- *J* instruction
  - Control signal *mpc_jimm_e*, generated at the m14k_mpc_dec module, is 1 for *J* instructions. This signal is used for several tasks.
  - Specifically, let´s analyze generation of signal *preiva_p*, at module m14k_edp. Figure 2 from Lab 18 shows a highly simplified version of this signal, where it is assigned directly from the sequential address (*edp_iva_i*+4). However, in the soft-core this signal is assigned from a 10-1 multiplexer (*_preiva_p_31_0_*). One input to that multiplexer, which is selected when *mpc_jimm_e*=1, is *jaddr_e*.
  - *jaddr_e* is computed as follows:

    {edp_iva_i[31:28], mpc_ir_e[25:0], 2'h0}

- *JAL* instruction
  - Control signal *mpc_jimm_e* is also 1 for *JAL* instructions at the E-Stage, and the effect is the same.
  - Besides, at *m14k_mpc_dec*, signal *lnk31_e* is set to 1 for *JAL* instructions at the E-Stage. This signal has 2 important implications:
    1. *dest_e*, the destination register to write at W-Stage, is set to 11111.
    2. Signal *maj_vd_e* is set, which implies that the Write Strobe for the RF will be set at W-Stage.
  - Two multiplexers, not included in previous figures (at Figure 1 of Lab 16, these multiplexers are connected between signals *edp_alu_m[31:0]* and *res_m[31:0]*, as shown in Figure 1 below), are used for computing the value to write to the Register File (*edp_wrdata_w*) in case of a JAL instruction, at the M-Stage:
    - *_sp_m_31_0_*: Uses as control signal *mpc_lnksel_m*, which is 1 for a JAL instruction at the M-Stage, selecting input signal *x_pc_hold*, which contains the address of the second instruction following the branch.
    - *_asp_nodsp_m_31_0_*: Uses as control signal *mpc_alusel_m*, which is 0 for a JAL instruction at the M-Stage, selecting input signal *sp_m*, which is the output of the previous multiplexer (*_sp_m_31_0_*).

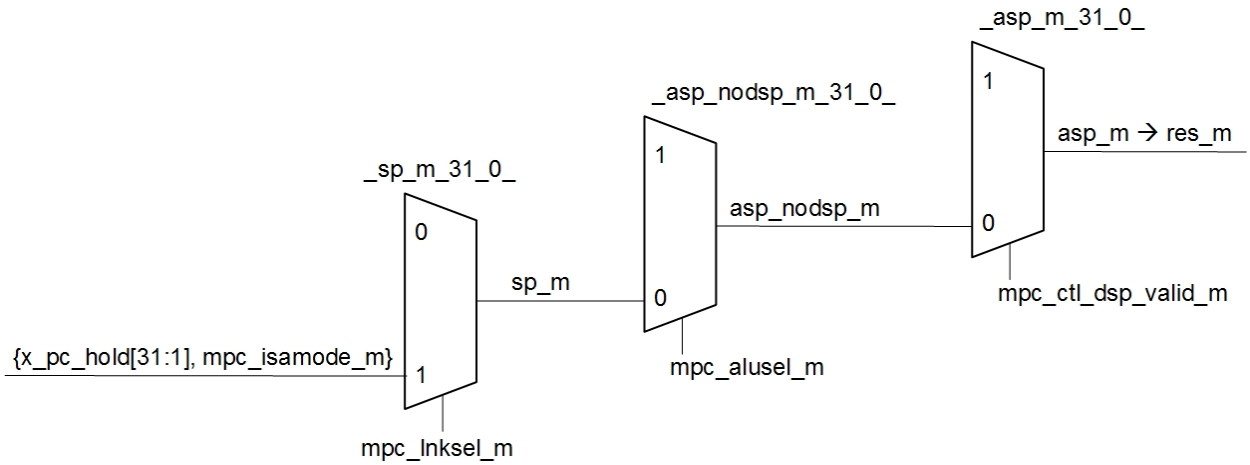

Figure 1. New multiplexers used by the JAL instruction.

- Other conditional branches
  - Each conditional branch has its own signal, computed at m14k_dec, which has a different effect in the condition computation.

```
1886    // br_gr: Branch on greater than zero (+L)
1887    assign br_gr =  (mpc_ir_e[31:26] == 6'o07) || (mpc_ir_e[31:26] == 6'o27);
1888
1889    // br_le: Branch on Less than or equal to zero (+L)
1890    assign br_le =  (mpc_ir_e[31:26] == 6'o06) || (mpc_ir_e[31:26] == 6'o26);
1891
1892    // br_ne: Branch on not equal (+L)
1893    assign br_ne =  (mpc_ir_e[31:26] == 6'o05) || (mpc_ir_e[31:26] == 6'o25);
1894
1895    // br_eq: Branch on equal (+L)
1896    assign br_eq =  (mpc_ir_e[31:26] == 6'o04) || (mpc_ir_e[31:26] == 6'o24);
```

```
2613    // Condition will be true if edp_cndeq_e == 1
2614    assign cnd_eq_en = mpc_irval_e  & (br_eq | br_le);
2615
2616    // Condition will be true if edp_cndeq_e == 0
2617    assign cnd_neq_en = mpc_irval_e & (br_ne | (br_gr & ~asign));
```