# SoC/OpenRISC Development Interface

*Author: Igor Mohor*

*IgorM@opencores.org*

***Rev. 1.5***

***October 10, 2002***

# Revision History

| Rev. | Date | Author | Description |
|---|---|---|---|
| 0.1 | 02/02/01 | Igor Mohor | First Draft |
| 0.2 | 05/04/01 | IM | Trace port added |
| 0.3 | 16/04/01 | IM | WP and BP number changed, trace modified |
| 0.4 | 01/05/01 | IM | Title changed, DEBUG instruction added, scan chains changed, IO ports changed |
| 0.5 | 05/05/01 | IM | TSEL and QSEL register changed |
| 0.6 | 06/05/01 | IM | Ports connected to the OpenRISC changed |
| 0.7 | 14/05/01 | IM | MODER register changed, trace scan chain changed; SSEL register added |
| 0.8 | 18/05/01 | IM | RESET bit and signal added; STALLR changed to RISCOP |
| 0.9 | 23/05/01 | IM | RISC changed to OpenRISC; WISHBONE interface added, SPR and memory access added |
| 0.10 | 01/06/01 | IM | Meaning of Instruction status and Load/store status changed in all registers; more details added to Appendix A |
| 0.11 | 10/09/01 | IM | Register and OpenRISC scan chain operation changed |
| 1.0 | 19/09/01 | IM | Some registers deleted |
| 1.1 | 15/10/01 | IM | WISHBONE interface added; RISC Stall signal is set by breakpoint and reset by writing 0 to RISCOP register |
| 1.2 | 03/12/01 | IM | Chain length changed so additional CRC checking can be performed |
| 1.3 | 21/01/02 | Jeanne Wiegelmann | Document revised. |
| 1.4 | 07/05/02 | IM | Register MONCNTL added. |
| 1.5 | 10/10/02 | IM | WISHBONE Scan Chain changed to show state of the access. |
| | | | |

# Contents

# 1

# Introduction

The Development Interface is used for development purposes (Boundary Scan testing and debugging) and is as such an interface between the OpenRISC, peripheral cores, and any commercial debugger/emulator or BS testing device. The external debugger or BS tester connects to the core via a fully IEEE 1149.1 compatible JTAG port. The Development Interface also contains a trace buffer and support for tracing the program flow, execution coverage, and profiling the code.

# 2

# IO Ports

## 2.1 External Device Ports

| Port | Width | Direction | Description |
|---|---|---|---|
| TCK_PAD_I | 1 | Input | Test Clock is the clock for the JTAG interface. |
| TMS_PAD_I | 1 | Input | Test Mode Select is the signal that controls the TAP machine sequence in the JTAG interface. |
| TDI_PAD_I | 1 | Input | Test Data Input is the input data for the JTAG interface. |
| TDO_PAD_I | 1 | Output | Test Data Out is the output data from the JTAG interface. |
| TRST_PAD_I | 1 | Input | Test Reset is an active low input signal that asynchronously resets the TAP machine in the JTAG interface. |
| DIROUT | 32 | Output | Direct Outputs are outputs from different blocks and are development port independent. Which blocks (signals) are connected to the DIROUT is selectable with the DIRSEL signals. It is up to the system integrator to connect the signals he wishes to be observed to the DIROUT signals. |
| DIRSEL | 3 | Input | Direct Signals Selection select signalsfor the output multiplexer so many different signals can be connected to the DIROUT signals. |

| Port | Width | Direction | Description |
|---|---|---|---|
| FORCESTALLIN | 1 | Input | Force Stall Input is an external signal that stalls the OpenRISC. |
| STALLSTAT | 1 | Output | Stall Status Output is connected to the board. It is used for telling “the world” that the OpenRISC is stalled. |

**Table 1: External Device Ports**

## 2.2 OpenRISC Ports

| Port | Width | Direction | Description |
|---|---|---|---|
| RISC_CLK_I | 1 | Input | RISC Clock signal. |
| WP_I | 11 | Input | Watchpoint Status is an input signal that reports the OpenRISC watchpoint generation. |
| BP_I | 1 | Input | Breakpoint Status is an input signal that reports the OpenRISC breakpoint generation. |
| RISC_DATA_I | 32 | Input | Data Input transfers the data from the OpenRISC to the development interface. |
| RISC_DATA_O | 32 | Output | Data Output transfers the data from the development interface to the OpenRISC. |
| RISC_ADDR_O | 32 | Output | Address of the special-purpose register to be read or written. |
| OPSELECT_O | 4 | Output | Operation Select asynchronously selects the operation of the OpenRISC debug port. |
| LSSTATUS_I | 4 | Input | Load/Store Status synchronously shows the status of the Load/Store unit. |
| ISTATUS_I | 4 | Input | Instruction Status synchronously shows the status of the Instruction Fetch unit. |
| RISCSTALL_O | 1 | Output | CPU Stall synchronously stalls the OpenRISC’s |

| Port | Width | Direction | Description |
|---|---|---|---|
| | | | core. |
| RESET_O | 1 | Output | Reset Output is connected to the OpenRISC (OpenRISC's reset logic). When set, the OpenRISC is reset. |

**Table 2: OpenRISC Ports**

## 2.3 WISHBONE Interface Ports

| Port | Width | Direction | Description |
|---|---|---|---|
| WB_CLK_I | 1 | Input | Clock Input (WISHBONE clock). |
| WB_RST_I | 1 | Input | Reset Input (WISHBONE reset). |
| WB_ACK_I | 1 | Input | Acknowledgment Input indicates a normal cycle termination. |
| WB_ADR_O | 32 | Output | Address Output array. |
| WB_CYC_O | 1 | Output | Cycle output encapsulates a valid transfer cycle. |
| WB_DAT_I | 32 | Input | Data Input array. |
| WB_DAT_O | 32 | Output | Data Output array. |
| WB_ERR_I | 1 | Input | Error acknowledgment input indicates an abnormal cycle termination. |
| WB_SEL_O | 4 | Output | Select Output indicates which bytes are valid on the data bus. |
| WB_STB_O | 1 | Output | Strobe Output indicates a valid transfer. |
| WB_WE_O | 1 | Output | Write Enable indicates a Write Cycle when asserted high. |
| WB_CAB_O | 1 | Output | Consecutive Address Burst indicates a burst cycle. |

**Table 3: WISHBONE Interface Ports**

## 2.4 Boundary Scan Chain Ports

| Port | Width | Direction | Description |
|---|---|---|---|
| CAPTURE_DR_O | 1 | Output | CaptureDR state of the TAP controller |
| SHIFT_DR_O | 1 | Output | ShiftDR state of the TAP controller |
| UPDATE_DR_O | 1 | Output | UpdateDR state of the TAP controller |
| EXTEST_SELECTED_O | 32 | Output | EXTESTSelected instruction external test selected |
| BS_CHAIN_I | 1 | Input | BS_CHAIN_I hooks for boundary scan chain |
| BS_CHAIN_O | 1 | Output | BS_CHAIN_O hooks for boundary scan chain |

**Table 4: Boundary Scan Chain Ports**

# 3 Registers

This section specifies all registers in the Development Interface.

## 3.1 Registers List

| Name | Address | Width | Access | Description |
|---|---|---|---|---|
| MODER | 0x0 | 32 | R/W | Mode Register |
| TSEL | 0x4 | 32 | R/W | Trigger Selection for the Trace Buffer |
| QSEL | 0x8 | 32 | R/W | Qualifier Selection for the Trace Buffer |
| SSEL | 0xC | 32 | R/W | Stop Select Register |
| RISCOP | 0x10 | 32 | R/W | OpenRISC Operation Register |
| | | | | |
| | | | | |
| RECSEL | 0x40 | 32 | R/W | Record Selection |
| MONCNTL | 0x44 | 32 | R/W | Monitor Control |

**Table 5: Register List**

## 3.2 MODER (Mode Register)

| Bit # | Access | Description |
|---|---|---|
| 31:18 | | Reserved |
| 17 | R/W | CONTIN – Continuous Mode<br>0 = Recording is suspended while the trace buffer is full<br>1 = Old samples are overwritten with new samples once the trace buffer is full |
| 16 | R/W | ENABLE – Trace Enable<br>0 = Trace is disabled<br>1 = Trace is enabled |
| 15:1 | | Reserved |
| 0 | R | TRACE PRESENT – Trace Present<br>0 = Trace not present (Development Interface does not include trace)<br>1 = Trace present (Development Interface includes trace) |

**Table 6: MODER Register**

Reset Value:

MODER: 00000000h

## 3.3 TSEL (Trigger Select Register)

| Bit # | Access | Description |
|---|---|---|
| 31:30 | R/W | TRIGOP – Trigger Operation<br>00 = Any (trigger is always active)<br>10 = All valid groups in the TSEL register are OR-ed |

| Bit # | Access | Description |
|---|---|---|
| | | 11 = All valid groups in the TSEL register are AND-ed |
| 29:24 | | Reserved |
| 23 | R/W | ISTRIGVALID – Instruction Status Trigger valid<br>0 = Don't care (Instruction Status Trigger is not valid)<br>1 = Instruction Status Trigger valid |
| 22:21 | R/W | ISTRIG – Instruction Status Trigger<br>00 = No instruction fetch in progress<br>01 = Normal instruction fetch<br>10 = Executing branch instruction<br>11 = Fetching instruction in delay slot |
| 20 | R/W | LSSTRIGVALID – Load/Store Status Trigger Valid<br>0 = Don't care (Load/Store Status Trigger is not valid)<br>1 = Load/Store Status Trigger valid |
| 19:16 | R/W | LSSTRIG – Load/Store Status Trigger<br>0000 = No load/store instruction in execution<br>0001 = Reserved for load doubleword<br>0010 = Load byte and zero extend<br>0011 = Load byte and sign extend<br>0100 = Load halfword and zero extend<br>0101 = Load halfword and sign extend<br>0110 = Load singleword and zero extend<br>0111 = Load singleword and sign extend<br>1000 = Reserved for store doubleword<br>1001 = Reserved<br>1010 = Store byte<br>1011 = Reserved<br>1100 = Store halfword<br>1101 = Reserved<br>1110 = Reserved<br>1111 = Reserved |
| 15:14 | | Reserved |

| Bit # | Access | Description |
|---|---|---|
| 13 | R/W | BPTRIGVALID – Breakpoint Trigger valid<br>0 = Don't care (BP trigger not valid)<br>1 = BP trigger valid |
| 12 | R/W | BPTRIG – Breakpoint Trigger<br>0 = Breakpoint does not start recording the trace unit<br>1 = BP[0] starts recording the trace unit |
| 11 | R/W | WPTRIGVALID – Watchpoint Trigger valid<br>0 = Don't care (WP Trigger not valid)<br>1 = WP Trigger valid |
| 10:0 | R/W | WPTRIG – Watchpoint Trigger<br>00000000000 = Watchpoints do not start recording the trace unit<br>00000000001 = WP[0] starts recording the trace unit<br>00000000010 = WP[1] starts recording the trace unit<br>.<br>11111111111 = Any WP[10:0] starts recording the trace unit |

**Table 7: TSEL Register**

Reset Value:

TSEL: 00000000h

## 3.4 QSEL (Qualifier Select Register)

| Bit # | Access | Description |
|---|---|---|
| 31:30 | R/W | QUALIFOP – Qualifier Operation<br>00 = Any (Qualifier is always active, all samples will be recorded)<br>10 = All valid groups in the QSEL register are OR-ed |

| Bit # | Access | Description |
|---|---|---|
| | | 11 = All valid groups in the QSEL register are AND-ed |
| 29:24 | | Reserved |
| 23 | R/W | ISTQUALIFVALID – Instruction Status Qualifier valid<br>0 = Don't care (Instruction Status Qualifier not valid)<br>1 = Instruction Status Qualifier valid |
| 22:21 | R/W | ISTQUALIF – Instruction Status Qualifier<br>00 = No instruction fetch in progress<br>01 = Normal instruction fetch<br>10 = Executing branch instruction<br>11 = Fetching instruction in delay slot |
| 20 | R/W | LSSQUALIFVALID – Load/Store Status Qualifier valid<br>0 = Don't care (Load/Store Status Qualifier not valid)<br>1 = Load/Store Status Qualifier valid |
| 19:16 | R/W | LSSQUALIF – Load/Store Status Qualifier<br>0000 = No load/store instruction in execution<br>0001 = Reserved for load doubleword<br>0010 = Load byte and zero extend<br>0011 = Load byte and sign extend<br>0100 = Load halfword and zero extend<br>0101 = Load halfword and sign extend<br>0110 = Load singleword and zero extend<br>0111 = Load singleword and sign extend<br>1000 = Reserved for store doubleword<br>1001 = Reserved<br>1010 = Store byte<br>1011 = Reserved<br>1100 = Store halfword<br>1101 = Reserved<br>1110 = Reserved<br>1111 = Reserved |
| 15:14 | | Reserved |

| Bit # | Access | Description |
|---|---|---|
| 13 | R/W | BPQUALIFVALID – Breakpoint Qualifier valid<br>0 = Don't care (BP qualifier not valid)<br>1 = BP Qualifier valid |
| 12 | R/W | BPQUALIF – Breakpoint Qualifier<br>0 = Breakpoint does not enable recording of the current sample<br>1 = BP[0] enables recording of the current sample |
| 11 | R/W | WPQUALIFVALID – Watchpoint Qualifier valid<br>0 = Don't care (WP qualifier not valid)<br>1 = WP qualifier valid |
| 10:0 | R/W | WPQUALIF – Watchpoint Qualifier<br>00000000000 = Watchpoints do not enable recording of the current sample<br>00000000001 = WP[0] enables recording of the current sample<br>00000000010 = WP[1] enables recording of the current sample<br>.<br>11111111111 = Any WP[10:0] enables recording of the current sample |

**Table 8: QSEL Register**

Reset Value:

QSEL: 00000000h

## 3.5 SSEL (Stop Select Register)

| Bit # | Access | Description |
|---|---|---|
| 31:30 | R/W | STOPOP – Stop Operation<br>00 = Nothing (recording cannot be stopped by selecting one of the |

| Bit # | Access | Description |
|---|---|---|
| | | following groups)<br>10 = All valid groups in the SSEL register are OR-ed<br>11 = All valid groups in the SSEL register are AND-ed |
| 29:24 | | Reserved |
| 23 | R/W | ISTSTOPVALID – Instruction Status Stop valid<br>0 = Don't care (Instruction Status Stop is valid)<br>1 = Instruction Status Stop valid |
| 22:21 | R/W | ISTSTOP – Instruction Status Stop<br>00 = No instruction fetch in progress<br>01 = Normal instruction fetch<br>10 = Executing branch instruction<br>11 = Fetching instruction in delay slot |
| 20 | R/W | LSSSTOPVALID – Load/Store Status Stop valid<br>0 = Don't care (Load/Store Status Stop is valid)<br>1 = Load/Store Status Stop is valid |
| 19:16 | R/W | LSSSTOP – Load/Store Status Stop<br>0000 = No load/store instruction in execution<br>0001 = Reserved for load doubleword<br>0010 = Load byte and zero extend<br>0011 = Load byte and sign extend<br>0100 = Load halfword and zero extend<br>0101 = Load halfword and sign extend<br>0110 = Load singleword and zero extend<br>0111 = Load singleword and sign extend<br>1000 = Reserved for store doubleword<br>1001 = Reserved<br>1010 = Store byte<br>1011 = Reserved<br>1100 = Store halfword<br>1101 = Reserved<br>1110 = Reserved |

| Bit # | Access | Description |
|---|---|---|
| | | 1111 = Reserved |
| 15:14 | | Reserved |
| 13 | R/W | BPSTOPVALID – Breakpoint Stop valid<br>0 = Don't care (BP Stop not valid)<br>1 = BP Stop is valid. |
| 12 | R/W | BPSTOP – Breakpoint Stop<br>0 = Breakpoint does not stop recording<br>1 = BP[0] stops recording |
| 11 | R/W | WPSTOPVALID – Watchpoint Stop valid<br>0 = Don't care (WP Stop not valid)<br>1 = WP Stop valid |
| 10:0 | R/W | WPSTOP – Watchpoint Stop<br>00000000000 = Watchpoints do not stop recording<br>00000000001 = WP[0] stops recording<br>00000000010 = WP[1] stops recording<br>.<br>11111111111 = Any WP[10:0] stops recording |

**Table 9: SSEL Register**

Reset Value:

SSEL: 00000000h

## 3.6 RISCOP (OpenRISC Operation Register)

| Bit # | Access | Description |
|---|---|---|
| 31:2 | | Reserved |
| 1 | R | RESET – Reset OpenRISC |

| Bit # | Access | Description |
|---|---|---|
| | | 0 = normal<br>1 = reset |
| 0 | R/W | RISCSTALL – OpenRISC Stall<br>0 = normal operation<br>1 = Stall OpenRISC. Risc can also set this bit by BP_I signal. |

**Table 10: RISCOP Register**

Reset Value:

RISCOP: 00000000h

## 3.7 RECSEL (Record Selection Register)

| Bit # | Access | Description |
|---|---|---|
| 31:7 | | Reserved |
| 6 | R/W | RECINSTR – Record Instruction in the Execution Pipeline<br>0 = Don't save INSTR to the trace buffer<br>1 = Save INSTR to the trace buffer |
| 5 | R/W | RECWRITESPR – Record Writing SPR<br>0 = Don't save WRITESPR to the trace buffer<br>1 = Save WRITESPR to the trace buffer |
| 4 | R/W | RECREADSPR – Record Reading SPR<br>0 = Don't save READSPR to the trace buffer<br>1 = Save READSPR to the trace buffer |
| 3 | R/W | RECSDATA – Record Store Data<br>0 = Don't save SDATA to the trace buffer<br>1 = Save SDATA to the trace buffer |
| 2 | R/W | RECLDATA – Record Load Data |

| Bit # | Access | Description |
|---|---|---|
| | | 0 = Don't save LDATA to the trace buffer<br>1 = Save LDATA to the trace buffer |
| 1 | R/W | RECLSEA – Record LSEA (Load/Store Effective Address)<br>0 = Don't save LSEA to the trace buffer<br>1 = Save LSEA to the trace buffer |
| 0 | R/W | RECPC – Record PC (Program Counter)<br>0 = Don't save PC to the trace buffer<br>1 = Save PC to the trace buffer |

**Table 11: RECSEL Register**

Reset Value:

RECSEL: 00000000h

## 3.8 MONCNTL (Monitor Control Register)

| Bit # | Access | Description |
|---|---|---|
| 31:4 | | Reserved |
| 3:0 | R/W | CLK_SEL – Clock Selection<br>Selects which clock signal is monitored by controlling monitor multiplexer. |

**Table 12: MONCNTL Register**

Reset Value:

MONCNTL: 00000000h

# 4

# Operation

This section describes the operation of the Development Interface. It discusses the visibility of internal signals, the JTAG interface with the TAP controller and supported instructions, scan chain configuration, watchpoints and breakpoints, trace, memory, SPR and GPR interface, and finally how to enter and exit the DEBUG mode.

## 4.1 Visibility of Internal Signals

The state of several internal signals can be monitored through the DIROUT[31:0] signals. The monitoring is development port independent. You can use the DIRSEL[2:0] signals to select which set of signals the multiplexer connects to the output pins. It is up to the system integrator to choose which signals need to be observed and how to connect them to the multiplexer. He also must decide whether to use dedicated pins for the DIROUT and DIRSEL signals or to multiplex them with some other pins.

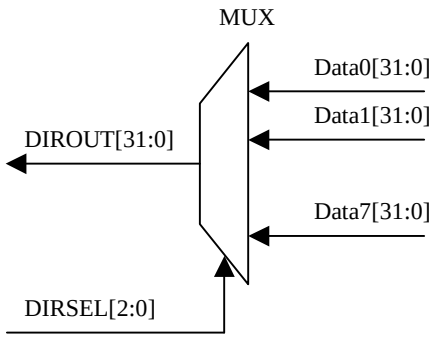


**Figure 1: Selection of Observed Signals**

## 4.2 JTAG Interface with the TAP Controller and Instructions

The JTAG Interface is fully IEEE Std.1149.1 compliant and supports the following instructions:

| Instruction | Code |
|---|---|
| EXTEST | 0000 |
| SAMPLE/PRELOAD | 0001 |
| IDCODE | 0010 |
| CHAIN_SELECT | 0011 |
| INTEST | 0100 |
| CLAMP | 0101 |
| CLAMPZ | 0110 |
| HIGHZ | 0111 |
| DEBUG | 1000 |
| BYPASS | 1111 |

**Table 13: TAP Instruction Set**

Which instructions will be supported remains TBD.

### 4.2.1 EXTEST (IR=0000)

The EXTEST instruction connects the selected chain, which is in TEST mode, between the TDI and TDO. During EXTEST instruction, the boundary-scan register is accessed to drive test data off-chip via the boundary outputs and to receive test data in-chip via the boundary inputs. The bit code of this instruction is defined as all zeroes by IEEE Std. 1149.1.

- ✓ CaptureDR state: The outputs from the system logic (test vector) are captured.
- ✓ ShiftDR state: The captured test vector is shifted out via TDO output while a new test vector is shifted in via the TDI input.

- ✓ UpdateDR state: The data shifted in via TDI is applied to Output and Control cells (output pins are driven with 0, 1, or highZ).

### 4.2.2 SAMPLE/PRELOAD (IR=0001)

The SAMPLE/PRELOAD instruction allows the IC to remain in its functional mode and to select the boundary-scan register to be connected between TDI and TDO. During this instruction, you can access the boundary-scan register via a data scan operation to take a sample of the functional data entering and leaving the IC. The instruction is also used to preload test data into the boundary-scan register before loading an EXTEST, CLAMP, or CLAMPZ instruction. This instruction should only be used for production tests.

- ✓ CaptureDR state: The inputs from the system logic (test vector) are captured.
- ✓ ShiftDR state: The captured test vector is shifted out via TDO output while a new test vector is shifted in via TDI input.
- ✓ UpdateDR state: No changes.

### 4.2.3. IDCODE (IR=0010)

The IDCODE instruction allows the IC to remain in its functional mode and selects the device identification register (ID register) to be connected between TDI and TDO. The device identification register is a 32-bit read-only register containing information regarding the IC manufacturer, device type, and version code. Accessing the device identification register does not interfere with the operation of the IC. Also, access to the device identification register should be immediately available via a TAP data-scan operation, after power-up of the IC, or after the TAP has been reset by using the optional TRSTn pin or by otherwise moving to the Test-Logic-Reset state.

- ✓ CaptureDR state: The ID value is captured from the ID register.
- ✓ ShiftDR state: The captured ID value is shifted out via TDO output.
- ✓ UpdateDR state: The data shifted in via TDI is ignored (ID is a read-only register).

### 4.2.4 CHAIN_SELECT (IR=0011)

With the CHAIN_SELECT instruction, different scan chains can be connected between the TDI and TDO while the IC remains in the same mode. On reset, scan chain 0 is selected by default (for Boundary Scan testing purposes).

- ✓ CaptureDR state: A fixed value 5'b01100 is loaded into the shift register (for testing purposes).
- ✓ ShiftDR state: The scan chain identification number is shifted in via TDI, while the fixed value is shifted out via TDO.
- ✓ UpdateDR state: The identified scan chain is connected between the TDI and TDO.

### 4.2.5 INTEST (IR=0100)

The INTEST instruction is used for internal testing (IC, core, or some other parts). The selected scan chain connects between TDI and TDO and is put into TEST mode.

- ✓ CaptureDR state: Both, the input signals (from the system logic) and the output signals (from the core) are captured.
- ✓ ShiftDR state: The captured data is shifted out via TDO output while new test data (from the system logic) is shifted in via the TDI input.
- ✓ UpdateDR state: The data shifted in via TDI is applied to the core inputs.
- ✓ Run/Test-Idle state: For each cycle in the Run/Test-Idle state, one clock pulse is applied to the core (for single step).

### 4.2.6 CLAMP (IR=0101)

The CLAMP instruction is used for setting all the outputs to the pre-loaded values (values that remain from the previous shifting or are loaded with the SAMPLE/PRELOAD instruction). The bypass register is connected between the TDI and TDO.

- ✓ CaptureDR state: A logical 0 is captured in the bypass register.
- ✓ ShiftDR state: Input data shifted in via TDI is shifted out via TDO after a one-clock delay.
- ✓ UpdateDR state: No changes.

### 4.2.7 CLAMPZ (IR=0110)

Same as CLAMP instruction. The difference lies in that the 3-state outputs are placed in their inactive state. This instruction is used during the production test (each output can be put in inactive state regardless of its data value).

### 4.2.8 HIGHZ (IR=0111)

The HIGHZ instruction sets all outputs (including two-state as well as three-state types) of an IC to a disabled (high-impedance) state and selects the bypass register to be connected between TDI and TDO. During this instruction, data can be shifted through the bypass register from TDI to TDO without affecting the condition of the IC outputs.

- ✓ CaptureDR state: A logical 0 is captured in the bypass register.
- ✓ ShiftDR state: Input data shifted in via TDI is shifted out via TDO after a one-clock delay.
- ✓ UpdateDR state: No changes.

### 4.2.9 DEBUG (IR=1000)

The DEBUG instruction must be used when the debugging is in progress (OpenRISC, register, or trace scan chain must be previously selected). After a read or write is performed in DEBUG mode, the result is known in the same shift cycle (while address and data are shifted in, the result is already shifted out). There is no need to perform an additional cycle to shift out the results (it will not work).

### 4.2.10 BYPASS (IR=1111)

The BYPASS instruction keeps the IC in a functional mode and selects that the bypass register will be connected between TDI and TDO. It allows serial data to be transferred through the IC from TDI to TDO without affecting the operation of the IC. The bit code of this instruction is defined as all ones by IEEE Std. 1149.1. Usage of an unimplemented instruction will result in the BYPASS instruction.

- ✓ CaptureDR state: A logical 0 is captured in the bypass register.

- ✓ ShiftDR state: Input data shifted in via TDI is shifted out via TDO after a one-clock delay.
- ✓ UpdateDR state: No changes.

## 4.3 Scan Chains

For BS testing, OpenRISC debugging, trace operation, observation of other cores (at the connection to the WISHBONE), identification of the chip version, etc., you can use several scan chains:

- ✓ Global boundary scan chain
- ✓ OpenRISC Debug Interface scan chain
- ✓ OpenRISC test chain
- ✓ Trace scan chain
- ✓ Register scan chain
- ✓ Block scan chain
- ✓ Optional scan chains

To select a chain, issue a CHAIN_SELECT instruction followed by the chain's unique ID value (and CRC). All of the following instructions apply to the selected chain:

| Chain | ID | Description |
|---|---|---|
| Global BS Chain | 0000 | For boundary scan testing |
| OpenRISC Debug Interface Scan Chain | 0001 | For OpenRISC debugging |
| OpenRISC Test Chain | 0010 | For OpenRISC testing (factory tests) |
| Trace Scan Chain | 0011 | For tracing the program flow |
| Register Scan Chain | 0100 | For internal register access |
| WISHBONE Scan Chain | 0101 | For WISHBONE access |
| Block Scan Chain | 0110 - 0111 | For peripheral cores testings |
| Optional Scan Chains (can be connected to any core, register, counter, etc.) | 1000 - 1111 | For additional testings |

**Table 14: Chains Identification**

**Note:** Not all instructions are supported when certain scan chains are selected.

## 4.4 Watchpoints and Breakpoints

Watchpoints WP[10:0] and breakpoints BP[0] are output signals from the OpenRISC debug module and input signals to the SoC/OpenRISC Development Interface. They are also connected to the output pads and can be used for informing external devices when certain events occur (i.e. they can set a trigger, enable/disable a device, put a device into SLEEP mode, etc.).

Watchpoints and breakpoints are used as triggers or/and qualifiers for the trace module.

## 4.5 Trace

Trace is the block that records the OpenRISC activities to the internal buffer (executed instructions, loaded/stored data, program counter, SPR access, etc.). The trace has a 1024 x 36 (n x 36) buffer built-in to store all information needed for program tracing, execution coverage, and profiling (measuring the time that subroutines need for their execution). Recording starts as soon as both trigger and qualifier occur. When the buffer

is full, the recording can be suspended or old samples can be overwritten. The recorded data is read out through the trace scan chain.

In order to start tracing, enable the trace and set the following:

- ✓ The trigger (TSEL register on page 12)
- ✓ The qualifier (QSEL register on page 14)
- ✓ The configuration of the sample – the kind of information to be stored in the buffer (RECSEL register on page 19)
- ✓ The stop condition (SSEL register on page 16)
- ✓ The operation mode (MODER register on page 12).

The following shortly explains the purpose of trigger, qualifier, and sample configuration to prevent mixing their use:

- ✓ The trigger starts the trace (once started, it might record the samples).
- ✓ The qualifier defines which samples are going to be saved.
- ✓ The sample configuration defines which parts of the sample are going to be stored (i.e. either PC or PC and instruction address).

### 4.5.1 Trigger

The trace is enabled for recording as soon as the trigger occurs and remains active until disabled. The trigger can be a certain watchpoint, breakpoint, load/store status, instruction status or their combination. For this purpose, you need to set the trigger selection register (TSEL register on page 12).

Note that very complex conditions might occur when using watchpoints and breakpoints as a trigger. Conditions for the occurrence of certain watchpoints and breakpoints need to be set at OpenRISC level (for details, please refer to the *OpenRISC 1000 System Architecture Manual*).

### 4.5.2 Qualifier

The qualifier defines which samples will be stored to the buffer. This is very useful when only a selection is to be stored (for example only write accesses to a specific address). The qualifier is defined in the QSEL register (page 14).

Note that very complex conditions might occur when using watchpoints and breakpoints as qualifier. Conditions for the occurrence of certain watchpoints and breakpoints need to be set at OpenRISC level (for details, please refer to the *OpenRISC 1000 System Architecture Manual*).

### 4.5.3 Stop Recording

The trace will stop recording when the following conditions are met:

- ✓ The buffer is full and the trace is set to normal mode (the CONTIN bit in the MODER register is not set). In this case, the recording is only suspended and will continue as soon as a sample is read out from the buffer (the buffer is not full).
- ✓ The trace is disabled (the ENABLE bit in the MODER register is set to 0).
- ✓ The stop condition occurs (described in the SSET register on page 16).

### 4.5.4 Sample Configuration (data selection for recording)

When the OpenRISC makes a step, the PC, instruction address, load/store date, etc. change. The RECSEL register (page 19) defines which information will compose a sample to be written to the buffer.

The sample is watchpoint and breakpoint independent.

### 4.5.5 Operation Modes

By setting the MODER, TSEL, QSEL, and SSEL registers, you can define several modes of operation. The most important modes are explained below:

- ✓ Post event recording: trace starts recording after an event occurred
- ✓ Prior event recording: trace records samples until a certain event occurs
- ✓ Post – prior event recording: trace records samples between two events.

#### 4.5.5.1 Post Event Recording

First, in the TSEL register, define the event that will activate the trigger. Thereupon, trace starts recording after the trigger-activating event occurred. If you want all samples to be stored to the buffer, set the qualifier to ANY; otherwise, choose the events you want to record (i.e. write to an address, read of the SPR, etc.). Once the buffer is full, two conditions can occur:

- ✓ If the CONTIN bit in the MODER register is set to 0, recording is suspended and the OpenRISC stalled. Once the stored sample is read (and therefore cleared) from the buffer, the operation resumes.
- ✓ If the CONTIN bit is set to 1, old samples are overwritten. To stop recording, disable trace – set the ENABLE bit (in the MODER) to 0. Then, the samples can be read out.

#### 4.5.5.2 Prior Event Recording

To start recording immediately prior to a certain event, set the ENABLE bit to 1 and set trigger to ANY. In the SSEL register, define the event upon which recording is to stop. To always overwrite old samples, set the CONTIN bit to 1. Once the defined event occurs, recording stops automatically. The samples can then be read out.

#### 4.5.5.3 Post - prior event recording

This is a combination of the previous two modes: Trace records samples that occur between two specific events. The start event sets the trigger and starts recording while the stop event stops recording. To not overwrite samples when the buffer is full, set the CONTIN bit to 0.

### 4.5.6 Reading Out Recorded Samples

The recorded data can be read out through the trace scan chain (please refer to the trace scan chain section on page 39). Prior to reading the data, you must select the trace scan chain.

Each sample written to the buffer is 40 bits wide:

- ✓ 32-bit data

- ✓ 4 bits for data type identification (information about the nature of the 32 bits– PC, load/store address, instruction address, etc.)
- ✓ 3 reserved bits
- ✓ 1 valid bit (marks valid samples)

This 40-bit sample is part of the trace scan chain as seen in Figure 6. The CRC bits and the valid bit are added to the sample while it is read out. The valid bit signals whether the sample is valid or not. This is useful because the samples can be read out any time and even if nothing was recorded. When a valid sample is read out, the pointer to the sample in the buffer is incremented automatically.

### 4.5.7 Stalling the OpenRISC

The OpenRISC clock is also used to clock the recording to the trace buffer. When a sample needs to be recorded, the trace asserts the CPUSTALL signal and stalls the OpenRISC for 8 clock cycles. Once the OpenRISC stops, the trace uses OPSELECT signals to change the data on the DATAIN lines. Samples are sequentially written to the trace buffer. After the last sample is written, CPUSTALL is de-asserted.

The OpenRISC is also stalled when the buffer is full and the trace is in normal operation mode (the CONTIN bit in the MODER is set to 0). When space is cleared in the buffer (data is read out), the stall signal is de-asserted.

## 4.6 Stalling the OpenRISC

The OpenRISC can be stalled in three ways:

- ✓ You deliberately set bit 0 of the RISCOP register to 1 (page 18). Clearing this bit again restarts the RISC.
- ✓ A breakpoint automatically stops the OpenRISC and sets bit 0 of the RISCOP register to 1. Clearing this bit again restarts the RISC.
- ✓ Trace stalls the OpenRISC.

## 4.7 Resetting the OpenRISC

The Development Interface puts the OpenRISC to reset by setting the RESET bit in the RISCOP register to 1. Clearing this bit to 0 deactivates the reset signal.

## 4.8 Enabling/Disabling DEBUG Mode

After reset, DEBUG mode (and OpenRISC stalling) is always enabled. Software can disable this mode by setting the Disable External Force Watchpoint DXFW bit in the debug mode register DMR (in the OpenRISC). For more details, please refer to the *OpenRISC 1000 System Architecture Manual*. Once the DEBUG mode has been disabled, the FORCEDBGIN input is ignored.

## 4.9 WISHBONE Interface (Memory Access)

The WISHBONE scan chain (WISHBONE master interface) enables writing to/ reading from the memory. On access, all cycles must be finished by asserting an acknowledge or error signal.

## 4.10 OpenRISC Debug Interface (SPR)

The OpenRISC debug chain enables writing to and reading from the debug interface (SPR). 64KB of the memory space is located at the offset 0x0 (0x00000000 to 0x0000FFFF). On access to any location within this range, the SPRs is accessed.

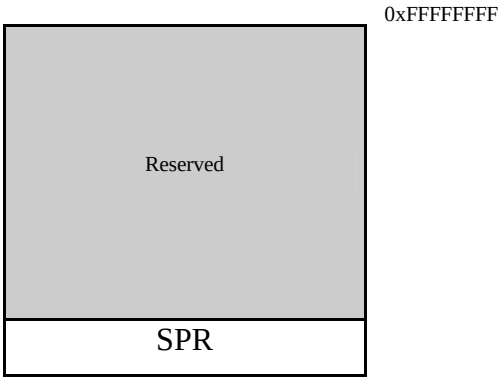


**Table 15: Memory Space**

According to the *OpenRISC 1000 System Architecture Manual*, the SPR registers address has a width of 16 bits. Five MSB are reserved for the group ID (GID), others define the register index within that group.

# 5

# Architecture

The Soc/OpenRISC Development Interface architecture is based on IEEE Std. 1149.1 Standard Test Access Port and Boundary Scan Architecture. Other signals are added to provide additional flexibility.

The interface consists of several parts (blocks):

- ✓ JTAG interface with the TAP Controller
- ✓ OpenRISC Debug Interface
- ✓ Trace
- ✓ Global BS (Boundary Scan) chain
- ✓ OpenRISC test chain
- ✓ Block scan chains
- ✓ WISHBONE scan chain
- ✓ Optional scan chains
- ✓ Block for monitoring internal signals

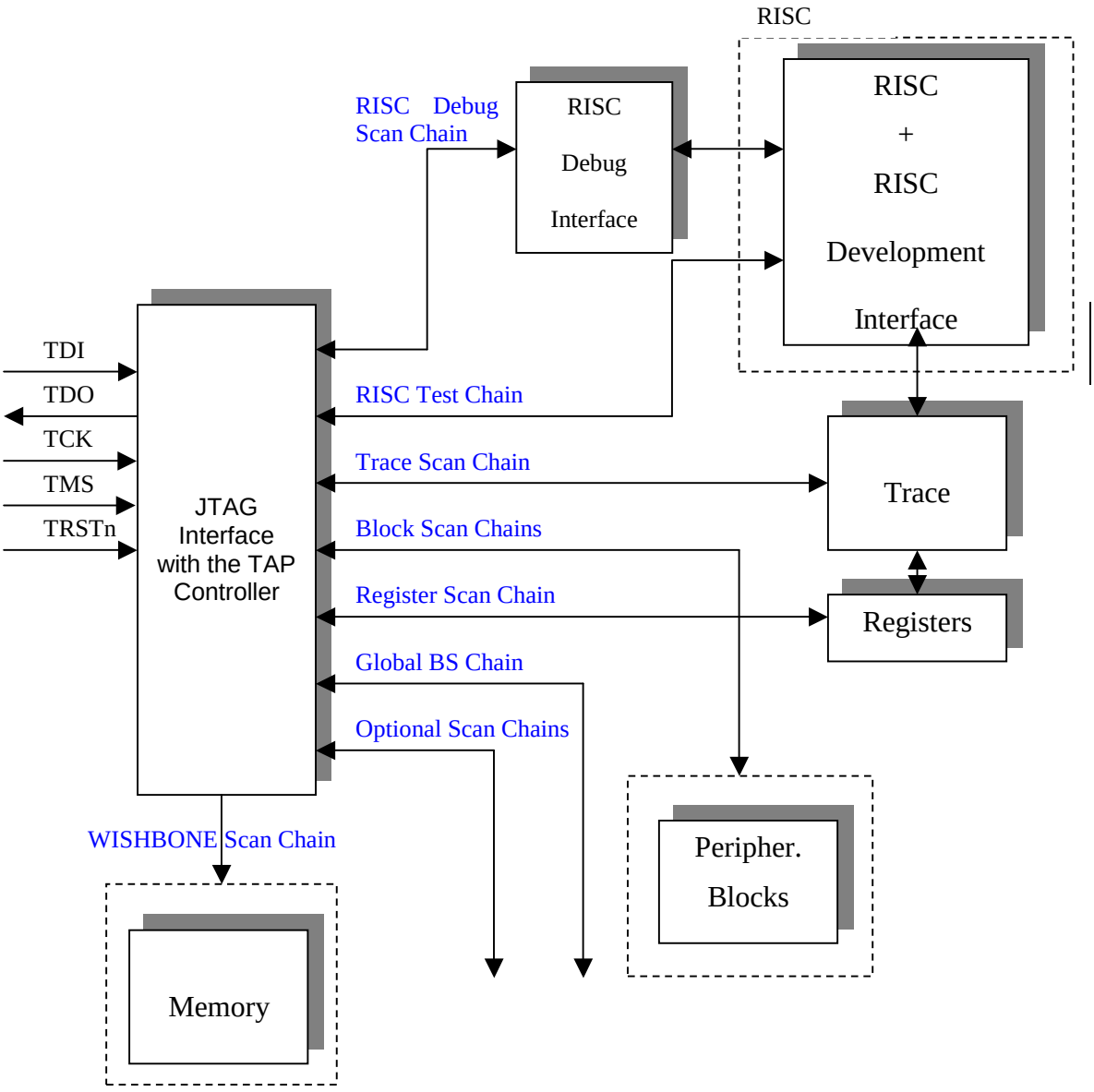


**Figure 2: Development Interface**

## 5.1 JTAG Interface with TAP Controller

The interface is fully IEEE Std. 1149.1 compliant. It is used for interfacing the chip to the external debugger.

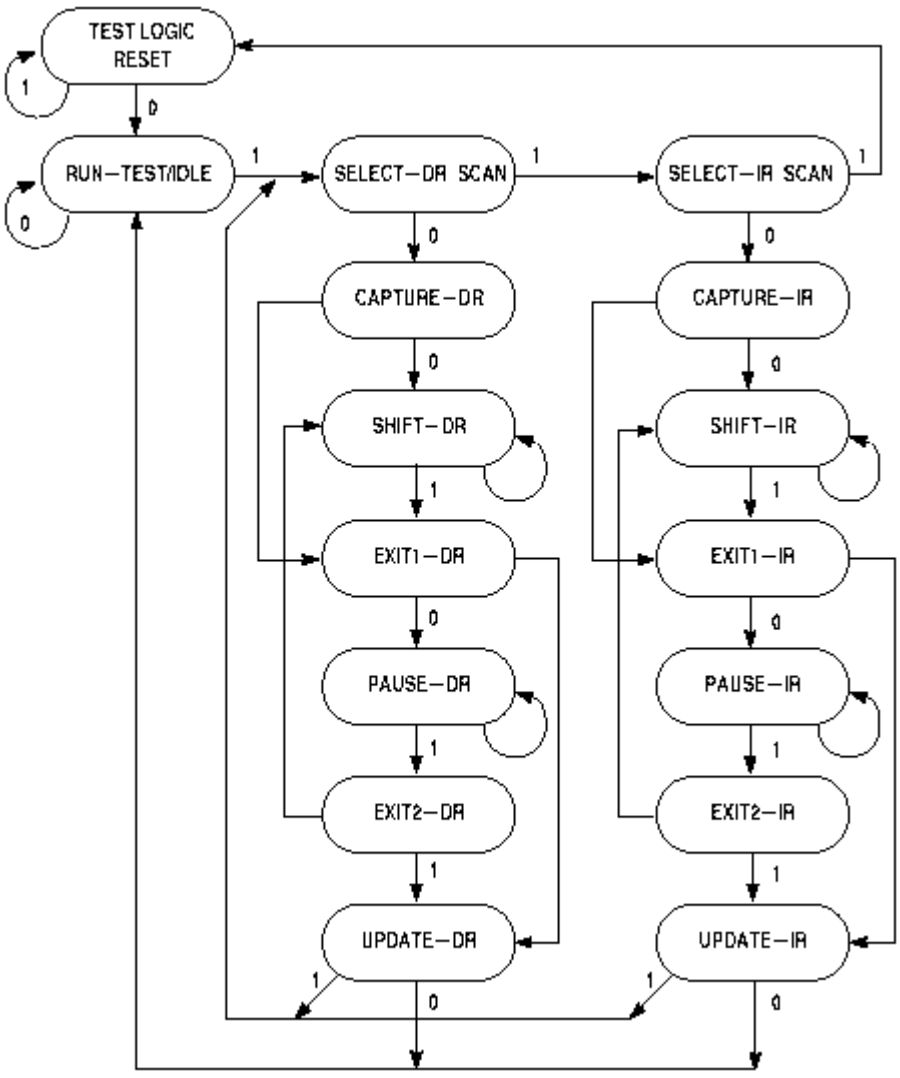


**Figure 3: TAP Controller**

## 5.2 Scan Chains

### 5.2.1 Global BS (Boundary Scan) Chain

This chain allows access to the entire OpenRISC periphery and is used for the boundary scan testing (interconnect test). The chain is automatically selected after the reset. When in BS TEST mode, two tests can be run: EXTEST when connections between BS devices are tested and INTEST when the IC functionality is tested.

### 5.2.2 OpenRISC Debug Interface Scan Chain

The OpenRISC Debug Interface scan chain is used for interfacing to the OpenRISC debug support. Using this chain, which is 74 bits long, data can be read from/written to the registers that are used for debugging purposes (watchpoint generation, breakpoint generation, memory read/write, SPR, GPR, etc.). The chain for shifting in is different from the chain for shifting out:

- ✓ Chain for shifting in: 32-bit address, R/W bit, 32-bit data, 8-bit CRC1, 1 reserved bit
- ✓ Chain for shifting out: 33 bits set to 0x0, 32-bit data, and 9-bit field used for CRC2

While shifting in is in progress on the first chain, shifting out is in progress on the second chain.

A read operation is performed when all data (address, data, RW, and CRC1) has been shifted in and input CRC1 equals to the internally calculated CRC. During the next shifting process, read data is latched and can be shifted out. The next CRC2 is shifted out jointly with the data.

When a write operation needs to be performed, the shifted-out CRC2 and the shifted-in CRC1 concur (one TCK clock delay). This is done so that debugging software can detect an occurring error condition and repeat the sequence. The write operation is also performed when the input CRC1 equals to the internally calculated CRC.

Two CRC codes are also shifted in and out:

- ✓ CRC1 is shifted in (Figure 4: OpenRISC Debug Interface Scan Chain (data shifted in)). The host calculates it from the address, the R/W bit, and data that is sent in.
- ✓ CRC2 is shifted out (Figure 5: OpenRISC Debug Interface Scan Chain (data shifted out)). It is calculated from the address, the R/W that were shifted in, and data that is

shifted out (data read from the register) when a read operation is in progress, or from a delayed CRC1 when a write operation is in progress.

After CRC1 has been shifted in, it is compared to the internally calculated CRC. If both CRC codes do not match, the TDO is set to 0 when the TAP is in the UpdateDR stage. If they do match, the TDO is set to 1. In this case, a read or write cycle is performed (after the UpdateDR stage).

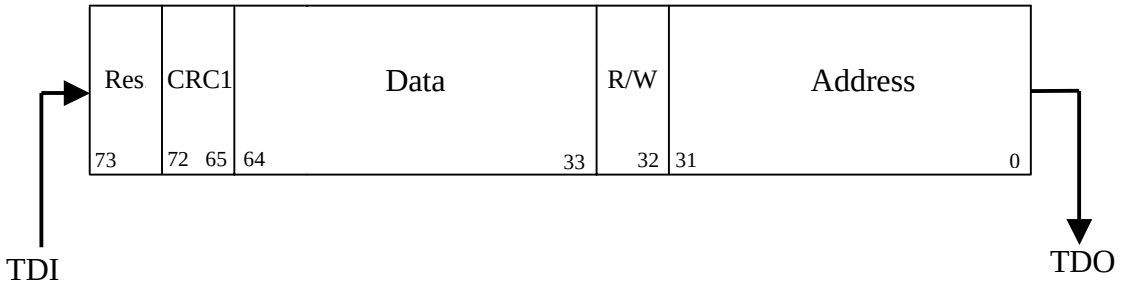


**Figure 4: OpenRISC Debug Interface Scan Chain (data shifted in)**

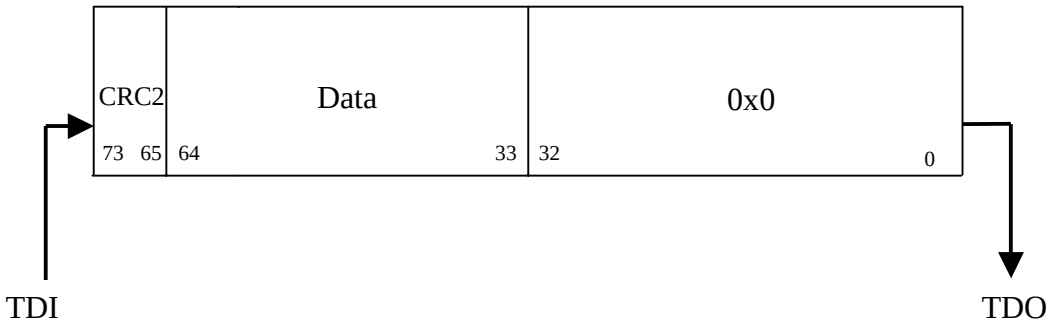


**Figure 5: OpenRISC Debug Interface Scan Chain (data shifted out)**

### 5.2.3 OpenRISC Test Chain

When this chain is selected, issuing the INTEST commands can test the OpenRISC functionality. Since the definition of the appropriate test chain is the system integrator's responsibility, the factory needs to perform the test.

### 5.2.4 Trace Scan Chain

The trace scan chain is used for reading the content of the trace buffer. The chain is 48 bits long – 8 bits for CRC, 36 bits for the recorded samples, 3 bits reserved for future use, and one bit for the sample valid status.

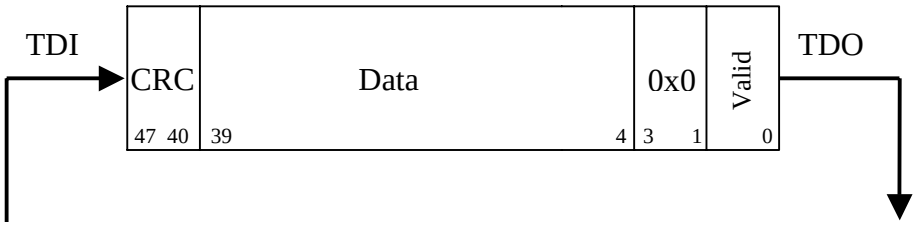


**Figure 6: Trace Scan Chain**

### 5.2.5 Register Scan Chain

The register scan chain is used for writing and reading the data to/from the registers used in this development interface. The chain is 47 bits long. The chain for shifting in differs from the chain for shifting out:

- ✓ Chain for shifting in: 5-bit address, R/W bit, 32-bit data, 8-bit CRC1, and 1 reserved bit
- ✓ Chain for shifting out: 6 bits set to 0x0, 32-bit data, and 9-bit CRC2

While shifting in is in progress on the first chain, shifting out is performed on the second chain.

A read operation is performed after all data (address, data, RW, and CRC1) has been shifted in and when the input CRC1 equals to the CRC calculated internally. Read data is

latched and can be shifted out in the next shifting process. The new CRC2 is shifted out together with the data.

When a write operation needs to be performed, the CRC2 that is shifted out equals the CRC1 that is shifted in (one TCK clock delay). This is done so that debugging software can detect an error condition and repeat the sequence. The write operation is also performed when the input CRC1 equals the one that is internally calculated.

Two CRC codes are shifted in and out:

- ✓ CRC1 is shifted in (Figure 7: Register Scan Chain (data shifted in)). The host calculates it from the address, R/W bit, and data that is sent in.
- ✓ CRC2 is shifted out (Figure 8: Register Scan Chain (data shifted out)). The host calculates it from the address, the R/W that are shifted in, and data that is shifted out (data read from a register) when a read operation is in progress, or from delayed CRC1 when a write operation is in progress.

When the CRC1 is shifted in, it is compared to the internally calculated CRC. If both CRC codes do not match, the TDO is set to 0 when the TAP is in the UpdateDR stage. If they do match, the TDO is set to 1. In this case, a read or write cycle is performed (after the UpdateDR stage).

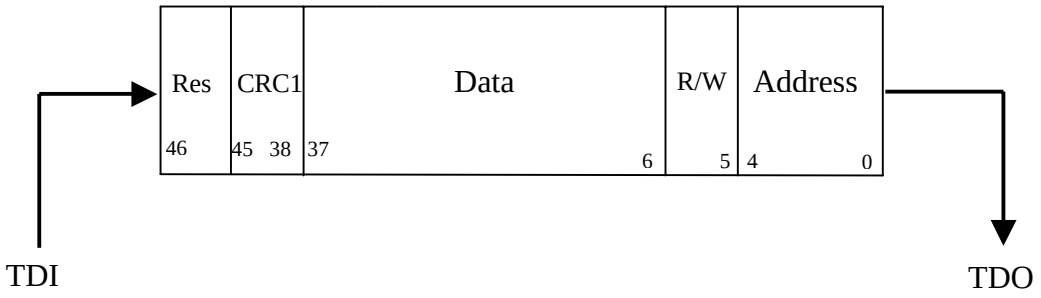


**Figure 7: Register Scan Chain (data shifted in)**

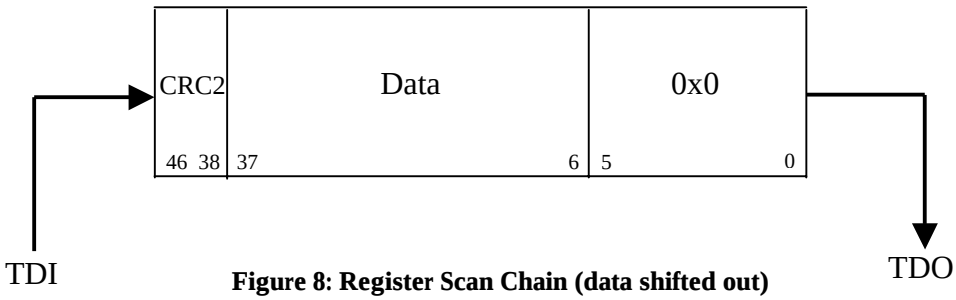


**Figure 8: Register Scan Chain (data shifted out)**

### 5.2.6 WISHBONE Scan Chain

The WISHBONE scan chain is used for interfacing WISHBONE slave devices – cores (memory). The chain is 74 bits long. The one for shifting in differs from the one for shifting out:

- ✓ Chain for shifting in: 32-bit address, R/W bit, 32-bit data, 8-bit CRC, 1 reserved bit
- ✓ Chain for shifting out: 1 bit indicating legal WISHBONE access (when access is finished with wb_err_i signal set to 1, this bit is set), 1 bit indicating that WISHBONE access is still in progress, 31 bits set to 0x0, 32-bit data, 9-bit CRC

While shifting in is in progress on the first chain, shifting out is performed on the second chain simultaneously.

A read operation is performed after all data (address, data, RW, and CRC1) have been shifted in and when the input CRC1 equals to the CRC that is calculated internally. Read data is latched and can be shifted out in the next shifting process. New CRC2 is shifted out simultaneously with the data. Bit Access Fail indicates legal (when set to 0) or illegal (when set to 1) access to the WISHBONE slave device. Bit InProgress indicates that access is still in progress (not finished by acknowledge or error signal). In this case any following accesses are ignored. After 256 WISHBONE clock cycles, bit InProgress and wb_cyc_o signal are cleared and the core is ready for next access.

When a write operation needs to be performed, the shifted-out CRC2 equals the shifted-in CRC1 (one TCK clock delay). This is done so that debugging software can detect an error condition and repeat the sequence. The write operation is also performed when the input CRC1 equals the one calculated internally.

Two CRC codes are also shifted in and out:

- ✓ CRC1 is shifted in (Figure 9: WISHBONE Scan Chain (data shifted in)). The host calculates it from the address, R/W bit, and data that are sent in.
- ✓ CRC2 is shifted out (Figure 10: WISHBONE Scan Chain (data shifted out)). It is calculated from the address, the R/W that were shifted in, data that is shifted out (data read from register) when a read operation is in progress, or from a delayed CRC1 when a write operation is in progress.

When the CRC1 has been shifted in, it is compared to the CRC that is internally calculated. If both CRC codes do not match, the TDO is set to 0 when the TAP is in the

UpdateDR stage. If they do match, the TDO is set to 1. In this case, a read or write cycle is performed (after the UpdateDR stage).

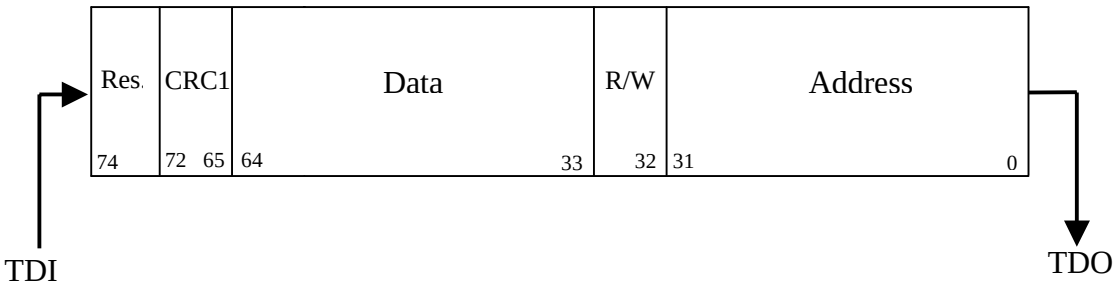


**Figure 9: WISHBONE Scan Chain (data shifted in)**

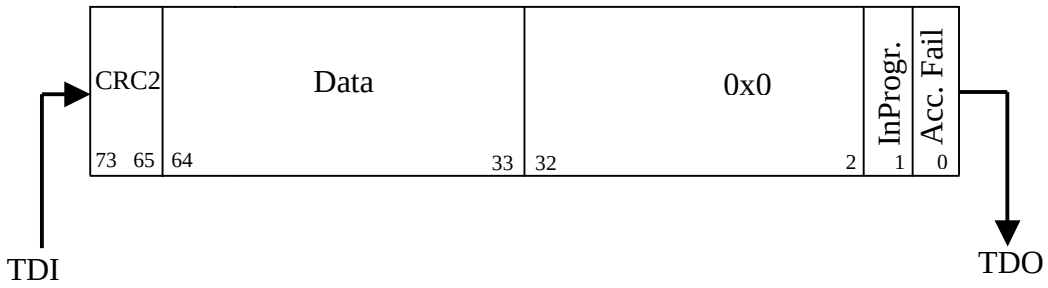


**Figure 10: WISHBONE Scan Chain (data shifted out)**

## 5.2.7 Block Scan Chains

Several block scan chains can be used in the SoC/OpenRISC for observing the operation of different peripheral cores. A block scan chain connects to the WISHBONE interface of the core rather than to the core itself and can only observe but not control the core's activity. The length of the scan chain depends on the number of the signals used as a WISHBONE interface. It is up to the system integrator to connect these scan chains to those in need of observation.

### 5.2.8 Optional Scan Chains

Optional scan chains can be used for both observing and controlling. Currently, they are reserved for future demands.

## 5.3 OpenRISC Debug Interface

The OpenRISC Debug Interface is used for interfacing the external devices (debugger) to the OpenRISC debug facilities.

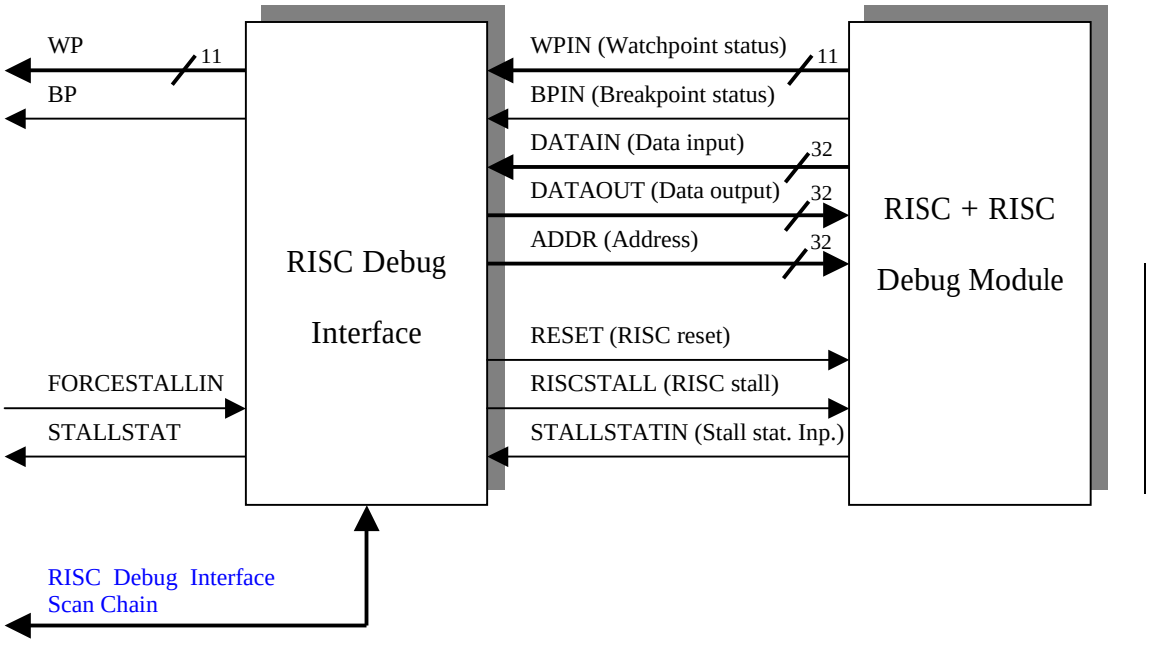


**Figure 11: OpenRISC Debug Interface**

## 5.4 Trace

Trace records selected samples to the trace buffer. The samples are read and passed to the external debugger using the trace scan chain.

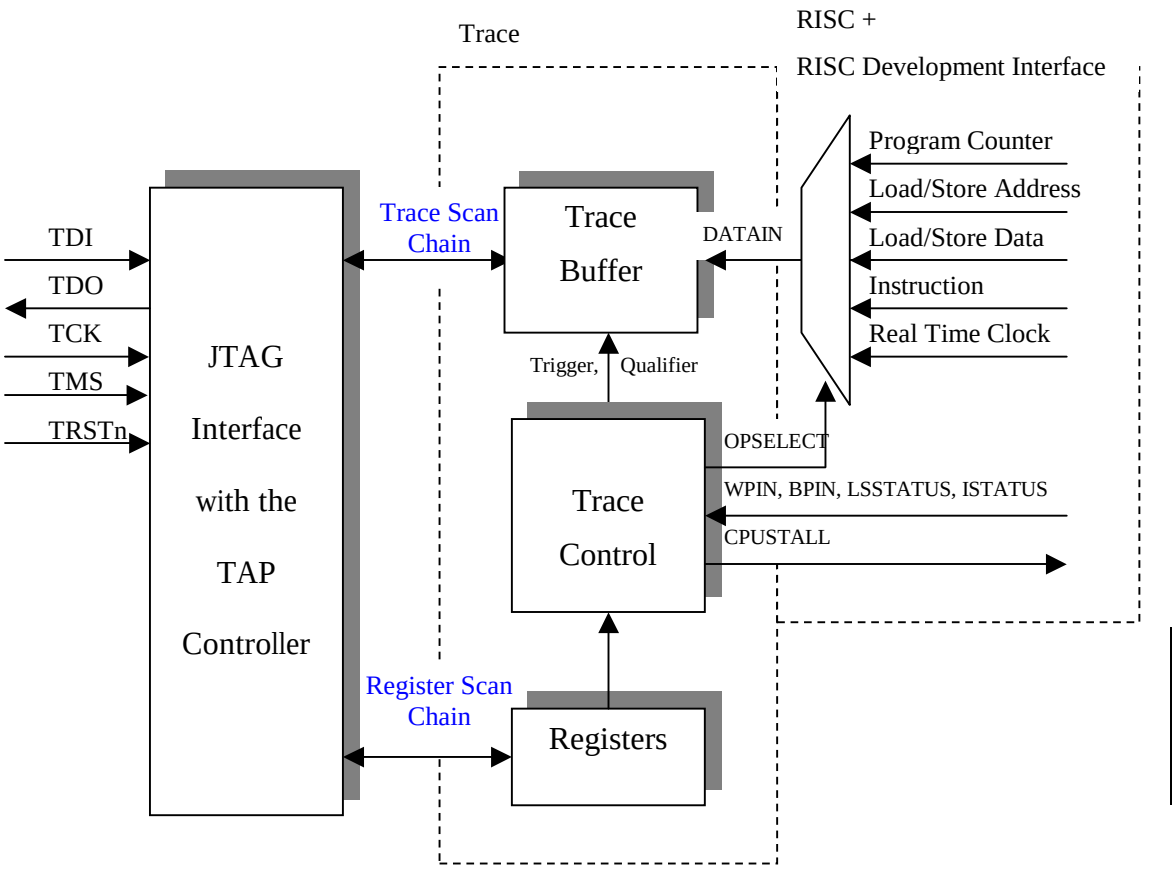


**Figure 12: Trace**

## 5.5 Observing internal signals

Through the DIROUT[31:0], you can monitor the state of several internal signals. The monitoring is development port independent. Select with the DIRSEL[2:0] which set of signals is connected. It is up to the system integrator to decide which signals need to be observed and how to connect them to the multiplexer. He also has to decide whether to use dedicated pins for the DIROUT and DIRSEL signals or to multiplex them with some other pins.

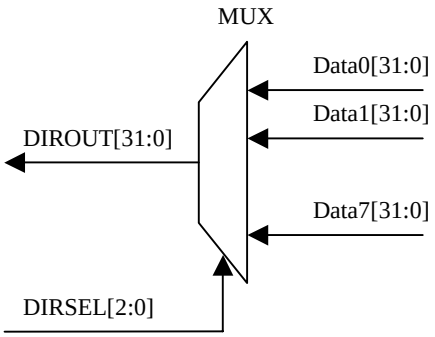


**Figure 13: Selection of Observed Signals**

# Appendix A

# Configuring Trace

Assuming you want to record all store data operations once the program enters the subroutine X, there are many ways to do so. Here is one example:

First, select the trace scan chain. The following events take place:

- ✓ The Instruction SELECT_CHAIN is shifted in to the TAP controller (see Table 13: TAP Instruction Set on page 22)
- ✓ The trace scan chain ID must be shifted in to the TAP controller (see Table 14: Chains Identification on page 27)

Then, put the development port to DEBUG mode. The following events take place:

- ✓ Instruction DEBUG is shifted in to the TAP controller (see Table 13: TAP Instruction Set on page 22)

Set the trigger:

- ✓ In the OpenRISC, set the watchpoint 0 to be asserted when the program executes the jump to the subroutine X. Please refer to the *OpenRISC 1000 System Architecture Manual* for more information.
- ✓ Set value 0xC0000801 to the TSEL register. This way, you instruct the trace to start recording when the WP0 occurs.

Set the qualifier:

- ✓ Set value 0xC01F0000 to the QSEL register. This way, you instruct the trace to record only when a store data operation occurs.

Set the record selection:

- ✓ Set value 0x00000008 to RECSEL so that samples will only consist of the stored data.

Set the TRACE mode and enable it:

- ✓ Set value 0x00010000 to the MODER register. This sets the trace to the normal mode (old samples will never be overwritten) and enables the trace.

Now, start the OpenRISC. The data can be read out through the trace scan chain. If the buffer contains valid records, the valid bit is set to 1. In case the buffer is full, the OpenRISC will be stalled until samples are not read out.