MIPS 弶 CPUµ CP0 □ ´¦

2025 年 11 月 12 日

目录

1

±¾ ĵµ MIPS 弶 CPU ´¦ 0£¨CP0£©µ £CP0 MIPS¼¹ ¿ □ ´¦ Ĺ ğ¿ 養 ¸º □ ¨ SYSCALL¡¢BREAK¡¢µ ´ ©º ¨ 絥 ©£¬²¢ »¤ ¼Ĵ

2 CP0ğ¿ [1]

2.1 ğ¿

CP0ğ¿ 飨 `cp0.v`£© □ ´¦ µĺ ğ¬ ¨ £º

Listing 1: CP0ğ¿ ¨

```
module cp0(
    input                clk,        //
    input                resetn,     // ¸´ źţ¬µ

    // 4 WB¼¶µĿ    ź
    input                mtc0,       // MTC0
    input                mfc0,       // MFC0
    input       [ 7:0]   cp0r_addr,  // CP0¼Ĵ      }¼Ĵ   [4:0],     [2:0]}
    input       [31:0]   wdata,      //    CP0µ   ¾

    //  □    ź
    input                syscall,    // SYSCALL
    input                eret,       // ERET
    input       [31:0]   pc,         // µ±ǰPC£¨    £´浽EPC£©
    input                wb_valid,   // WB¼¶    ź
    input                wb_over,    // WB¼¶    ź

    //    □    ¨4  WBµ     þ
    input                ex_valid_i,         //  □
    input       [ 4:0]   ex_code_i,          //  □±
    input                ex_bd_i,            //     □
    input       [31:0]   ex_pc_i,            // ·¢   □µ PC
    input                badvaddr_valid_i,   // ´
    input       [31:0]   badvaddr_i,         // ´

    // CP0¼Ĵ      ¾
    output      [31:0]   cp0r_rdata,         // CP0¼Ĵ      ¾ ¨     MFC0£©

    //  □´¦
    output               cancel,     //           ź
    output               exc_valid,  //  □     ź
    output      [31:0]   exc_pc,     //  □      « ERET · µ»
    // ¼Ĵ           □´¦
    output     [31:0] cp0r_status,      // STATUS¼Ĵ
    output     ]31:0] cp0r_cause,        // CAUSE¼Ĵ
    output     ]31:0] cp0r_epc,          // EPC¼Ĵ

    //
    output            c0_int             //        ź
);
```

2.2

CP0ğ¿

1. □ £º □ £¨° (SYSCALL¡¢BREAK¡¢µ ´ ©¶¼¹ µ □ ¨ `ex_valid_i, ex_code_i`
2. **¼Ĵ** ᶜ £º¹ MTC0/MFC0 CP0¼Ĵ □ WMASK£©¿
3. □ ¶£º WB¼¶½ □ £¡CP0 ã¬¡¶¨ ¶º ´«µ

.4 £ºCP0 ¿¼ £¬ źš£

3 CP0¼Ĵ

3.1 ¼Ĵ

±¾ º¬ CP0¼Ĵ

表 1: CP0¼Ĵ

¼Ĵ	³		¹¦
12	0	STATUS	¼Ĵ
13	0	CAUSE	□ Ĵ
14	0	EPC	□ ³
8	0	BADVADDR	´ ¼Ĵ
9	0	COUNT	¶¨ ¼Ĵ
11	0	COMPARE	¶¨ Ĵ

3.2 STATUS¼Ĵ Ĵ 12£©

STATUS¼Ĵ º £

Listing 2: STATUS¼Ĵ

```
// STATUS¼Ĵ
wire status_ie;          // bit 0: ȫ¾          (IE)
wire status_exl;         // bit 1:  □¼¶±   (EXL)
wire [7:0] status_im;    // bit 15:8:     q    (IM)

// STATUS¼Ĵ          □ ³  IE¡¢EXL¡¢IM £©
wire [31:0] STATUS_WMASK;
assign STATUS_WMASK = 32'h0000_8103; // bit 0(IE), bit 1(EXL), bit 15:8(IM)
```

1

- IE (bit 0)£ºȫ¾ ¡£µ±IE=0 £¬ » qP£
- EXL (bit 1)£º □ ¼¶± ¡£µ±EXL=1 £¬CPU´¦ □ ´¦ ģ £¬ µ □ ±» qP£
- IM[7:0] (bit 15:8)£º q ¡£ÿ ¶ ¸ £¬IM[7]¶ ¶¨ £

º

Listing 3: STATUS¼Ĵ

```
if (status_wen) begin
    status <= (status & ~STATUS_WMASK) | (wdata & STATUS_WMASK);
end
```

3.3 CAUSE¼Ĵ Ĵ 13£©

CAUSE¼Ĵ ¼ □ ¡£

Listing 4: CAUSE¼Ĵ

```
// CAUSE¼Ĵ
wire cause_bd;           // bit 31:        )BD)
wire cause_ti;           // bit 30: ¶¨          )TI)
wire [7:0] cause_ip;     // bit 15:8:          (IP)
wire [4:0] cause_excode; // bit 6:2:  □±     (ExcCode)

// CAUSE¼Ĵ         □ ³  IP[1:0] £©
wire [31:0] CAUSE_WMASK;
assign CAUSE_WMASK = 32'h0000_0300;  // bit 9:8(IP[1:0])
```

1

- BD (bit 31)£º ¡£µ± □ · £¡ £¬BD=1 ¢
- TI (bit 30)£º¶¨ ¡£µ±COUNT == COMPARE £¬TI=1¡£

- IP[7:0] (bit 15:8)£º ¡£IP[7]¶ ¶¨ ¬ ¼ ¯¸ ¡£
- ExcCode (bit 6:2)£º □ ± □ ± □ £

  □

表 2: □ ±

□ ±	□
0	(Interrupt)
4	µ ´ -¼ (AdEL)
5	µ ´ -´ 泋 (AdES)
8	µ (SYSCALL)
9	¶ (BREAK)
12	(OV)

3.4 EPC¼Ĵ Ĵ 14£©

EPC¼Ĵ 減 □ µij ¡£

Listing 5: EPC¼Ĵ

```
//    □´¦        □¶¼ ¹  ex_valid_i´«µ
if (ex_valid_i && wb_valid) begin
    status[1] <= 1'b1;                  // EXL
    cause[31] <= ex_bd_i;               // BD
    cause[6:2] <= ex_code_i;            // ExcCode
    epc <= ex_bd_i ? ex_pc_i : ex_pc_i;   //      PC»    PC
    if (badvaddr_valid_i) begin
        badvaddr <= badvaddr_i;
    end
end
```

- µ± □ · ¢ £¬EPC±£´ 減 □ µ ¡£
- □ · ¢ ¨BD=1£©£¬EPC±£´ ĵ ¡£
- ERET £¬CPU µ½EPC±£´ ĵ ¼ £

3.5 BADVADDR¼Ĵ Ĵ 8£©

BADVADDR¼Ĵ 涴 µ ´ ¡£

Listing 6: BADVADDR¼Ĵ

```
if (badvaddr_valid_i) begin
    badvaddr <= badvaddr_i;
end
```

- ´ ½ ☐ £¨AdEL/AdES£© ☐
- MEM¼¶¼ ¶ ¹ ☐ £¡CP0½ µ»

3.6 COUNT¼Ĵ Ĵ 9£©

COUNT¼Ĵ ¨ ¨ £

Listing 7: COUNT¼Ĵ

```
// ¶¨        £¨ÿ}¸          ´Σ¬½µµ       ©
reg time_tick;
always @(posedge clk) begin
    if (!resetn) begin
        time_tick <= 1'b0;
    end else begin
        time_tick <= ~time_tick;
    end
end

// COUNT¼Ĵ     £¬» ÿ}¸
always @(posedge clk) begin
    if (!resetn) begin
        count <= 32'd0;
    end else begin
        if (count_wen) begin
            count <= wdata;
        end else if (time_tick) begin
            count <= count + 1'b1;
        end
    end
end
```

- COUNT¼Ĵ ¹ MTC0 ☐
- ³£ £¬ÿ¸ 1£¨¹ `time_tick` · £©¡£
- µ±COUNT == COMPARE £¬´¥ · ¢¶¨ £

3.7 COMPARE¼Ĵ Ĵ 11£©

COMPARE¼Ĵ ¨ £¬ ¨ £

Listing 8: COMPARE¼Ĵ

```
// COMPARE¼Ĵ     £¬       ¶¨
always @(posedge clk) begin
    if (!resetn) begin
        compare <= 32'd0;
    end else begin
        if (compare_wen) begin
            compare <= wdata;
        end
    end
end

// ¶¨         £¨cause_ti_reg£©
always @(posedge clk) begin
    if (!resetn) begin
        cause_ti_reg <= 1'b0;
    end else begin
        if (compare_wen) begin
            cause_ti_reg <= 1'b0;  //    COMPARE
        end else if (count_eq_compare) begin
            cause_ti_reg <= 1'b1;   // COUNT == COMPARE
        end
    end
end
```

- COMPARE¼Ĵ ¹ MTC0 □
- COMPARE¼Ĵ ¶¨ £¨TI £©¡£
- µ±COUNT == COMPARE £¬ TI± £¬´¥ · ¢ £

4 □ ´¦

4.1 □ ¼

□ ¼ ¼ IJ»½ Σº

1. **ID¼¶£º¼** SYSCALL¡¢BREAK
2. **MEM¼¶£º¼** ¶ AdEL/AdES£©¡£

3. **WB¼¶£º** □ £¬·¶¨ ¶¡£

4. **CP0£º¼** £

4.2 WB¼¶ □

WB¼¶¸º □ £¬·¶¨ ¶º CP0£º

Listing 9: WB¼¶ □

```
// □ £¨ ¶£º > µ ´ > BREAK > SYSCALL£©
// · □£¨µ ´ BREAK¡¢SYSCALL£©
assign wb_ex_valid_no_int  = (mem_ex_adel_wb | mem_ex_ades_wb | brk_wb | syscall) ?
    WB_valid : 1'b0;
assign wb_ex_code_no_int   = mem_ex_adel_wb ? 5'd4 :  // AdEL
                             mem_ex_ades_wb ? 5'd5 :  // AdES
                             brk_wb ? 5'd9 :          // BREAK
                             syscall ? 5'd8 : 5'd0;   // SYSCALL

//    □   źţ¨     ¶   ©
assign wb_ex_valid = (cp0_int && WB_valid) | wb_ex_valid_no_int;
assign wb_ex_code  = cp0_int ? 5'd0 : wb_ex_code_no_int;  //    □  IO
assign wb_ex_bd    = 1'b0;        //       ½
assign wb_ex_pc    = pc;          //  □PC£¨µ ´   syscall¾ µ±ǰpc£©
```

□ ¶£º

1. ¶¨ ¨ ¶£©

2. µ ´ AdEL/AdES£©

.3 BREAK □

4. SYSCALL □

4.3 □ ´¦

µ± □ · ¢ £¬CP0 ²

1. **EXL** £ºSTATUS[1] = 1£¬½ □ ´¦ ģ ¡£

2. ±£´ **EPC**£º½« · ¢ □ µ PC±£´ 浽 EPC¼Ĵ

3. **CAUSE**£º □ ± □ ExcCode£©º £¨BD£©¡£

4. ±£´ **BADVADDR**£º ģ ´ £´ ¡£

5. □ £ºCPU µ½ □ £¡©0x0£¨£

Listing 10: CP0 □ ¦´

```
//    □´¦        □¶¼¹ ex_valid_i´«µ
if (ex_valid_i && wb_valid) begin
    status[1] <= 1'b1;                      // EXL
    cause[31] <= ex_bd_i;                   // BD
    cause[6:2] <= ex_code_i;                // ExcCode
    epc <= ex_bd_i ? ex_pc_i : ex_pc_i;   //      PC»    PC
    if (badvaddr_valid_i) begin
        badvaddr <= badvaddr_i;
    end
end
```

4.4 ERET

ERET □´¦ » º

Listing 11: ERET

```
// ERET      EXL
if (eret && wb_valid) begin
    status[1] <= 1'b0;   //    EXL
end

//   □/·µ»     ź
assign exc_pc = eret ? epc : `EXC_ENTER_ADDR;
```

- ERET £¬ STATUS[1]£¨EXL £©£¬ □´¦ ģ¡£
- CPU µ½EPC±£´ĵ¼ £

5 ¦

5.1 ¶¨

¶¨ CP0 ¿¼

Listing 12: ¶¨

```
// COUNT == COMPARE¼
assign count_eq_compare = (count == compare);

// ¶¨        £¨cause_ti_reg£©
always @(posedge clk) begin
    if (!resetn) begin
```

```
        cause_ti_reg <= 1'b0;
    end else begin
        if (compare_wen) begin
            cause_ti_reg <= 1'b0;  //     COMPARE
        end else if (count_eq_compare) begin
            cause_ti_reg <= 1'b1;   // COUNT == COMPARE
        end
    end
end

// CAUSE¼Ĵ
always @(posedge clk) begin
    // cause[30]: TI £¨¶¨        £©
    if (!ex_valid_i || !wb_valid) begin
        cause[30] <= cause_ti_reg;
    end

    // cause[15:8]: IP £¨       £©
    // IP[7] = TI£¨¶¨      ©
    if (!ex_valid_i || !wb_valid) begin
        cause[15:8] <= {cause_ti_reg, 5'd0, cause[9:8]};
    end
end
```

5.2

² ¥ · ¢£º

Listing 13:

```
//
//       £º          &&  ¶          &&  ō¾         &&  ²»    ☐¼¶±
assign c0_int = |(cause_ip[7:0] & status_im[7:0]) & status_ie & !status_exl;
```

£º

1. IP Ī1£©
2. ¶ q ¨IM Ī1£©
3. ō¾ ¨IE=1£©
4. ²» ☐ ¼¶± EXL=0£©

6 Ų¼

6.1 MEM->WB

MEM¼¶¹ « □ ´«µ WB¼¶£º

Listing 14: MEM->WB ¨

```
`define MEM_WB_BUS_WIDTH    153

// ) MEM->WB    ¬    mem_ex_adel, mem_ex_ades, badvaddr(dm_addr)
assign MEM_WB_bus = {rf_wen,rf_wdest,                    // WB     õ  ź
                     mem_result,                         //        » Ĵ       ¾
                     lo_result,                          // ³ ¨µ 32 ½
                     hi_write,lo_write,                  // HI/LO
                     mfhi,mflo,                          // WB     õ  ź
                     mtc0,mfc0,cp0r_addr,syscall,brk,eret,  // WB     õ  ź
                     mem_ex_adel, mem_ex_ades,           // µ   □±  £¨
                     dm_addr,                            // BADVADDR£¨
                     pc};                                // PC
```

- mem_ex_adel£ºLoadµ ´
- mem_ex_ades£ºStoreµ ´
- dm_addr£º · ô £¨ BADVADDR£©
- syscall, brk, eret£º □

6.2 WB->CP0 □

WB¼¶¹ □ « □ ´«µ CP0£º

Listing 15: WB->CP0 □

```
//    □    ¨´«µ  CP0£©
assign wb_ex_valid = (cp0_int && WB_valid) | wb_ex_valid_no_int;
assign wb_ex_code  = cp0_int ? 5'd0 : wb_ex_code_no_int;
assign wb_ex_bd    = 1'b0;
assign wb_ex_pc    = pc;
assign wb_badvaddr_valid = mem_ex_adel_wb | mem_ex_ades_wb;
assign wb_badvaddr = mem_badvaddr_wb;
```

7 ¸ ☐

7.1 SYSCALL ☐

¼ ãºID¼¶£¨ ☐

Listing 16: SYSCALL

```
// decode.v
assign inst_SYSCALL = (op == 6'b000000) & (funct == 6'b001100);
```

´¦ .º

1. ID¼¶¼ SYSCALL ½«syscall źŴ«µ ½WB¼¶¡£
2. WB¼¶ ã¬ ☐ Ĩ8¡£
3. CP0±£´ EPC£¨SYSCALL £©£¬ EXL=1£¬ µ½ ☐ £
.4 ☐
5. ´¦ EPC += 4£¬ERET · µ» £

7.2 BREAK ☐

¼ ãºID¼¶£¨ ☐

Listing 17: BREAK

```
// decode.v
assign inst_BREAK = (op == 6'b000000) & (funct == 6'b001101);
```

´¦ .º

1. ID¼¶¼ BREAK ½«brk źŴ«µ ½WB¼¶¡£
2. WB¼¶ ã¬ ☐ Ĩ9¡£
3. CP0±£´ EPC£¨BREAK £©£¬ EXL=1£¬ µ½ ☐ £
.4 ☐
5. ´¦ EPC += 4£¬ERET · µ» £

7.3 µ ´ ☐ £¨AdEL/AdES£©

¼ ãºMEM¼¶£¨ · ô Σ©

Listing 18: µ ¶

```
// mem.v
wire mem_ex_adel;  // load µ  ´
wire mem_ex_ades;  // store µ  ´
assign mem_ex_adel = MEM_valid && inst_load  && ls_word && (dm_addr[1:0]!=2'b00);
assign mem_ex_ades = MEM_valid && inst_store && ls_word && (dm_addr[1:0]!=2'b00);
```

´¦ .º

1. MEM¼¶¼ µ ²»¶ □ µ 2 ²»Ī00£©£¬ mem_ex_adel» mem_ex_ades¡£

2. ½«´ £¨dm_addr£©º □ £¡¶¼WB½ µ»´ ±

.3 WB¼¶ ã¬ □ Ī4£¨AdEL£©» 5£¨AdES£©¡£

4. CP0±£´ EPC£¨³ £©£¬±£´ BADVADDR£¨´ £©£¬ EXL=1£¬ µ½ □ £

.5 □

6. ´¦ EPC += 4£¬ERET · µ» £

7.4 ¶¨

¼ ãºCP0 ¿

Listing 19: ¶¨

```
// cp0.v
assign count_eq_compare = (count == compare);
assign c0_int = |(cause_ip[7:0] & status_im[7:0]) & status_ie & !status_exl;
```

´¦ .º

1. CP0¼ COUNT == COMPARE£¬ TI± ¡£

2. ¼ £¨IE=1, IM[7]=1, EXL=0£©¡£

3. 徥 c0_int źš£

4. WB¼¶ ã¬ □ Ī0£¨ ©¡£

5. CP0±£´ EPC£¨±» £©£¬ EXL=1£¬ µ½ □ £

.6 □

7. ´¦ COMPARE TI £¬ERET · µ» £

8 CP0¼Ĵ

8.1 MTC0 CP0¼Ĵ

MTC0 CP0¼Ĵ

Listing 20: MTC0

```
//       ź
wire mtc0_wr;  // MTC0   ¨ ų  □   □
assign mtc0_wr = mtc0 && wb_valid && !ex_valid_i; //  □ ²»

assign status_wen   = mtc0_wr && sel_status;
assign cause_wen    = mtc0_wr && sel_cause;
assign epc_wen      = mtc0_wr && sel_epc;
assign count_wen    = mtc0_wr && sel_count;
assign compare_wen  = mtc0_wr && sel_compare;
assign badvaddr_wen = mtc0_wr && sel_badvaddr;

//      £¨       □
if (status_wen) begin
    status <= (status & ~STATUS_WMASK) | (wdata & STATUS_WMASK);
end
```

- □ · ¢ £¬½ CP0¼Ĵ !ex_valid_i£©¡£
- □ WMASK£©¿
- STATUSº CAUSE¼Ĵ ¿ · ¿ ¡£

8.2 MFC0 ¶ CP0¼Ĵ

MFC0 CP0¼Ĵ

Listing 21: MFC0

```
// MFC0¶
assign cp0r_rdata = sel_status  ? status   :
                    sel_cause   ? cause    :
                    sel_epc     ? epc      :
                    sel_count   ? count    :
                    sel_compare ? compare  :
                    sel_badvaddr? badvaddr : 32'd0;
```

- ¸ ¾ CP0¼Ĵ £¨cp0r_addr£© µļĴ
- ¶ ¾ ¹ cp0r_rdata » ½ üĴ

9 ź

9.1 cancel ź

cancel ź ³

Listing 22: cancel ź ³

```
assign cancel = (ex_valid_i | eret | c0_int) && wb_over;
```

- µ± ☐ » ERET · ¢ £¬ ³
- cancel ź WB¼¶ £¨wb_over£© · ¢³

9.2 exc_validº exc_pc ź

exc_validº exc_pc ź ☐ º£

Listing 23: ☐ ź ³

```
assign exc_valid = (ex_valid_i | eret | c0_int) && wb_valid;
assign exc_pc    = eret ? epc : `EXC_ENTER_ADDR;
```

- exc_valid£º ☐ źţ¬¿ ¡£
- exc_pc£º Ŀ± £¬ERET · µ» EPC£¬ ☐ µ½ ☐ £¡©0x0£¨

10 ☐ ´¦

±¾ ½ ☐ ´¦ ı · ½ · ¨¡¢² · µ £¾ ²¹³ 輯

10.1 ☐ ´¦

²¹³ ´¨£ 酢

10.2 ²

£¨´ ²¹³ 酢

11

±¾ ĵµ MIPS 弶 CPU CP0 □ ´¦ µ ¬° (£º

- CP0ģ¿ ¹º
- 6¸ CP0¼Ĵ ½ ¨STATUS¡¢CAUSE¡¢EPC¡¢BADVADDR¡¢COUNT¡¢COMPARE£©
- □ û ¦´
- 4 □ ¨SYSCALL¡¢BREAK¡¢AdEL/AdES¡¢¶¨ ©
- Ų¼º źŴ«µ
- CP0¼Ĵ ¨MTC0/MFC0£©
- ź

¸ MIPS¼¹¹ 浗 £¬ş µ □ ´¦ ³¬Ĩ ş ¿µ □ ´¦ σ£