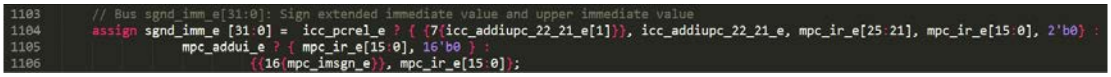
- ***SUB* instruction**
  - The same adder as the one used for the *ADD* instruction is used here (*m14k_edp_add_simple*).
  - In case of a SUB instruction, Source Operand B is *2's complemented* before being provided to the adder:
    1. Signal ***mpc_subtract_e*** is set to 1 for SUB instructions in *m14k_mpc_dec* module.
    2. Source Operand B is inverted when ***mpc_subtract_e***=1 before going into the adder (*1s complement*):

       ***bop_e***[31:0] = {32{***mpc_subtract_e***}} ^ ***bbus_imm_e***
    3. A Carry-In is inserted in the adder for performing the *2's complement*: *.ci(**mpc_subtract_e**)*.

- ***ADDIU* instruction**
  - Source B is provided from signal ***sgnd_imm_e*** instead of signal ***edp_bbus_e*** (from the RF) at multiplexer *_bbus_imm_e_31_0_* (Figure 5).
  - Signal ***sgnd_imm_e*** is computed in the following multiplexer:



```
        // Bus sgnd_imm_e[31:0]: Sign extended immediate value and upper immediate value
        assign sgnd_imm_e [31:0] =  icc_pcrel_e ? { {7{icc_addiupc_22_21_e[1]}}, icc_addiupc_22_21_e, mpc_ir_e[25:21], mpc_ir_e[15:0], 2'b0} :
                    mpc_addui_e ? { mpc_ir_e[15:0], 16'b0 } :
                        {{16{mpc_imsgn_e}}, mpc_ir_e[15:0]};
```

  where:
  1. ***icc_pcrel_e*** is always 0.
  2. ***mpc_addui_e***=0 for ADDIU instructions, as determined in module *m14k_mpc_dec*.

  thus:

  ***sgnd_imm_e*** = *{{16{**mpc_imsgn_e**}}, **mpc_ir_e**[15:0]}*

  where:
  1. ***mpc_imsgn_e*** is just equal to ***mpc_ir_e***[15] in this case.

- ***SLT* instruction**
  - The result of a *slt* instruction is computed at module *m14k_edp* in signal ***bit0_m***, as follows:

```
assign bit0_m = mpc_signed_m ? (pro31_m ^ car31_m) : ~car31_m;
```

  2 cases are possible: signed instructions (***mpc_signed_m***=1), such as *slt* or *slti*, and unsigned instructions (***mpc_signed_m***=0), such as *sltu* or *sltiu*.
  - Signal ***mpc_signed_m*** is registered from signal ***signed_e***, computed at ***m14k_mpc_dec*** at the E-Stage as:

```
assign signed_e = (mpc_ir_e[29] & ~mpc_ir_e[26]) | (~mpc_ir_e[29] & ~mpc_ir_e[26]
& ~mpc_ir_e[0]) | (~mpc_ir_e[29] & mpc_ir_e[26] & ~mpc_ir_e[16]);
```

- Note that ***car31_m*** is the carry out of the Adder (module m14k_edp_add_simple) and that:

  ```
  pro31_e = aop_e[31] ^ bop_e[31];
  ```

  Note also that the adder performs a subtraction for any SLT-Type instruction, given that:

  ```
          assign mpc_subtract_e =
              (mpc_ir_e[31:27] == 5'h05) ||            // SLTI(U)
              ((mpc_ir_e[31:26] == 6'h1) &&            // RegImm
                 (mpc_ir_e[20:19] == 2'b01)) ||        // trap immed
               ((mpc_ir_e[31:26] == 6'h0) &&           // SPECIAL
                 (mpc_ir_e[5:1] != 5'b10000));         // Add(U)
  ```

- This result is assigned to signal ***res_m*** in module *m14k_edp_buf_misc_pro* whenever we have this kind of instructions, computed by the following two multiplexers:

  ```
  mvp_mux2 #(32) _udislt_m_31_0_(udislt_m[31:0],mpc_udisel_m && edp_udi_present,
  {31'h0, bit0_m}, UDI_data_m[31:0]);
  mvp_mux2 #(32) _res_m_31_0_(res_m[31:0],mpc_udislt_sel_m, asp_m[31:0],
  udislt_m[31:0]);
  ```

  Signal ***mpc_udislt_sel_m*** is computed at *m14k_mpc_ctl* as follows:

  ```
  assign mpc_udislt_sel_m = slt_sel_m | mpc_udisel_m;
  ```

  Signal ***slt_sel_m*** is registered from signal ***slt_sel_e***, computed at *m14k_mpc_dec* as follows:

  ```
    assign slt_sel_e =   (mpc_ir_e[31:27] == 5'b001_01) ||  // slti, sltiu
     special_e && (mpc_ir_e[5:1] == 5'b101_01) ||             // slt, sltu
     special_e && (mpc_ir_e[5:2] == 4'b110_0)  || // tge, tgeu, tlt, tltu
     regimm_e && (mpc_ir_e[20:18] == 3'o2);    // tgei, tgeiu, tlti tltiu
  ```

  Thus, for a slt instruction, control signal ***slt_sel_e***=1