MIPS® iFlowtrace™ Architecture Specification

Document Number: MD00526
Revision 2.00
March 20, 2010

MIPS Technologies, Inc.
955 East Arques Avenue
Sunnyvale, CA 94085-4521

Template: nB1.03, Built with tags: 2B

Table of Contents

Chapter 1: Introduction to the iFlowtrace™ Architecture and Methodology 7
1.1: Overview of the iFlowtrace™ Method 7
1.2: Architectural Indicator for the Presence of the iFlowtrace™ Method 7
1.3: A Block-Level Overview of the iFlowtrace™ Method 7
1.4: Revisions of the iFlowtrace™ Method 8

Chapter 2: iFlowtrace™ Interface Signals 9
2.1: Trace Inputs 9
2.1.1: Implementation Notes 9
2.2: Normal Trace Mode Outputs 9
2.3: Special Trace Modes 10
2.3.1: Mode Descriptions 11
2.3.2: Special Trace Mode Outputs 12

Chapter 3: iFlowtrace™ Control Block (ITCB) 15
3.1: ITCB Storage Representation 15
3.2: ITCB Register Interface for Software Configurability 16
3.2.1: IFlowTrace Control/Status (IFCTL) Register (offset 0x3fc0) 16
3.2.2: ITCBTW Register (offset 0x3F80) 18
3.2.3: ITCBRDP Register (Offset 0x3f88) 18
3.2.4: ITCBWRP Register (Offset 0x3f90) 18
3.3: ITCB iFlowtrace Off-Chip Interface 19
3.4: iFlowTrace related COP0 Registers 20
3.4.1: The *UserTraceData1* and *UserTraceData2* Registers (CP0 Register 23 Select 3 and CP0 Register 24 Select 3) 20

Chapter 4: Breakpoint-Based Triggering of Tracing 21

Appendix A: References 23

Appendix B: Revision History 25

List of Figures

Figure 1.1: An Overview of iFlowtrace, ITCB, and Other Core Blocks 8
Figure 3.1: Control/Status Register 16
Figure 3.2: *ITCBTW* Register Format 18
Figure 3.3: *ITCBRDP* Register Format 18
Figure 3.4: *ITCBWRP* Register Format 19
Figure 3.5: *UserTraceData1* and *UserTraceData2* Register Format 20

List of Tables

Table 3.1: Tag Bit Encoding 15
Table 3.2: Control/Status Register Field Descriptions 16
Table 3.3: *ITCBTW* Register Field Descriptions 18
Table 3.4: *ITCBRDP* Register Field Descriptions 18
Table 3.5: *ITCBWRP* Register Field Descriptions 19
Table 3.6: *UserTraceData1* Register Field Descriptions 20
Table 3.7: *UserTraceData2* Register Field Descriptions 20
Table 4.1: drseg Registers that Enable/Disable Trace from Breakpoint-Based Triggers 21

Chapter 1

Introduction to the iFlowtrace™ Architecture and Methodology

The goal of this trace specification is to describe the information needed to reconstruct a simple instruction trace from an execution stream. Such a simple mechanism enables a small, efficient tracing methodology for any core. The PDtrace™ scheme that is currently supported by MIPS® core families can sometimes be too heavy weight for certain applications such as low-end microcontrollers or when it is desired to have a simpler tracing scheme in production chips.

1.1 Overview of the iFlowtrace™ Method

The light-weight instruction-only tracing scheme proposed here is sufficient to reconstruct the execution flow in a core under conditions as stated in this document. This tracing scheme has been kept very simple to minimize the impact on die size.

The iFlowtrace tracing scheme is not a strict subset of the PDtrace tracing methodology, and its trace format outputs differ from those of PDtrace. New trace formats, using simplified instruction state descriptors, were designed for the iFlowtrace trace to simplify the trace mechanism and to obtain better compression.

Tracing is disabled if the processor enters DebugMode (refer to the EJTAG specification for description of Debug-Mode). This is true for both Normal Trace Mode as well as Special Trace Mode (the modes are described in the next chapter).

1.2 Architectural Indicator for the Presence of the iFlowtrace™ Method

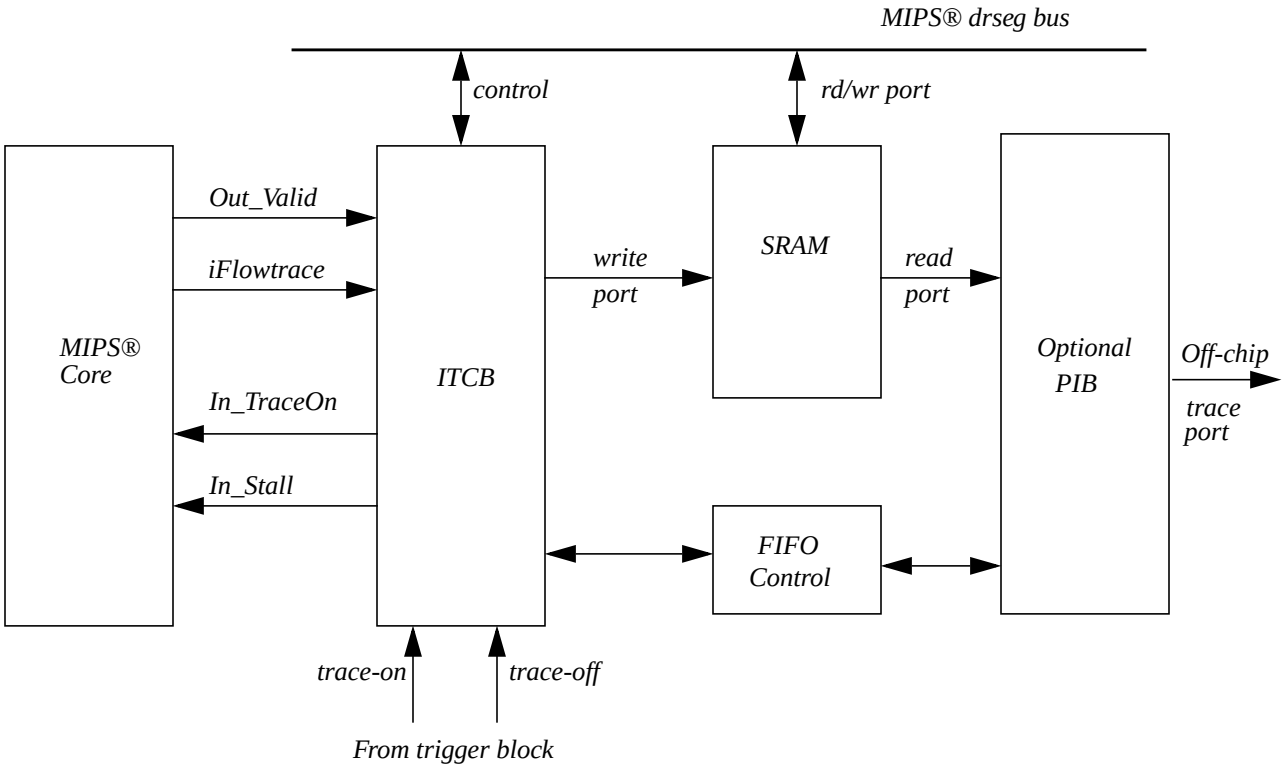
The presence of the iFlowtrace scheme in a core is indicated by a configuration bit, bit 8 (*ITL*) of *Config3* CP0 register (see the MIPS32 Architecture for Programmers, Volume III: The MIPS32 and microMIPS Privileged Resource Architecture, MD00090).

1.3 A Block-Level Overview of the iFlowtrace™ Method

A trace methodology can often be defined mainly by its inputs and outputs. Hence this scheme is described by the inputs to the core tracing logic and by the trace output format from the core. We assume here that the execution flow of the program is traced at the end of the execution path in the core similar to the PDtrace scheme. Figure 1.1 shows a block diagram of the flow of the trace from the tracing logic near the core to the iFlowtrace Control Block (ITCB).

The ITCB is responsible for accepting trace signals from the CPU core, formatting them, and storing them in an on-chip memory organized as a circular buffer. The figure shows the Probe Interface Block (PIB) as capable of emptying the SRAM/circular buffer and outputting the memory contents through a narrow off-chip trace port.

Figure 1.1 An Overview of iFlowtrace, ITCB, and Other Core Blocks



1.4 Revisions of the iFlowtrace™ Method

Revision 2.0 introduces the Special Tracing Modes. Please refer to Section 2.3 “Special Trace Modes”.

Chapter 2

iFlowtrace™ Interface Signals

This chapter describes iFlowtrace input and output signals and the iFlowtrace interface to the iFlowTrace Control Block (ITCB). Starting with iFlowtrace revision 2.0, special tracing modes are defined which can be used to trace information about certain events. These modes are enabled only when normal tracing is disabled.

2.1 Trace Inputs

1. *In_TraceOn*: When on, legal trace words are coming from the core and at the point when it is turned on, that is for the first traced instruction, a full PC value is output. When off, it cannot be assumed that legal trace words are available at the core interface.

2. *In_Stall*: This indicates that the processor should be stalled in order to avoid a buffer overflow that can lose trace information. When off, a buffer overflow will simply throw away trace data and start over again. When on, the processor is signalled from the tracing logic to stall until the buffer is sufficiently drained and then restart the pipeline.

2.1.1 Implementation Notes

Depending on the core pipeline and the ease with which the pipe can be stalled, the trace control block needs to know the latency between the assertion of the *In_Stall* signal and the maximum number of cycles before the pipe can be halted. This information is then used to determine how many empty trace FIFO entries are needed to store potential trace information after the stall is asserted and the *Out_Valid* signal will be de-asserted.

For a given core implementation, the maximum pipeline stall latency is known and this will be used by the ITCB (iFlowtrace Control Block, shown in Figure 1.1) for its worst case calculations on FIFO space requirements. Note that if tracing is turned on, stalls are enabled to ensure no lost trace data, and the code being run has a particularly large number of unpredictable jumps, then for a given FIFO size, it is possible to make the core stall quite often, and this will affect the use of the core and the performance that one would see on the core when running under these conditions. If it is anticipated that tracing will be enabled and stalls will be enabled for full traces as the default configuration, then it is essential to take typical code that will run under these situations and characterize the number of bits of trace that will be needed for say 100 instructions and correlate that back to both the size of the FIFO in the ITCB as well the expected rate at which this FIFO will be cleared, in order to prevent an excessive amount of stalling.

2.2 Normal Trace Mode Outputs

1. Stall cycles in the pipe are ignored by the tracing logic and are not traced. This is indicated by the signal *Out_Valid* that is turned off when no valid instruction is being traced. When *Out_Valid* is asserted, instructions are traced out as described in the rest of this section. The traced instruction PC is a virtual address.

2. In the output format, every sequentially executed instruction is traced as 1’b0.

3. Every instruction that is not sequential to the previous one is traced as either a 10 or an 11 (read this as a serial bitstream from left to right). This implies that the target instruction of a branch or jump is traced this way, not the actual branch or jump instruction (this is similar to PDtrace):

4. A 10 instruction implies a taken branch for a conditional branch instruction whose condition is unpredictable statically, but whose branch target can be computed statically and hence the new PC does not need to be traced out. Note that if this branch was not taken, it would have been indicated by a 0 bit, that is sequential flow.

5. A 11 instruction implies a taken branch for an indirect jump-like instruction whose branch target could not be computed statically and hence the taken branch address is now given in the trace. This includes, for example, instructions like jr, jalr, and interrupts:

    - 11 00 - followed by 8 bits of 1-bit shifted offset from the last PC. The bit assignments of this format on the bus between the core tracing logic and the ITCB is:

      [3:0] = 4’b0011
      [11:4] = PCdelta[8:1]

    - 11 01 - followed by 16 bits of 1-bit shifted offset from the last PC. The bit assignments of this format on the bus between the core tracing logic and the ITCB is:

      [3:0] = 4’b1011
      [19:4] = PCdelta[16:1]

    - 11 10 - followed by 31 of the most significant bits of the PC value, followed by a bit (NCC) that indicates no code compression. Note that for a MIPS32 or MIPS64 instruction, NCC=1, and for MIPS16e instruction NCC=0. This trace record will appear at all transition points between MIPS32/MIPS64 and MIPS16e instruction execution.
      This form is also a special case of the 11 format and it is used when the instruction is not a branch or jump, but nevertheless the full PC value needs to be reconstructed. This is used for synchronization purposes, similar to the Sync in PDtrace. In iFlowtrace rev 2.0 onwards, the sync period is user-defined, and is counted down and when an internal counter runs through all the values, this format is used. The bit assignments of this format on the bus between the core tracing logic and the ITCB is:

      [3:0] = 4’b0111
      [34:4] = PC[31:1]
      [35] = NCC

    - 11 11 - Used to indicate trace resumption after a discontinuity occurred. The next format is a 1110 that sends a full PC value. A discontinuity might happen due to various reasons, for example, an internal buffer overflow, and at trace-on/trace-off trigger action.

2.3 Special Trace Modes

iFlowtrace 2.0 adds special trace modes which can only be active when the normal tracing mode is disabled. An implementation can choose to implement any subset of these special trace modes. Software can determine which modes are supported by attempting to write the enable bits in the IFCTL register - unsupported options will not be settable and will read as 0. Additionally, implementations can choose which special trace modes can be simultaneously active. Software can check the Illegal bit in the IFCTL register - if an unsupported combination of modes is

requested, the bit will be set and the trace contents will be unpredictable. The special trace modes are described below.

2.3.1 Mode Descriptions

2.3.1.1 Delta Cycle Mode

This mode is specified in combination with the other special trace modes. It is enabled via the CYC bit in the Control/Status Register. When delta cycle reporting is enabled, each trace message will include a 10b delta cycle value which reports the number of cycles that have elapsed since the last message was generated. A value of 0 indicates that the two messages were generated in the same cycle. A value of 1 indicates that they were generated in consecutive cycles. If 1023 cycles elapse without an event being traced, a counter rollover message is generated.

Note: If the processor clocks stop due to execution of the WAIT instruction, the delta cycle counter will also stop and will report ‘active’ cycles between events rather than ‘total’ cycles.

2.3.1.2 Breakpoint Match Mode

This modes uses EJTAG data and instruction breakpoint hardware to enable a trace of PC values. Instead of starting or stopping trace, a triggerpoint will cause a single breakpoint match trace record. This record indicates that there was a triggerpoint match, the breakpoint ID of the matching breakpoint, and the PC value of an instruction that matched the instruction of data breakpoint.This mode can only be used when normal tracing mode is turned off. It is implementation dependent whether this mode can be used in conjunction with other special trace modes. This mode is enabled or disabled via the BM field in the Control/Status register (see Section 3.2 “ITCB Register Interface for Software Configurability”).

The breakpoints used in this mode must have the TE bit set (within the appropriate IBCn/DBCn register) to enable the match condition. If the BE bit is also set (within the appropriate IBCn/DBCn register), the behavior is **UNDEFINED**.

Software should avoid setting up overlapping breakpoints. The behavior when multiple matches occur on the same instruction is implementation dependent. The recommended behavior is to report a BreakpointID of 15.

2.3.1.3 Filtered Data Tracing Mode

This mode uses EJTAG data breakpoint hardware to enable a trace of data values. Rather than starting or stopping trace as in normal trace mode, a data triggerpoint will cause a filtered data trace record. This record indicates that there was a data triggerpoint match, the breakpoint ID of the matching breakpoint, whether it was a load or store, the size of the request, low order address bits, and the data value. This mode can only be used when normal tracing mode is turned off. It is implementation dependent whether this mode can be used in conjunction with other special trace modes. This mode can be enabled or disabled via the FDT bit in the Control/Status register (see Section 3.2 “ITCB Register Interface for Software Configurability”).

The corresponding data breakpoint must have the TE bit set (within the appropriate DBCn register) to enable the match condition. If the BE bit is also set (within the appropriate DBCn register), the behavior is **UNDEFINED**.

Software should avoid setting up overlapping data breakpoints. The behavior when multiple matches on one load or store are detected is implementation dependent. The recommended behavior is to report a BreakpointID of 15.

EJTAG Tuple breakpoints (where both an instruction and data trigger conditions are met), are treated as a single data triggerpoint.

2.3.1.4 Function Call/Return and Exception Tracing Mode

In this mode, the PC value of function calls and returns and/or exceptions and returns are traced out. This mode can only be used when normal tracing mode is turned off. It is implementation dependent whether this mode can be used in conjunction with other special trace modes. The function call/return and exception/return are independently enabled or disabled via the FCR and ER bits in the Control//Status register (see Section 3.2 "ITCB Register Interface for Software Configurability").

These events are reported for the following instructions:

- MIPS32 and MIPS64 Function calls: JAL, JALR, JALR.HB, JALX
- MIPS16e Function calls: JAL, JALR, JALX, JALRC
- microMIPS Function calls: JAL, JALR, JALR.HB, JALX, JALR16, JALRS16, JALRS, JALRS.HB, JALS
- MIPS32 and MIPS64 Function Returns: JR, JR.HB
- MIPS16e Function Returns: JR, JRC
- microMIPS Function Returns: JR, JR.HB, JRC, JRADDIUSP, JR16
- Exceptions: Reported on the first instruction of the exception handler
- Exception returns: ERET
- MCU ASE Interrupt returns: IRET

2.3.1.5 Other Trace Messages

In any of the special trace modes, it is possible to embed messages into the trace stream directly from a program. This is done by writing to the UserTraceData1 or UseTraceData2 Cop0 registers. When UserTraceData1 register is written, a trace message of type "User Triggered Message 1" (UTM1) is generated. When UserTraceData2 register is written, a trace message of type "User Triggered Message 2" (UTM2) is generated. Please refer to Section 3.4.1 "The UserTraceData1 and UserTraceData2 Registers (CP0 Register 23 Select 3 and CP0 Register 24 Select 3)".

Additionally, overflow messages can be generated when tracing offchip if the IO control bit is 0 and trace data is generated faster than it is consumed. No overflow will be generated when using onchip trace.

2.3.2 Special Trace Mode Outputs

The normal and special trace modes cannot be enabled at the same time because the trace message encoding is not unique between the two modes. The software reading the trace stream must be aware of which mode is selected to know how to interpret the bits in the trace stream. Some implementations may choose to support multiple special trace modes simultaneously so there are unique message types for each type of special trace message.

- 00 (as above, read a bitstream from left to right) - Delta Cycle Rollover message. The output format is:
  [1:0] = 2'b00
- 010 - User Trace Message. The format of this type of message is:
  [2:0] = 3'b010
  [34:3] = Data[31:0]

[35] = UTM2/UTM1 (1=UTM2, 0=UTM1)
[45:36] = DeltaCycle (if enabled)

- 011 - Reserved

- 10 - Breakpoint Match Message. The output format during this trace mode is:
  [1:0] = 2'b01
  [5:2] = BreakpointID
  [6] = Instruction Breakpooint
  [37:7] = MatchingPC[31:1]
  [38] = NCC
  [48:39] = DeltaCycle (if enabled)
  Note that for a MIPS32 or MIPS64 instruction, NCC=1, and for MIPS16e instruction NCC=0.

- 110 - Filtered Data Message. The output format during this trace mode is:
  [2:0] = 3'b011
  [6:3] = BreakpointID
  [7] = Load/Store (1=Load, 0=Store)
  [8] = FullWord (1=32b data, 0= <32b)
  [14:5] = Addr[7:2]
  [46:15] = {32b data value} OR
  [46:15] = {BE[3:0], 4'b0, 24b data value} OR
  [46:15] = {BE[3:0], 12'b0, 16b data value} OR
  [46:15] = {BE[3:0],20'b0, 8b data value}
  [56:47] = DeltaCycle (if enabled)

- 1110 - Function Call/Return/Exception Tracing. The output format during this trace mode is:
  [3:0] = 4'b0111
  [4] = FC
  [5] = Ex
  [6] = R
  [37:8] = PC[31:1]
  [38] = NCC
  [48:39] = Delta Cycle (if enabled)
  Note that for a MIPS32 or MIPS64 instruction, NCC=1, and for MIPS16e instruction NCC=0. FC=1 implies a function call, Ex=1 implies the start of an exception handler, and R=1 implies a function or exception return.

- 1111- Used to indicate trace resumption after a discontinuity occurred. A discontinuity might happen due to various reasons, for example, an internal buffer overflow, and at trace-on/trace-off trigger action.

Chapter 3

iFlowtrace™ Control Block (ITCB)

The ITCB is responsible for accepting the data stream from the iFlowtrace output and sending it to the memory or appropriate core output interface.

3.1 ITCB Storage Representation

Records from iFlowtrace are inserted into a memory stream exactly as they appear in the iFlowtrace data output. Records are concatenated into a continuous stream starting at the LSB. When a trace word is filled, it is written to memory along with some tag bits. Each record consists of a 64-bit word, which comprises 58 message bits and 6 tag bits or header bits that clarify information about the message in that word.

The ITCB includes a 58-bit shift register to accumulate trace messages. Once 58 or more bits are accumulated, the 58 bits and 6 tag bits are sent to the memory write interface. Messages may span a trace word boundary. In this case, the 6 tag bits indicate the bit number of the first full trace message in the 58-bit data field.

The tag bits are slightly encoded so they can serve a secondary purpose of indicating to off-chip trace hardware when a valid trace word transmission begins. The encoding ensures that at least one of the 4 LSBs of the tag is always a 1 for a valid trace message. The tag values are shown in Table 3.1. The longest trace message is 57 bits (filtered data trace in special trace mode with delta cycle), so the starting position indicated by the tag bits is always between 0 and 56.

Table 3.1 Tag Bit Encoding

Starting Bit of First Full Trace Message	Encoding (decimal)
0	58
16	59
32	60
48	61
Unused	0,16,32,48
Reserved	62,63
Others	StartingBit

When trace stops (ON set to zero), any partially filled trace words are written to memory. Any unused space above the final message is filled with 1's. The decoder distinguishes 1111 patterns used for fill in this position from an 1111 overflow message by recognizing that it is the last trace word.

These trace formats are written to a trace memory that is either on-chip or off-chip. No particular size of SRAM is specified; the size is user selectable based on the application needs and area trade-offs. Each trace word can typically store about 20 to 30 instructions in normal trace mode, so a 1 KWord trace memory could store the history of 20K to 30K executed instructions.

The on-chip SRAM or trace memory is written continuously as a circular buffer. It is accessible via drseg address mapped registers. There are registers for the read pointer, write pointer, and trace word. The write pointer register includes a wrap bit that indicates that the pointer has wrapped since the last time the register was written. Before starting trace, the write pointer would typically be set to 0. To read the trace memory, the read pointer should be set to 0 if there has not been a wrap, or to the value of the write pointer if there has been. Reading the trace word register will read the entry pointed to by the read pointer and will automatically increment the read pointer. Software can continue reading until all valid entries have been read out.

3.2 ITCB Register Interface for Software Configurability

The ITCB includes a drseg memory interface to allow software to set up tracing and read the current status. If an on-chip trace buffer is also implemented, there are additional registers included for accessing it.

3.2.1 IFlowTrace Control/Status (IFCTL) Register (offset 0x3fc0)

The Control/Status register provides the mechanism for turning on the different trace modes. Figure 3.1 has the format of the register and Table 3.2 describes the register fields.

Figure 3.1 Control/Status Register

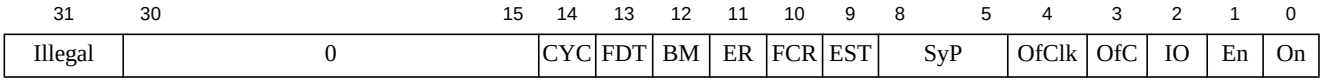


Table 3.2 Control/Status Register Field Descriptions

Fields		Description	Read / Write	Reset State	Compliance
Name	Bits				
0	30:19	Reserved for future use. Read as zeros, must be written as zeros	R	0	Required
Illegal	31	This bit is set by hardware and indicates if the currently enabled trace output modes are an illegal combination. Since an implementation can choose to allow or disallow the use of a combination of the special tracing modes, this bit provides an indication whether the currently chosen set of tracing modes is supported by hardware. A value of 1 indicates an unsupported setting. A value of 0 indicates that the currently selected settings are legal.	R	0	Required
CYC	14	Delta Cycle Mode: This mode can be set in combination with the EST special trace modes. When set, a delta cycle value is included in each of the trace messages and indicates the number of cycles since the last message was generated. If this tracing mode is not implemented, the field is read-only and read as zero.	R/W	0	Optional for iFlowTrace rev 2.0+
FDT	13	Filtered Data Trace mode. If set, on a data breakpoint match, the data value of the matching breakpoint is traced. Normal tracing is inhibited when this mode is active. If this tracing mode is not implemented, the field is read-only and read as zero.	R/W	0	Optional for iFlowTrace rev 2.0+

Table 3.2 Control/Status Register Field Descriptions (Continued)

Fields		Description	Read / Write	Reset State	Compliance
Name	Bits				
BM	12	Breakpoint Match. If set, only instructions that match instruction or data breakpoints are traced. Normal tracing is inhibited when this mode is active. If this tracing mode is not implemented, the field is read-only and read as zero.	R/W	0	Optional for iFlowTrace rev 2.0+
ER	11	Trace exceptions and exception returns. If set, trace includes markers for exceptions and exception returns. Can be used in conjunction with the FCR bit. Inhibits normal tracing. If this tracing mode is not implemented, the field is read-only and read as zero..	R/W	0	Optional for iFlowTrace rev 2.0+
FCR	10	Trace Function Calls and Returns. If set, trace includes markers for function calls and returns. Can be used in conjunction with the ER bit. If this tracing mode is not implemented, the field is read-only and read as zero..	R/W	0	Optional for iFlowTrace rev 2.0+
EST	9	Enable Special Tracing Modes. If set, normal tracing is inhibited, allowing the user to choose one of several special tracing modes. Setting this bit inhibits normal trace mode. An implementation may implement a subset of the special tracing modes and may disallow some or all combinations of multiple modes. If no special tracing modes are implemented, this field is read-only, and read as zero.	R/W	0	Optional for iFlowTrace rev 2.0+
SyP	8:5	Synchronization Period. The synchronization period is set to $2^{(SyP+8)}$ instructions. Thus a value of 0x0 implies 256 instructions, and a value of 0xF implies 8M instructions.	R/W	0	Required for iFlowTrace rev 2.0+
OfClk	4	Controls the Off-chip clock ratio. When the bit is set, this implies 1:2, that is, the trace clock is running at 1/2 the core clock, and when the bit is clear, implies 1:4 ratio, that is, the trace clock is at 1/4 the core clock. Ignored unless OfC is also set.	R/W	0	Required
OfC	3	Offchip. 1 enables the PIB (if present) to unload the trace memory. 0 disables the PIB and would be used when on-chip storage is desired or if a PIB is not present. This bit is settable only if the design supports both on-chip and off-chip modes. Otherwise is a read-only bit indicating which mode is supported.	R/W or R	Preset	Required
IO	2	Inhibit overflow. If set, the CPU is stalled whenever the trace memory is full. Ignored unless OfC is also set.	R/W	0	Required
En	1	Trace enable. This bit may be set by software or by Trace-on/Trace-off action bits from the Complex Trigger block. Software writes EN with the desired initial state of tracing when the ITCB is first turned on and EN is controlled by hardware thereafter. EN turning on and off does not flush partly filled trace words.	R/W	0	Required
On	0	Software control of trace collection. 0 disables all collection and flushes out any partially filled trace words.	R/W	0	Required

3.2.2 ITCBTW Register (offset 0x3F80)

The *ITCBTW* register is used to read Trace Words from the on-chip trace memory. The TW read is the TW pointed to by the *ITCBRDP* register. A side effect of reading the *ITCBTW* register is that the *ITCBRDP* register increments to the next TW in the on-chip trace memory. If *ITCBRDP* is at the max size of the on-chip trace memory, the increment wraps back to address zero.

Note that this is a 64b register.On a 32b processor, software must read the upper word (offset 0x3F84) first as the address increment takes place on a read of the lower word (0x3F80).

The format of the *ITCBTW* register is shown below, and the field is described in Table 3.3.

Figure 3.2 *ITCBTW* Register Format

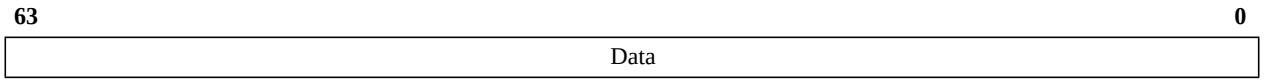


Table 3.3 *ITCBTW* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Data	63:0	Trace Word	R	Undefined	Required

3.2.3 ITCBRDP Register (Offset 0x3f88)

The *ITCBRDP* register is the address pointer to on-chip trace memory. It points to the TW read when reading the *ITCBTW* register. This value will be automatically incremented after a read of the *ITCBTW* register.

The format of the *ITCBRDP* register is shown below, and the field is described in Table 3.3. The value of n depends on the size of the on-chip trace memory. As the address points to a 64-bit TW, lower three bits are always zero.

Figure 3.3 *ITCBRDP* Register Format

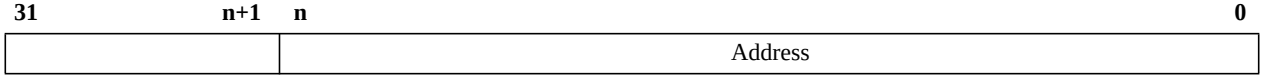


Table 3.4 *ITCBRDP* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Data	31:(n+1)	Reserved. Must be written zero, reads back zero.	0	0	Required
Address	n:0	Byte address of on-chip trace memory word.	R/W	0	Required

3.2.4 ITCBWRP Register (Offset 0x3f90)

The *ITCBWRP* register is the address pointer to on-chip trace memory. It points to the location where the next new TW for on-chip trace will be written. The top bit in the register indicates whether the pointer has wrapped. If it has,

then the write pointer will also point to the oldest trace word. and the read pointer can be set to that to read the entire array in order. If it is cleared, then the read pointer can be set to 0 to read up to the write pointer position.

The format of the *ITCBWRP* register is shown below, and the field is described in Table 3.3. The value of n depends on the size of the on-chip trace memory. As the address points to a 64-bit TW, lower three bits are always zero.

Figure 3.4 *ITCBWRP* Register Format

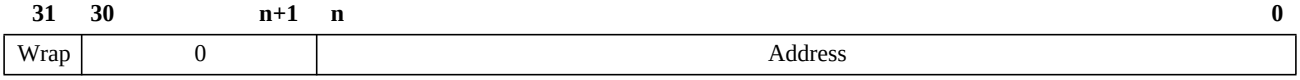


Table 3.5 *ITCBWRP* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Wrap	31	Indicates that the entire array has been written at least once	R/W	0	Required
0	30:(n+1)	Reserved. Must be written zero, reads back zero.	0	0	Required
Address	n:0	Byte address of the next on-chip trace memory word to be written	R/W	0	Required

3.3 ITCB iFlowtrace Off-Chip Interface

The off-chip interface consists of a 4-bit data port (*TR_DATA*) and a trace clock (*TR_CLK*). *TR_CLK* can be a DDR clock; that is, both edges are significant. *TR_DATA* and *TR_CLK* follow the same timing and have the same output structure as the PDtrace TCB described in MIPS specifications. The trace clock is synchronous to the system clock but running at a divided frequency. The *OfClk* bit in the *Control/Status* register indicates the ratio between the trace clock and the core clock. The Trace clock is always 1/2 of the trace port data rate, hence the “full speed” ITCB outputs data at the CPU core clock rate but the trace clock is half that, hence the 1:2 OfClk value is the full speed, and the 1:4 OfClk ratio is half-speed.

When a 64-bit trace word is ready to transmit, the PIB reads it from the FIFO and begins sending it out on *TR_DATA*. It is sent in 4-bit increments starting at the LSBs. In a valid trace word, the 4 LSBs are never all zero, so a probe listening on the *TR_DATA* port can easily determine when the transmission begins and then count 15 additional cycles to collect the whole 64-bit word. Between valid transmissions, *TR_DATA* Is held at zero and *TR_CLK* continues to run.

TR_CLK runs continuously whenever a probe is connected. An optional signal *TR_PROBE_N* may be pulled high when a probe is not connected and could be used to disable the off-chip trace port. If not present, this signal must be tied low at the Probe Interface Block (PIB) input.

The following encoding is used for the 6 tag bits to tell the PIB receiver that a valid transmission is starting:

```
//   if (srcount == 0), EncodedSrCount = 111010 = 58
//   else if (srcount == 16) EncodedSrCount = 111011 = 59
//   else if (srcount == 32) EncodedSrCount = 111100 = 60
//   else if (srcount == 48) EncodedSrCount = 111101 = 61
//   else EncodedSrCount = srcount
```

3.4 iFlowTrace related COP0 Registers

3.4.1 The *UserTraceData1* and *UserTraceData2* Registers (CP0 Register 23 Select 3 and CP0 Register 24 Select 3)

User defined trace messages are introduced in iFlowtrace revision 2.00.

A software write to any bits in the *UserTraceData1* register will trigger a trace message to be created indicating a type 1 user format. Similarly, a write by software to any bits in the *UserTraceData2* register will trigger a trace message to be created indicating a type 2 user format. It is implementation dependent whether or not writes to this register cause dependency stalling, or the latency between writes to the register and the subsequent generation of the trace record. Please read the core-specific implementation specification for this information. Please note that since these two registers are in CP0 register space, the access to these registers is ruled by CP0 access restrictions imposed by the system. For example, when a processor is under the control of an operating system such as Linux, these registers cannot be written by code executing in user-level privilege mode.

Figure 3.5 *UserTraceData1* and *UserTraceData2* Register Format

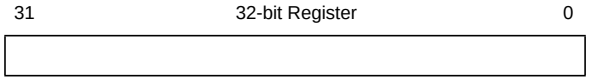


Table 3.6 *UserTraceData1* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Name	Bits				
Data	31:0 or 63:0	Software readable/writable data. When written, this triggers a user format trace message type1 out into the trace stream to be written into the trace memory.	R/W	0	Required after iFlowtrace spec 02.00 and higher.

Table 3.7 *UserTraceData2* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Name	Bits				
Data	31:0 or 63:0	Software readable/writable data. When written, this triggers a user format trace message type 2 out into the trace stream to be written to trace memory.	R/W	0	Required after iFlowtrace spec 02.00 and higher

Chapter 4

Breakpoint-Based Triggering of Tracing

Each hardware breakpoint in the EJTAG block (see the MIPS EJTAG Specification, MD00047, revision 4.14) has a control bit associated with it that enables a trigger signal to be generated on a break match condition. In special trace mode, this trigger can be used to insert an event record into the trace stream. In normal trace mode, this trigger signal can be used to turn trace on or off, thus allowing a user to control the trace on/off functionality using breakpoints. Similar to the TraceIBPC and TraceDBPC registers in PDtrace, registers are defined to control the start and stop of iFlowtrace. The details on the actual register names and drseg addresses are shown in Table 4.1.

Table 4.1 drseg Registers that Enable/Disable Trace from Breakpoint-Based Triggers

Register Name	drseg Address	Reset Value	Description
ITrigiFlowTrcEn	0x3FD0	0	Register that controls whether or not hardware instruction breakpoints can trigger iFlowtrace tracing functionality
DTrigiFlowTrcEn	0x3FD8	0	Register that controls whether or not hardware data and tuple breakpoints can trigger iFlowtrace tracing functionality

The bits in each register are defined as follows:

- Bit 28 (IE/DE) : Used to specify whether the trigger signal from EJTAG simple or complex instruction (data or tuple) break should trigger iFlowTrace tracing functions or not. A value of 0 disables trigger signals from EJTAG instruction breaks, and 1 enables triggers for the same.

- Bits 14..0 (IBrk/DBrk): Used to explicitly specify which instruction (data or tuple) breaks enable or disable iFlowTrace. A value of 0 implies that trace is turned off (unconditional trace stop) and a value of 1 specifies that the trigger enables trace (unconditional trace start).

Chapter A

References

This appendix lists other documents available from MIPS Technologies that are referenced elsewhere in this document. These documents may be included in the `$MIPS_HOME/$MIPS_CORE/doc` area of a typical CoreName soft or hard core release, or in some cases may be available on the MIPS web site, http://www.mips.com.

1. MIPS® EJTAG Specification
   MIPS document: MD00047

2. MIPS® EJTAG Trace Control Block Specification
   MIPS document: MD00148

3. MIPS® PDtrace™ Interface Specification
   MIPS document: MD00136

4. MIPS® Architecture For Programmers, Volume III: The MIPS32® and microMIPS32™ Privileged Resource Architecture
   MIPS document: MD0090

Appendix B

Revision History

Revision	Date	Description
1.00	June 26, 2006	Initial release.
1.01	July 15, 2008	Add Reference appendix.
1.80	February 25, 2009	• Add special tracing modes. • Sync period is now user-defined
1.81	July 21, 2009	• Special mode overflow behavior is the same as Normal/PC mode. • Added comment that Special-Mode tracing is disabled when the CPU is in DebugMode, just like Normal Mode tracing. • Added comment that tuple breakpoint is treaded like a data breakpoint.
1.82	September 11, 2009	• Special mode overflow message doesn't include PC.
1.83	November 09, 2009	• Breakpoint Match Mode and Filtered Data Mode are only defined when appropriate $BreakPointControl_{BE}$=0. • User Trace Message - added identifier bit for UTM1 & UTM2 • Added description of UserTraceData1 & UserTraceData2 COP0 registers.
2.00	March 20, 2009	• Function Call/Return section - 2.3.1.4 - Added ISA specific instructions - MIPS16e, microMIPS, MCU ASE. • Updated References Document names • Mention 2.0 changes in Chapter 1