SoC/OpenRISC Development Interface

Author: Igor Mohor
igorm@opencores.org

Rev. 0.11
September 10, 2001

Preliminary Draft

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	The meaning of Instruction status and Load/store status changed in all registers. More details added to the Appendix A.
0.11	10/09/01	IM	Register and OpenRISC Scan Chain operation changed.

Contents:

1

Introduction

Development Interface is used for development purposes (Boundary Scan testing and debugging). It is 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 JTAG port that is fully IEEE 1149.1 compatible. The Development Port also contains a trace buffer and support for tracing the program flow, execution coverage and profiling the code.

2

IO ports

2.1 Ports connected to the external device (debugger)

Port	Width	Direction	Description
TCLK	1	Input	**Test Clock** is the clock for the JTAG interface.
TMS	1	Input	**Test Mode Select** is the signal that controls the TAP machine sequence in the JTAG interface.
TDI	1	Input	**Test Data Input** is the input data for the JTAG interface.
TDO	1	Output	**Test Data Out** is the output data from the JTAG interface.
TRSTn	1	Input	**Test Reset** is an active low input signal that asynchronously resets the TAP machine in the JTAG interface.
MCLK	1	Input	**Master Clock** signal. This clock signal is equal to the OpenRISC clock.
WP	11	Output	**Watchpoint Status** is an output signal that reports the watchpoint generation.
BP	1	Output	**Breakpoint Status** is an output signal that reports the breakpoint generation.
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 that wishes to be observed to the DIROUT signals.
DIRSEL	3	Input	**Direct Signals Selection** is select signals for the output multiplexer so many different signals can be connected to the DIROUT signals.
FORCESTALLIN	1	Input	**Force Stall Input** is 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 OpenRISC is stalled

Table 1: Ports connected to the external device

2.2 Ports connected to internal devices (OpenRISC, Reset Logic, WISHBONE interface)

Port	Width	Direction	Description
WPIN	11	Input	**Watchpoint Status** is an input signal that reports the OpenRISC watchpoint generation.
BPIN	1	Input	**Breakpoint Status** is an input signal that reports the OpenRISC breakpoint generation.
DATAIN	32	Input	**Data Input** transfers the data from OpenRISC to the development interface.
DATAOUT	32	Output	**Data Output** transfers the data from development interface to the OpenRISC.
ADDR	32	Output	**Address** of the special-purpose register to be read or written.
OPSELECT	4	Output	**Operation Select** asynchronously selects the operation of the OpenRISC debug port.
LSSTATUS	4	Input	**Load/Store Status** synchronously shows the status of the Load/Store unit.
ISTATUS	4	Input	**Instruction Status** synchronously shows the status of the Instruction Fetch unit.
CPUSTALL	1	Output	**CPU Stall** synchronously stalls the OpenRISC's core.
STALLSTATIN	1	Input	**Stall Status Input** is connected to the OpenRISC. When the OpenRISC is stalled, it asserts this signal.
RISC_CS	1	Output	**Chip Select** signal. Activates read/write operation of the OpenRISC registers. Active high.
RISC_RW	1	Output	**Read/Write** select when accessing OpenRISC registers.
RESET	1	Output	**Reset Output** is connected to the OpenRISC (OpenRISC's reset logic). When set, OpenRISC is in reset.
CLK_I	1	Input	**Clock Input** (WISHBONE clock)
RST_I	1	Input	**Reset Input** (WISHBONE reset)
ACK_I	1	Input	**Acknowledgment Input** indicates a normal cycle termination.
ADR_O	32	Output	**Address Output array**
CYC_O	1	Output	**Cycle output** encapsulates a valid transfer cycle.
DAT_I	32	Input	**Data Input array**
DAT_O	32	Output	**Data Output array**
ERR_I	1	Input	**Error acknowledgment input** indicates an

			abnormal cycle termination.
RTY_I	1	Input	**Retry Input** indicates that the memory is not ready, and the master should retry this operation.
SEL_O	4	Output	**Select Output** indicates which bytes are valid on the data bus.
STB_O	1	Output	**Strobe Output** indicates a valid transfer.
WE_O	1	Output	**Write Enable** indicates a Write Cycle when asserted high.

Table 2: Ports connected to the OpenRISC, Reset Logic and WISHBONE Interface

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	0x1	32	R/W	Trigger Selection for the Trace Buffer.
QSEL	0x2	32	R/W	Qualifier Selection for the Trace Buffer.
SSEL	0x3	32	R/W	Stop Select Register
RISCOP	0x9	32	R/W	OpenRISC Operation Register
RECWP0	0x10	32	R/W	Record Selection for WP[0].
RECWP1	0x11	32	R/W	Record Selection for WP[1].
RECWP2	0x12	32	R/W	Record Selection for WP[2].
RECWP3	0x13	32	R/W	Record Selection for WP[3].
RECWP4	0x14	32	R/W	Record Selection for WP[4].
RECWP5	0x15	32	R/W	Record Selection for WP[5].
RECWP6	0x16	32	R/W	Record Selection for WP[6].
RECWP7	0x17	32	R/W	Record Selection for WP[7].
RECWP8	0x18	32	R/W	Record Selection for WP[8].
RECWP9	0x19	32	R/W	Record Selection for WP[9].
RECWP10	0x1A	32	R/W	Record Selection for WP[10].
RECBP0	0x1B	32	R/W	Record Selection for BP[0].

Table 3: Register list

3.2 MODER (Mode Register)

Bit #	Access	Description
31:3		Reserved
2	R/W	RECSELDEPEND – Record Selection Dependency 0 = Record Selection is watchpoint and breakpoint independent. RECWP0 is used for global record selection. 1 = Record Selection is watchpoint and breakpoint dependent (RECWPn and RECBP0 registers are valid).
1	R/W	ENABLE – Trace Enable 0 = Trace is disabled 1 = Trace is enabled
0	R/W	CONTIN – Continuous Mode 0 = Recording is suspended while the trace buffer is full. 1 = Old samples are overwritten with new samples once the trace buffer is full.

Table 4: MODER Register

Reset Value:
MODER: 00000000h

3.3 TSEL (Trigger Select Register)

Bit #	Access	Description
31:30	R/W	TRIGOP – Trigger Operation 00 = Any (trigger is always active) 10 = All valid groups in the TSEL register are OR-ed. 11 = All valid groups in the TSEL register are AND-ed.
29:24		Reserved
23	R/W	ISTRIGVALID – Instruction Status Trigger Valid 0 = Don't care (Instruction Status Trigger is not valid). 1 = Instruction Status Trigger is valid.
22:21	R/W	ISTRIG – Instruction Status Trigger 00 = No instruction fetch in progress. 01 = Normal instruction fetch. 10 = Executing branch instruction. 11 = Fetching instruction in delay slot.
20	R/W	LSSTRIGVALID – Load/Store Status Trigger Valid 0 = Don't care (Load/Store Status Trigger is not valid).

		1 = Load/Store Status Trigger is valid.
19:16	R/W	LSSTRIG – Load/Store Status Trigger 0000 = No load/store instruction in execution. 0001 = Reserved for load doubleword. 0010 = Load byte and zero extend. 0011 = Load byte and sign extend. 0100 = Load halfword and zero extend. 0101 = Load halfword and sign extend. 0110 = Load singleword and zero extend. 0111 = Load singleword and sign extend. 1000 = Reserved for store doubleword. 1001 = Reserved. 1010 = Store byte. 1011 = Reserved. 1100 = Store halfword. 1101 = Reserved. 1110 = Reserved. 1111 = Reserved.
15:14		Reserved
13	R/W	BPTRIGVALID – Breakpoint Trigger Valid 0 = Don't care (BP trigger is not valid). 1 = BP trigger is valid.
12	R/W	BPTRIG – Breakpoint Trigger 0 = Breakpoint doesn't start recording of the trace unit. 1 = BP[0] starts recording of the trace unit.
11	R/W	WPTRIGVALID – Watchpoint Trigger Valid 0 = Don't care (WP trigger is not valid). 1 = WP trigger is valid.
10:0	R/W	WPTRIG – Watchpoint Trigger 00000000000 = Watchpoints don't start recording of the trace unit. 00000000001 = WP[0] starts recording of the trace unit. 00000000010 = WP[1] starts recording of the trace unit. . 11111111111 = Any WP[10:0] starts recording of the trace unit.

Table 5: TSEL Register

Reset Value:

TSEL: 00000000h

3.4 QSEL (Qualifier Select Register)

Bit #	Access	Description
31:30	R/W	QUALIFOP – Qualifier Operation 00 = Any (Qualifier is always active, all samples will be recorded) 10 = All valid groups in the QSEL register are OR-ed. 11 = All valid groups in the QSEL register are AND-ed.
29:24		Reserved
23	R/W	ISTQUALIFVALID – Instruction Status Qualifier Valid 0 = Don't care (Instruction Status Qualifier is not valid). 1 = Instruction Status Qualifier is valid.
22:21	R/W	ISTQUALIF – Instruction Status Qualifier 00 = No instruction fetch in progress. 01 = Normal instruction fetch. 10 = Executing branch instruction. 11 = Fetching instruction in delay slot.
20	R/W	LSSQUALIFVALID – Load/Store Status Qualifier Valid 0 = Don't care (Load/Store Status Qualifier is not valid). 1 = Load/Store Status Qualifier is valid.
19:16	R/W	LSSQUALIF – Load/Store Status Qualifier 0000 = No load/store instruction in execution. 0001 = Reserved for load doubleword. 0010 = Load byte and zero extend. 0011 = Load byte and sign extend. 0100 = Load halfword and zero extend. 0101 = Load halfword and sign extend. 0110 = Load singleword and zero extend. 0111 = Load singleword and sign extend. 1000 = Reserved for store doubleword. 1001 = Reserved. 1010 = Store byte. 1011 = Reserved. 1100 = Store halfword. 1101 = Reserved. 1110 = Reserved. 1111 = Reserved.
15:14		Reserved
13	R/W	BPQUALIFVALID – Breakpoint Qualifier Valid 0 = Don't care (BP qualifier is not valid). 1 = BP qualifier is valid.
12	R/W	BPQUALIF – Breakpoint Qualifier 0 = Breakpoint doesn't enable recording of the current sample. 1 = BP[0] enables recording of the current sample.
11	R/W	WPQUALIFVALID – Watchpoint Qualifier Valid 0 = Don't care (WP qualifier is not valid).

		1 = WP qualifier is valid.
10:0	R/W	WPQUALIF – Watchpoint Qualifier 00000000000 = Watchpoints don't enable recording of the current sample. 00000000001 = WP[0] enables recording of the current sample. 00000000010 = WP[1] enables recording of the current sample. . 11111111111 = Any WP[10:0] enables recording of the current sample.

Table 6: QSEL Register

Reset Value:
QSEL: 00000000h

3.5 SSEL (Stop Select Register)

Bit #	Access	Description
31:30	R/W	STOPOP – Stop Operation 00 = Nothing (Recording can't be stopped by selecting one of the following groups) 10 = All valid groups in the SSEL register are OR-ed. 11 = All valid groups in the SSEL register are AND-ed.
29:24		Reserved
23	R/W	ISTSTOPVALID – Instruction Status Stop Valid 0 = Don't care (Instruction Status Stop is not valid). 1 = Instruction Status Stop is valid.
22:21	R/W	ISTSTOP – Instruction Status Stop 00 = No instruction fetch in progress. 01 = Normal instruction fetch. 10 = Executing branch instruction. 11 = Fetching instruction in delay slot.
20	R/W	LSSSTOPVALID – Load/Store Status Stop Valid 0 = Don't care (Load/Store Status Stop is not valid). 1 = Load/Store Status Stop is valid.
19:16	R/W	LSSSTOP – Load/Store Status Stop 0000 = No load/store instruction in execution. 0001 = Reserved for load doubleword. 0010 = Load byte and zero extend. 0011 = Load byte and sign extend.

		0100 = Load halfword and zero extend. 0101 = Load halfword and sign extend. 0110 = Load singleword and zero extend. 0111 = Load singleword and sign extend. 1000 = Reserved for store doubleword. 1001 = Reserved. 1010 = Store byte. 1011 = Reserved. 1100 = Store halfword. 1101 = Reserved. 1110 = Reserved. 1111 = Reserved.
15:14		Reserved
13	R/W	BPSTOPVALID – Breakpoint Stop Valid 0 = Don't care (BP Stop is not valid). 1 = BP Stop is valid.
12	R/W	BPSTOP – Breakpoint Stop 0 = Breakpoint doesn't stop recording. 1 = BP[0] stops recording.
11	R/W	WPSTOPVALID – Watchpoint Stop Valid 0 = Don't care (WP Stop is not valid). 1 = WP Stop is valid.
10:0	R/W	WPSTOP – Watchpoint Stop 00000000000 = Watchpoints don't stop recording. 00000000001 = WP[0] stops recording. 00000000010 = WP[1] stops recording. . 11111111111 = Any WP[10:0] stop recording.

Table 7: SSEL Register

Reset Value:

SSEL: 00000000h

3.6 RISCOP (OpenRISC Operation Register)

Bit #	Access	Description
31:2		Reserved
1	R	RESET – Reset OpenRISC 0 = normal 1 = reset

0	R/W	RISCSTALL – OpenRISC Stall 0 = normal operation 1 = Stall OpenRISC

Table 8: RISCOP Register

Reset Value:
RISCOP: 00000000h

3.7 RECWPn (Record Selection Register for WP[n])

Bit #	Access	Description
31:7		Reserved
6	R/W	RECINSTR – Record Instruction in the Execution Pipeline 0 = Don't save INSTR to the trace buffer 1 = Save INSTR to the trace buffer
5	R/W	RECWRITESPR – Record Writing SPR 0 = Don't save WRITESPR to the trace buffer 1 = Save WRITESPR to the trace buffer
4	R/W	RECREADSPR – Record Reading SPR 0 = Don't save READSPR to the trace buffer 1 = Save READSPR to the trace buffer
3	R/W	RECSDATA – Record Store Data 0 = Don't save SDATA to the trace buffer 1 = Save SDATA to the trace buffer
2	R/W	RECLDATA – Record Load Data 0 = Don't save LDATA to the trace buffer 1 = Save LDATA to the trace buffer
1	R/W	RECLSEA – Record LSEA (Load/Store Effective Address) 0 = Don't save LSEA to the trace buffer 1 = Save LSEA to the trace buffer
0	R/W	RECPC – Record PC (Program Counter) 0 = Don't save PC to the trace buffer 1 = Save PC to the trace buffer

Table 9: RECWPn Register

Reset Value:
RECWPn: 00000000h

3.8 RECBP0 (Record Selection Register for BP[0])

Bit #	Access	Description
31:7		Reserved
6	R/W	RECINSTR – Record Instruction in the Execution Pipeline 0 = Don't save INSTR to the trace buffer 1 = Save INSTR to the trace buffer
5	R/W	RECWRITESPR – Record Writing SPR 0 = Don't save WRITESPR to the trace buffer 1 = Save WRITESPR to the trace buffer
4	R/W	RECREADSPR – Record Reading SPR 0 = Don't save READSPR to the trace buffer 1 = Save READSPR to the trace buffer
3	R/W	RECSDATA – Record Store Data 0 = Don't save SDATA to the trace buffer 1 = Save SDATA to the trace buffer
2	R/W	RECLDATA – Record Load Data 0 = Don't save LDATA to the trace buffer 1 = Save LDATA to the trace buffer
1	R/W	RECLSEA – Record LSEA (Load/Store Effective Address) 0 = Don't save LSEA to the trace buffer 1 = Save LSEA to the trace buffer
0	R/W	RECPC – Record PC (Program Counter) 0 = Don't save PC to the trace buffer 1 = Save PC to the trace buffer

Table 10: RECBP0 Register

Reset Value:

RECBP0: 00000000h

4

Operation

This section describes the operation of the development interface. It discusses about visibility of the internal signals, JTAG interface with the TAP controller and supported instructions, scan chain configuration, watchpoints and breakpoints, trace, memory, SPR and GPR interface, and at the end how to enter and exit the debug mode.

4.1 Visibility of the internal signals

The state of several internal signals can be monitored through the DIROUT[31:0] signals. The monitoring is development port independent. Which set of signals is connected through the multiplexer to the output pins is selected with the DIRSEL[2:0] signals. 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.

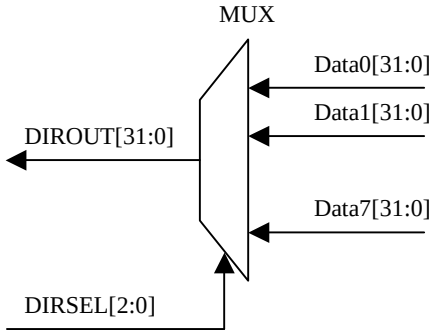

Figure 1: Selection of the observed signals

4.2 JTAG Interface with the TAP Controller and Instructions

JTAG Interface is fully IEEE Std.1149.1 compliant. It 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 11: TAP Instruction Set

Which instructions will be supported is still TBD.

4.2.1 EXTEST (IR=0000)

The EXTEST instruction connects the selected chain between the TDI and TDO. The selected chain is put in test mode. During EXTEST instruction, the boundary-scan register is accessed to drive test data off-chip via the boundary outputs and 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 selects the boundary-scan register to be connected between TDI and TDO. During this instruction, the boundary-scan register can be accessed via a data scan operation, to take a sample of the functional data entering and leaving the IC. This 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 the 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 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 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 is connected between TDI and TDO and put into test mode.

- CaptureDR state: Both, the input (from the system logic) and the output (from the core) signals 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)

CLAMP instruction is used for setting all the outputs to the pre-loaded values (values that remain from the previous shifting or loaded with the SAMPLE/PRELOAD instruction). 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 except 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 to 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 the 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 additional cycle to shift out the results (it won't work).

4.2.10 BYPASS (IR=1111)

The BYPASS instruction allows the IC to remain in a functional mode and selects the bypass register to be connected between TDI and TDO. The BYPASS instruction 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 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

There are several scan chains in the chip that can be used for BS testing, OpenRISC debugging, trace operation, other cores observing (at the connection to the WISHBONE), identifying the chip version, etc. They are:

- Global Boundary Scan Chain
- OpenRISC Debug Interface Scan Chain
- OpenRISC Test Chain
- Trace Scan Chain
- Register Scan Chain
- Block Scan Chain
- Optional Scan Chains

A chain is selected by issuing a CHAIN_SELECT instruction followed by the chains unique ID value (and CRC). All 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 registers accessing
Block Scan Chains	0101 - 0111	For peripheral cores testing
Optional Scan Chains (can be connected to any core, register, counter, etc.)	1000 - 1111	For additional testing

Table 12: Chains Identification

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

4.4 Watchpoints and Breakpoints

Watchpoints WP[10:0] and breakpoint BP[0] are output signals from the OpenRISC debug module and input signals to the SoC/OpenRISC Development Interface. These

signals are also connected to the output pads and can be used for informing the external devices when certain events occur (they can set a trigger, enable/disable a device, put device into the sleep mode, etc.).
They are used as triggers or/and qualifiers for the trace module.

4.5 Trace

One block in the Development Interface is Trace. It is a 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 that is needed for program tracing, execution coverage and profiling (measuring the time that subroutines need for their execution). The recording starts as soon as the trigger and qualifier both occur. After 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, several things need to be done:

- Trigger needs to be set (TSEL register on page 11).
- Qualifier needs to be set (QSEL register on page 12).
- The configuration of the sample need to be set – what kind of information will be stored in the buffer (RECWPn and RECBP0 registers on page 16).
- The stop condition needs to be set (SSEL register on page 14).
- The operation mode needs to be set and trace must be enabled (MODER register on page 10).

To prevent mixing the purpose of the trigger, qualifier and sample configuration let's explain their use in plain words:

- Trigger starts the trace (Once started it might record the samples)
- Qualifier defines which samples are going to be saved (a sample changes each time when the OpenRISC makes a step).
- Sample configuration defines which parts of the sample are going to be stored (i.e. we can store just the 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 the trace is disabled. The trigger can be certain watchpoint, breakpoint, load/store status, instruction status, their combination or anything. For this purpose the trigger selection register needs to be set (TSEL register on page 11).
Using watchpoints and breakpoints for trigger can set very complex conditions. The conditions on which certain watchpoint or breakpoint occurs need to be set in the OpenRISC. For this purpose refer to the OpenRISC documentation ("OpenRISC 1000 System Architecture Manual").

4.5.2 Qualifier

Qualifier defines which samples will be stored to the buffer. This is very useful when we don't want to store everything but only selected things (for example just write accesses to a certain address). The qualifier is defined in the QSEL register (page 12).
Using watchpoints and breakpoints for qualifier can set very complex conditions in which samples will be stored. The conditions on which a certain watchpoint or breakpoint occurs needs to be set in the OpenRISC. For this purpose refer to the OpenRISC documentation ("OpenRISC 1000 System Architecture Manual").

4.5.3 Stop recording

There are several occasions in which the trace will stop recording:

- Buffer is full and trace is set to the normal mode (CONTIN bit in the MODER register is not set). In this case the recording is just suspended and will be continued as soon as a sample is read out from the buffer (buffer is not full).
- Trace is disabled (ENABLE bit in the MODER register is set to 0).
- The stop condition occurs (described in the SSET register on page 14).

4.5.4 Sample configuration (selecting the data for recording)

When OpenRISC makes a step, PC, instruction address, load/store date, etc., are changed. The RECWPn and RECBP0 registers (page 16) define, which information is going to compose a sample that will be written to the buffer.
The sample can be watchpoint and breakpoint dependent or independent. This is defined in the MODER register (RECSELDEPEND).

If the bit is cleared, then the selection of the data for recording (sample) is done in the RECWP0 register and is WP and BP independent.

If the bit is set, than the sample selection is changed each time a certain WP or a BP occurs. The twelve record selection registers (RECWPn and RECBP0) are all valid. Each register is related to one breakpoint or watchpoint (i.e. after a certain watchpoint occurs, the content of the sample changes). Thus the bits in the RECWPn and RECBP0 registers define how big a sample is going to be.

4.5.5 Operation modes

Setting the MODER, TSEL, QSEL and SSEL registers define several modes of operation. The few most important are mentioned here others are left to the users imagination:

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

4.5.5.1 Post event recording

Trace starts recording after an event that activates the trigger occurs. Which event starts the trigger needs to be set in the TSEL register. Once the recording starts, samples are stored to the buffer. If we want to store all samples, than qualifier needs to be set to ANY, otherwise it has to be set to what we want to record (i.e. write to an address, read of the SPR, etc.). Once the buffer is full, two things might happen:

- If the CONTIN bit in the MODER register is set to 0, then recording is suspended and OpenRISC stalled. Once we read the stored sample from the buffer (the buffer is not full any more), operation is resumed.
- If the CONTIN bit is set to 1, old samples are overwritten. When we want to stop recording, the trace must be disabled - ENABLE bit (in the MODER) must be set to 0. Then the samples can be read out.

4.5.5.2 Prior event recording

Trace starts recording after the trace is enabled (ENABLE set to 1) and records until a certain event happens. Trigger must be set to ANY in order to start recording immediately. Event for the stop condition needs to be set in the SSEL register. The CONTIN bit must be set to 1, so that old samples are always overwritten. Once the stop event occurs, recording is automatically stopped. Then the samples can be read out.

4.5.5.3 Post - prior event recording

Trace records samples between the two events occur. This is a combination of the previous two modes. Start event sets the trigger while stop event stops the recording. The CONTIN bit must be set to 0 in order we don’t overwrite samples when the buffer is full.

4.5.6 Reading out the recorded samples

The recorded data can be read out through the trace scan chain (Trace scan chain section on page 32). Prior to reading the data, the trace scan chain must be selected.
Each sample written to the buffer is 36-bit width:

- 32-bit data
- 4-bit for the data type identification (what are the 32 bits representing - PC, load/store address, instruction address, etc.)

This 36-bit sample is part of the trace scan chain as seen on the 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 anytime 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 recording to the trace buffer is clocked with the same clock as OpenRISC uses. When a sample needs to be recorded, the trace asserts 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 the normal operation mode (CONTIN bit in the MODER is set to 0). As soon as the buffer is not full (data is read out), the stall signal is de-asserted.

4.6 Forcing OpenRISC to stall

Development Interface stalls the OpenRISC in three cases:

- Trace is enabled and samples are stored to the buffer (stall is activated and deactivated automatically).
- External signal FORCESTALLIN is asserted for at least one MCLK clock cycle. To run the OpenRISC again, deactivate the FORCESTALLIN signal and clear RISCSTALL bit in the RISCOP register.
- Bit RISCSTALL in the RISCOP register is set to 1. Clear this bit to 0 in order to run the RISC again.

The stall status can be read through the same RISCSTALL bit (i.e. bit RISCSTALL reflects the state of the STALLSTATIN signal).

4.7 Forcing OpenRISC to reset

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

4.8 Enabling/Disabling debug mode

Debug mode (and stalling the OpenRISC) is always enabled after the reset. It is up to software to disable it by setting the Disable External Force Watchpoint DXFW bit in the Debug Mode Register DMR (in OpenRISC). Check the “OpenRISC 1000 System

Architecture Manual" for more details. Once it is disabled, the FORCEDBGIN input is ignored.

4.9 WISHBONE Interface (Memory Access)

Writing and reading to/from the memory is achieved through the OpenRISC debug chain. 1Gbyte of the memory space is located at the 2Gbyte offset (0x80000000 to 0xBFFFFFFF). When accesses to any location within that range will occur, memory will be accessed through the WISHBONE (master) interface. (See Table 13: Memory space).

4.10 OpenRISC Debug Interface (SPR)

Writing and reading to/from the debug interface (SPR) is achieved through the OpenRISC debug chain. 64Kbyte of the memory space is located at the offset 0x0 (0x00000000 to 0x0000FFFF). When accesses to any location within that range will occur, the SPRs will be accessed.

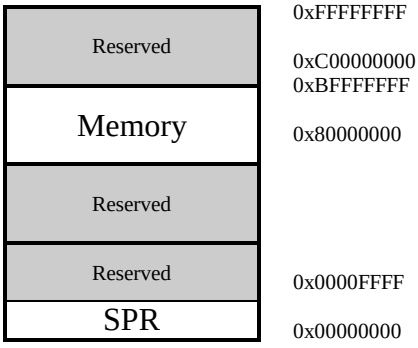

Table 13: Memory space

According to the "OpenRISC 1000 System Architecture Manual", SPR registers address is 16-bit width. Five MSB are reserved for the group ID (GID), others define 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
- Optional scan chains
- Block for monitoring internal signals

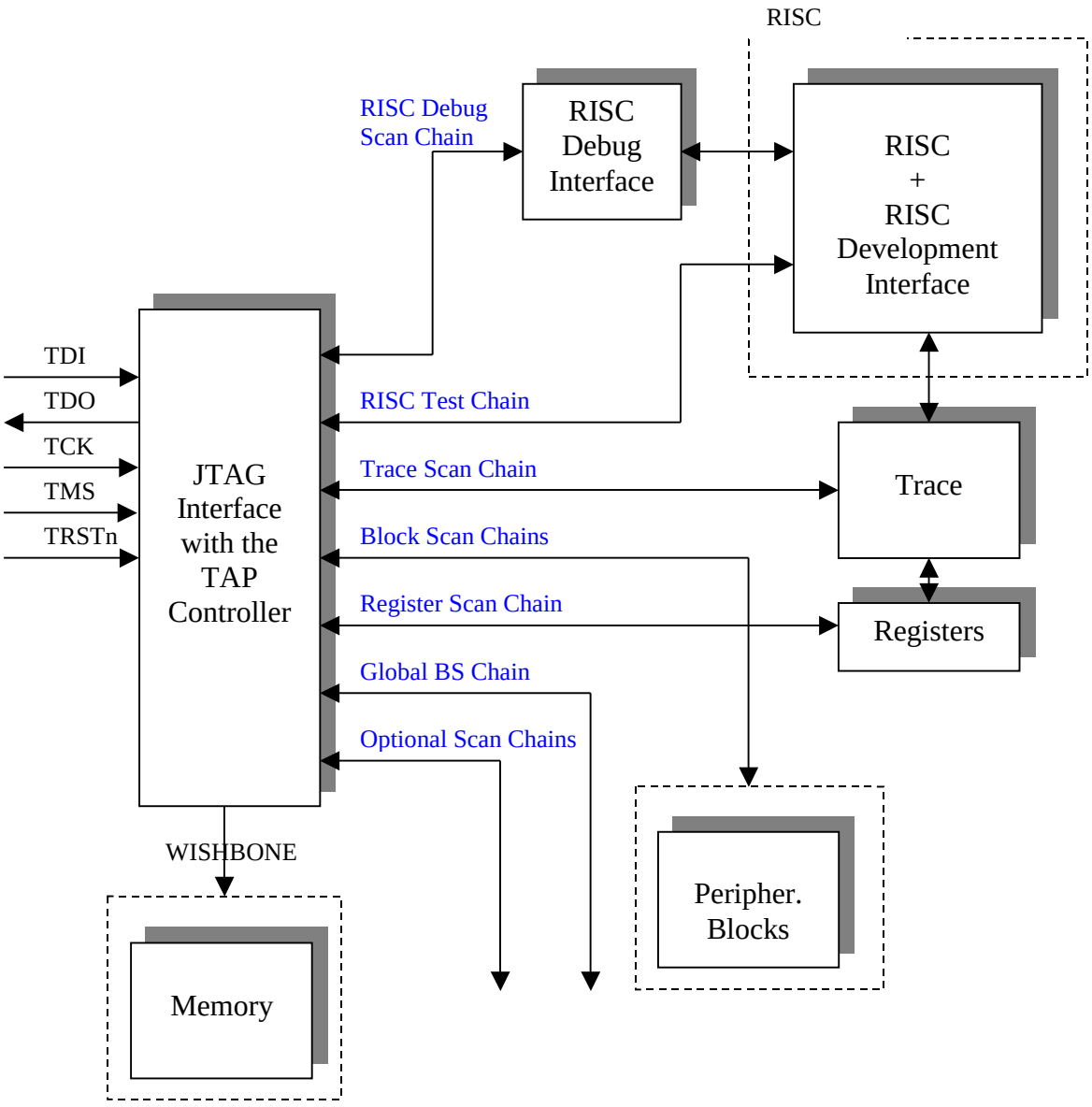

Figure 2: Development Interface

5.1 JTAG interface with the TAP Controller

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

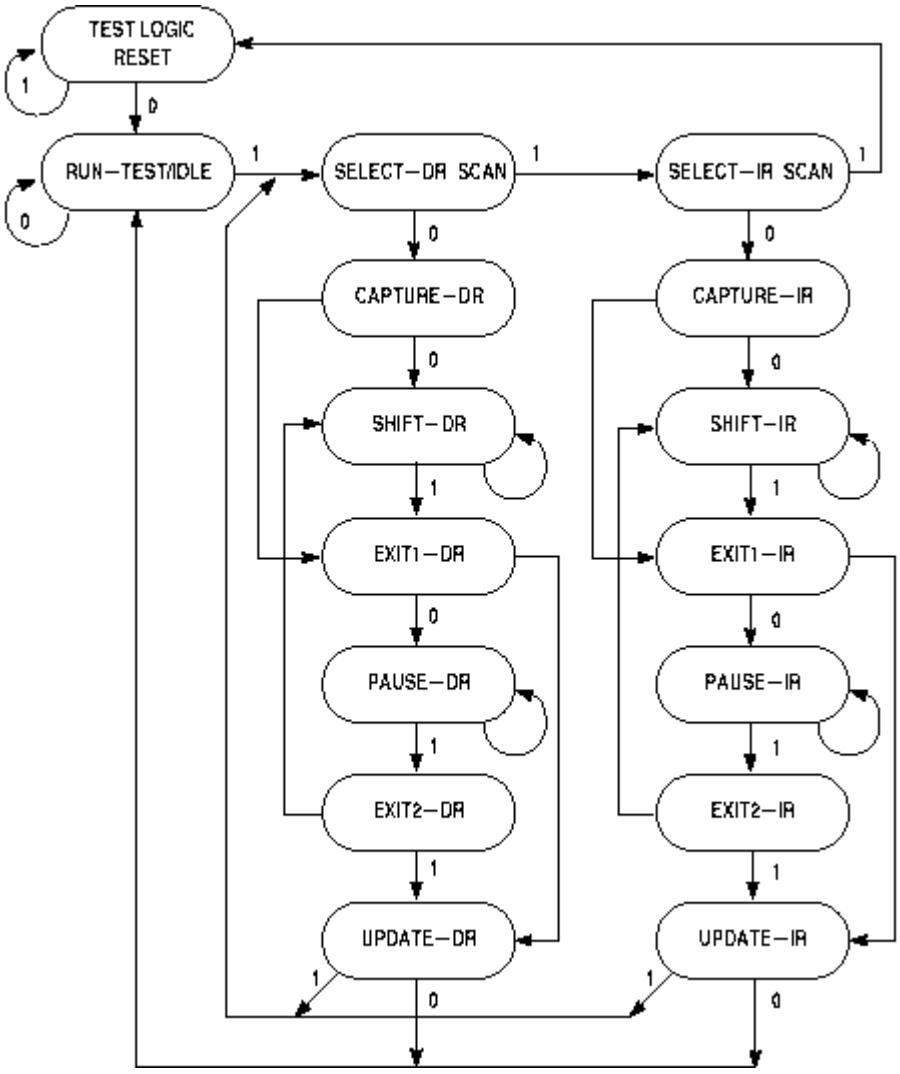

Figure 3: TAP Controller

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

OpenRISC Debug Interface Scan Chain is used for interfacing to the OpenRISC debug support. Using this chain, data can be read/write from/to the registers that are used for

debugging purposes (watchpoint generation, breakpoint generation, memory read/write, SPR, GPR, etc.). The chain is 73-bit long. The chain for shifting is different from the chain for shifting out:

- Chain for shifting in: 32-bit address, R/W bit, 32-bit data and 8-bit CRC.
- Chain for shifting out: 33 bits set to 0x0, 32-bit data and 8-bit CRC.

While shifting in is in progress on first chain, shifting out is in progress on second chain. Read or write operation is performed after all data (address, data, RW and CRC) is shifted in. In case of a read cycle the read data is latched and can be shifted out in the next shifting process. Two CRC codes are also shifted in and out:

- CRC1 (Figure 4: OpenRISC Debug Interface Scan Chain (data shifted in)) is shifted in. Host calculates it from the address, R/W bit and data that are send in.
- CRC2 (Figure 5: OpenRISC Debug Interface Scan Chain (data shifted out)) is shifted out. It is calculated from the address and the R/W that were shifted in and data that is shifted out (data read from register).

After the CRC1 is shifted in, it is compared to the CRC that is internally calculated. If both CRC codes don't match then TDO is set to 0 when the TAP is in the UpdateDR stage. If they do match, then the TDO is set to 1. When the CRC codes match read or write cycle is performed (after the UpdateDR stage).

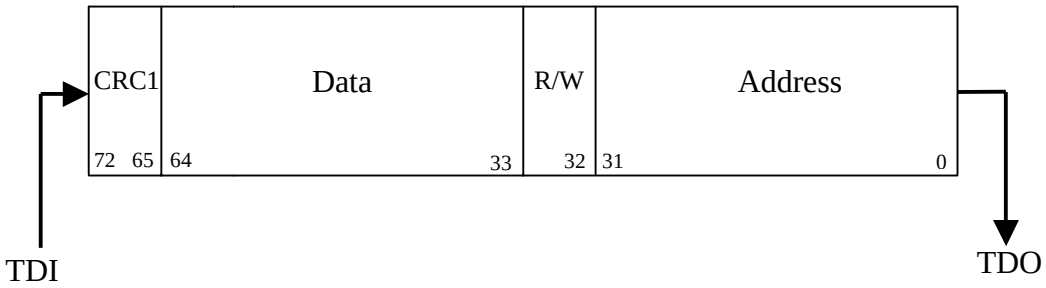

Figure 4: OpenRISC Debug Interface Scan Chain (data shifted in)

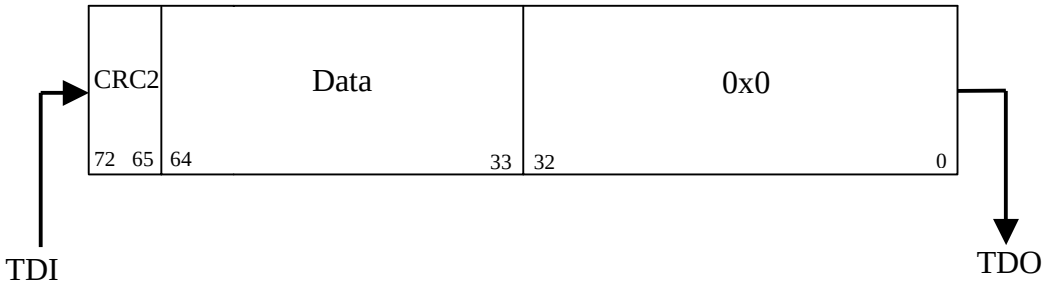

Figure 5: OpenRISC Debug Interface Scan Chain (data shifted out)

5.2.3 OpenRISC Test Chain

When this chain is selected, the OpenRISC functionality can be tested by issuing the INTEST commands. The test needs to be performed in the factory, so it system integrator's responsibility to define the appropriate test chain.

5.2.4 Trace Scan Chain

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

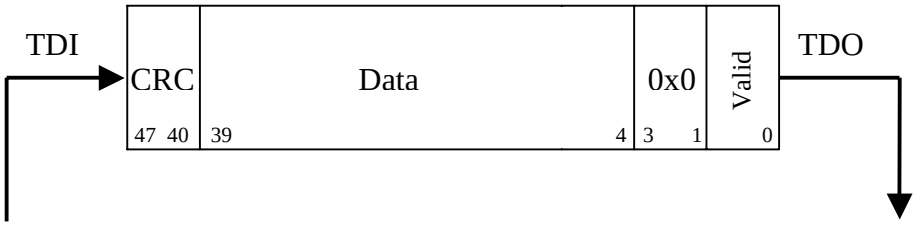

Figure 6: Trace Scan Chain

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 46-bit long. The chain for shifting is different from the chain for shifting out:

- Chain for shifting in: 5-bit address, R/W bit, 32-bit data and 8-bit CRC.
- Chain for shifting out: 6 bits set to 0x0, 32-bit data and 8-bit CRC.

Read or write operation is performed after all data (address, data, RW and CRC) is shifted in. In case of a read cycle the read data is latched and can be shifted out in the next shifting process. Two CRC codes are shifted in and out:

- CRC1 (Figure 7: Register Scan Chain (data shifted in)) is shifted in. Host calculates it from the address, R/W bit and data that are send in.
- CRC2 (Figure 8: Register Scan Chain (data shifted out)) is shifted out. It is calculated from the address and the R/W that were shifted in and data that is shifted out (data read from register).

After the CRC1 is shifted in, it is compared to the CRC that is internally calculated. If both CRC codes don't match then TDO is set to 0 when the TAP is in the UpdateDR stage. If they do match, then the TDO is set to 1. When the CRC codes matche read or write cycle is performed (after the UpdateDR stage).

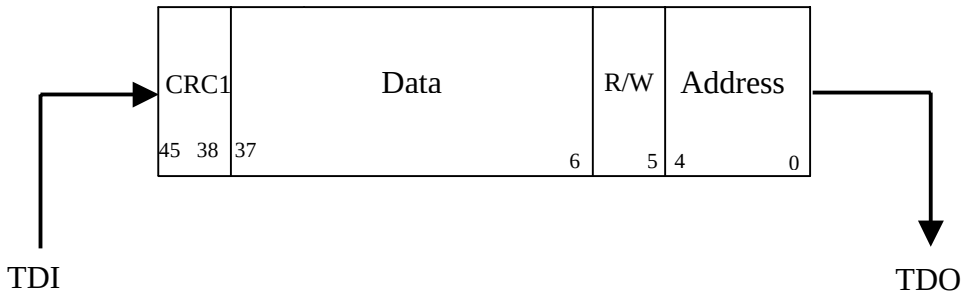

Figure 7: Register Scan Chain (data shifted in)

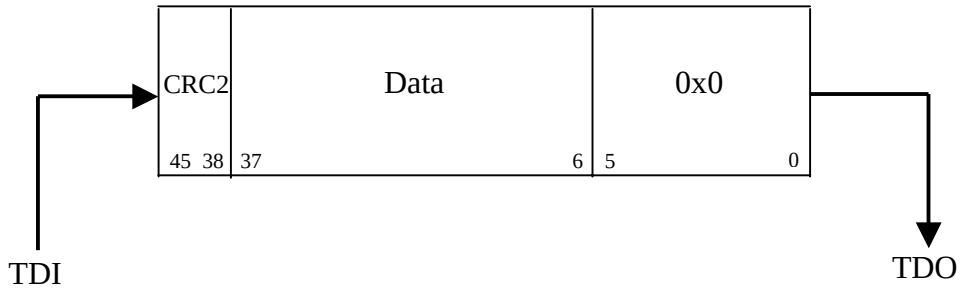

Figure 8: Register Scan Chain (data shifted out)

5.2.6 Block Scan Chains

Several block scan chains can be used in 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. The cores activity can only be observed and not controlled by this scan chain. The length of the scan chain depends upon the number of the signals used as a WISHBONE interface. It is up to the system integrator to connect this scan chains to those cores that need observation.

5.2.7 Optional Scan Chains

Optional scan chains can be used for both observing and controlling. So far they are reserved for the future demands.

5.3 OpenRISC Debug Interface

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

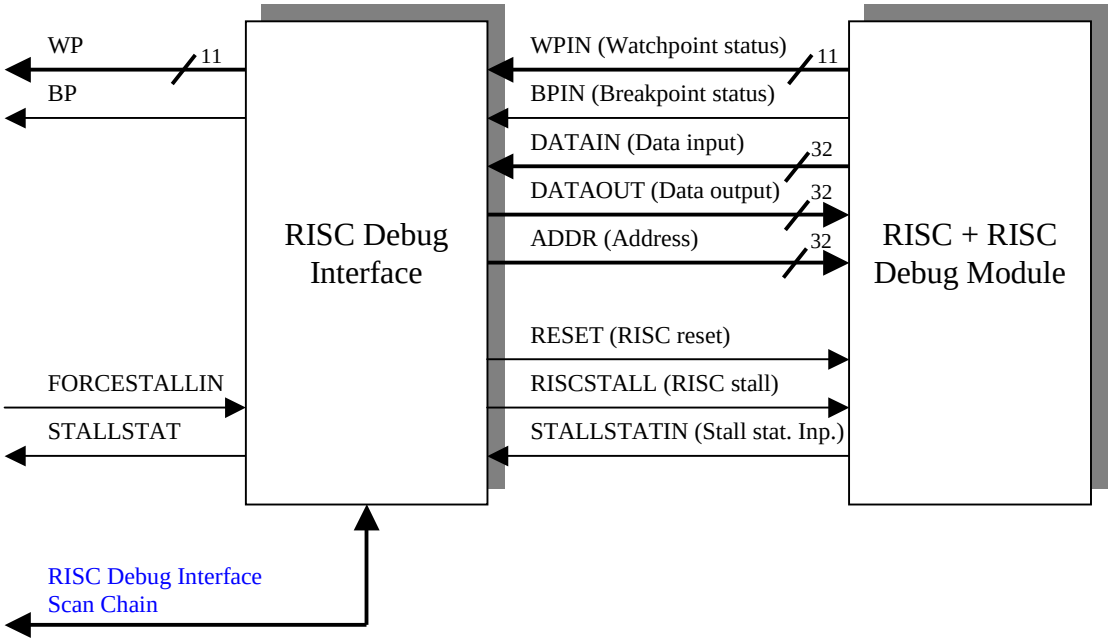

Figure 9: OpenRISC Debug Interface

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.

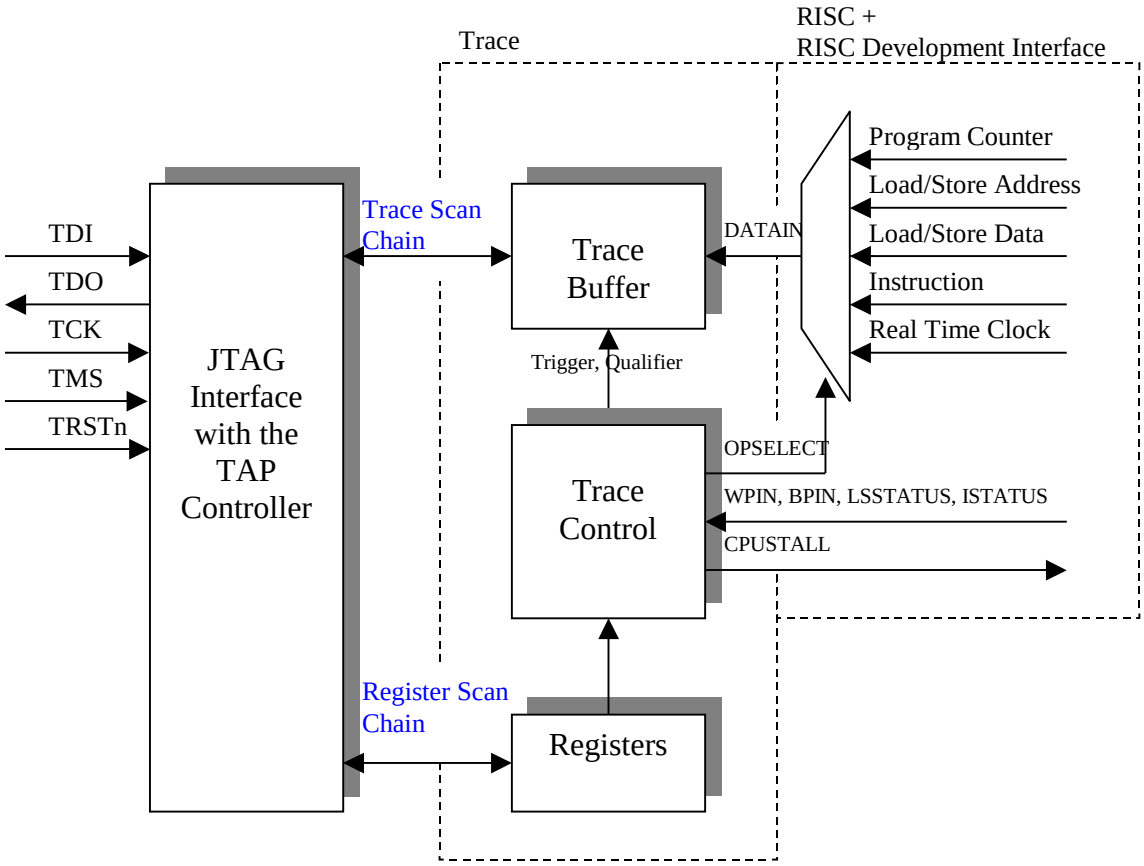

Figure 10: Trace

5.4 Observing internal signals

The state of several internal signals can be monitored through the DIROUT[31:0] signals. The monitoring is development port independent. Which set of signals is connected through the multiplexer to the output pins is selected with the DIRSEL[2:0] signals. 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.

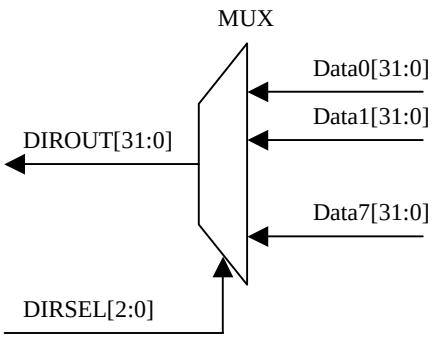

Figure 11: Selection of the observed signals

Appendix A

Configuring trace

Let's say that we want to record all store data operations once the program enters the subroutine X.

There are many solutions how to achieve that. Here is just one example:

First the trace scan chain must be selected:

- Instruction SELECT_CHAIN is shifted in to the TAP controller (see Table 11: TAP Instruction Set on page 19)
- Trace scan chain ID must be shifted in to the TAP controller (see Table 12: Chains Identification on page 22)

Development port must be put to DEBUG mode:

- Instruction DEBUG is shifted in to the TAP controller (see Table 11: TAP Instruction Set on page 19)

Trigger must be set:

- Set the watchpoint 0 to be asserted when the program executes the jump to the subroutine X. This needs to be done in the OpenRISC. Refer to the OpenRISC documentation ("OpenRISC 1000 System Architecture Manual") for more information on that.
- Set value 0xC0000801 to the TSEL register. By doing so you instructed the trace to start recording when the WP0 occurs.

Qualifier must be set:

- Set value 0xC01F0000 to the QSEL register. By doing so you instructed the trace to record only when a store data operations occurs.

Record selection must be set:

- Set value 0x00000008 to the RECWP0. This means that the samples will only consist of the stored data.

Set the trace mode and enable it:

- Set value 0x00000002 to the MODER register. This sets the trace to the normal mode (old samples will never be overwritten), sets the sample configuration to be watchpoint and breakpoint independent (in case that another WP or BP occurs, sample won't be changed) and finally enables the trace.

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