MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this document
by AN1500/D

AN1500

IEEE Std. 1149.1 Boundary Scan for H4C™ Arrays

With H4CPlus™ and H4EPlus™ Supplement

Prepared by: Roy Jones and Nick Spence
Edited by: Clarence Nakata
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 H4C 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. describes the macros which have been added to the H4C library to facilitate designing boundary scan circuitry into an H4C gate array.

Section 3. 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. describes how to add boundary scan to a chip whose system logic has been designed using conventional scan techniques. Mustang™, Motorola’s scan ATPG (Automatic Test Pattern Generation) tool, is used to test the JTAG circuitry as well as the system scan circuitry.

Section 5. 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. describes the CAD design flows used when designing an H4C array which incorporates JTAG boundary scan.

Appendix A lists the ERC (Electrical Rule Checker) 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 “Mustang-Compatible JTAG” discussed in Section 4.

Appendix C describes how to build the EDIFMERGE “Attribute file” for JTAG designs using Synopsys logic synthesis.

Appendix D provides H4CP specific procedures.

2. H4C JTAG Macro Descriptions

Technical data, including logic diagrams, for all JTAG macros in the H4C library can be found in the H4C 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.

Non-JTAG hi-drive output and bidirectional macros have always carried a “hi-drive” property to differentiate them from their normal-drive counterparts. However, hi-drive versions of the JTAG output and JTAG bidirectional macros have no such hi-drive property. Instead, there is a separate macro for each JTAG hi-drive so that the timing of these macros can be modeled correctly.

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_TAPC).

The IDREG and BPREG are hard macros in the H4C library. Motorola will assign device identification codes according to the following format:

```
Bit #:   31-28 27---22  21----------1211----------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 by GateEnsemble™. 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_TAPC 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_TAPC has been placed and routed as a single entity by Gate Ensemble. 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_TAPC is shown in Figure B-4 in Appendix B, which discusses this macro in detail.

2.3 Special Purpose Macros

2.3.1 TAP macros: TCK, TMS, TRSTB, TDI, TDO and TDOA.

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.

A functional diagram of the TDO macro is shown in Figure 2-1. The 'IR' port receives scan data from the TDO port of the Instruction Register. The 'DR' port receives scan data from whichever JTAG data register is activated by the current JTAG instruction. Therefore the multiplexer contained in the TDO macro selects either the Instruction Register or the currently active JTAG data register to be shifted out through the TDO pin.

The TDO macro is used with the "small array" scheme for distribution of the JTAG control signals. (See Section 3.3.)

The TDOA macro is functionally identical to TDO. TDOA is used with the "large array" scheme for distribution of the JTAG control signals. (See Section 3.2.)

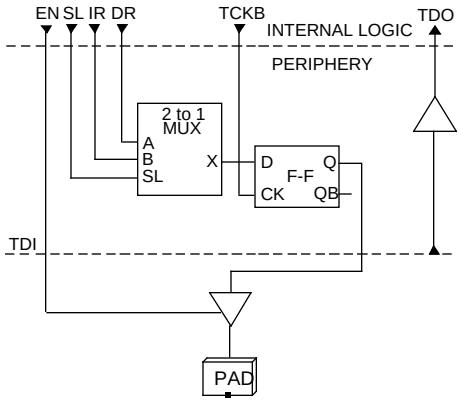

Figure 2-1 TDO Macro (TDO, TDOA)

2.3.2 ENSCANI/J/P

The ENSCAN BSC's are used to drive the enable port of 3-state bidirectional and output BSC's. The ENSCANI, ENSCANJ, and ENSCANP are functionally identical. The differences among the three are:

ENSCANJ must reside on a non-power I/O site.

ENSCANP must reside on a power I/O site.

ENSCANI must reside in the array core.

In the example of Figure 2-2 an ENSCANP 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.

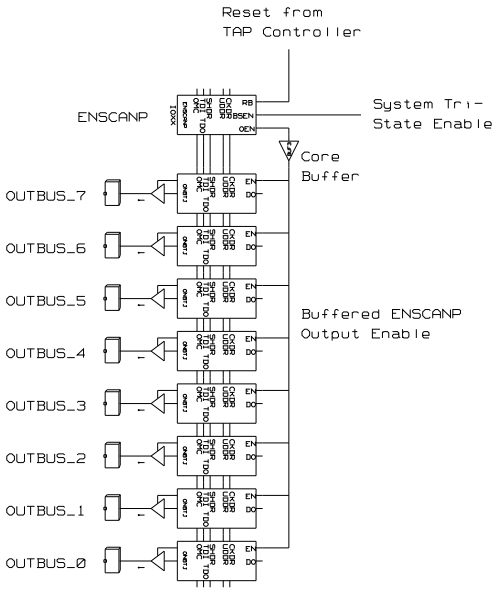

Figure 2-2 ENSCANP Driving 3-State Enable of 8-bit Output Bus

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 TDO/A macro, passes through all ENSCANI's, then gets multiplexed with the scan paths from all other JTAG data registers before passing to the DR port of the TDO macro 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 ENSCANP's or ENSCANJ's.

2.3.3 TDBUF(P)

If consecutive peripheral BSC's are separated by more than seven I/O sites, a TDBUF or TDBUFP buffer macro must be inserted between them since a BSC's TDO output has limited drive strength. The TDBUF(P) 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."

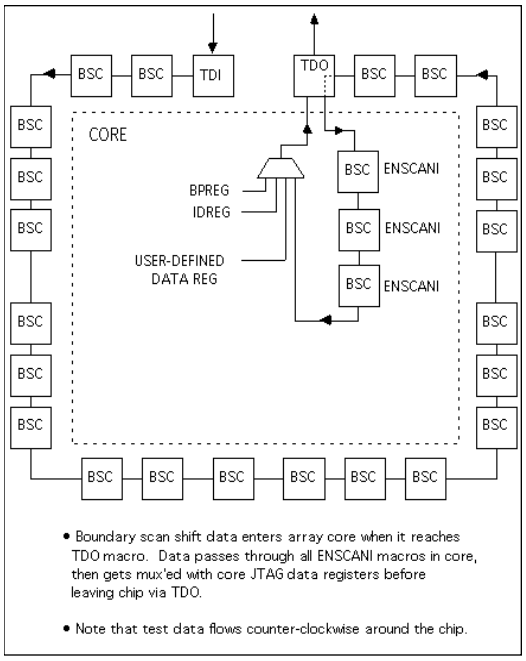

Figure 2-3 Position of ENSCANI 3-State Enable Macros within I/O Boundary Scan Register

3. JTAG Clock & Control Signal Distribution

3.1 Overview

On Motorola's sub-micron H4C 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
   i) 100% utilization in the periphery versus 60-70% in the core (i. e., no unused gates in the periphery).
   ii) transistor sizes can be optimized to their small, non-varying loads
   iii) 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 cannot occur when shifting the boundary scan register since all BSC's share a common clock net.

The five JTAG clock and control signals are CKDR, SHDR, UDDR, IMC and OMC. Two methods are provided for distributing these signals to the boundary scan cells located in the periphery of H4C arrays. The "large or high speed array" scheme can be used to maximize performance on any array size, but it must be used on the larger arrays (H4C086 and above) to ensure that edge-rate limits are not violated on the peripheral clock and control lines, which have a large number of BSC loads. The "small or low speed array" scheme is more simply implemented and can be used on smaller arrays which have no such edge-rate problem (H4C057 and below), when maximum performance is not required. The "large or high speed array" scheme should enable the boundary scan circuitry to operate at > 25Mhz.

In Figures 3-1 to 3-6, 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:

i) maximize the drive capability of these buffers by utilizing the large transistors that normally drive off-chip, and to
ii) 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.

Use of the "P" versions of these buffers allows them to be placed on power sites, which conserves I/O sites for other uses such as hi-drive outputs.

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 Large or High Speed Arrays

3.2.1 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 3-2, 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. In addition, the Gate Ensemble place and route software can correctly model metal interconnect resistance and capacitance (RC's) on this net only if it is not a closed loop. (Gate Ensemble must see only one path from a net's driver to any given load on that net.) 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 TDOA macro. However, closing the CKDR ring on the opposite side of the chip from TDOA is important because it guarantees that, over any range of operating conditions, there will never be race conditions/hold time violations during shifting of the I/O boundary scan register. This is true due to the fact that all I/O BSC's share a common clock (CKDR) net. Nor is there a "bus contention" problem on the CKDR ring, due to the wide separation (roughly two sides of the chip) and minimal skew between the CKDRCC1 and CKDRCC2 buffers which drive this ring. However, Gate Ensemble cannot correctly model distributed RC's for nets driven by more than one source if these sources are active simultaneously. The solution is to capture the schematic, or write the HDL circuit description, such that, as in Figure 3-1, CKDRCC1 drives all BSC's on CKDRNET1 and CKDRCC2 drives all BSC's on CKDRNET2, so that the CAD system actually sees a gap in the ring. The performance of the CKDR ring is identical with or without the gap under the following conditions (refer to Figure 3-1):

i) branches (a) and (c) are perfectly balanced in terms of number of loads and length of metal interconnect

ii) branches (b) and (d) are perfectly balanced in terms of number of loads and length of metal interconnect

iii) CKDRMID is perfectly centered between CKDRCC1 and CKDRCC2.

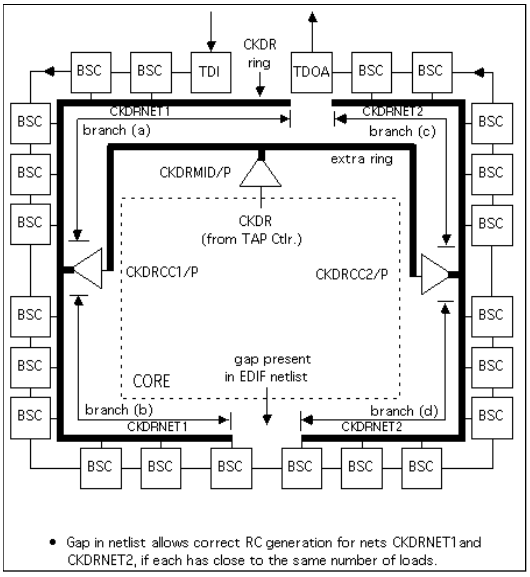

Figure 3-1 CKDR Distribution (Netlist Interconnect for Large or Fast Arrays)

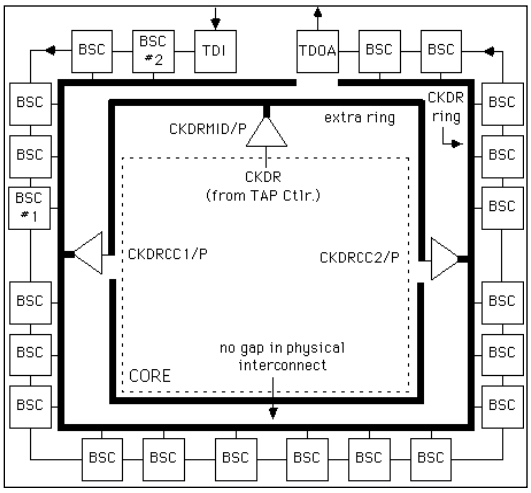

Figure 3-2 CKDR Physical Distribution (Physical Interconnect for Large/Fast Arrays)

In reality, the ASIC designer will not be able to achieve this perfect balance. However, RC's for loads on the CKDR ring, which has no gap in silicon, can be calculated within 5% error by Gate Ensemble if:

iv) the number of loads on branch (a) is within 15% of the number of loads on branch (c)

v) the number of loads on branch (b) is within 15% of the number of loads on branch (d)

vi) the number of I/O sites between CKDRMID and CKDRCC1 is within 15% of the number of I/O sites between CKDRMID and CKDRCC2.

ERC errors are generated if the above conditions are not met. In addition, ERC warnings are generated if the following two conditions are not met:

vii) the number of loads on branch (a) is within 15% of the number of loads on branch (b)

viii) the number of loads on branch (c) is within 15% of the number of loads on branch (d)

Simulation accuracy does not suffer if (vii) and (viii) are violated, therefore the chip designer may choose to ignore these two ERC warnings. However best performance is achieved by adhering to these two conditions.

Note: Since the gap in Figure 3-1 is not a gap in silicon, it does not correspond to any particular I/O site and therefore its placement is not constrained by the availability of an unused I/O site. The designer has complete freedom as to where this gap is "placed," which is done by connecting some BSC's to CKDRCC1 and the others to CKDRCC2 as in Figure 3-1. Consequently, the imbalance between the number of loads on CKDRNET1 and the number of loads on CKDRNET2 should never be greater than one (which would occur if there are an odd number of peripheral JTAG macros driven by CKDR).

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.

3.2.2 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 3-3).

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. Gap 1 and gap 2 are both actual physical cuts in the SHDR ring, unlike the gap in the CKDR ring. As a result the ASIC designer does not need to balance the number of BSC loads on nets 1 and 2 in order for Gate Ensemble to correctly model the distributed RC's for these loads. Gap 1 is designed into the TDOA macro. To create gap 2 the ASIC designer must place a special "ISO" macro on a power or I/O site near the point diametrically opposite from the TDOA macro. This ISO macro cuts the SHDR and UDDR rings. A break in these lines does not cause timing problems as it would in the CKDR line, and it frees the designer from the need to balance loads on these lines as he has to do on CKDR. Even so, ERC warnings are generated if the following two conditions are not met:

i) the number of loads on branch (a) is within 15% of the number of loads on branch (b)

ii) the number of loads on branch (c) is within 15% of the number of loads on branch (d)

As stated above, simulation accuracy does not suffer if (i) and (ii) are violated, therefore the chip designer may choose to ignore these two ERC warnings. However best performance is achieved by adhering to these two conditions. The ERC warnings are simply to alert the designer that his buffer placement will not achieve minimum insertion delay and maximum shift speed. **Note: The ISO macro has nothing to do with the CKDR gap in Figure 3-1.**

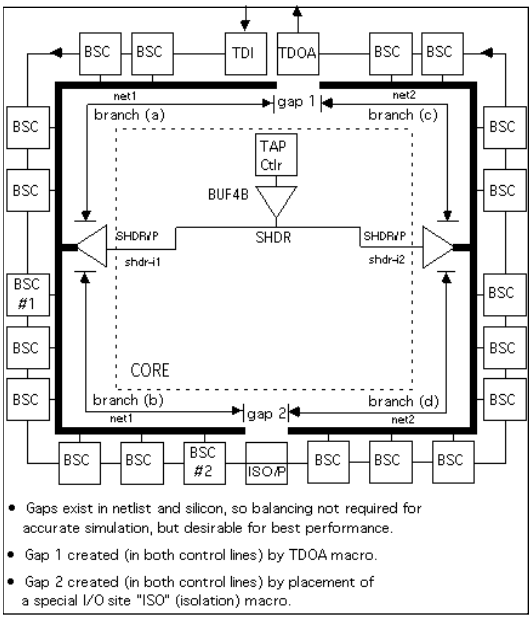

Figure 3-3 SHDR, UDDR Distribution (Large/Fast Arrays)

3.2.3 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 TDOA and ISO 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 Figures 3-3 and 3-4), 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.

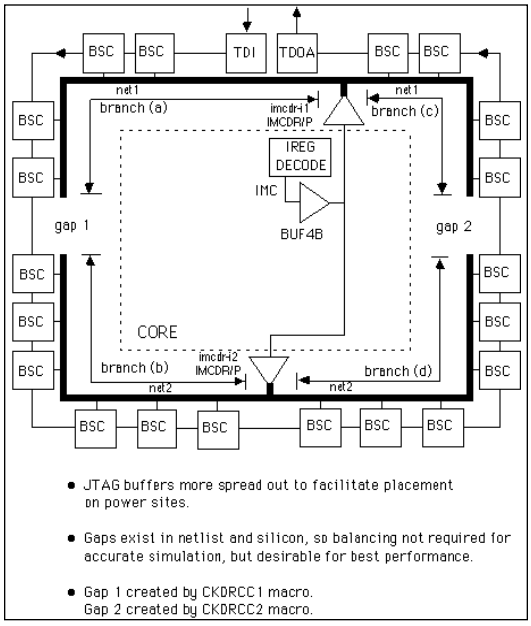

Figure 3-4 IMC, OMC Distribution (Large/Fast Arrays)

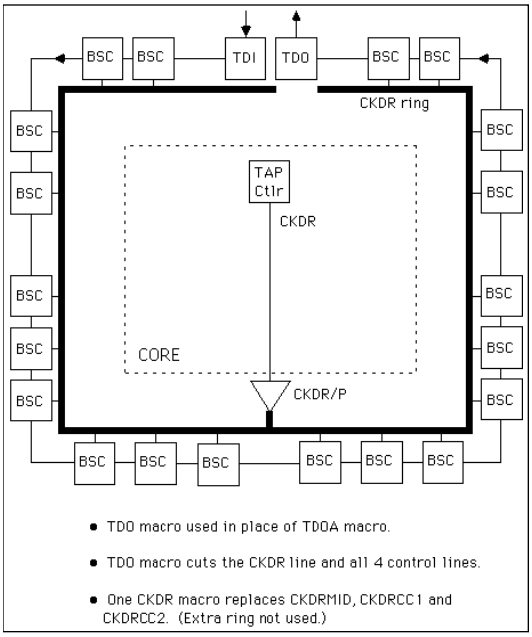

Figure 3-5 CKDR Distribution (Small/Slow Arrays)

3.3 Small or Low Speed Arrays

3.3.1 CKDR Distribution

As shown in Figure 3-5, the TAP controller drives one CKDR buffer, which drives the CKDR ports of all JTAG I/O cells on the chip via the CKDR ring. The CKDRCC1 and CKDRCC2 buffers, and therefore the extra ring, are not needed. Consequently there is no balancing of loads to be done, as for large or high speed arrays. There is a physical cut in the CKDR ring within the TDO macro so that Gate Ensemble can correctly model distributed RC's for BSC loads on this ring. The netlist interconnect matches the physical interconnect, unlike the large/high speed array scheme. The TDO macro replaces the TDOA macro used in the large/high speed array scheme.

3.3.2 SHDR, UDDR, IMC, OMC Distribution

SHDR, UDDR, IMC and OMC all use the same distribution scheme, which is also the same as the CKDR scheme since the extra ring is not being used. As shown in Figure 3-6, the TAP controller drives one SHDR buffer which drives the SHDR ports of all JTAG I/O cells on the chip via the SHDR ring. The ISO macro is not used so that gap 2 in the SHDR and UDDR lines (see Figure 3-3) does not exist. Likewise, gaps 1 and 2 in the IMC and OMC lines (see Figure 3-4) do not exist, since CKDRCC1 and CKDRCC2 are not used in the CKDR scheme. There is only one gap in each control line, and that gap occurs in the TDO macro.

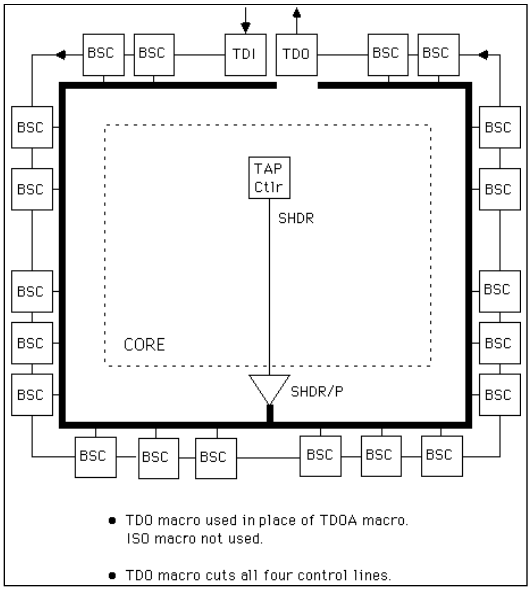

Figure 3-6 SHDR, UDDR, IMC, OMC (Small/Slow Arrays)

4. Mustang-Compatible JTAG

4.1 Introduction

On the Motorola H4C family of CMOS arrays, JTAG boundary scan circuitry has been designed to be compatible with Motorola's Mustang ATPG product, which was designed to do automatic test pattern generation for conventional scan designs. 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 H4C arrays in order to allow it to be tested by Mustang.

The user is assumed to have a working knowledge of Mustang. For more information on Mustang itself refer to Tim Boland's Application Note, number AN1096, entitled "Guidelines for Using the Mustang ATPG System," and to the Mustang User's Guide portion of Motorola's Open Architecture CAD System™ (OACS) documentation.

4.2 Design Overview

Motorola has designed scan-compatibility into the JTAG circuitry in two respects:

1. Special Mustang modeling has been done for the boundary scan cell, which contains a non-scannable latch.
2. The TAP Controller has been modified to include scannable flip-flops, and to satisfy some Mustang timing requirements. For detailed explanations of items 1 and 2 see Appendix B. In addition, the user is responsible for:

- ensuring that no timing problems arise due to clock skews, and
- interconnecting all JTAG circuitry such that Mustang compatibility is maintained.

4.2.1 Handling of Clock Skew

Motorola's H4C 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 Mustang 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 "Mustang test mode" in this document, must be added to the chip to logically reconfigure the JTAG circuitry when Mustang 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 Mustang 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 Mustang compatibility.

4.3 Handling of Clock Skew

The JTAG design can suffer from clock skew problems during Mustang 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 Mustang test mode all flip-flops must 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. (A Mustang scan test consists of three parts: shift in of scan stimulus, pulsing of zero or one of the clocks -- 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 Mustang 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 Mustang 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 Mustang clock pulse mode. (This flop is labeled “TCK/TCKB Select Flop” in Figure B-4 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.

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. The UDIR signal can now be applied separately from the CKIR signal to remove the possibility of skew.

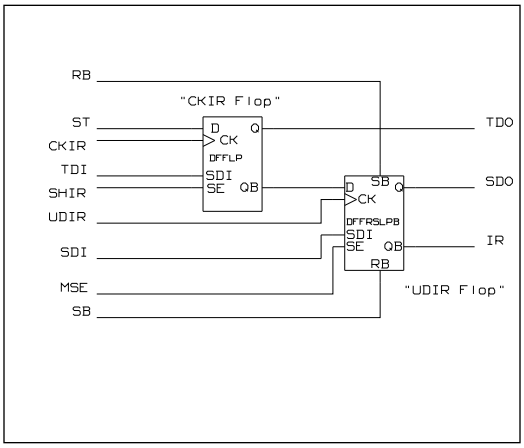

Figure 4-1 Single-Bit Instruction Register Cell (MC_IREG).

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 Mustang 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 Mustang-compatible TAP Controller in Appendix B, Figure B-4, note the inclusion of the following circuitry:

1. A “TCK/TCKB Select Flop” to control the clock gating in Mustang 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 because these flops have separate clocks which pass through different gating logic. A buffer was also added to the clock for the “TCK/TCKB Select Flop” for similar reasons.

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

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 Mustang test mode, either:

i) the clock TCK should be different from the system clock, or

ii) circuitry similar to the “TCK/TCKB Select Flop” and “Clock Select” gates in the TAP Controller should be implemented to prevent Mustang from pulsing both TCK and the system clock in the same clock cycle.

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. Veritime is able to take into account the effects of variations in process/voltage/temperature across a chip.

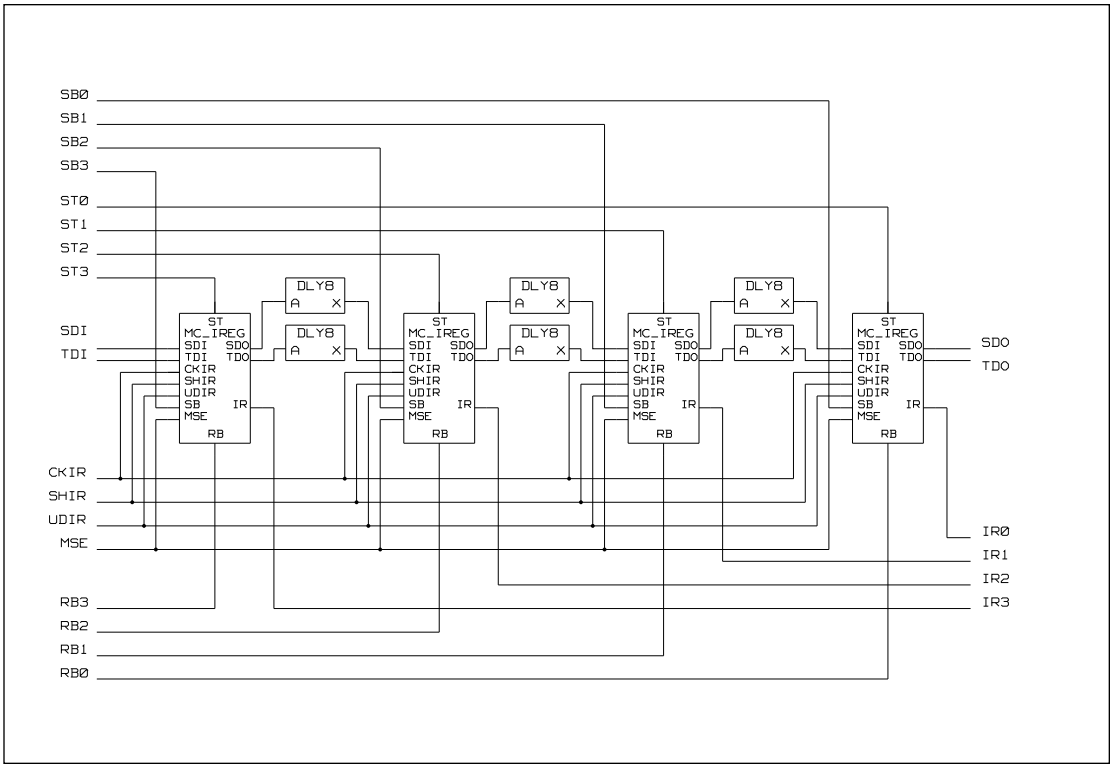

Figure 4-2 MC_IREG4 Functional Diagram

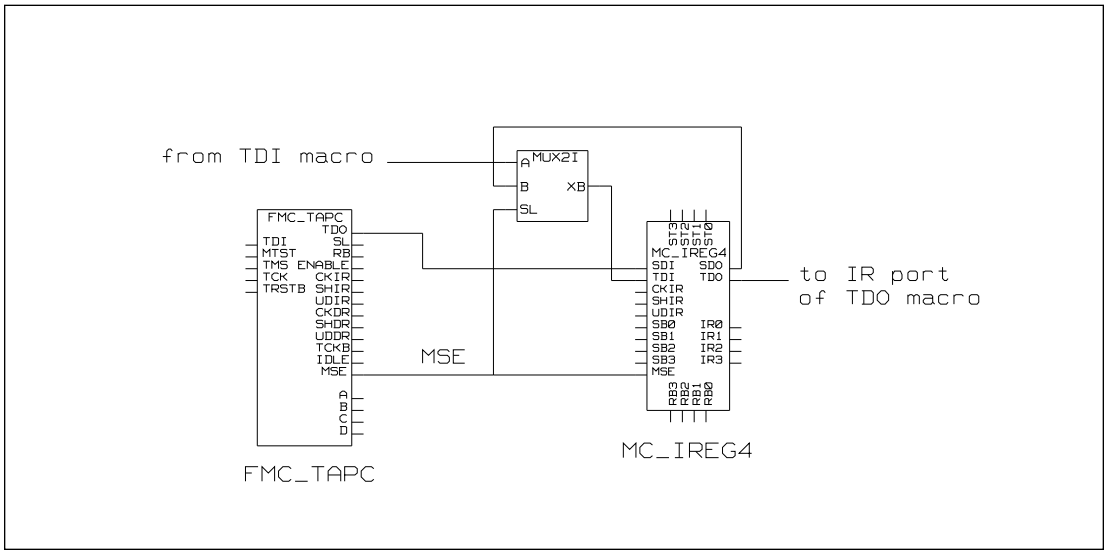

Figure 4-3 Scan Chain Hook-up of MC_IREG4

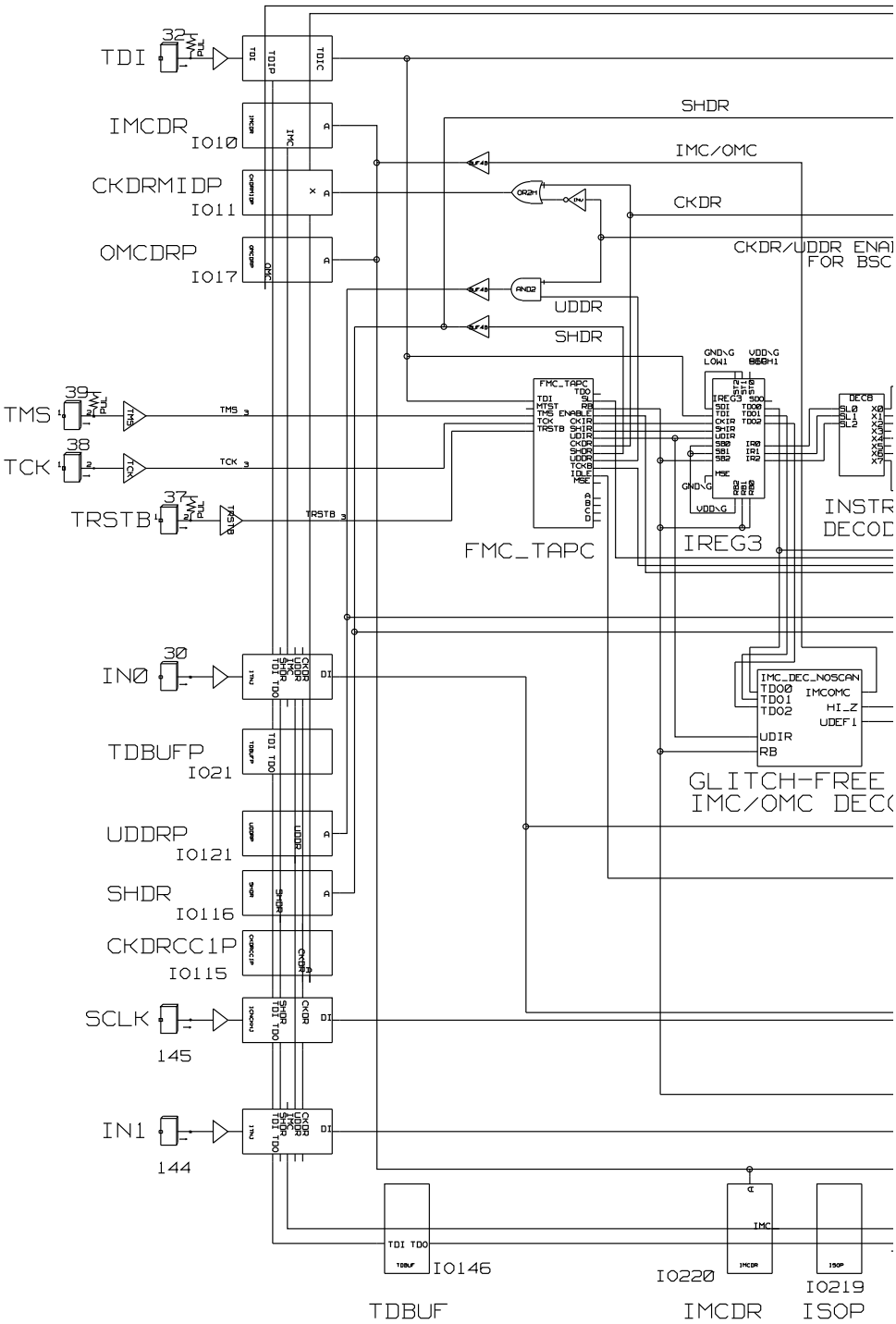

Figure 4-4 Non-Scan JTAG Example Circuit

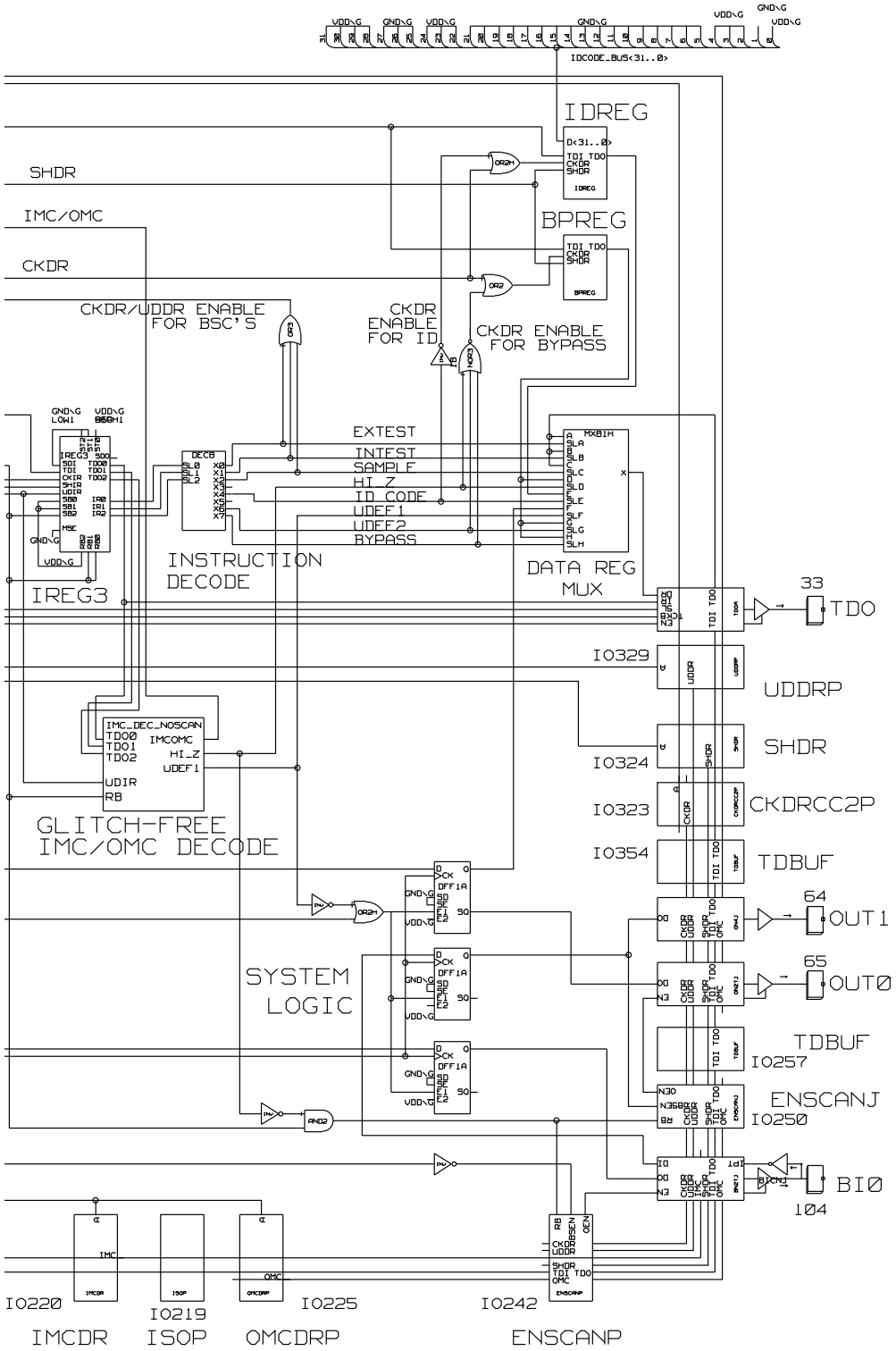

Figure 4-4 Non-Scan JTAG Example Circuit (continued)

4.4 JTAG Circuitry Interconnection

First a non-scan, JTAG design will be discussed to show the essential circuitry required to implement JTAG on H4C arrays. Afterwards, the requirements for a Mustang/scan-compatible JTAG design will be discussed and illustrated.

4.4.1 Non-Scan JTAG Interconnection

If the system logic is not a scan design, Mustang cannot be used for ATPG. 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 H4C 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 H4C library. However, in this example a DEC8 macro from the H4C library is sufficient to implement the Instruction Register decoder.

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 4-4 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.

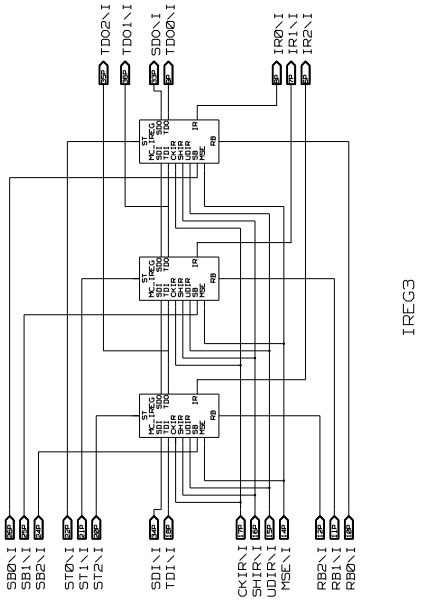

Figure 4-5 IREG3 3-Bit Instruction Register

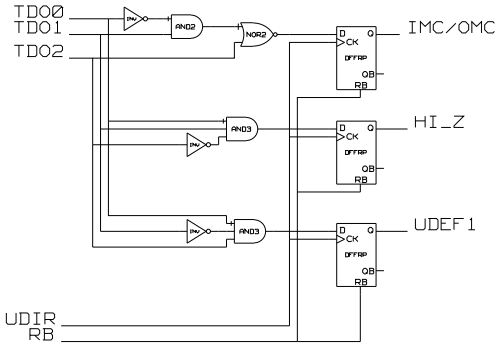

Figure 4-6 Glitch-Free IMC/OMC Decode Block

4.4.2 Mustang-Compatible JTAG Interconnection

In order to make a JTAG design Mustang-compatible the FMC_TAPC TAP Controller of Figure B-4 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 Mustang test mode. The requirements, which are illustrated in Figure 4-7, are as follows:

1. An extra MTST input pin, which will only be active during Mustang 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 Mustang 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 Mustang test mode, during which they are part of one scan chain between TDI and TDO.
3. The Mustang 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 Mustang 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 Mustang 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. Mustang requires all asynchronous control lines to be controlled only by an external input. An asynchronous reset line which enters the chip through a normal input BSC can be controlled by the jlatch (see Appendix B) within that BSC. Therefore the reset line must be gated with MTST in the array core so that the reset will be disabled during Mustang testing. Alternatively, the asynchronous reset can enter the chip through a "sample-only" cell. In this case gating the reset signal with MTST is not required, however the reset is no longer controllable from the JTAG BSC ring.

4.5 Mustang-Compatible JTAG Example Circuit

In Figure 4-7 the non-scan JTAG design in Figure 4-4 has been modified, according to the requirements of Section 4.4-2, to create a Mustang-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 Mustang 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.) As in Figure 4-4, the BSC's and peripheral JTAG buffers are located around the periphery of the schematic. **Note that test data must shift counterclockwise through the BSC's around the periphery of the chip.** 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.

Mustang test mode is established by forcing the MTST pin high. During Mustang test mode the TMS pin controls scan mode (MTST and TMS high), during which bidirectional pins must be disabled. Using TRSTB, instead of TMS, to disable the bidirectionals during scan mode improves the fault coverage of the bidirectionals. 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_TAPC, MC_IREG4, TDO. Also, during Mustang 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 Mustang control file used is as follows:

```
SCAN_OUTPUT TDO
SCAN_OUTPUT OUT0
SCAN_MODE  TMS  1
```

```
SCAN_CLOCK   TCK   1 1
SCAN_CLOCK   SCLK 1 2
SYSTEM_CLOCK TCK   1 1
SYSTEM_CLOCK SCLK 1 2
ASSERT MTST 1
BIDIRECT_CONTROL TRSTB 1
SCAN_MODE       TRSTB 1
```

The fault coverage obtained for this example circuit was >95%. 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 a real-world chip design would be much higher, since the vast majority of the circuitry would be scannable system logic with close to 100% fault coverage.

The implementation of the “System Scan Reg” shown in Figure 4-7 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.

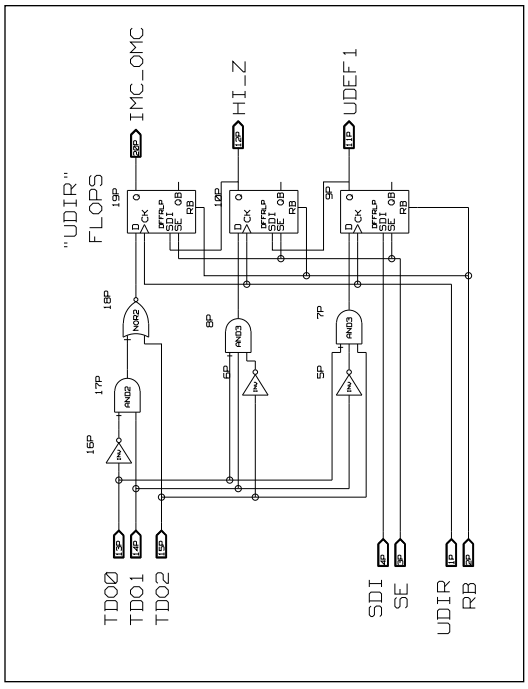

Figure 4-8 Glitch-Free IMC/OMC Decode Block

4.6 Conclusions

The methodology presented here allows Mustang 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 with Mustang. However, Mustang cannot test the gate (G) input to the BSC shadow latches. Nor can Mustang test all of the JTAG Instruction Register decoding logic. Testing of these areas should be achieved by supplementing the Mustang 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 Mustang vectors is accomplished by TESTPAS, which is one of the OACS 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

For each array size there are 32 fixed pad pairs which can interface to 64 high-speed scan channels on the Typhoon tester. These pad pairs are marked with asterisks on each Pad-to-Pin cross reference table in the H4C Design Reference Guide. Test Access Port (TAP) pins TMS, TCK, TDI and TDO each must be assigned to a pin which is bonded to one of these “*” pads. 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/P, ENSCANJ/P, ISO/P, and the special buffers for CKDR, SHDR, UDDR, IMC and OMC:

CKDRMID, CKDRMIDP (large or high speed arrays only)
CKDRCC1, CKDRCC1P (large or high speed arrays only)
CKDRCC2, CKDRCC2P (large or high speed arrays only)
CKDR, CKDRP (small or low speed arrays only)
SHDR, SHDRP
UDDR, UDDRP
IMCDR, IMCDRP
OMCDR, OMCDRP
ISO, ISOP

Referring to the “H4C123 160 QFP PAD-to-Pin Cross Reference” in the H4C Series Design Reference Guide, each non-bonded macro must reside on a specific power or I/O site which:

- does not connect to a pad (e. g., I/O sites 58 & 59), or
- connects to a pad which is not bonded out to a package pin (e. g., I/O sites 47 and 48), or
- connects to a power/ground pad (e. g., I/O sites 50 and 51); the “P” version of the macro is used at power/ground sites.

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

5.1.3 Placement of JTAG I/O Macros within Schematic Capture

The user must assign a package pin number to the "IO_PIN1" property on each I/O macro, including input, output and bidirectional BSC's and the TAP macros. In addition, a FIX property whose initial value is "IOXX" is already attached to each non-bonded JTAG macro, which the user places by substituting the I/O site # for the "XX" portion of the FIX value. In Figure 4-7 the JTAG buffers have been placed according to the "large or fast array" scheme (see Section 3.2), using the "Pad to Pin" for an H4C123 array in a 160QFP package. Note the inclusion of TDBUF's to buffer BSC TDO output ports where the following BSC is >=7 I/O sites away. In determining the pin-out and I/O site placement, follow the guidelines in Section 5.2.

5.1.4 Placement of JTAG I/O Macros within Synopsys/Verilog HDL Flow

For Verilog HDL design entry followed by Synopsys logic synthesis, the designer creates a Verilog netlist for the JTAG I/O and the non-bonded macros. The designer also creates the EDIFMERGE "Attribute" file, which contains I/O site placement information in the form of FIX properties for all of the peripheral JTAG macros. The Verilog netlists for the JTAG I/O, the core JTAG logic, and the system logic are combined into one EDIF netlist using Synopsys. This "Synopsys EDIF" netlist, along with the Attribute file, are subsequently input to EDIFMERGE to create the "Motorola EDIF" netlist required by the OACS tools. (EDIFMERGE creates a Motorola EDIF netlist from the Synopsys EDIF netlist by adding properties which are specific to Motorola's H4C technology. For more information on EDIFMERGE see the Synopsys/EDIFMERGE Application Note for OACS.)

Had Verilog HDL been used instead of schematic capture to enter the circuit in Figure 4-7, the JTAG portion of the Attribute file would appear as shown in Fig. C-1 (Appendix C).

5.2 Guidelines for Finding an ERC-Compatible Chip Pin-out

In both procedures that follow, the JTAG clock and control signal buffers should be placed on power sites to the extent possible.

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 TDOA on a pair of adjacent high-speed scan tester pins. TDI must be counterclockwise from TDO. In the same general area as TDI and TDOA, place TCK and TMS on high-speed tester pins and TRSTB on a normal pin.
2. Place remaining system pins, and any additional power/ground pins required, in conformance with the ERC 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/P 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/P must be inserted on an I/O site within 7 sites of the BSC whose TDO port drives the other's TDI port. (TDBUF/P 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/P and CKDRCC2/P buffers such that, in Figure 3-1, nets CKDRNET1 and CKDRNET2 are balanced as described in Section 3.1.
6. Place the CKDRMID/P buffer equidistant between CKDRCC1/P and CKDRCC2/P as shown in Figure 3-1.
7. Place the ISO/P macro diametrically opposite from TDOA (approximately), as in Figure 3-3.
8. Place the SHDR/P buffers approximately in the center of nets 1 and 2 as shown in Figure 3-3. Do the same for the UDDR/P buffers.
9. Place the IMCDR/P buffers approximately halfway between the CKDRCC1/P and CKDRCC2/P buffers, which create gap 1 and gap 2 in Figure 3-4. Do the same for the OMCDR/P 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, TDOA, TMS, TCK, and TRSTB), and any additional power/ground pins required, in conformance with the ERC 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/P 3-state control BSC's on available I/O or power sites.
3. Choose a pair of adjacent high-speed scan tester pins for TDI and TDOA (with TDI counterclockwise from TDO) such that a line drawn from TDOA 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. In the same general area as TDI and TDOA, place TCK and TMS on high-speed tester pins and TRSTB on any pin.

4. Between every pair of BSC's separated by >7 non-BSC I/O sites, a TDBUF/P must be inserted on an I/O site within 7 sites of the BSC whose TDO port drives the other's TDI port. (TDBUF/P 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/P and CKDRCC2/P buffers such that between each buffer and TDOA 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/P buffer equidistant between CKDRCC1/P and CKDRCC2/P (typically near TDOA). Refer to Figure 3-1.
7. Place the ISO/P macro diametrically opposite from TDOA (approximately), as in Figure 3-3.
8. Place the SHDR/P buffers approximately halfway between the TDOA and ISO/P, in terms of BSC's (see Figure 3-3). Do the same for the UDDR/P buffers.
9. Place the IMCDR/P buffers approximately halfway between the CKDRCC1/P and CKDRCC2/P buffers, in terms of BSC's (see Figure 3-4). Do the same for the OMCDR/P buffers.

6. CAD Design Flows

For Mentor QSIM simulation on the HP/Apollo platform, entry of an H4C design would likely be done via schematic capture. The corresponding "Schematic Capture/QSIM Design Flow" in Section 6.1 follows a "bottom-up" design methodology.

For Verilog simulation on either the Sun or HP/Apollo platform, 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 6.2 and the "Synopsys/Verilog Design Flow" in Section 6.3 both follow a "top-down" design methodology where the chip is initially described behaviorally using Verilog HDL. In Section 6.2 the HDL is manually converted into gates using schematic capture, whereas in Section 6.3 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 footprint as in Figure 4-7.

During pre-layout 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, since the peripheral nets have been shown by SPICE to have no edge-rate problems.** After layout the actual resistance and capacitance values are used, at which time no edge-rate errors should occur on peripheral nets.

6.1 Schematic Capture/QSIM Design Flow

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

1. Use CAPTURE to capture a schematic of the JTAG I/O, following the "Guidelines for Finding an ERC-Compatible Chip Pin-Out" in Section 5.2.
2. Verify JTAG I/O conform to electrical design rules:
   a. Run FLATTEN to generate a QSIM database.
   b. Run NETLIST to generate an EDIF netlist (used by ERC).
   c. Run ERC for peripheral rule checks. Repeat steps 1 and 2 until ERC passes.

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

A. Design Individual Sub-Blocks

3. Use CAPTURE to capture a schematic for one system sub-block "Y".
4. Run FLATTEN to generate QSIM database for Y.
5. (Optional) Verify Y via unit-delay QSIM simulation. Repeat steps 3-5 until Y's functionality is correct.
6. Run NETLIST to create the required Motorola netlists:
   a. an EDIF netlist (used by ERC and DECAL)
   b. a TEGAS/TDL netlist (used by MUSTANG) if the chip is a scan design.
7. Run ERC to verify Y conforms to electrical design rules. If violations occur return to step 3 to correct the schematic.
8. If the chip is a scan design, run the MUSTANG 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 3 to correct the schematic.
9. Verify Y via real-time simulation:
   a. Run DECAL to calculate real-time delays. If edge-rate violations occur, return to step 3 to correct the schematic.
   b. Run INSERT_DELAYS to insert real-time delays into the QSIM database.
   c. Run QSIM. If Y's functionality or timing is incorrect, return to step 3 to correct the schematic.
10. Repeat steps 3-9 for each system sub-block on the chip.

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

11. Use CAPTURE to combine the sub-blocks for all system logic on the chip.
12. Run FLATTEN to generate a QSIM database.
13. Run NETLIST to create an EDIF netlist. Also create a TEGAS/TDL netlist if the chip is a scan design.
14. Run ERC and, if the chip is a scan design, run MUSTANG; then run DECAL and INSERT_DELAYS followed by QSIM real-time simulation.

If violations occur in any of these tools, make schematic corrections for all erroneous sub-blocks. Re-verify each corrected sub-block individually to the extent desired in section II, part A, then return to step 11.

VI. Combine JTAG Circuitry with Chip's System Logic

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

Verify All JTAG Circuitry by Itself (Optional)

16. In CAPTURE, combine the core JTAG logic with the JTAG I/O from step 1.
17. Run FLATTEN to generate a QSIM database.
18. Do unit-delay QSIM simulation, if desired. If step 18 is done, repeat steps 16-18 until functionality is correct.
19. Run NETLIST to create an EDIF netlist. Also create a TEGAS/TDL netlist if the chip is a scan design.
20. Run ERC and, if the chip is a scan design, run the MUSTANG design rule checker; then run DECAL and INSERT_DELAYS followed by QSIM real-time simulation. If violations occur in any of these tools, return to step 16 to correct the schematic.

Verify Combined System and JTAG Circuitry

21. Use CAPTURE to combine the core JTAG logic, JTAG I/O, and system logic.
22. Create required netlists for the entire chip:
    a. Run FLATTEN to generate a QSIM database.
    b. Run NETLIST to create the following:
        i) an EDIF netlist (used by ERC and DECAL)
        ii) a TEGAS/TDL netlist (used by MUSTANG) if the chip is a scan design.
        iii) an "Actual.RC" file (used by DECAL) for each firm macro, such as the FMC_TAPC
23. Run ERC to verify the entire chip conforms to electrical design rules.
24. If the chip is a scan design, run the MUSTANG design rule checker to verify the entire chip conforms to scan design rules.
25. Verify entire chip via real-time simulation:
    a. Run DECAL to calculate real-time delays.
    b. Run INSERT_DELAYS to insert real-time delays into the QSIM database.
    c. Run QSIM.

If errors occur in any of steps 23-25, return to step 15 to fix the core JTAG logic, or return to step 3 to fix any erroneous system sub-blocks. Re-verify each corrected sub-block individually to the extent desired in section II, part A, then continue at step 11, 15 or 21 as desired.

26. If the chip is a scan design, run MUSTANG to generate scan test patterns.
27. Run TESTPAS to combine the functional and scan test patterns from steps 25 and 26, respectively.

6.2 Schematic Capture/Verilog Design Flow

Veritime timing analysis is recommended as a complement to real-time simulations (those using DECAL delays instead of unit-delays).

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 H4C chip.

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

2. Use ASIC_GED to capture a schematic of the JTAG I/O, following the "Guidelines for Finding an ERC-Compatible Chip Pin-Out" in Section 5.2.
3. Verify JTAG I/O conform to electrical design rules:
    a. Run NETLIST to generate an EDIF netlist (used by ERC).
    b. Run ERC for peripheral rule checks. Repeat steps 2 and 3 until ERC passes.

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

A. Convert Behavioral Description to an 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 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. Convert Sub-Block RTL Descriptions to Gate-Level Netlists

8. Use ASIC_GED to capture a schematic for one system sub-block "X".
9. Run NETLIST to create the required netlists for X:
    a. an EDIF netlist (used by ERC and DECAL) and a Verilog netlist
    b. a TEGAS/TDL netlist (used by MUSTANG) if the chip is a scan design.
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 5. If X's functionality is incorrect return to step 8 to correct the

schematic, or correct X's RTL description and return to step 5, 6 or 7 to verify the correction.

11. (Optional) 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. Handle bugs as prescribed in step 10.
12. Run ERC to verify X conforms to electrical design rules. If violations occur return to step 8 to correct the schematic.
13. If the chip is a scan design, run the MUSTANG 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 8 to correct the schematic.
14. 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.
15. Repeat steps 8-14 for each system sub-block on the chip.

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

16. Use ASIC_GED to combine the sub-blocks for all system logic on the chip.
17. Run NETLIST to create Verilog and EDIF netlists. Also create a TEGAS/TDL netlist if the chip is a scan design.
18. Run ERC and, if the chip is a scan design, run MUSTANG. 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:
    i) return to step 8 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.
    iii) Re-verify each corrected sub-block individually to the extent desired in section III, part B, then return to step 16.

IV. Combine JTAG Circuitry with Chip's System Logic

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

Verify All JTAG Circuitry by Itself (Optional)

20. In ASIC_GED, combine the core JTAG logic with the JTAG I/O from step 2.
21. Run NETLIST to create EDIF and Verilog netlists. Also create a TEGAS/TDL netlist if the chip is a scan design.
22. Run ERC and, if the chip is a scan design, run the MUSTANG design rule checker; then run DECAL followed by Verilog real-time simulation. If violations occur in any of these tools, return to step 20 to correct the schematic.

Verify Combined System and JTAG Circuitry

23. Use ASIC_GED to combine the core JTAG logic, JTAG I/O, and system logic.
24. Run NETLIST for the entire chip to create the following:
    a. an EDIF netlist (used by ERC and DECAL) and a Verilog netlist
    b. an "Actual.RC" file (used by DECAL) for each firm macro, such as the FMC_TAPC
    c. a TEGAS/TDL netlist (used by MUSTANG) if the chip is a scan design.
25. Run ERC to verify entire chip conforms to electrical design rules.
26. If the chip is a scan design, run the MUSTANG design rule checker to verify entire chip conforms to scan design rules.
27. 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 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 25-27, return to step 19 to correct the core JTAG logic, or do one of the following for each erroneous sub-block:
        i) return to step 8 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, then continue at step 16, 19 or 23 as desired.
28. If the chip is a scan design, run MUSTANG to generate scan test patterns.
29. Run TESTPAS to combine the functional and scan test patterns from steps 27 and 28, respectively.

6.3 Synopsys/Verilog Design Flow

Tieoff's and buses require special handling when a Verilog netlist created by the OACS "NETLIST" tool is to be used in Synopsys (e. g. in steps16, 23 or 25 below). In an OACS netlist, the two statements which define VDD and VSS tieoff's are not recognized by Synopsys and must be modified as described in a preliminary application note entitled "High-Level Design Methodology for OACS 2.0." Also, in an OACS netlist buses are separated into individual bits, which need to be recombined into buses to properly connect to other Verilog HDL modules within Synopsys. An example of this is shown in the "High-Level Design" Application Note.

The "Synopsys delays" mentioned below are the macrocell prop delays calculated by Synopsys during the process of synthesizing a sub-block. These delays must be written out to a "Verilog.timing" file in order to be used during simulation of a synthesized sub-block. Synopsys delays are accurate to within approximately 5% of DECAL delays.

Veritime timing analysis is recommended as a complement to real-time simulations (those using either DECAL or Synopsys delays instead of unit-delays).

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 H4C chip.

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

2. Create both a Verilog netlist and a "Motorola EDIF" netlist for the JTAG I/O only (no core module instantiation).
3. Verify JTAG I/O conform to electrical design rules:
   a. Run ERC for peripheral rule checks. If violations occur (other than those due to the absence of a core module instantiation) return to step 2 to correct the JTAG I/O netlists.

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, or
   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 flat EDIF netlist.
    b. Run EDIFMERGE to generate a "Motorola EDIF" netlist (used by ERC and DECAL).
    c. Run NETLIST to create a Verilog netlist from the "Motorola EDIF" netlist. Also generate a TEGAS/TDL netlist for MUSTANG if the chip is a scan design.
12. Run ERC 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.
13. If the chip is a scan design, run the MUSTANG 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.
14. 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.
15. 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

8a. Use ASIC_GED to capture a schematic for one system sub-block "Y".

9a. Run NETLIST to create the following netlists for Y:

a. an EDIF netlist (used by ERC and DECAL) and a Verilog netlist

b. a TEGAS/TDL netlist (used by MUSTANG) if the chip is a scan design

10a. (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:

a. return to step 8a to correct the schematic, or

b. correct Y's RTL description and return to step 5, 6 or 7 to verify the correction.

11a. (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 10a.

12a. Run ERC to verify Y conforms to electrical design rules. If violations occur return to step 8a to correct the schematic.

13a. If the chip is a scan design, run the MUSTANG 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 8a to correct the schematic.

14a. 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 10a, using DECAL delays instead of unit delays.

c. Repeat step 11a, using DECAL delays instead of unit delays.

15a. Repeat steps 8a-14a 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

16. 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 and one EDIF netlist which contain all system logic on the chip.

17. 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 8a 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 16.

OACS Verification

18. Run EDIFMERGE to create a "Motorola EDIF" netlist, then run NETLIST to create a TEGAS/TDL netlist for MUSTANG (if the chip is a scan design) and a Verilog netlist.

19. Run ERC. Handle violations as prescribed in step 17.

20. If the chip is a scan design, run MUSTANG. Handle violations as prescribed in step 17.

21. Run DECAL and then repeat step 17, using DECAL delays instead of Synopsys delays.

IV. Combine JTAG Circuitry with Chip's System Logic

OACS Verification

22. 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 run NETLIST.

b. write a Verilog netlist manually.

Verify All JTAG Circuitry by Itself (Optional)

23. 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_TAPC.)

24. 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 22 to correct the schematic.

Verify Combined System and JTAG Circuitry

25. Create the required netlists for the entire chip:

a. Run Synopsys with logic optimization turned off (i. e., no "compile") to generate a flat EDIF netlist (except for firm macros) from the Verilog netlists for the system logic, core JTAG logic, and JTAG I/O. (As in step 23 above, the Verilog netlists for "soft" and "firm" macros must be read into Synopsys. However, firm macros must remain hierarchical until step 'c' below. For more detail see the preliminary application note "High-Level Design Methodology for OACS 2.0".)

b. Run EDIFMERGE to generate an “edif.net” netlist. Rename it as “edif.hnet” since it contains hierarchical firm macros which have not yet been flattened.

c. Run NETLIST, with the “edif.hnet” as input, to create:

i) “complete” or “flat” EDIF and Verilog netlists which include each firm macro’s internal circuitry

ii) an “Actual.RC” file (used be DECAL) for each firm macro, such as the FMC_TAPC

iii) a TEGAS/TDL netlist for MUSTANG, if the chip is a scan design.

26. Run ERC to verify entire chip conforms to electrical design rules.

27. If the chip is a scan design, run the MUSTANG design rule checker to verify entire chip conforms to scan design rules.

28. Verify entire chip via real-time simulation and timing analysis:

a. Run DECAL to calculate real-time delays (uses “complete” EDIF netlist).

b. 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 26-28, 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 8a 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 16, 22 or 25 as desired.

29. If the chip is a scan design, run MUSTANG to generate scan test patterns.

30. Run TESTPAS to combine the functional and scan test patterns from steps 28 and 29, respectively.

Appendix A: Electronic Rule Checker (ERC) 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 H4C

1. (E) BUFXP and INVXP macros must be placed on power or ground sites.
2. (E) No macro may have both an IO_PIN1 and a FIX property.
3. (E) The IO_PIN1 property must be used on all IO macros which contain a pad port.
4. (E, W)All the ERC rules that apply to non-JTAG input macros apply to input BSC macros and to TCK, TMS, TDI, TRSTB macros.
5. (E, W)All the ERC rules that apply to non-JTAG output macros apply to output BSC macros and to TDO, TDOA macros.
6. (E, W)All the ERC 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.
7. (E, W)All the ERC rules that apply to non-JTAG bidirectional output macros apply to bidirectional output BSC macros.
8. (E, W)All the ERC rules that apply to non-JTAG bidirectional input macros apply to bidirectional input BSC macros.
9. (E, W)All the ERC rules that apply to non-JTAG oscillator macros apply to oscillator BSC macros.
10. (E) For peripheral JTAG macros, an I/O site can be shared only in the following ways.

a. Any normal drive input BSC macro, excluding ICNJA, 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, but a TDBUFP cannot. (TDBUF(P) is built with the input buffer portion of an I/O site.)

c. A bidirectional BSC instance is constructed from a bidirectional output BSC macro and a bidirectional input BSC macro. An IO_PIN1 property is associated with a bidirectional output BSC macro. Neither an IO_PIN1 nor a FIX property is associated with a bidirectional input

BSC macro. A bidirectional input BSC macro shares a site with the bidirectional output BSC macro to which it gets connected. Note that the ENSCANJ cannot share an I/O site, even though it does not use the input or output buffer portion of the I/O site. In addition, neither the ISO(P) macro nor any of the special buffers for CKDR, SHDR, UDDR, IMC and OMC can share an I/O site.

11. (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 EDIF2TANGATE to place all BUFX and INVX macros used in the design.
12. (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, TCKS, TCKH, TCKHT, TCKHS} in the design.
2. (E) There must be one and only one macro from the set {TMS, TMST, TMSS} in the design.
3. (E) There must be one and only one macro from the set {TDI, TDIT, TDIS} in the design.
4. (E) There must be one and only one TDO or TDOA macro in the design.
5. (E) There must not be more than one macro from the set {TRSTB, TRSTBT, TRSTBS, ICNJA} in the design.
6. (E) The PAD inputs of {TCK, TCKT, TCKS, TCKH, TCKHT, TCKHS, TMS, TMST, TMSS, TDI, TDIT, TDIS} macros must be connected to high speed scan pads. The PAD outputs of TDO and TDOA macros must be connected to high speed scan pads.
7. (E) There must be a pullup resistor connected to the IC ports of {TMS, TMST, TMSS, TDI, TDIT, TDIS, TRSTB, TRSTBT, TRSTBS, ICNJA} 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, ENSCANP, 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 H4C array implementation of JTAG.

1. (E) No BSC JTAG macros may be placed between a macro from {TDI, TDIT, TDIS} and a macro from {TDO, TDOA} in the direction clockwise of a macro from {TDI, TDIT, TDIS}.
2. (E) Instances of peripheral JTAG buffer macros must have valid FIX property values that locate them in the periphery of the design.
3. (E) All 'xxxP' JTAG macros must be placed on power and ground IO sites.
4. (E) No 'non-xxxP' JTAG macro can be placed on a power or ground site.
5. (E) The fanout of the TDIP port of a macro from {TDI, TDIT, TDIS} must be one. The TDIP port of a macro from {TDI, TDIT, TDIS} must be connected to the TDI port of a peripheral BSC or the TDI port of a macro from {TDO, TDOA, TDBUF, TDBUFP}.
6. (E) The fanout of the TDO port of every peripheral BSC or TDBUF/P 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 macro from {TDBUF, TDBUFP, TDO, TDOA}.
7. (E) The fan-in of the TDI port of every peripheral BSC and TDBUF/P macro must be one. The TDI port of such a macro must be connected to the TDO port of another peripheral BSC, TDBUF/P, or the TDIP port of a macro from {TDI, TDIT, TDIS}.
8. (E) The fanin of the TDI port of the TDO/TDOA macro must be one, and must be connected to the TDO port of a peripheral BSC, TDBUF/P, or the TDIP port of a macro from {TDI, TDIT, TDIS}.
9. (E) Starting from the {TDI, TDIT, TDIS} 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, TDIT, TDIS}.
10. (W)If there are any unused I/O sites then there should be zero ENSCANI macros.

11.(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: Hi-Drive Outputs

1. (E) There must be no HIDRIVE property on any BSC output or bidirectional macro, or on the TDO/TDOA macro.

Appendix A.6: JTAG Clock & Control Signal Distribution

The rules in Sections A.6.1, A.6.2 and A.6.3 are split into two cases. Case I should be used for large and/or high speed arrays. Case II can be used for small and/or low speed arrays. The case type can be specified in the “design_info” file. A given design must follow either Case I or Case II rules for all six signals, namely CKDR, SHDR, UDDR, IMC and OMC. Case types cannot be mixed on the same chip. Also, 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.6.1: CKDR Distribution

The rules in this section verify proper distribution of the CKDR signal in the periphery. Case I uses a central CKDRMID/P macro driving a CKDRCC1/P and a CKDRCC2/P macro as shown in Figure 3-1. Case II only requires a single CKDR macro as shown in Figure 3-5.

CASE I - Large and/or High Speed Arrays (see Figure 3-1)

1. (E) There must be one and only one occurrence of each of the following macros: CKDRCC1/P, CKDRCC2/P, CKDRMID/P, and TDOA. There must be no occurrences of the CKDR/P macro.
2. (E) The CKDRMID/P must be driven by a core macro.
3. (E) CKDRMID/P can only drive CKDRCC1/P and CKDRCC2/P.
4. (E) CKDRCC1/P and CKDRCC2/P can only be driven by a CKDRMID/P.
5. (E) CKDRCC1/P and CKDRCC2/P cannot drive same net.
6. (E) CKDRCC1/P and CKDRCC2/P 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/P or CKDRCC2/P.
7. (E) CKDRCC1/P must drive only the CKDRNET1 net.
8. (E) CKDRCC2/P must drive only the CKDRNET2 net.
9. (E) The CKDRMID/P must reside in an I/O site between TDI and CKDRCC1/P, or between CKDRCC2/P and TDOA, or between TDOA 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/P must reside at an IO/power/ground site covered by physical CKDRNET1. CKDRCC2/P must reside at an IO/power/ground site covered by physical CKDRNET2.

In Figure 3-1, branches (a) and (b) comprise CKDRNET1 and branches (c) and (d) comprise CKDRNET2.

12. (E) |MID2CC1 - MID2CC2|/[(MID2CC1 + MID2CC2)/2] <15%, where MID2CC1 = # I/O sites between CKDRMID/P and CKDRCC1/P. MID2CC2 = # I/O sites between CKDRMID/P and CKDRCC2/P.
13. (E) $\frac{|\text{\# loads on branch (a) - \# loads on branch (c)}|}{[\text{\# loads on branch (a) + \# loads on branch (c)}]1/2} < 15\%$
14. (E) $\frac{|\text{\# loads on branch (b) - \# loads on branch (d)}|}{[\text{\# loads on branch (b) + \# loads on branch (d)}]/2} < 15\%$
15. (E) $\frac{|(\text{\#loads onCKDRNET1})-(\text{\#loads onCKDRNET2})|}{(\text{\# loads on CKDRNET1 + \# loads on CKDRNET2})/2} < 15\%$
16. (W) $\frac{|\text{\# loads on branch (a) - \# loads on branch (b)}|}{[\text{\# loads on branch (a) + \# loads on branch (b)}]/2} < 15\%$
17. (W) $\frac{|\text{\# loads on branch (c) - \# loads on branch (d)}|}{[\text{\# loads on branch (c) + \# loads on branch (d)}]/2} < 15\%$

CASE II - Small and/or Low Speed Arrays (see Figure 3-5)

1. (E) There must be one and only one occurrence of the CKDR/P macro.
2. (E) There must be no occurrences of the CKDRCC1/P, CKDRCC2/P, or CKDRMID/P macros.
3. (E) The CKDR/P must be driven by a core macro.
4. (E) The CKDR/P can only drive the CKDR port of peripheral BSC's. Conversely, the CKDR port of peripheral BSC's can only be driven by the CKDR/P.
5. (E) The TDO macro must be used instead of the TDOA macro.

Appendix A.6.2: SHDR, UDDR Distribution

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

CASE I - Large and/or High Speed Arrays (see Figure 3-3)

1. (E) There must be two and only two occurrences of the SHDR/P macro.
2. (E) Both SHDR/P's cannot drive the same net.

3. (E) Each SHDR/P must be driven by a core macro.
4. (E) There must be one and only one occurrence of ISO/P macro.
5. (E) Each SHDR/P 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/P.

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

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

In Figure 3-3, branches (a) and (b) comprise net1 and branches (c) and (d) comprise net2.

8. (W) $\frac{|\#\text{ loads on branch(a) - \# loads on branch(b)}|}{[\#\text{ loads on branch (a) + \# loads on branch (b)}]/2} < 15\%$
9. (W) $\frac{|\#\text{ loads on branch(c) - \# loads on branc (d)}|}{[\#\text{ loads on branch (c) + \# loads on branch (d)}]/2} < 15\%$
10. (W) $\frac{|(\#\text{ loads on net 1) - (\# loads on net 2)}|}{(\#\text{ loads on net 1 + \# loads on net 2})/2} < 15\%$

CASE II - Small and/or Low Speed Arrays (see Figure 3-6)

1. (E) There must be one and only one occurrence of the SHDR/P macro.
2. (E) The SHDR/P must be driven by a core macro.
3. (E) The SHDR/P 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/P.
4. (E) There must be no occurrences of the ISO/P macro.

Appendix A.6.3: IMC, OMC Distribution

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

CASE I - Large and/or High Speed Arrays (see Figure 3-4)

1. (E) There must be two and only two occurrences of IMCDR/P.
2. (E) Both IMCDR/P's cannot drive the same net.
3. (E) An IMCDR/P must be driven by a core macro.
4. (E) Each IMCDR/P 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/P.

In Figure 3-4, the IMCDR/P driving net1 is called ‘imcdr-i1’ and the IMCDR/P 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/P and CKDRCC2/P, as I/O sites are traversed clockwise from CKDRCC1/P. imcdr-i2 must reside in an IO/power/ground site between CKDRCC1/P and CKDRCC2/P, as I/O sites are traversed counter-clockwise from CKDRCC1/P.
6. (E) All the peripheral BSC's on IO sites between CKDRCC1/P and CKDRCC2/P, as I/O sites are traversed clockwise from CKDRCC1/P, must have their IMC ports driven by imcdr-i1. All the peripheral BSC's on IO sites between CKDRCC1/P and CKDRCC2/P, as I/O sites are traversed counter-clockwise from CKDRCC1/P, must have their IMC ports driven by imcdr-i2.
7. (W) $\frac{|\#\text{ loads on branch(a) - \# loads on branch(c)}|}{[\#\text{ loads on branch (a) + \# loads on branch (c)}]/2} < 15\%$
8. (W) $\frac{|\#\text{ loads on branch(b) - \# loads on branch(d)}|}{[\#\text{ loads on branch (b) + \# loads on branch (d)}]/2} < 15\%$

CASE II - Small and/or Low Speed Arrays (see Figure 3-6)

1. (E) There must be one and only one occurrence of the IMCDR/P macro.
2. (E) The IMCDR/P must be driven by a core macro.
3. (E) The IMCDR/P can only drive the IMC port of peripheral BSC's. Conversely, the IMC port of peripheral BSC's can only be driven by the IMCDR/P.

Appendix B: BSC Modeling & TAP Controller Design for Mustang Compatibility

Appendix B.1: BSC Mustang Model

The peripheral boundary scan cells use a shadow register structure similar to that shown in Figure B-1.

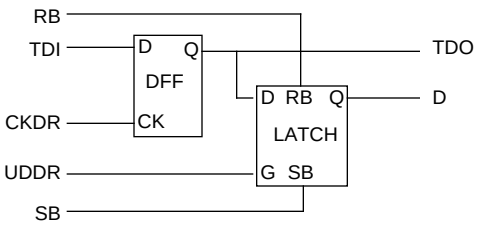

Figure B-1 General JTAG Shadow Register Structure.

Mustang cannot correctly model this functionality because the shadow latch is not on a scan chain, so a more limited Mustang-compatible model is used. The shadow latch is modeled as a combinatorial element instead of being static. This is done by creating a primitive with the truth table shown in Table B-1:

Table B-1 Truth Table for JTAG Combinatorial "jlatch"

D	G	RB	SB	Q
X	X	0	1	0
X	X	1	0	1
0	1	1	1	0
1	1	1	1	1
All Other States				X

The primitive operates with the constraint that either one of the reset / set pins (RB, SB) must be active or the gate (G) must be active with the data (D) input known. Any other combination will result in an X being generated at the output.

The JTAG boundary scan cell Mustang models have been constructed using this primitive. As a result, Mustang can detect all faults within the BSC except a stuck-at-one fault on the gate input of the latch.

Appendix B.2: TAP Controller

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:

i) it contains static elements that are not scannable

ii) it contains signals which are gated by the TCK clock

iii) 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 Mustang-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 Mustang 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 Mustang 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 shadow register structure shown in Figure B-1. The latch within this BSC is driven by UDDR, which is derived from clock TCK in Figure B-2. Since the latch in the BSC must be modeled as a combinatorial "JLATCH," Mustang sees a clock derivative (UDDR) propagating through the JLATCH to drive the data input of the system flop. This condition violates scan design rules; state element data inputs cannot be derived from a clock. 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 Mustang requirement that the data input to a flop be an NRZ (non-pulsed) waveform. (UDDR still drives the data input of the core/system flop via the combinatorial JLATCH.)

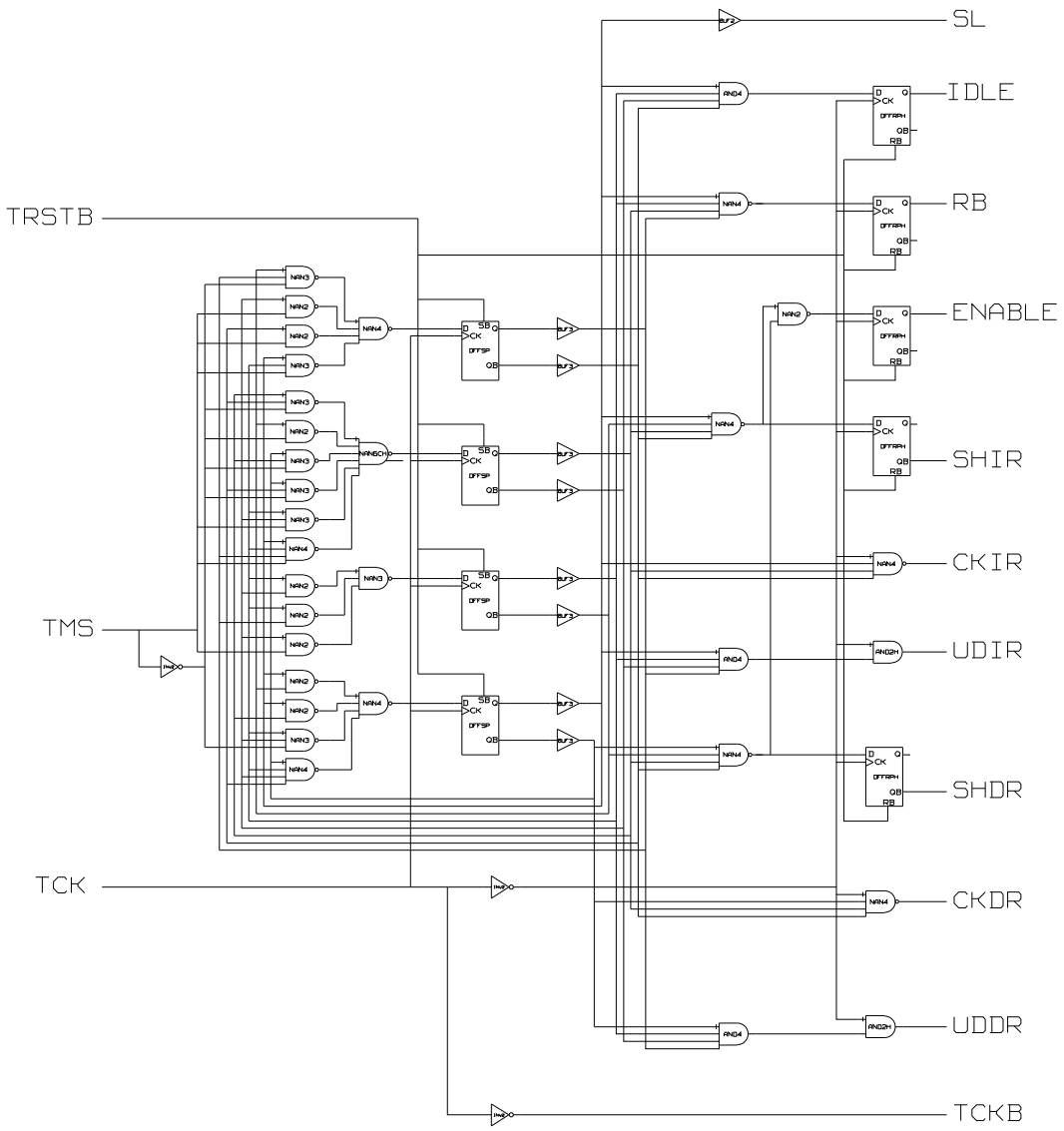

Figure B-2 TAP Controller Shown in IEEE 1149.1 JTAG Specification

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 combinatorial JLATCH. The reason is that Mustang requires flops to have pulsed clocks.

The new functionality for UDDR is shown in the waveform diagram of Figure B-3, which demonstrates the loading of a data register.

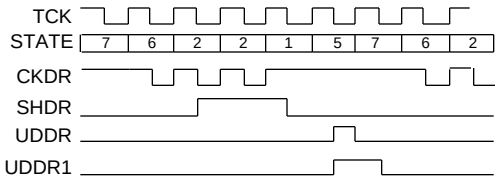

Figure B-3 JTAG Control Signals Waveform Diagram

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-4. 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 Mustang 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 Mustang test mode because Mustang 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.

Appendix C: EDIFMERGE Attribute File Entries for Peripheral JTAG Macros

Figure C-1 shows the JTAG portion of the Attribute file for the circuit in Section 4.5, Figure 4-7 (if this circuit had been entered using Verilog HDL instead of schematic capture).

In Figure C-1, comment lines are denoted by an asterisk as the leading character. The keyword "-INSTANCE" is followed by a Fix entry for each non-bonded macro in the design. In each FIX entry the macro instance name is copied from the Synopsys EDIF netlist, and is the instance "name" with any leading non-alphabetic characters stripped off. (The instance "rename," which is not used, is enclosed in quotes and follows the instance "name" in the netlist.)

The ISOP macro requires two entries in the Attribute file. In the entry which follows the "-NONLOGICCELL" keyword, the instance name is arbitrarily chosen. This instance name is then used in the second ISOP entry, which follows the "-INSTANCE" keyword along with the entries for the other non-bonded macros.

Note that the TAP macros are fix-placed via the I0_PIN1 property, not the FIX property, since these are input macros which connect to package pins.

```
-NONLOGICCELL
*cell type          instance name
 ISOP               isop;

-PORT
*Signal Name        property        Pin #
 TMS                IO_PIN1         "39";
 TCK                IO_PIN1         "38";
 TDOA               IO_PIN1         "33";
 TDI                IO_PIN1         "32";

-INSTANCE
*instance name      property        I/O site #
 imcdr_1            FIX             "IO10";
 ckdrmid p          FIX             "IO11";
 omcdrp_1           FIX             "IO17";
 ckdrcc1p           FIX             "IO115";
 shdr_1             FIX             "IO116";
 uddrp_1            FIX             "IO121";
 isop               FIX             "IO219";
 imcdr_2            FIX             "IO220";
 omcdrp_2           FIX             "IO225";
 ckdrcc2p           FIX             "IO323";
 shdr_2             FIX             "IO324";
 uddrp_2            FIX             "IO329";
```

Figure C-1 JTAG-Macro Portion of Attribute File for H4C123 in 160 QFP(CD) or MicroCool

Non-JTAG hi-drive outputs and bidirectionals are designated by a HIDRIVE property. However each JTAG hi-drive is a separate macro, which has a COMPLEX_HIDRIVE property instead of a HIDRIVE property. COMPLEX_HIDRIVE properties are added to the EDIF netlist automatically by the NETLIST program, therefore a JTAG hi-drive has no entry in the Attribute file for either a HIDRIVE property or a COMPLEX_HIDRIVE property.

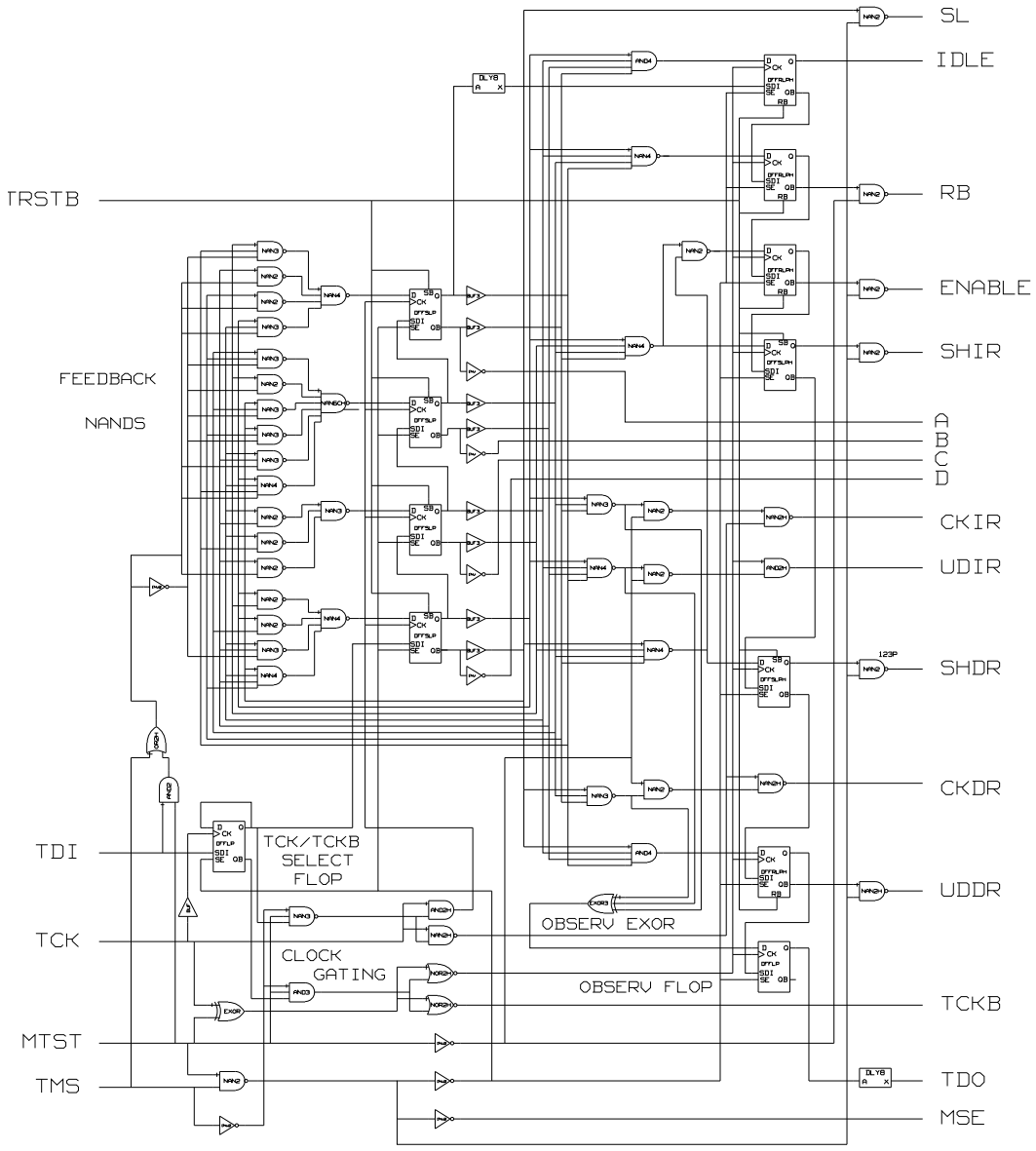

Figure B-4 Mustang-Compatible TAP Controller (FMC_TAPC).

Appendix D: JTAG for H4CPlus and H4EPlus

The H4CPlus and H4EPlus JTAG implementation is identical to the H4C JTAG implementation with the exeption of the changes described below.

1. The TDO and CKDR macros do not exist because the “Small or Low-Speed Array” distribution scheme (see section 3.3) has not been implemented.
2. The H4C TDOA macro has been renamed “TDOUT” in H4CPlus and H4EPlus, and the mux and flop (see Figure 2-1) have been removed from the TDOUT macro. This logic must now be implemented in the array core, where the customer can modify it if he wishes.
3. Only the “non-P” version exists for the following macros:
   - ISOR
   - CKDRCC1
   - CKDRCC2
   - CKDRMID
   - SHDR
   - UDDR
   - IMCDR
   - OMCDR
   - ENSCANJ
   - TDBUF

   The ISOR, CKDRCC1, CKDRCC2, ENSCANJ and TDBUF macros can now be placed on normal I/O sites as well as on output-power/gnd sites. However, they cannot be placed on input-power/gnd sites. All the rest of these macros can be placed by themselves on normal I/O sites, output-power/gnd sites, and input-power/gnd sites.
4. The ISOR macro now makes a physical cut in the CKDR ring, in addition to the SHDR and UDDR rings as in H4C. The ISOR also contains a flop and an inverter such that the flop is clocked on the falling edge of CKDR (see Figure 7-1). This flop is placed in the shift path of the boundary scan chain to prevent hold time violations due to clock skew between the two physically separate CKDR nets. Unlike H4C, all BSC’s no longer share a common clock net, therefore hold time violations between BSC’s are of concern. However, the falling-edge flop inside the ISOR macro allows CKDRNET2 in Figure 7-1 to be skewed from CKDRNET1 by up to half a cycle of CKDR without causing a hold time violation. The benefit realized is that there is no fake cut required in the netlist for accurate simulation (see section 3.2), therefore the need for tight balancing of the CKDR buffers is eliminated, along with the associated ERC rules. Simulation 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.
5. The customer must use PREDIX to compute the actual RC’s for peripheral JTAG nets, in order to do accurate JTAG simulations. DECAL merges the PREDIX peripheral RC’s with either DECAL-estimated RC’s for the array core (for pre-layout simulations) or with Gate Ensemble-generated actual RC’s for the array core (for pre-layout simulations).
6. The paralleled output buffer portion of a hi-drive, such as an ON32, can reside on an ouptut-gnd I/O site if that gnd is an OVSSP macro.

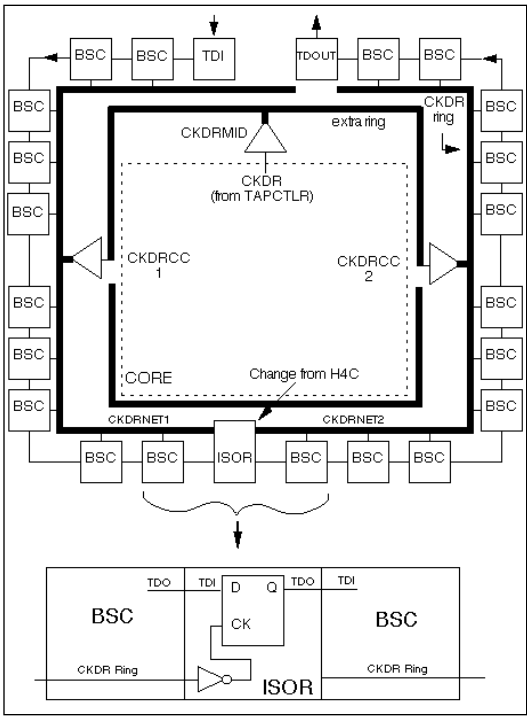

Figure D-1 CKDR Distribution to I/O Boundary Scan Cells on H4CPlus and H4EPlus Arrays

ASIC REGIONAL DESIGN CENTERS - U.S.A.

California, San Jose
(408) 749-0510

Illinois, Chicago
(708) 490-9500

Massachusetts, Marlborough
(508) 481-8100

ASIC REGIONAL DESIGN CENTERS - International

European Headquarters,
Germany, Munich
(089) 92103-0

England, Aylesbury, Bucks
(0296) 395252

France, Vanves
(01) 40355877

Holland, Eindhoven
(04998) 61211

Hong Kong, Kwai Chung
480 8333

Italy, Milan
(02) 82201

Japan, Tokyo
(03) 440-3311

Sweden, Stockholm
(08) 734-8800

Literature Distribution Centers:

USA:	Motorola Literature Distribution; P.O. Box 20912, Phoenix, Arizona 85036
EUROPE:	Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands, Miltion Keynes MK14 5BP, England
JAPAN:	Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan
ASIA-PACIFIC:	Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong

Trademarks

H4C, H4CPlus and H4EPlus, DECAL, Mustang, 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.