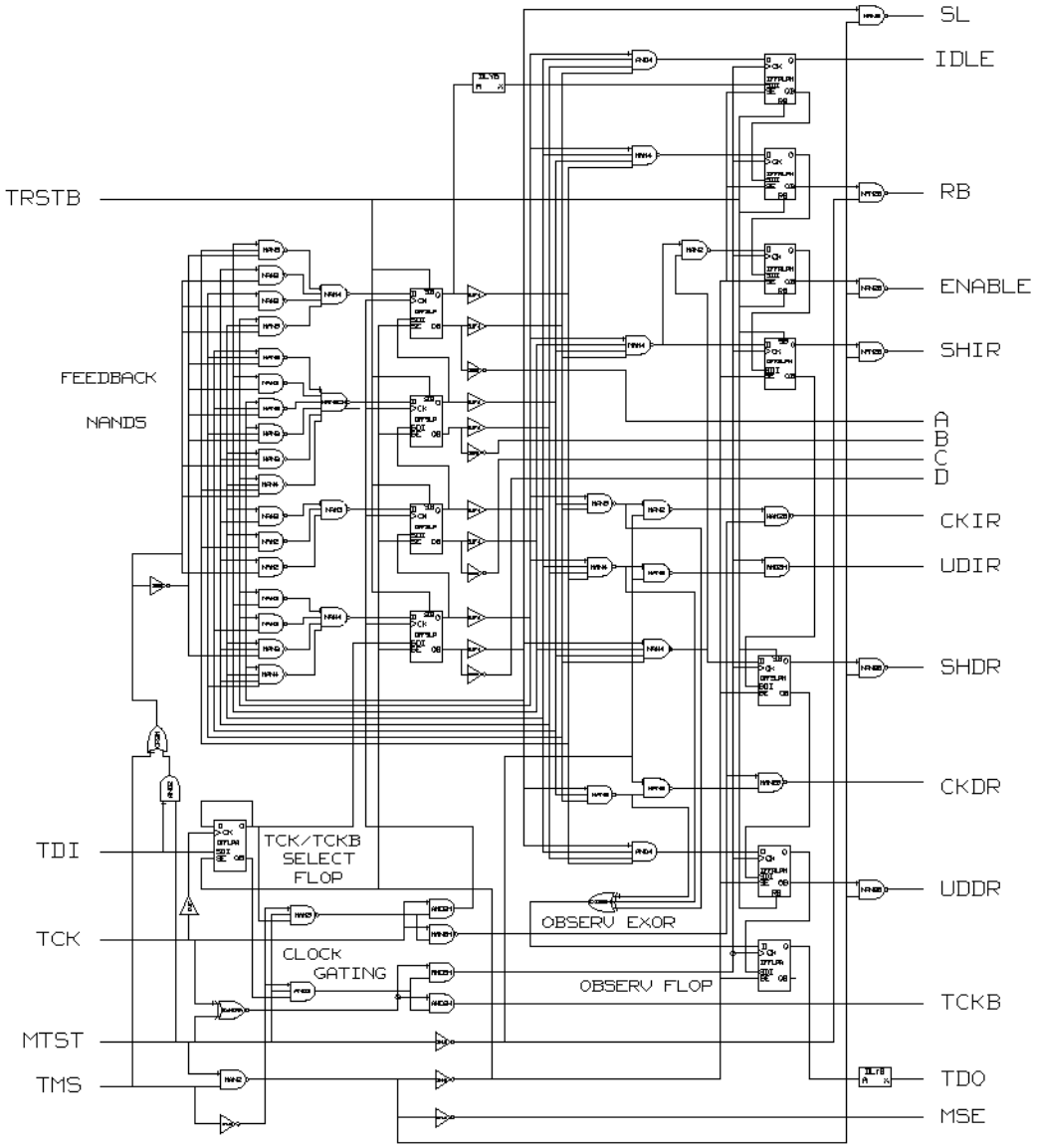

Figure B-3 ATPG-Compatible TAP Controller (FMC_TAPCB).

Notes:

ASIC REGIONAL DESIGN CENTERS – U.S.A.

California, San Jose
(408) 749-0510

Georgia, Atlanta
(404) 729-7100

Illinois, Chicago
(847) 413-2500

Massachusetts, Marlborough
(508) 481-8100

ASIC REGIONAL DESIGN CENTERS – International

European Headquarters

Germany, Munich
(089) 92103-306

England, Aylesbury, Bucks
(0)1296) 395252

France, Velizy
(01) 3463900

Holland, Best
(04998) 61211

Hong Kong, Tai Po
(852)2666-8333

Israel, Tel Aviv
(09) 590-303

Italy, Milan
(02) 82201

Japan, Tokyo
(03) 440-3311

Sweden, Stockholm
(08) 734-8800

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447 or 602-303-5454

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-81-3521-8315

MFAX: RMFAX0@email.sps.mot.com -TOUCHTONE (602) 244-6609
INTERNET: http://Design-NET.com

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

Trademarks
H4EPlus, DECAL, and TestPAS are trademarks of Motorola, Inc.
Verilog and Gate Ensemble are trademarks of Cadence Design Systems, Inc.
Synopsys is a registered trademark of Synopsys, Inc.