Debug module in TEU

Titto Thomas

May 31, 2016

1 Introduction

The debug module inside TEU will be responsible for storing the information about breakpoints and watchpoints and also informing the stream corrector to stop the program execution.

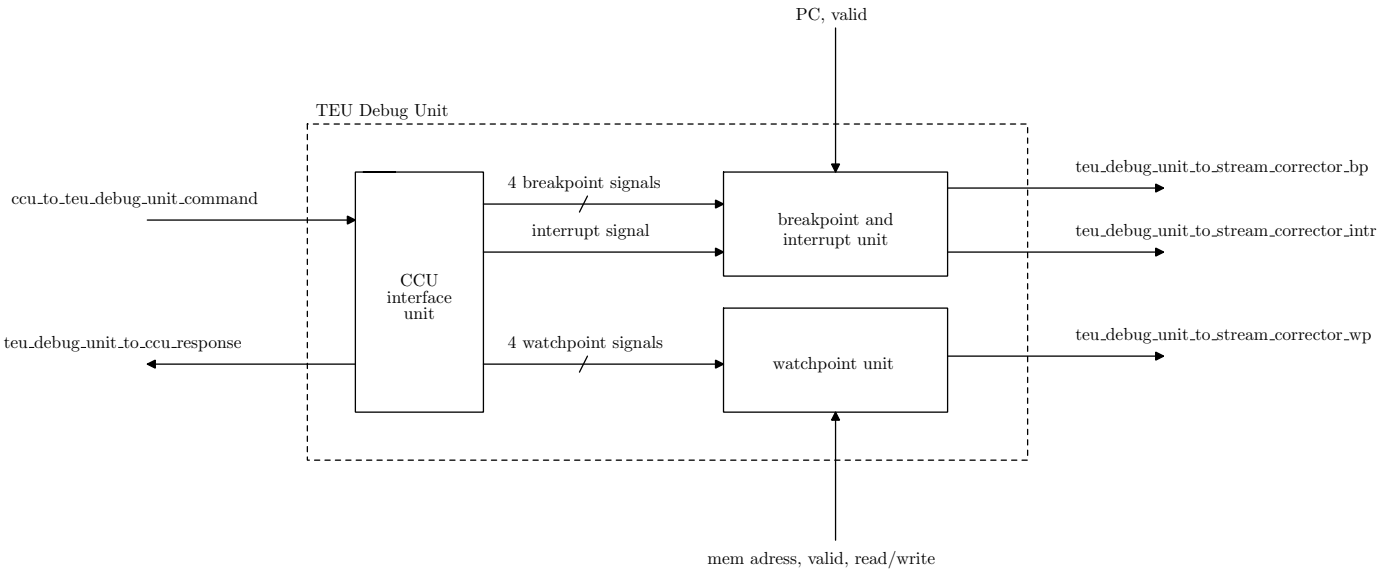

Figure 1: TEU debug module architecture

2 Expected behaviours

Expected behaviours of the two blocks are represented in the form of pseudo code in the following sections.

2.1 CCU interface unit

The pseudocode is described in the Algorithm 1 below.

2.1.1 Inputs

The input messages for the daemon will be of the format shown in Fig 2. The fields could have the values,

1. interrupt : Interrupt the processor execution flow, if this bit is set (kill request from GDB).
2. break or watch : 0 is for breakpoint register write and 1 for watchpoint register write.

Algorithm 1 ccu interface daemon

```
1: function WRITE_DEBUG_REGISTERS_DAEMON
2:     while 1 do                                                    ▷ Infinite loop
3:         ccu_debug_msg ← ccu_to_teu_debug_unit_command
4:         command, value ← Decode (ccu_debug_msg)
5:         if (command = update_breakpoint) then
6:             break_point_reg[reg_id] ← value
7:         else if (command = update_watchpoint) then
8:             watch_point_reg[reg_id] ← value
9:         else if (command = update_interrupt) and (prev_interrupt = FALSE) then
10:            currnt_interrupt ← TRUE
11:            prev_interrupt ← TRUE
12:        else
13:            currnt_interrupt ← FALSE
14:        end if
15:        for reg_id in all breakpoints do
16:            signal_breakpoint_reg_id ← break_point_reg[reg_id]
17:        end for
18:        for reg_id in all watchpoints do
19:            signal_watchpoint_reg_id ← watch_point_reg[reg_id]
20:        end for
21:        signal_interrupt ← currnt_interrupt
22:        if (command = valid) then
23:            teu_debug_unit_to_ccu_response ← SUCCESS
24:        end if
25:    end while
26: end function
```

3. set or clear : 0 is for clearing the register and 1 is for setting up a new watchpoint/breakpoint.

4. reg_id : The 2-bit register id.

5. watch_reg_type : The type of watchpoint register to be set up. Here 1 is for watching just the write access, 2 for read access, 3 is for watching both of them.

6. reg_val : The 32-bit address to be stored in the registers.

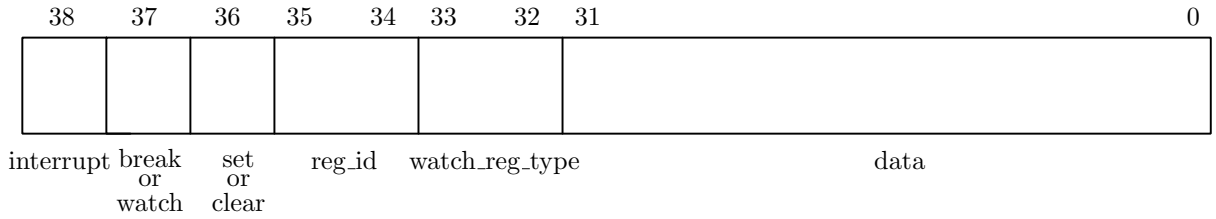

Figure 2: ccu_to_teu_debug_unit_command message format

2.1.2 Output

The teu_to_ccu_break_watch_register_access_response contains a single bit indicating if the previous command was successfully executed or not.

The block gives out 4 signals corresponding to current watchpoints (35-bit each), 4 breakpoints (33-bit each), and one corresponding to the interrupt. The formats of these signals are shown in Fig. 3.

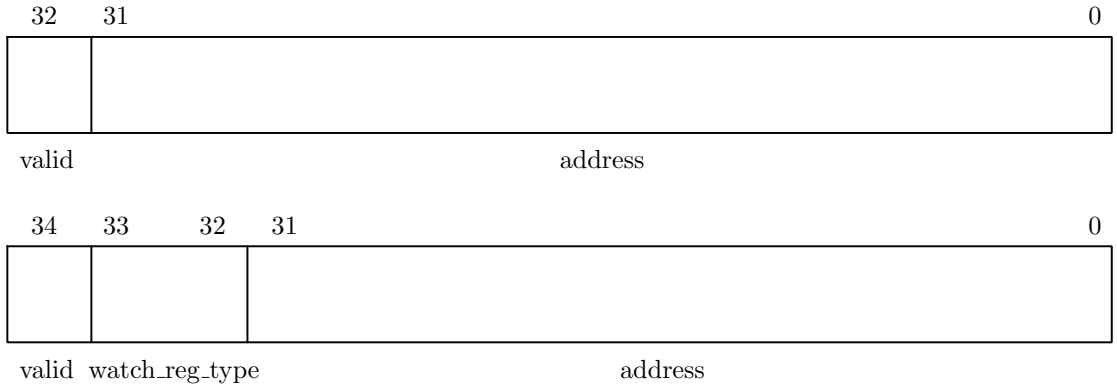

Figure 3: (a) breakpoint signal format (b) watchpoint signal format

2.2 debug compare unit

The unit is split into two parts, one which checks the breakpoints & interrupt and comparing the watchpoints. The pseudocodes are described in the Algorithm 2 and 3 below.

Algorithm 2 breakpoint and interrupt unit

```
1: function BP_AND_INTR_CHECK_DAEMON
2:     while 1 do                                                    ▷ Infinite loop
3:         idispatch_msg ← idispatch_to_teu_debug_unit
4:         intr_hit ← Decode (idispatch_msg)
5:         if (not intr_hit) then
6:             PC, valid ← Decode (idispatch_msg)
7:             bp_hit, reg_id ← Compare(PC, valid breakpoint registers)
8:         end if
9:         if (intr_hit) then
10:            teu_debug_unit_to_stream_corrector_intr ← TRUE
11:        else if (bp_hit and valid) then
12:            sc_msg ← Encode (bp_hit, reg_id)
13:            teu_debug_unit_to_stream_corrector_bp ← sc_msg
14:        end if
15:    end while
16: end function
```

Algorithm 3 watchpoint unit

```
1: function WP_CHECK_DAEMON
2:     while 1 do                                                    ▷ Infinite loop
3:         iunit_msg ← iunit_to_teu_debug_unit
4:         address, access_type ← Decode (iunit_msg)
5:         wp_hit, reg_id, hit_type ← Compare(address, valid breakpoint registers, access_type)
6:         if (wp_hit) then
7:             sc_msg ← Encode (wp_hit, reg_id, wp_hit_type)
8:             teu_debug_unit_to_stream_corrector_wp ← sc_msg
9:         end if
10:    end while
11: end function
```