Appendice B

La progettazione in *VHDL*

Il componente *Read Out Slice*

E' riportato in seguito il file in *VHDL* strutturale del componente *Read Out Slice*. Tale file consente la lettura dalle 4 *FIFO* che ogni *Read Out Slice* può leggere. In particolare, il componente gestisce i segnali di *Empty Fifo**, *Output Enable** e *Read Enable** e permette di costruire pacchetti di parole da 24 bit. Ai 18 bit in ingresso sono aggiunti bit di controllo quali, ad esempio, i due bit di indirizzo della *FIFO* letta, un bit di errore e il bit di validazione dei dati.

```
library ieee;
use ieee.std_logic_1164.all;

entity edge_det2 is
    port( reset, clk, data_in: IN std_logic;
          data_out      : out std_logic);
  end;

architecture edge_det2 of edge_det2 is
  signal q_next, q_out: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q_out<='1';
     elsif ( clk'event and clk='1' ) then
              q_out<=q_next;
      end if;
     end process;

   q_next<=data_in;
   data_out<=data_in and(not q_out);
   end;

configuration edge_det2_conf of edge_det2 is
 for edge_det2
 end for;
end edge_det2_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity reg_en is
    port( reset, clk, header_en, pad_header, cm_head, cm_shead, cm_footer, ok_data, pad_ft_up: IN std_logic;
```

```
        data_out      : out std_logic);
  end;

architecture reg_en of reg_en is
  signal q_next, q_out, reg_enable: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q_out<='0';
      elsif ( clk'event and clk='1' ) then
              q_out<=q_next;
      end if;
     end process;

   q_next<=header_en;
   reg_enable<=( pad_header or (cm_head or (cm_shead or (cm_footer or (ok_data or pad_ft_up)))));
   data_out<=(q_out and reg_enable);
   end;

configuration reg_en_conf of reg_en is
 for reg_en
 end for;
end reg_en_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity header_en is
    port( reset, clk, ren0, ren1, ren2, ren3: IN std_logic;
        data_out      : out std_logic);
  end;

architecture header_en of header_en is
  signal q_next: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              data_out<='0';
      elsif ( clk'event and clk='1' ) then
              data_out<=q_next;
      end if;
     end process;

   q_next<=not ( ren0 and (ren1 and (ren2 and ren3)));
   end;

configuration header_en_conf of header_en is
 for header_en
 end for;
end header_en_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity empty_fifo_man is
    port( reset, clk, r_en, f_empty: IN std_logic;
         data_val     : out std_logic);
  end;

architecture empty_fifo_man of empty_fifo_man is
  signal q_next, q_out: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q_out<='0';
      elsif ( clk'event and clk='1' ) then
              q_out<=q_next;
      end if;
     end process;

   q_next<=r_en;
   data_val<= q_out and f_empty;
   end;

configuration empty_fifo_man_conf of empty_fifo_man is
```

```
 for empty_fifo_man
 end for;
end empty_fifo_man_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

 entity other_net is
  port( rst, ren_0, ren_1, ren_2, ren_3, o_n_clk, read_en, fifo_e, padfooter, padheader, cmhead, cmshead, cmfooter, okdata: IN
std_logic;
        padft_up, headeren: INOUT std_logic;
        register_en, valid_data: out std_logic);
 end;

 architecture other_net of other_net is

 component reg_en is
    port( reset, clk, header_en, pad_header, cm_head, cm_shead, cm_footer, ok_data, pad_ft_up: IN std_logic;
          data_out      : out std_logic);
  end component;

  component edge_det2 is
    port( reset, clk, data_in: IN std_logic;
          data_out      : out std_logic);
  end component;

 component header_en is
    port( reset, clk, ren0, ren1, ren2, ren3: IN std_logic;
          data_out      : out std_logic);
  end component;

 component empty_fifo_man is
    port( reset, clk, r_en, f_empty: IN std_logic;
          data_val      : out std_logic);
  end component;


 begin

  comp_a : reg_en PORT MAP (reset=>rst, clk=>o_n_clk, header_en=>headeren, pad_header=>padheader, cm_head=>cmhead,
cm_shead=>cmshead, cm_footer=>cmfooter, ok_data=>okdata, pad_ft_up=>padft_up, data_out=>register_en);
  comp_b : edge_det2 PORT MAP (reset=>rst, clk=>o_n_clk, data_in=>padfooter, data_out=>padft_up);
  comp_c : header_en PORT MAP (reset=>rst, clk=>o_n_clk, ren0=>ren_0, ren1=>ren_1, ren2=>ren_2, ren3=>ren_3,
data_out=>headeren);
  comp_d : empty_fifo_man PORT MAP (reset=>rst, clk=>o_n_clk, r_en=>read_en, f_empty=>fifo_e, data_val=>valid_data);

end;

 configuration other_net_conf of other_net is

   for other_net

    for comp_a: reg_en use configuration work.reg_en_conf;
    end for;

    for comp_b: edge_det2 use configuration work.edge_det2_conf;
    end for;

    for comp_c: header_en use configuration work.header_en_conf;
    end for;

    for comp_d: empty_fifo_man use configuration work.empty_fifo_man_conf;
    end for;

 end for;
 end other_net_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity or_2 is

    port( a, b : IN std_logic;
          c    : out std_logic);
  end;

architecture or_2 of or_2 is
   begin
```

```
    c<=a or b;
  end;

 configuration or_2_conf of or_2 is
  for or_2
  end for;
 end or_2_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity and_2 is

   port( a, b : IN std_logic;
         c    : out std_logic);
 end;

architecture and_2 of and_2 is
  begin
    c<=a and b;
  end;

 configuration and_2_conf of and_2 is
  for and_2
  end for;
 end and_2_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity nand_4 is

   port( a, b, c, d : IN std_logic;
         e    : out std_logic);
 end;

architecture nand_4 of nand_4 is
  begin
    e<=not (a and ( b and ( c and d)));
  end;

 configuration nand_4_conf of nand_4 is
  for nand_4
  end for;
 end nand_4_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity and_3_b2 is

   port( a, b, c : IN std_logic;
         d    : out std_logic);
 end;

architecture and_3_b2 of and_3_b2 is
  begin
    d<=a and ((not b) and (not c));
  end;

 configuration and_3_b2_conf of and_3_b2 is
  for and_3_b2
  end for;
 end and_3_b2_conf;

 -----

library ieee;
use ieee.std_logic_1164.all;

entity and_3_b1 is

   port( a, b, c : IN std_logic;
         d    : out std_logic);
 end;
```

```
architecture and_3_b1 of and_3_b1 is
  begin
    d<=a and ( b and (not c));
  end;

 configuration and_3_b1_conf of and_3_b1 is
  for and_3_b1
  end for;
 end and_3_b1_conf;

 -----
library ieee;
use ieee.std_logic_1164.all;

entity re_man is
    port( reset, clk, first_rst, second_rst, abilitaz, re: IN std_logic;
          read_en      : out std_logic);
  end;

architecture re_man of re_man is
  signal n_next, n_out: std_logic;
  begin
   process(clk, reset)
    begin
     if reset='0' then
              n_out<='0';
     elsif ( clk'event and clk='1' ) then
              n_out<=n_next;
     end if;
    end process;

  n_next<=(n_out or re) and ((not first_rst) and ( not second_rst));
  read_en<=not(n_out and (abilitaz and (not second_rst)));
  end;

configuration re_man_conf of re_man is
 for re_man
 end for;
end re_man_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity edge_det is
    port( reset, clk, data_in: IN std_logic;
          data_out      : out std_logic);
  end;


architecture edge_det of edge_det is
  signal q_next, q_out: std_logic;
  begin
   process(clk, reset)
    begin
     if reset='0' then
              q_out<='0';
     elsif ( clk'event and clk='1' ) then
              q_out<=q_next;
     end if;
    end process;

  q_next<=data_in;
  data_out<=q_out and (not data_in);
  end;

configuration edge_det_conf of edge_det is
 for edge_det
 end for;
end edge_det_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity oe_2re is
    port( reset, clk, oe0, oe1, oe2, oe3, readen, addr0, addr1: IN std_logic;
          re0, re1, re2, re3: out std_logic);
  end;
```

```
architecture oe_2re of oe_2re is
  signal q1_next, q1_out, q2_next, q2_out, enab: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q1_out<='0';
              q2_out<='0';
      elsif ( clk'event and clk='1' ) then
              q1_out<=q1_next;
              q2_out<=q2_next;
      end if;
     end process;

   q1_next<=not (oe0 and( oe1 and ( oe2 and ( oe3))));
   q2_next<=q1_out;
   enab<=q2_out and readen;

demux1: process (enab, addr0, addr1)
  begin
 if enab='1' then
  if (addr0='0' and addr1='0') then
      re0<=enab;
      re1<='0';
      re2<='0';
      re3<='0';
  elsif (addr0='1' and addr1='0') then
      re0<='0';
      re1<=enab;
      re2<='0';
      re3<='0';
  elsif (addr0='0' and addr1='1') then
      re0<='0';
      re1<='0';
      re2<=enab;
      re3<='0';
  else
      re0<='0';
      re1<='0';
      re2<='0';
      re3<=enab;
    end if;
 else
      re0<='0';
      re1<='0';
      re2<='0';
      re3<='0';
 end if;
   end process;

   end;

configuration oe_2re_conf of oe_2re is
 for oe_2re
 end for;
end oe_2re_conf;

-----
library ieee;
use ieee.std_logic_1164.all;

 entity enable_man2 is
  port( rst, o_e_0, o_e_1, o_e_2, o_e_3, empty_f, addr_0, addr_1, abilit, read_en, second_reset, en_clock : IN std_logic;
        re_0, re_1, re_2, re_3: out std_logic);
 end;

 architecture enable_man2 of enable_man2 is

  component oe_2re is
    port( reset, clk, oe0, oe1, oe2, oe3, readen, addr0, addr1: IN std_logic;
          re0, re1, re2, re3: out std_logic);
  end component;

component edge_det is
    port( reset, clk, data_in: IN std_logic;

          data_out      : out std_logic);
  end component;

component re_man is
    port( reset, clk, first_rst, second_rst, abilitaz, re: IN std_logic;
          read_en       : out std_logic);
```

```
  end component;


signal re_n0, re_n1, re_n2, re_n3, first_reset: std_logic;

 begin

  comp_a : oe_2re PORT MAP (reset=>rst, clk=>en_clock, oe0=>o_e_0, oe1=>o_e_1, oe2=>o_e_2, oe3=>o_e_3,
readen=>read_en, addr0=>addr_0, addr1=>addr_1, re0=>re_n0, re1=>re_n1, re2=>re_n2, re3=>re_n3);
  comp_b : edge_det PORT MAP (reset=>rst, clk=>en_clock, data_in=>empty_f, data_out=>first_reset);
  comp_c : re_man PORT MAP (reset=>rst, clk=>en_clock, first_rst=>first_reset, second_rst=>second_reset, abilitaz=>abilit,
re=>re_n0, read_en=>re_0);
  comp_d : re_man PORT MAP (reset=>rst, clk=>en_clock, first_rst=>first_reset, second_rst=>second_reset, abilitaz=>abilit,
re=>re_n1, read_en=>re_1);
  comp_e : re_man PORT MAP (reset=>rst, clk=>en_clock, first_rst=>first_reset, second_rst=>second_reset, abilitaz=>abilit,
re=>re_n2, read_en=>re_2);
  comp_f : re_man PORT MAP (reset=>rst, clk=>en_clock, first_rst=>first_reset, second_rst=>second_reset, abilitaz=>abilit,
re=>re_n3, read_en=>re_3);

end;

 configuration enable_man2_conf of enable_man2 is

   for enable_man2

    for comp_a: oe_2re use configuration work.oe_2re_conf;
    end for;

    for comp_b: edge_det use configuration work.edge_det_conf;
    end for;

    for comp_c: re_man use configuration work.re_man_conf;
    end for;


    for comp_d: re_man use configuration work.re_man_conf;
    end for;

    for comp_e: re_man use configuration work.re_man_conf;
    end for;

    for comp_f: re_man use configuration work.re_man_conf;
    end for;

  end for;
 end enable_man2_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity and_2_b1 is

    port( a, b : IN std_logic;
          c    : out std_logic);
  end;

architecture and_2_b1 of and_2_b1 is
   begin
     c<=a and ( not b);
   end;

 configuration and_2_b1_conf of and_2_b1 is
   for and_2_b1
   end for;
 end and_2_b1_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity or4_b1 is

    port( a, b, c, d : IN std_logic;
          e    : out std_logic);
  end;

architecture or4_b1 of or4_b1 is
   begin
     e<=a or (b or (c or (not d)));
```

```
  end;

 configuration or4_b1_conf of or4_b1 is
   for or4_b1
   end for;
 end or4_b1_conf;

library ieee;
use ieee.std_logic_1164.all;

entity or3_b1 is

    port( a, b, c : IN std_logic;
          d    : out std_logic);
  end;

architecture or3_b1 of or3_b1 is
   begin
     d<=a or (b or (not c));
   end;

 configuration or3_b1_conf of or3_b1 is
   for or3_b1
   end for;
 end or3_b1_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity or2_b1 is

    port( a, b : IN std_logic;
          c    : out std_logic);
  end;

architecture or2_b1 of or2_b1 is
   begin
     c<=a or (not b);
   end;

 configuration or2_b1_conf of or2_b1 is
   for or2_b1
   end for;
 end or2_b1_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity or1_b1 is

    port( a : IN std_logic;
          b    : out std_logic);
  end;

architecture or1_b1 of or1_b1 is
   begin
     b<= not a;
   end;

 configuration or1_b1_conf of or1_b1 is
   for or1_b1
   end for;
 end or1_b1_conf;


-----

library ieee;
use ieee.std_logic_1164.all;

 entity ef_mux is
  port ( add0, add1, ef0, ef1, ef2, ef3: IN std_logic;
               empty_f: OUT std_logic);
  end ef_mux;

  architecture ef_mux of ef_mux is
   begin
```

```
mux1: process (add0, add1, ef0, ef1, ef2, ef3)
 begin
 if (add0='0' and add1='0') then
    empty_f<=ef0;
 elsif (add0='1' and add1='0') then
    empty_f<=ef1;
 elsif (add0='0' and add1='1') then
    empty_f<=ef2;
 else
    empty_f<=ef3;
 end if;
  end process;
 end;

configuration ef_mux_conf of ef_mux is
 for ef_mux
 end for;
end ef_mux_conf;


-----

library ieee;
use ieee.std_logic_1164.all;

 entity test_4ef2 is
  port ( reset, ef_clk, ef, end_cm, cm_head, err_time, epf, enable: IN std_logic;
        read_en, err_rst, time_en, end_cycle, err, miss_fifo: OUT std_logic);
  end test_4ef2;

  architecture test_4ef2 of test_4ef2 is
  type stato is (
                          s0, s1, s2, s3, s4, s5, s6, s7, s8);
  signal state, next_state : stato;
  signal read_en_next, err_rst_next, time_en_next, end_cycle_next, err_next, miss_fifo_next: std_logic;
  begin
  reg: process(ef_clk, reset)
  begin
  if reset='0' then
   state<=s0;
   read_en<='0';
   err_rst<='1';
   time_en<='0';
   end_cycle<='0';
   err<='0';
   miss_fifo<='0';
  elsif ( ef_clk'event and ef_clk='1' ) then
   state<=next_state;
   read_en<=not(read_en_next);
   err_rst<=err_rst_next;
   time_en<=time_en_next;
   end_cycle<=end_cycle_next;
   err<=err_next;
   miss_fifo<=miss_fifo_next;
  end if;
 end process reg;

 comb: process(ef, end_cm, cm_head, err_time, epf, enable, state)
 begin
  case state is
   when s0=>
    if enable='0' then
     next_state<=s0;
     read_en_next<='1';
     err_rst_next<='1';
     time_en_next<='0';
     end_cycle_next<='0';
     err_next<='0';
     miss_fifo_next<='0';
    else
     if ef='1' then
      next_state<=s1;
      read_en_next<='0';
      err_rst_next<='1';
      time_en_next<='0';
      end_cycle_next<='0';
      err_next<='0';
      miss_fifo_next<='0';
     else
      next_state<=s2;
      read_en_next<='1';
      err_rst_next<='0';
```

```
    time_en_next<='1';
    end_cycle_next<='0';
    err_next<='0';
    miss_fifo_next<='1';
   end if;
  end if;
 when s1=>
  if end_cm='1' then
   next_state<=s4;
   read_en_next<='0';
   err_rst_next<='1';
   time_en_next<='0';
   end_cycle_next<='0';
   err_next<='0';
   miss_fifo_next<='0';
  else
   if ef='1' then
    next_state<=s1;
    read_en_next<='0';
    err_rst_next<='1';
    time_en_next<='0';
    end_cycle_next<='0';
    err_next<='0';
    miss_fifo_next<='0';
   else
    next_state<=s3;
    read_en_next<='0';
    err_rst_next<='0';
    time_en_next<='1';
    end_cycle_next<='0';
    err_next<='0';
    miss_fifo_next<='0';
   end if;
  end if;
 when s2=>
  if err_time='1' then
   next_state<=s7;
   read_en_next<='1';
   err_rst_next<='1';
   time_en_next<='0';
   end_cycle_next<='0';
   err_next<='1';
   miss_fifo_next<='0';
  else
   if ef='1' then
    next_state<=s1;
    read_en_next<='0';
    err_rst_next<='1';
    time_en_next<='0';
    end_cycle_next<='0';
    err_next<='0';
    miss_fifo_next<='0';
   else
     next_state<=s2;
     read_en_next<='1';
     err_rst_next<='0';
     time_en_next<='1';
     end_cycle_next<='0';
     err_next<='0';
     miss_fifo_next<='1';
   end if;
  end if;
 when s3=>
  if err_time='1' then
   next_state<=s7;
   read_en_next<='1';
   err_rst_next<='1';
   time_en_next<='0';
   end_cycle_next<='0';
   err_next<='1';
   miss_fifo_next<='0';
  else
   if ef='1' then
    next_state<=s1;
    read_en_next<='0';
    err_rst_next<='1';
    time_en_next<='0';
    end_cycle_next<='0';
    err_next<='0';
    miss_fifo_next<='0';
   else
    next_state<=s3;
```

```
   read_en_next<='1';
   err_rst_next<='0';
   time_en_next<='1';
   end_cycle_next<='0';
   err_next<='0';
   miss_fifo_next<='0';
  end if;
 end if;
when s4=>
 if ef='0' then
  next_state<=s6;
  read_en_next<='1';
  err_rst_next<='0';
  time_en_next<='1';
  end_cycle_next<='0';
  err_next<='0';
  miss_fifo_next<='0';
 else
   if epf='1' then
    next_state<=s8;
    read_en_next<='1';
    err_rst_next<='1';
    time_en_next<='0';
    end_cycle_next<='1';
    err_next<='0';
    miss_fifo_next<='0';
   else
    if cm_head='1' then
     next_state<=s1;
     read_en_next<='0';
     err_rst_next<='1';
     time_en_next<='0';
     end_cycle_next<='0';
     err_next<='0';
     miss_fifo_next<='0';
    else
     next_state<=s5;
     read_en_next<='0';
     err_rst_next<='0';
     time_en_next<='1';
     end_cycle_next<='0';
     err_next<='0';
      miss_fifo_next<='0';
    end if;
   end if;
  end if;
  when s5=>
   if err_time='1' then
    next_state<=s7;
    read_en_next<='1';
    err_rst_next<='1';
    time_en_next<='0';
    end_cycle_next<='0';
    err_next<='1';
    miss_fifo_next<='0';
   else
    if ( epf='0' and cm_head='0') then
     next_state<=s5;
     read_en_next<='0';
     err_rst_next<='0';
     time_en_next<='1';
     end_cycle_next<='0';
     err_next<='0';
     miss_fifo_next<='0';
    else
     if cm_head='1' then
      next_state<=s1;
      read_en_next<='0';
      err_rst_next<='1';
      time_en_next<='0';
      end_cycle_next<='0';
      err_next<='0';
      miss_fifo_next<='0';
     else
      next_state<=s8;
      read_en_next<='1';
      err_rst_next<='1';
      time_en_next<='0';
      end_cycle_next<='1';
      err_next<='0';
      miss_fifo_next<='0';
     end if;
```

```
       end if;
      end if;
     when s6=>
      if err_time='1' then
       next_state<=s7;
       read_en_next<='1';
       err_rst_next<='1';
       time_en_next<='0';
       end_cycle_next<='0';
       err_next<='1';
       miss_fifo_next<='0';
      else
       if ef='1' then
        next_state<=s4;
        read_en_next<='0';
        err_rst_next<='1';
        time_en_next<='0';
        end_cycle_next<='0';
        err_next<='0';
        miss_fifo_next<='0';
       else
        next_state<=s6;
        read_en_next<='1';
        err_rst_next<='0';
        time_en_next<='1';
        end_cycle_next<='0';
        err_next<='0';
        miss_fifo_next<='0';
       end if;
      end if;
     when s7=>
        next_state<=s8;
        read_en_next<='1';
        err_rst_next<='1';
        time_en_next<='0';
        end_cycle_next<='1';
        err_next<='0';
        miss_fifo_next<='0';
      when s8=>
        next_state<=s0;
        read_en_next<='1';
        err_rst_next<='1';
        time_en_next<='0';
        end_cycle_next<='0';
        err_next<='0';
        miss_fifo_next<='0';
      end case;
     end process comb;
    end test_4ef2;

  configuration test_4ef2_conf of test_4ef2 is
    for test_4ef2
    end for;
   end test_4ef2_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

 entity fifo_select is
  port ( reset, clk, en, f0_yes, f1_yes, f2_yes, f3_yes, end_cycle: IN std_logic;
             addr0, addr1, enable, end_tour: OUT std_logic );
  end fifo_select;

  architecture fifo_select of fifo_select is
  type stato is (
                          s0, s1, s2, s3, s4, s5);
  signal state, next_state : stato;
  signal addr0_next, addr1_next, enable_next, end_tour_next: std_logic;
  begin
  reg: process(clk, reset)
  begin
  if reset='0' then
   state<=s0;
   addr0<='0';
   addr1<='0';
   enable<='0';
   end_tour<='0';
  elsif ( clk'event and clk='1' ) then
    state<=next_state;
    addr0<=addr0_next;
```

```
  addr1<=addr1_next;
  enable<=enable_next;
  end_tour<=end_tour_next;
 end if;
end process reg;

comb: process(end_cycle, en, f0_yes, f1_yes, f2_yes, f3_yes, state)
begin
 case state is
  when s0=>
   if(en='1' and f0_yes='1') then
    next_state<=s1;
    addr0_next<='0';
    addr1_next<='0';
    enable_next<='1';
    end_tour_next<='0';
   elsif(en='1' and f1_yes='1') then
     next_state<=s2;
     addr0_next<='1';
     addr1_next<='0';
     enable_next<='1';
     end_tour_next<='0';
    elsif(en='1' and f2_yes='1') then
      next_state<=s3;
      addr0_next<='0';
      addr1_next<='1';
      enable_next<='1';
      end_tour_next<='0';
     elsif(en='1' and f3_yes='1') then
       next_state<=s4;
       addr0_next<='1';
       addr1_next<='1';
       enable_next<='1';
       end_tour_next<='0';
      else
       next_state<=s5;
       addr0_next<='0';
       addr1_next<='0';
       enable_next<='0';
       end_tour_next<='1';
      end if;
  when s1=>
   if end_cycle='0' then
    next_state<=s1;
    addr0_next<='0';
    addr1_next<='0';
    enable_next<='1';
    end_tour_next<='0';
   else
    if(end_cycle='1' and f1_yes='1') then
     next_state<=s2;
     addr0_next<='1';
     addr1_next<='0';
     enable_next<='1';
     end_tour_next<='0';
    else
     if(end_cycle='1' and f2_yes='1') then
      next_state<=s3;
      addr0_next<='0';
      addr1_next<='1';
      enable_next<='1';
      end_tour_next<='0';
     else
      if(end_cycle='1' and f3_yes='1') then
       next_state<=s4;
       addr0_next<='1';
       addr1_next<='1';
       enable_next<='1';
       end_tour_next<='0';
      else
       next_state<=s5;
       addr0_next<='0';
       addr1_next<='0';
       enable_next<='0';
       end_tour_next<='1';
      end if;
     end if;
    end if;
   end if;
  when s2=>
   if end_cycle='0' then
    next_state<=s2;
```

```
      addr0_next<='1';
      addr1_next<='0';
      enable_next<='1';
      end_tour_next<='0';
     else
      if(end_cycle='1' and f2_yes='1') then
       next_state<=s3;
       addr0_next<='0';
       addr1_next<='1';
       enable_next<='1';
       end_tour_next<='0';
      else
       if(end_cycle='1' and f3_yes='1') then
        next_state<=s4;
        addr0_next<='1';
        addr1_next<='1';
        enable_next<='1';
        end_tour_next<='0';
       else
        next_state<=s5;
        addr0_next<='0';
        addr1_next<='0';
        enable_next<='0';
        end_tour_next<='1';
       end if;
      end if;
     end if;
    when s3=>
     if end_cycle='0' then
      next_state<=s3;
      addr0_next<='0';
      addr1_next<='1';
      enable_next<='1';
      end_tour_next<='0';
    else
       if(end_cycle='1' and f3_yes='1') then
        next_state<=s4;
        addr0_next<='1';
        addr1_next<='1';
        enable_next<='1';
        end_tour_next<='0';
       else
        next_state<=s5;
        addr0_next<='0';
        addr1_next<='0';
        enable_next<='0';
        end_tour_next<='1';
       end if;
      end if;
     when s4=>
      if end_cycle='0' then
        next_state<=s4;
        addr0_next<='1';
        addr1_next<='1';
        enable_next<='1';
        end_tour_next<='0';
       else
        next_state<=s5;
        addr0_next<='0';
        addr1_next<='0';
        enable_next<='0';
        end_tour_next<='1';
       end if;
      when s5=>
       next_state<=s0;
       addr0_next<='0';
       addr1_next<='0';
       enable_next<='0';
       end_tour_next<='0';
      end case;
     end process comb;
    end fifo_select;

  configuration fifo_select_conf of fifo_select is
    for fifo_select
    end for;
  end fifo_select_conf;

-----
library ieee;
use ieee.std_logic_1164.all;
```

```
entity fifo_letta_man is
    port( reset, clk, f_letta_rst, end_cycle, oe: IN std_logic;
          fifo_letta      : inout std_logic);
  end;

architecture fifo_letta_man of fifo_letta_man is
  signal n_next: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              fifo_letta<='0';
      elsif ( clk'event and clk='1' ) then
              fifo_letta<=n_next;
      end if;
     end process;

   n_next<=((fifo_letta or (end_cycle and ( not oe))) and (not f_letta_rst));
   end;

configuration fifo_letta_man_conf of fifo_letta_man is
 for fifo_letta_man
 end for;
end fifo_letta_man_conf;


-----

library ieee;
use ieee.std_logic_1164.all;

entity f_rst is
    port( reset, clk, end_tour: IN std_logic;
          fifo_letta_rst: out std_logic);
  end;

architecture f_rst of f_rst is
  signal q1_next, q1_out, q2_next, q2_out: std_logic;
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q1_out<='0';
              q2_out<='0';
      elsif ( clk'event and clk='1' ) then
              q1_out<=q1_next;
              q2_out<=q2_next;
      end if;
     end process;

   q1_next<=end_tour;
   q2_next<=q1_out;
   fifo_letta_rst<=(not q2_out) and q1_out;
   end;

configuration f_rst_conf of f_rst is
 for f_rst
 end for;
end f_rst_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

entity or_2_b1 is

    port( a, b : IN std_logic;
          c    : out std_logic);
  end;

architecture or_2_b1 of or_2_b1 is
   begin
     c<=a or (not b);
   end;

 configuration or_2_b1_conf of or_2_b1 is
   for or_2_b1
   end for;
 end or_2_b1_conf;

-----------
```

```
library ieee;
use ieee.std_logic_1164.all;

 entity f_letta_man2 is
  port( f_reset, o_e_0, o_e_1, o_e_2, o_e_3, end_cy, end_tur, yes_fifo0, yes_fifo1, yes_fifo2, yes_fifo3, clok: IN std_logic;
        fifoletta0, fifoletta1, fifoletta2, fifoletta3: out std_logic);
 end;

 architecture f_letta_man2 of f_letta_man2 is


  component fifo_letta_man is
    port( reset, clk, f_letta_rst, end_cycle, oe: IN std_logic;
          fifo_letta      : inout std_logic);
  end component;


component or_2_b1 is
    port( a, b : IN std_logic;
          c    : out std_logic);
  end component;


component f_rst is
    port( reset, clk, end_tour: IN std_logic;
          fifo_letta_rst: out std_logic);
  end component;


signal f_letta_reset, f_letta0, f_letta1, f_letta2, f_letta3: std_logic;

 begin

  comp_a : fifo_letta_man PORT MAP (reset=>f_reset, clk=>clok, f_letta_rst=>f_letta_reset, end_cycle=>end_cy, oe=>o_e_0,
fifo_letta=>f_letta0);
  comp_b : fifo_letta_man PORT MAP (reset=>f_reset, clk=>clok, f_letta_rst=>f_letta_reset, end_cycle=>end_cy, oe=>o_e_1,
fifo_letta=>f_letta1);
  comp_c : fifo_letta_man PORT MAP (reset=>f_reset, clk=>clok, f_letta_rst=>f_letta_reset, end_cycle=>end_cy, oe=>o_e_2,
fifo_letta=>f_letta2);
  comp_d : fifo_letta_man PORT MAP (reset=>f_reset, clk=>clok, f_letta_rst=>f_letta_reset, end_cycle=>end_cy, oe=>o_e_3,
fifo_letta=>f_letta3);
  comp_e : or_2_b1 PORT MAP (a=>f_letta0, b=>yes_fifo0, c=>fifoletta0);
  comp_f : or_2_b1 PORT MAP (a=>f_letta1, b=>yes_fifo1, c=>fifoletta1);
  comp_g : or_2_b1 PORT MAP (a=>f_letta2, b=>yes_fifo2, c=>fifoletta2);
  comp_h : or_2_b1 PORT MAP (a=>f_letta3, b=>yes_fifo3, c=>fifoletta3);
  comp_i : f_rst PORT MAP (reset=>f_reset, clk=>clok, end_tour=>end_tur, fifo_letta_rst=>f_letta_reset);


end;

 configuration f_letta_man2_conf of f_letta_man2 is

   for f_letta_man2

    for comp_a: fifo_letta_man use configuration work.fifo_letta_man_conf;
    end for;

    for comp_b: fifo_letta_man use configuration work.fifo_letta_man_conf;
    end for;

    for comp_c: fifo_letta_man use configuration work.fifo_letta_man_conf;
    end for;

    for comp_d: fifo_letta_man use configuration work.fifo_letta_man_conf;
    end for;

    for comp_e: or_2_b1 use configuration work.or_2_b1_conf;
    end for;

    for comp_f: or_2_b1 use configuration work.or_2_b1_conf;
    end for;

    for comp_g: or_2_b1 use configuration work.or_2_b1_conf;
    end for;

    for comp_h: or_2_b1 use configuration work.or_2_b1_conf;
    end for;

    for comp_i: f_rst use configuration work.f_rst_conf;
    end for;
```

```
 end for;
 end f_letta_man2_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

 entity struct_select_fifo is
  port( ef_0, ef_1, ef_2, ef_3, man_en, cmhead, cmend, timeerr, end_pf, rst, ef_clock : IN std_logic;
        empty_fifo, addr_0, addr_1: INOUT std_logic;
        oe_0, oe_1, oe_2, oe_3, endcycle: INOUT std_logic;
        fifomask0, fifomask1, fifomask2, fifomask3 : IN std_logic;
        error, rd_en, error_rst, count_en, missing_fifo: out std_logic);
 end;

 architecture struct_select_fifo of struct_select_fifo is

  component or4_b1 is
    port( a, b, c, d : IN std_logic;
          e    : out std_logic);
  end component;

component or3_b1 is
    port( a, b, c : IN std_logic;
          d    : out std_logic);
  end component;


component and_2_b1 is
    port( a, b : IN std_logic;
          c    : out std_logic);
  end component;

component or_2_b1 is
    port( a, b : IN std_logic;
          c    : out std_logic);
  end component;


component or1_b1 is
    port( a : IN std_logic;
          b    : out std_logic);
  end component;

component ef_mux is
  port ( add0, add1, ef0, ef1, ef2, ef3: IN std_logic;
               empty_f: OUT std_logic);
  end component;

component fifo_select is
port ( reset, clk, en, f0_yes, f1_yes, f2_yes, f3_yes, end_cycle: IN std_logic;
             addr0, addr1, enable, end_tour: OUT std_logic );
end component;

component test_4ef2 is
port ( reset, ef_clk, ef, end_cm, cm_head, err_time, epf, enable : IN std_logic ;
      read_en, err_rst, time_en, end_cycle, err, miss_fifo: OUT std_logic);
end component;

component f_letta_man2 is
  port( f_reset, o_e_0, o_e_1, o_e_2, o_e_3, end_cy, end_tur, yes_fifo0, yes_fifo1, yes_fifo2, yes_fifo3, clok: IN std_logic;
        fifoletta0, fifoletta1, fifoletta2, fifoletta3: out std_logic);
 end component;



signal endtour, test_enable, fifo_letta0, fifo_letta1, fifo_letta2, fifo_letta3, n0, n1, n2, n3: std_logic;

 begin

  comp_a : or4_b1 PORT MAP (a=>n0, b=>n1, c=>n2, d=>n3, e=>oe_3);
  comp_b : or3_b1 PORT MAP (a=>n0, b=>n1, c=>n2, d=>oe_2);
  comp_c : or_2_b1 PORT MAP (a=>n0, b=>n1, c=>oe_1);
  comp_d : or1_b1 PORT MAP (a=>n0, b=>oe_0);
  comp_e : ef_mux PORT MAP (add0=>addr_0, add1=>addr_1, ef0=>ef_0, ef1=>ef_1, ef2=>ef_2, ef3=>ef_3,
empty_f=>empty_fifo);
  comp_f : fifo_select PORT MAP (reset=>rst, clk=>ef_clock, en=>man_en, f0_yes=>fifomask0, f1_yes=>fifomask1,
f2_yes=>fifomask2, f3_yes=>fifomask3, end_cycle=>endcycle, addr0=>addr_0, addr1=>addr_1, enable=>test_enable,
end_tour=>endtour);
```

```
  comp_g : test_4ef2 PORT MAP (reset=>rst, ef_clk=>ef_clock, ef=>empty_fifo, end_cm=>cmend, cm_head=>cmhead,
err_time=>timeerr, epf=>end_pf, enable=>test_enable, read_en=>rd_en, err_rst=>error_rst, time_en=>count_en,
end_cycle=>endcycle, err=>error, miss_fifo=>missing_fifo);
  comp_h : and_2_b1 PORT MAP (a=>ef_0, b=>fifo_letta0, c=>n0);
  comp_i : and_2_b1 PORT MAP (a=>ef_1, b=>fifo_letta1, c=>n1);
  comp_l : and_2_b1 PORT MAP (a=>ef_2, b=>fifo_letta2, c=>n2);
  comp_m : and_2_b1 PORT MAP (a=>ef_3, b=>fifo_letta3, c=>n3);
  comp_n : f_letta_man2 PORT MAP (f_reset=>rst, o_e_0=>oe_0, o_e_1=>oe_1, o_e_2=>oe_2, o_e_3=>oe_3, end_cy=>endcycle,
end_tur=>endtour, yes_fifo0=>fifomask0, yes_fifo1=>fifomask1, yes_fifo2=>fifomask2, yes_fifo3=>fifomask3, clok=>ef_clock,
fifoletta0=>fifo_letta0, fifoletta1=>fifo_letta1, fifoletta2=>fifo_letta2, fifoletta3=>fifo_letta3);


  end;

 configuration struct_select_fifo_conf of struct_select_fifo is

   for struct_select_fifo

    for comp_a: or4_b1 use configuration work.or4_b1_conf;
    end for;

    for comp_b: or3_b1 use configuration work.or3_b1_conf;
    end for;

    for comp_c: or_2_b1 use configuration work.or_2_b1_conf;
    end for;

    for comp_d: or1_b1 use configuration work.or1_b1_conf;
    end for;

    for comp_e: ef_mux use configuration work.ef_mux_conf;
    end for;

    for comp_f: fifo_select use configuration work.fifo_select_conf;
    end for;

    for comp_g: test_4ef2 use configuration work.test_4ef2_conf;
    end for;

    for comp_h: and_2_b1 use configuration work.and_2_b1_conf;
    end for;

    for comp_i: and_2_b1 use configuration work.and_2_b1_conf;
    end for;

    for comp_l: and_2_b1 use configuration work.and_2_b1_conf;
    end for;

    for comp_m: and_2_b1 use configuration work.and_2_b1_conf;
    end for;

    for comp_n: f_letta_man2 use configuration work.f_letta_man2_conf;
    end for;


  end for;
 end struct_select_fifo_conf;


------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

 entity header_man is
  port ( rst, clk, man_en, bit_31, bit_30, bit_29, bit_28: IN std_logic;
        pad_hd, cm_hd, cm_shd, cm_ft, empty_cm, pad_ft, err, data_ok : OUT std_logic);
 end header_man;

 architecture header_man of header_man is
 type stato is ( s0, s1, s2, s3, s4, s5, s6, s7);
 signal state, next_state : stato;
 signal pad_hd_next, cm_hd_next, cm_shd_next, cm_ft_next, empty_cm_next, pad_ft_next, err_next, data_ok_next: std_logic;

 begin
 reg: process( clk, rst, man_en)
 begin
  if rst='0' then
   state<=s0;
   pad_hd<='0';
   cm_hd<='0';
   cm_shd<='0';
```

```
  cm_ft<='0';
  pad_ft<='1';
  err<='0';
  data_ok<='0';
  empty_cm<='0';
 elsif ( clk'event and clk='1' ) then
  if (man_en='1') then
   state<=next_state;
   pad_hd<=pad_hd_next;
   cm_hd<=cm_hd_next;
   cm_shd<=cm_shd_next;
   cm_ft<=cm_ft_next;
   pad_ft<=pad_ft_next;
   err<=err_next;
   data_ok<=data_ok_next;
   empty_cm<=empty_cm_next;
  end if;
 end if;
end process reg;

comb: process ( bit_31, bit_30, bit_29, bit_28, state)
begin
 case state is
  when s0 =>
   if ( bit_31='0' and bit_30='1' and bit_29='0' and bit_28='1') then
    next_state<=s1;
    pad_hd_next<='1';
    cm_hd_next<='0';
    cm_shd_next<='0';
    cm_ft_next<='0';
    pad_ft_next<='0';
    err_next<='0';
    data_ok_next<='0';
    empty_cm_next<='0';
   else
    next_state<=s0;
    pad_hd_next<='0';
    cm_hd_next<='0';
    cm_shd_next<='0';
    cm_ft_next<='0';
    pad_ft_next<='1';
    err_next<='0';
    data_ok_next<='0';
    empty_cm_next<='0';
   end if;
  when s1 =>
   if ( bit_31='1' and bit_30='1' and bit_29='0' and bit_28='0') then
    next_state<=s2;
    pad_hd_next<='0';
    cm_hd_next<='1';
    cm_shd_next<='0';
    cm_ft_next<='0';
    pad_ft_next<='0';
    err_next<='0';
    data_ok_next<='0';
    empty_cm_next<='0';
   else
    next_state<=s7;
    pad_hd_next<='0';
    cm_hd_next<='0';
    cm_shd_next<='0';
    cm_ft_next<='0';
    pad_ft_next<='0';
    err_next<='1';
    data_ok_next<='0';
    empty_cm_next<='0';
   end if;
  when s2 =>
   if ( bit_31='1' and bit_30='0' and bit_29='0' and bit_28='0') then
    next_state<=s3;
    pad_hd_next<='0';
    cm_hd_next<='0';
    cm_shd_next<='1';
    cm_ft_next<='0';
    pad_ft_next<='0';
    err_next<='0';
    data_ok_next<='0';
    empty_cm_next<='0';
   else
    next_state<=s7;
    pad_hd_next<='0';
    cm_hd_next<='0';
```

```
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='0';
  err_next<='1';
  data_ok_next<='0';
  empty_cm_next<='0';
 end if;
when s3 =>
 if ( bit_31='0' and bit_30='1' and bit_29='0' and bit_28='0') then
  next_state<=s6;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='0';
  err_next<='0';
  data_ok_next<='0';
  empty_cm_next<='1';
 elsif ( bit_31='0' and bit_30='0') then
  next_state<=s4;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='0';
  err_next<='0';
  data_ok_next<='1';
  empty_cm_next<='0';
 else
  next_state<=s7;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='0';
  err_next<='1';
  data_ok_next<='0';
  empty_cm_next<='0';
 end if;
when s4 =>
 if ( bit_31='0' and bit_30='1' and bit_29='0' and bit_28='0') then
  next_state<=s5;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='1';
  pad_ft_next<='0';
  err_next<='0';
  data_ok_next<='0';
  empty_cm_next<='0';
 elsif ( bit_31='0' and bit_30='0') then
  next_state<=s4;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='0';
  err_next<='0';
  data_ok_next<='1';
  empty_cm_next<='0';
 else
  next_state<=s7;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='0';
  err_next<='1';
  data_ok_next<='0';
  empty_cm_next<='0';
 end if;
when s5 =>
 if ( bit_31='0' and bit_30='1' and bit_29='1' and bit_28='1') then
  next_state<=s0;
  pad_hd_next<='0';
  cm_hd_next<='0';
  cm_shd_next<='0';
  cm_ft_next<='0';
  pad_ft_next<='1';
  err_next<='0';
  data_ok_next<='0';
  empty_cm_next<='0';
```

```
    elsif ( bit_31='1' and bit_30='1' and bit_29='0' and bit_28='0') then
     next_state<=s2;
     pad_hd_next<='0';
     cm_hd_next<='1';
     cm_shd_next<='0';
     cm_ft_next<='0';
     pad_ft_next<='0';
     err_next<='0';
     data_ok_next<='0';
     empty_cm_next<='0';
    else
     next_state<=s7;
     pad_hd_next<='0';
     cm_hd_next<='0';
     cm_shd_next<='0';
     cm_ft_next<='0';
     pad_ft_next<='0';
     err_next<='1';
     data_ok_next<='0';
     empty_cm_next<='0';
    end if;
   when s6 =>
    if ( bit_31='0' and bit_30='1' and bit_29='1' and bit_28='1') then
     next_state<=s0;
     pad_hd_next<='0';
     cm_hd_next<='0';
     cm_shd_next<='0';
     cm_ft_next<='0';
     pad_ft_next<='1';
     err_next<='0';
     data_ok_next<='0';
     empty_cm_next<='0';
    else
     next_state<=s7;
     pad_hd_next<='0';
     cm_hd_next<='0';
     cm_shd_next<='0';
     cm_ft_next<='0';
     pad_ft_next<='0';
     err_next<='1';
     data_ok_next<='0';
     empty_cm_next<='0';
    end if;
   when s7 =>
     next_state<=s0;
     pad_hd_next<='0';
     cm_hd_next<='0';
     cm_shd_next<='0';
     cm_ft_next<='0';
     pad_ft_next<='1';
     err_next<='0';
     data_ok_next<='0';
     empty_cm_next<='0';
   end case;
   end process comb;
  end header_man;


configuration header_man_conf of header_man is
     for header_man
     end for;
   end header_man_conf;


-----

library ieee;
use ieee.std_logic_1164.all;

entity or_3 is

    port( a, b, c : IN std_logic;
          d    : out std_logic);
  end;

architecture or_3 of or_3 is
   begin
     d<=a or (b or c);
   end;

 configuration or_3_conf of or_3 is
   for or_3
```

```
  end for;
end or_3_conf;

-------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity data_read_man2 is
 port( reset, f_e0, f_e1, f_e2, f_e3, r_data17, r_data16, r_data15, r_data14, enable, time_err, ok_fifo0, ok_fifo1, ok_fifo2,
ok_fifo3, rd_clk: IN std_logic;
      validdata, oen0, oen1, oen2, oen3, addr0, addr1, ren0, ren1, ren2, ren3: INOUT std_logic;
          err, regen, error_reset, counten, missingfifo: out std_logic);
end;

architecture data_read_man2 of data_read_man2 is

 component struct_select_fifo is
 port( ef_0, ef_1, ef_2, ef_3, man_en, cmhead, cmend, timeerr, end_pf, rst, ef_clock : IN std_logic;
      empty_fifo, addr_0, addr_1: INOUT std_logic;
      oe_0, oe_1, oe_2, oe_3, endcycle: INOUT std_logic;
      fifomask0, fifomask1, fifomask2, fifomask3 : IN std_logic;
      error, rd_en, error_rst, count_en, missing_fifo: out std_logic);
end component;

component and_2 is

   port( a, b : IN std_logic;
        c    : out std_logic);
 end component;


component or_2 is

   port( a, b : IN std_logic;
        c    : out std_logic);
 end component;

component or_3 is

   port( a, b, c : IN std_logic;
        d    : out std_logic);
 end component;

component header_man is
 port ( rst, clk, man_en, bit_31, bit_30, bit_29, bit_28: IN std_logic;
       pad_hd, cm_hd, cm_shd, cm_ft, empty_cm, pad_ft, err, data_ok : OUT std_logic);
end component;

component enable_man2 is
 port( rst, o_e_0, o_e_1, o_e_2, o_e_3, empty_f, addr_0, addr_1, abilit, read_en, second_reset, en_clock : IN std_logic;
      re_0, re_1, re_2, re_3: out std_logic);
end component;


component other_net is
 port( rst, ren_0, ren_1, ren_2, ren_3, o_n_clk, read_en, fifo_e, padfooter, padheader, cmhead, cmshead, cmfooter, okdata: IN
std_logic;
      padft_up, headeren: INOUT std_logic;
      register_en, valid_data: out std_logic);
end component;


signal end_cycle, pad_header, cm_head, cm_shead, cm_empty, cm_footer, pad_footer, pad_ft_up, fifo_empty, read_enable,
fifo_err, data_err, ok_data, man_enable, second_rst, header_en: std_logic;

begin

 comp_a : struct_select_fifo PORT MAP (rst=>reset, ef_clock=>rd_clk, ef_0=>f_e0, ef_1=>f_e1, ef_2=>f_e2, ef_3=>f_e3,
man_en=>enable, cmhead=>cm_head, cmend=>cm_footer, timeerr=>time_err, end_pf=>pad_ft_up, empty_fifo=>fifo_empty,
fifomask0=>ok_fifo0, fifomask1=>ok_fifo1, fifomask2=>ok_fifo2, fifomask3=>ok_fifo3, addr_0=>addr0, addr_1=>addr1,
oe_0=>oen0, oe_1=>oen1, oe_2=>oen2, oe_3=>oen3, error=>fifo_err, rd_en=>read_enable, error_rst=>error_reset,
count_en=>counten, endcycle=>end_cycle, missing_fifo=>missingfifo);
 comp_b_dataread : and_2 PORT MAP (a=>validdata, b=>header_en, c=>man_enable);
 comp_c : or_3 PORT MAP (a=>cm_footer, b=>end_cycle, c=>pad_ft_up, d=>second_rst);
 comp_d : header_man PORT MAP (rst=>reset, clk=>rd_clk, man_en=>man_enable, bit_31=>r_data17, bit_30=>r_data16,
bit_29=>r_data15, bit_28=>r_data14, pad_hd=>pad_header, cm_hd=>cm_head, cm_shd=>cm_shead, cm_ft=>cm_footer,
empty_cm=>cm_empty, pad_ft=>pad_footer, err=>data_err, data_ok=>ok_data);
 comp_e : enable_man2 PORT MAP (rst=>reset, o_e_0=>oen0, o_e_1=>oen1, o_e_2=>oen2, o_e_3=>oen3,
empty_f=>fifo_empty, addr_0=>addr0, addr_1=>addr1, abilit=>enable, read_en=>read_enable, second_reset=>second_rst,
en_clock=>rd_clk, re_0=>ren0, re_1=>ren1, re_2=>ren2, re_3=>ren3);
```

```
  comp_f : other_net PORT MAP (rst=>reset, ren_0=>ren0, ren_1=>ren1, ren_2=>ren2, ren_3=>ren3, o_n_clk=>rd_clk,
read_en=>read_enable, fifo_e=>fifo_empty, padfooter=>pad_footer, padheader=>pad_header, cmhead=>cm_head,
cmshead=>cm_shead, cmfooter=>cm_footer, okdata=>ok_data, padft_up=>pad_ft_up, headeren=>header_en,
register_en=>regen, valid_data=>validdata);
  comp_g : or_2 PORT MAP (a=>fifo_err, b=>data_err, c=>err);

end;

 configuration data_read_man2_conf of data_read_man2 is

   for data_read_man2

    for comp_a: struct_select_fifo use configuration work.struct_select_fifo_conf;
    end for;

    for comp_b_dataread: and_2 use configuration work.and_2_conf;
    end for;

    for comp_c: or_3 use configuration work.or_3_conf;
    end for;

    for comp_d: header_man use configuration work.header_man_conf;
    end for;

    for comp_e: enable_man2 use configuration work.enable_man2_conf;
    end for;

    for comp_f: other_net use configuration work.other_net_conf;
    end for;

    for comp_g: or_2 use configuration work.or_2_conf;
    end for;


  end for;
 end data_read_man2_conf;

--------------------------

 library ieee;
 use ieee.std_logic_1164.all;
 entity fdce_comp is
    port( reset, clk, clkenn, d_in: IN std_logic;
          q_out      : inout std_logic);
  end;

 architecture fdce_comp of fdce_comp is
 signal d_in_next:std_logic;
   begin
    process(clk, reset, d_in, q_out, clkenn)
     begin
      if reset='0' then
              q_out<='0';
      elsif ( clk'event and clk='1' ) then
              q_out<=d_in_next;
      end if;

     if clkenn='1' then
      d_in_next<=d_in;
     else
      d_in_next<=q_out;
     end if;
    end process;

   end;

 configuration fdce_comp_conf of fdce_comp is
  for fdce_comp
  end for;
 end fdce_comp_conf;

 -----

 library ieee;
 use ieee.std_logic_1164.all;

  entity fdce18_comp is
   port( rst, clken, d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, fdce18_clk: IN std_logic;
        qout0, qout1, qout2, qout3, qout4, qout5, qout6, qout7, qout8, qout9, qout10, qout11, qout12, qout13, qout14, qout15,
qout16, qout17: inout std_logic);
  end;
```

```
architecture fdce18_comp of fdce18_comp is

component fdce_comp is
  port( reset, clk, clkenn, d_in: IN std_logic;
        q_out      : inout std_logic);
end component;

begin

 comp_a0 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d0, q_out=>qout0);
 comp_a1 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d1, q_out=>qout1);
 comp_a2 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d2, q_out=>qout2);
 comp_a3 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d3, q_out=>qout3);
 comp_a4 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d4, q_out=>qout4);
 comp_a5 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d5, q_out=>qout5);
 comp_a6 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d6, q_out=>qout6);
 comp_a7 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d7, q_out=>qout7);
 comp_a8 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d8, q_out=>qout8);
 comp_a9 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d9, q_out=>qout9);
 comp_a10 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d10, q_out=>qout10);
 comp_a11 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d11, q_out=>qout11);
 comp_a12 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d12, q_out=>qout12);
 comp_a13 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d13, q_out=>qout13);
 comp_a14 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d14, q_out=>qout14);
 comp_a15 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d15, q_out=>qout15);
 comp_a16 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d16, q_out=>qout16);

 comp_a17 : fdce_comp PORT MAP (reset=>rst, clk=>fdce18_clk, clkenn=>clken, d_in=>d17, q_out=>qout17);


end;

 configuration fdce18_comp_conf of fdce18_comp is

  for fdce18_comp

   for comp_a0: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a1: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a2: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a3: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a4: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a5: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a6: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a7: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a8: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a9: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a10: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a11: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a12: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a13: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a14: fdce_comp use configuration work.fdce_comp_conf;
   end for;

for comp_a15: fdce_comp use configuration work.fdce_comp_conf;
```

```
    end for;

for comp_a16: fdce_comp use configuration work.fdce_comp_conf;
    end for;

for comp_a17: fdce_comp use configuration work.fdce_comp_conf;
    end for;


  end for;
 end fdce18_comp_conf;

--------

library ieee;
use ieee.std_logic_1164.all;

entity fd_comp is
    port( reset, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end;

architecture fd_comp of fd_comp is
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q_out<='0';
      elsif ( clk'event and clk='1' ) then
              q_out<=d_in;
      end if;

    end process;

   end;

configuration fd_comp_conf of fd_comp is
 for fd_comp
 end for;
end fd_comp_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

 entity shift_reg_box is
  port( highv, reg_enable, rest, din0, din1, din2, din3, din4, din5, din6, din7, din8, din9, din10, din11, din12, din13, din14,
din15, din16, din17, shift_clk: IN std_logic;
       q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17: inout std_logic);
 end;

 architecture shift_reg_box of shift_reg_box is

 component fdce18_comp is
  port( rst, clken, d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, fdce18_clk: IN std_logic;
       qout0, qout1, qout2, qout3, qout4, qout5, qout6, qout7, qout8, qout9, qout10, qout11, qout12, qout13, qout14, qout15,
qout16, qout17: inout std_logic);
 end component;

 component fd_comp is
    port( reset, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end component;

 signal shift_en, int10, int11, int12, int13, int14, int15, int16, int17, int18, int19, int110, int111, int112, int113, int114,
int115, int116, int117, int20, int21, int22, int23, int24, int25, int26, int27, int28, int29, int210, int211, int212, int213, int214,
int215, int216, int217: std_logic;

 begin

  compa0 : fdce18_comp PORT MAP (rst=>rest, fdce18_clk=>shift_clk, clken=>highv, d0=>din0, d1=>din1, d2=>din2,
d3=>din3, d4=>din4, d5=>din5, d6=>din6, d7=>din7, d8=>din8, d9=>din9, d10=>din10, d11=>din11, d12=>din12,
d13=>din13, d14=>din14, d15=>din15, d16=>din16, d17=>din17, qout0=>int10, qout1=>int11, qout2=>int12, qout3=>int13,
qout4=>int14, qout5=>int15, qout6=>int16, qout7=>int17, qout8=>int18, qout9=>int19, qout10=>int110, qout11=>int111,
qout12=>int112, qout13=>int113, qout14=>int114, qout15=>int115, qout16=>int116, qout17=>int117);
  compa1 : fdce18_comp PORT MAP (rst=>rest, fdce18_clk=>shift_clk, clken=>reg_enable, d0=>int10, d1=>int11, d2=>int12,
d3=>int13, d4=>int14, d5=>int15, d6=>int16, d7=>int17, d8=>int18, d9=>int19, d10=>int110, d11=>int111, d12=>int112,
d13=>int113, d14=>int114, d15=>int115, d16=>int116, d17=>int117, qout0=>int20, qout1=>int21, qout2=>int22,
qout3=>int23, qout4=>int24, qout5=>int25, qout6=>int26, qout7=>int27, qout8=>int28, qout9=>int29, qout10=>int210,
qout11=>int211, qout12=>int212, qout13=>int213, qout14=>int214, qout15=>int215, qout16=>int216, qout17=>int217);
```

```
  compa2 : fdce18_comp PORT MAP (rst=>rest, fdce18_clk=>shift_clk, clken=>shift_en, d0=>int20, d1=>int21, d2=>int22,
d3=>int23, d4=>int24, d5=>int25, d6=>int26, d7=>int27, d8=>int28, d9=>int29, d10=>int210, d11=>int211, d12=>int212,
d13=>int213, d14=>int214, d15=>int215, d16=>int216, d17=>int217, qout0=>q0, qout1=>q1, qout2=>q2, qout3=>q3,
qout4=>q4, qout5=>q5, qout6=>q6, qout7=>q7, qout8=>q8, qout9=>q9, qout10=>q10, qout11=>q11, qout12=>q12,
qout13=>q13, qout14=>q14, qout15=>q15, qout16=>q16, qout17=>q17);
  compa3 : fd_comp PORT MAP (reset=>rest, clk=>shift_clk, d_in=>reg_enable, q_out=>shift_en);


end;

 configuration shift_reg_box_conf of shift_reg_box is

   for shift_reg_box

     for compa0: fdce18_comp use configuration work.fdce18_comp_conf;
    end for;

     for compa1: fdce18_comp use configuration work.fdce18_comp_conf;
    end for;

     for compa2: fdce18_comp use configuration work.fdce18_comp_conf;
    end for;

     for compa3: fd_comp use configuration work.fd_comp_conf;
    end for;

   end for;

 end shift_reg_box_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

 entity pack_man_comp is
  port( pack_enable, rest, din0, din1, din2, din3, din4, din5, din6, din7, din8, din9, din10, din11, din12, din13, din14, din15,
din16, din17, addr0, addr1, error, missing_fifo, pack_clk: IN std_logic;
       q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17, q18, q19, q20, q21, q22: inout std_logic);
 end;

 architecture pack_man_comp of pack_man_comp is

 component fdce18_comp is
  port( rst, clken, d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, fdce18_clk: IN std_logic;
       qout0, qout1, qout2, qout3, qout4, qout5, qout6, qout7, qout8, qout9, qout10, qout11, qout12, qout13, qout14, qout15,
qout16, qout17: inout std_logic);
 end component;

 component fd_comp is
    port( reset, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end component;

 begin

  compa0 : fdce18_comp PORT MAP (rst=>rest, fdce18_clk=>pack_clk, clken=>pack_enable, d0=>din0, d1=>din1, d2=>din2,
d3=>din3, d4=>din4, d5=>din5, d6=>din6, d7=>din7, d8=>din8, d9=>din9, d10=>din10, d11=>din11, d12=>din12,
d13=>din13, d14=>din14, d15=>din15, d16=>din16, d17=>din17, qout0=>q0, qout1=>q1, qout2=>q2, qout3=>q3, qout4=>q4,
qout5=>q5, qout6=>q6, qout7=>q7, qout8=>q8, qout9=>q9, qout10=>q10, qout11=>q11, qout12=>q12, qout13=>q13,
qout14=>q14, qout15=>q15, qout16=>q16, qout17=>q17);
  compa1 : fd_comp PORT MAP (reset=>rest, clk=>pack_clk, d_in=>addr0, q_out=>q18);
  compa2 : fd_comp PORT MAP (reset=>rest, clk=>pack_clk, d_in=>addr1, q_out=>q19);
  compa3 : fd_comp PORT MAP (reset=>rest, clk=>pack_clk, d_in=>error, q_out=>q20);
  compa4 : fd_comp PORT MAP (reset=>rest, clk=>pack_clk, d_in=>missing_fifo, q_out=>q21);
  compa5 : fd_comp PORT MAP (reset=>rest, clk=>pack_clk, d_in=>pack_enable, q_out=>q22);


end;

 configuration pack_man_comp_conf of pack_man_comp is

   for pack_man_comp

     for compa0: fdce18_comp use configuration work.fdce18_comp_conf;
    end for;

     for compa1: fd_comp use configuration work.fd_comp_conf;
    end for;

     for compa2: fd_comp use configuration work.fd_comp_conf;
    end for;
```

```
    for compa3: fd_comp use configuration work.fd_comp_conf;
   end for;

   for compa4: fd_comp use configuration work.fd_comp_conf;
   end for;

   for compa5: fd_comp use configuration work.fd_comp_conf;
   end for;

  end for;

end pack_man_comp_conf;

-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter_7bit is
 port (clk, resetn, count_en, count_rst: IN std_logic;
       cout:                     OUT std_logic);
 end;

 architecture counter_7bit of counter_7bit is
 signal count: std_logic_vector(6 downto 0);
 begin
  process (clk, resetn, count_en, count_rst)
  begin
   if resetn='0' then
     count<=(others=>'0');
    elsif clk'event and clk='1' then
     if count_en='1' then
       if count_rst='0' then
         count<=(others=>'0');
        else
         count<=count+1;
        end if;
      end if;
   end if;
  end process;

  cout<='1' when (count=127 and count_en='1') else '0';
 end;

configuration counter_7bit_conf of counter_7bit is
 for counter_7bit
 end for;
end counter_7bit_conf;

-----------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

 entity data_pack_man2_comp is
  port( high_voltage, pack_enable, rst, datain0, datain1, datain2, datain3, datain4, datain5, datain6, datain7, datain8,
datain9, datain10, datain11, datain12, datain13, datain14, datain15, datain16, datain17, add0, add1, err, miss_fifo,
count_enable, counter_rst, pack_man_clk: IN std_logic;
       qout0, qout1, qout2, qout3, qout4, qout5, qout6, qout7, qout8, qout9, qout10, qout11, qout12, qout13, qout14, qout15,
qout16, qout17, qout18, qout19, qout20, qout21, qout22, time_error: inout std_logic);
 end;

 architecture data_pack_man2_comp of data_pack_man2_comp is

 component shift_reg_box is
  port( highv, reg_enable, rest, din0, din1, din2, din3, din4, din5, din6, din7, din8, din9, din10, din11, din12, din13, din14,
din15, din16, din17, shift_clk: IN std_logic;
       q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17: inout std_logic);
 end component;

 component pack_man_comp is
  port( pack_enable, rest, din0, din1, din2, din3, din4, din5, din6, din7, din8, din9, din10, din11, din12, din13, din14, din15,
din16, din17, addr0, addr1, error, missing_fifo, pack_clk: IN std_logic;
       q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17, q18, q19, q20, q21, q22: inout std_logic);
 end component;

  component counter_7bit is
 port (clk, resetn, count_en, count_rst: IN std_logic;
       cout:                     OUT std_logic);
  end component;
```

```
  component fd_comp is
    port( reset, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end component;

 signal int10, int11, int12, int13, int14, int15, int16, int17, int18, int19, int110, int111, int112, int113, int114, int115, int116,
int117, pack_en, intermediate1: std_logic;

 begin

  compa0 : shift_reg_box PORT MAP (highv=> high_voltage, rest=>rst, shift_clk=>pack_man_clk, reg_enable=>pack_enable,
din0=>datain0, din1=>datain1, din2=>datain2, din3=>datain3, din4=>datain4, din5=>datain5, din6=>datain6,
din7=>datain7, din8=>datain8, din9=>datain9, din10=>datain10, din11=>datain11, din12=>datain12, din13=>datain13,
din14=>datain14, din15=>datain15, din16=>datain16, din17=>datain17, q0=>int10, q1=>int11, q2=>int12, q3=>int13,
q4=>int14, q5=>int15, q6=>int16, q7=>int17, q8=>int18, q9=>int19, q10=>int110, q11=>int111, q12=>int112, q13=>int113,
q14=>int114, q15=>int115, q16=>int116, q17=>int117);
  compa1 : pack_man_comp PORT MAP (rest=>rst, pack_clk=>pack_man_clk, pack_enable=>pack_en, addr0=>add0,
addr1=>add1, error=>err, missing_fifo=>miss_fifo, din0=>int10, din1=>int11, din2=>int12, din3=>int13, din4=>int14,
din5=>int15, din6=>int16, din7=>int17, din8=>int18, din9=>int19, din10=>int110, din11=>int111, din12=>int112,
din13=>int113, din14=>int114, din15=>int115, din16=>int116, din17=>int117, q0=>qout0, q1=>qout1, q2=>qout2,
q3=>qout3, q4=>qout4, q5=>qout5, q6=>qout6, q7=>qout7, q8=>qout8, q9=>qout9, q10=>qout10, q11=>qout11, q12=>qout12,
q13=>qout13, q14=>qout14, q15=>qout15, q16=>qout16, q17=>qout17, q18=>qout18, q19=>qout19, q20=>qout20,
q21=>qout21, q22=>qout22);
  compa2 : counter_7bit PORT MAP (clk=>pack_man_clk, resetn=>rst, count_en=>count_enable, count_rst=>counter_rst,
cout=>time_error);
  compa3 : fd_comp PORT MAP (reset=>rst, clk=>pack_man_clk, d_in=>pack_enable, q_out=>intermediate1);
  compa4 : fd_comp PORT MAP (reset=>rst, clk=>pack_man_clk, d_in=>intermediate1, q_out=>pack_en);

end;

 configuration data_pack_man2_comp_conf of data_pack_man2_comp is

   for data_pack_man2_comp

    for compa0: shift_reg_box use configuration work.shift_reg_box_conf;
    end for;

    for compa1: pack_man_comp use configuration work.pack_man_comp_conf;
    end for;

    for compa2: counter_7bit use configuration work.counter_7bit_conf;
    end for;

    for compa3: fd_comp use configuration work.fd_comp_conf;
    end for;

    for compa4: fd_comp use configuration work.fd_comp_conf;
    end for;


   end for;

 end data_pack_man2_comp_conf;

-----------------------------

library ieee;
use ieee.std_logic_1164.all;

 entity fifo_reader_comp is
  port( high_level, frame_enable, rest, ef0, ef1, ef2, ef3, din0, din1, din2, din3, din4, din5, din6, din7, din8, din9, din10, din11,
din12, din13, din14, din15, din16, din17, fifo0, fifo1, fifo2, fifo3, fifo_read_clk: IN std_logic;
       rd_en0, rd_en1, rd_en2, rd_en3, out_en0, out_en1, out_en2, out_en3, data_enable, dout0, dout1, dout2, dout3, dout4,
dout5, dout6, dout7, dout8, dout9, dout10, dout11, dout12, dout13, dout14, dout15, dout16, dout17, dout18, dout19, dout20,
dout21, dout22: inout std_logic);
 end;

 architecture fifo_reader_comp of fifo_reader_comp is

 component data_read_man2 is
  port( reset, f_e0, f_e1, f_e2, f_e3, r_data17, r_data16, r_data15, r_data14, enable, time_err, ok_fifo0, ok_fifo1, ok_fifo2,
ok_fifo3, rd_clk: IN std_logic;
        validdata, oen0, oen1, oen2, oen3, addr0, addr1, ren0, ren1, ren2, ren3: INOUT std_logic;
             err, regen, error_reset, counten, missingfifo: out std_logic);

 end component;


component data_pack_man2_comp is
```

```
  port( high_voltage, pack_enable, rst, datain0, datain1, datain2, datain3, datain4, datain5, datain6, datain7, datain8,
datain9, datain10, datain11, datain12, datain13, datain14, datain15, datain16, datain17, add0, add1, err, miss_fifo,
count_enable, counter_rst, pack_man_clk: IN std_logic;
       qout0, qout1, qout2, qout3, qout4, qout5, qout6, qout7, qout8, qout9, qout10, qout11, qout12, qout13, qout14, qout15,
qout16, qout17, qout18, qout19, qout20, qout21, qout22, time_error: inout std_logic);
 end component;


signal t_err, address0, address1, data_err, en_reg, counter_reset, counter_enable, missing_fifo : std_logic;

 begin

  compa0 : data_read_man2 PORT MAP (reset=>rest, rd_clk=>fifo_read_clk, enable=>frame_enable, f_e0=>ef0, f_e1=>ef1,
f_e2=>ef2, f_e3=>ef3, r_data17=>din17, r_data16=>din16, r_data15=>din15, r_data14=>din14, time_err=>t_err,
ok_fifo0=>fifo0, ok_fifo1=>fifo1, ok_fifo2=>fifo2, ok_fifo3=>fifo3, validdata=>data_enable, oen0=>out_en0, oen1=>out_en1,
oen2=>out_en2, oen3=>out_en3, addr0=>address0, addr1=>address1, ren0=>rd_en0, ren1=>rd_en1, ren2=>rd_en2,
ren3=>rd_en3, err=>data_err, regen=>en_reg, error_reset=>counter_reset, counten=>counter_enable,
missingfifo=>missing_fifo);
  compa1 : data_pack_man2_comp PORT MAP (high_voltage=>high_level, rst=>rest, pack_man_clk=>fifo_read_clk,
pack_enable=>en_reg, add0=>address0, add1=>address1, err=>data_err, miss_fifo=>missing_fifo,
count_enable=>counter_enable, counter_rst=>counter_reset, time_error=>t_err, datain0=>din0, datain1=>din1,
datain2=>din2, datain3=>din3, datain4=>din4, datain5=>din5, datain6=>din6, datain7=>din7, datain8=>din8,
datain9=>din9, datain10=>din10, datain11=>din11, datain12=>din12, datain13=>din13, datain14=>din14, datain15=>din15,
datain16=>din16, datain17=>din17, qout0=>dout0, qout1=>dout1, qout2=>dout2, qout3=>dout3, qout4=>dout4,
qout5=>dout5, qout6=>dout6, qout7=>dout7, qout8=>dout8, qout9=>dout9, qout10=>dout10, qout11=>dout11,
qout12=>dout12, qout13=>dout13, qout14=>dout14, qout15=>dout15, qout16=>dout16, qout17=>dout17, qout18=>dout18,
qout19=>dout19, qout20=>dout20, qout21=>dout21, qout22=>dout22);

end;

 configuration fifo_reader_comp_conf of fifo_reader_comp is

   for fifo_reader_comp

     for compa0: data_read_man2 use configuration work.data_read_man2_conf;
    end for;

     for compa1: data_pack_man2_comp use configuration work.data_pack_man2_comp_conf;
    end for;

   end for;

 end fifo_reader_comp_conf;
library ieee;
use ieee.std_logic_1164.all;

entity ifd_comp is
    port( reset, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end;

architecture ifd_comp of ifd_comp is
   begin
    process(clk, reset)
     begin
      if reset='0' then
              q_out<='0';
      elsif ( clk'event and clk='1' ) then
              q_out<=d_in;
      end if;
     end process;

   end;

configuration ifd_comp_conf of ifd_comp is
 for ifd_comp
 end for;
end ifd_comp_conf;

--------------------

library ieee;
use ieee.std_logic_1164.all;

entity afd_comp is
    port( rst, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end;

architecture afd_comp of afd_comp is
   begin
    process(clk, rst)
```

```
    begin
     if rst='0' then
              q_out<='0';
     elsif ( clk'event and clk='1' ) then
              q_out<=d_in;
     end if;
    end process;

  end;

configuration afd_comp_conf of afd_comp is
 for afd_comp
 end for;
end afd_comp_conf;

-----

library ieee;
use ieee.std_logic_1164.all;

 entity ifd4_comp is
  port( rst, d0, d1, d2, d3, ifd_clk: IN std_logic;
       q0, q1, q2, q3: out std_logic);
 end;

 architecture ifd4_comp of ifd4_comp is

 component ifd_comp is
    port( reset, clk, d_in: IN std_logic;
          q_out      : out std_logic);
  end component;


 begin

  comp_a : ifd_comp PORT MAP (reset=>rst, clk=>ifd_clk, d_in=>d0, q_out=>q0);
  comp_b : ifd_comp PORT MAP (reset=>rst, clk=>ifd_clk, d_in=>d1, q_out=>q1);
  comp_c : ifd_comp PORT MAP (reset=>rst, clk=>ifd_clk, d_in=>d2, q_out=>q2);
  comp_d : ifd_comp PORT MAP (reset=>rst, clk=>ifd_clk, d_in=>d3, q_out=>q3);

end;

 configuration ifd4_comp_conf of ifd4_comp is

   for ifd4_comp

    for comp_a: ifd_comp use configuration work.ifd_comp_conf;
    end for;

    for comp_b: ifd_comp use configuration work.ifd_comp_conf;
    end for;

    for comp_c: ifd_comp use configuration work.ifd_comp_conf;
    end for;

    for comp_d: ifd_comp use configuration work.ifd_comp_conf;
    end for;

  end for;
 end ifd4_comp_conf;

-------------------------

library ieee;
use ieee.std_logic_1164.all;

entity ifdx_comp is
    port( reset, clk, c_en, d_in: IN std_logic;
          q_out      : inout std_logic);
  end;

architecture ifdx_comp of ifdx_comp is
signal d_in_next:std_logic;
   begin
    process(clk, reset, c_en, d_in, q_out)
     begin
      if reset='0' then
              q_out<='0';
      elsif ( clk'event and clk='1' ) then
              q_out<=d_in_next;
      end if;
```

```
    if c_en='1' then
     d_in_next<=d_in;
    else
     d_in_next<=q_out;
    end if;
   end process;

  end;

configuration ifdx_comp_conf of ifdx_comp is
 for ifdx_comp
 end for;
end ifdx_comp_conf;


library ieee;
use ieee.std_logic_1164.all;

 entity ifdx18_comp is
  port( rst, c_e, d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, ifdx_clk: IN std_logic;
       q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17: inout std_logic);
 end;

 architecture ifdx18_comp of ifdx18_comp is

 component ifdx_comp is
   port( reset, clk, c_en, d_in: IN std_logic;
         q_out      : inout std_logic);
 end component;


 begin
  comp_a0 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d0, q_out=>q0);
  comp_a1 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d1, q_out=>q1);
  comp_a2 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d2, q_out=>q2);
  comp_a3 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d3, q_out=>q3);
  comp_a4 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d4, q_out=>q4);
  comp_a5 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d5, q_out=>q5);
  comp_a6 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d6, q_out=>q6);
  comp_a7 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d7, q_out=>q7);
  comp_a8 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d8, q_out=>q8);
  comp_a9 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d9, q_out=>q9);
  comp_a10 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d10, q_out=>q10);
  comp_a11 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d11, q_out=>q11);
  comp_a12 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d12, q_out=>q12);
  comp_a13 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d13, q_out=>q13);
  comp_a14 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d14, q_out=>q14);
  comp_a15 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d15, q_out=>q15);
  comp_a16 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d16, q_out=>q16);
  comp_a17 : ifdx_comp PORT MAP (reset=>rst, clk=>ifdx_clk, c_en=>c_e, d_in=>d17, q_out=>q17);


end;

 configuration ifdx18_comp_conf of ifdx18_comp is

   for ifdx18_comp

    for comp_a0: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a1: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a2: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a3: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a4: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a5: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a6: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;

for comp_a7: ifdx_comp use configuration work.ifdx_comp_conf;
    end for;
```

```
for comp_a8: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a9: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a10: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a11: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a12: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a13: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a14: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a15: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a16: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;

for comp_a17: ifdx_comp use configuration work.ifdx_comp_conf;
   end for;


  end for;
 end ifdx18_comp_conf;

-----------------------------

library ieee;
use ieee.std_logic_1164.all;

 entity fpga_core_comp is
  port( hl, enable, reset, ef_0, ef_1, ef_2, ef_3, datain0, datain1, datain2, datain3, datain4, datain5, datain6, datain7, datain8,
datain9, datain10, datain11, datain12, datain13, datain14, datain15, datain16, datain17, fifomask0, fifomask1, fifomask2,
fifomask3, fpga_core_clk: IN std_logic;
       read_en0, read_en1, read_en2, read_en3, output_en0, output_en1, output_en2, output_en3, d0, d1, d2, d3, d4, d5, d6, d7,
d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23: inout std_logic);
 end;

 architecture fpga_core_comp of fpga_core_comp is

 component fifo_reader_comp is
  port( high_level, frame_enable, rest, ef0, ef1, ef2, ef3, din0, din1, din2, din3, din4, din5, din6, din7, din8, din9, din10, din11,
din12, din13, din14, din15, din16, din17, fifo0, fifo1, fifo2, fifo3, fifo_read_clk: IN std_logic;
       rd_en0, rd_en1, rd_en2, rd_en3, out_en0, out_en1, out_en2, out_en3, data_enable, dout0, dout1, dout2, dout3, dout4,
dout5, dout6, dout7, dout8, dout9, dout10, dout11, dout12, dout13, dout14, dout15, dout16, dout17, dout18, dout19, dout20,
dout21, dout22: inout std_logic);
 end component;



component ifdx18_comp is
  port( rst, c_e, d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, ifdx_clk: IN std_logic;
       q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17: inout std_logic);
 end component;


 component ifd4_comp is
  port( rst, d0, d1, d2, d3, ifd_clk: IN std_logic;
       q0, q1, q2, q3: out std_logic);
 end component;

 component afd_comp is
    port( rst, clk, d_in: IN std_logic;
          q_out       : out std_logic);
  end component;



signal enable_data, e_f0, e_f1, e_f2, e_f3, ren0, ren1, ren2, ren3, oen0, oen1, oen2, oen3, q00, q01, q02, q03, q04, q05, q06, q07,
q08, q09, q010, q011, q012, q013, q014, q015, q016, q017 : std_logic;

 begin
```

```
  compa0 : fifo_reader_comp PORT MAP (high_level=>hl, frame_enable=>enable, rest=>reset, ef0=>e_f0, ef1=>e_f1,
ef2=>e_f2, ef3=>e_f3, din0=>q00, din1=>q01, din2=>q02, din3=>q03, din4=>q04, din5=>q05, din6=>q06, din7=>q07,
din8=>q08, din9=>q09, din10=>q010, din11=>q011, din12=>q012, din13=>q013, din14=>q014, din15=>q015, din16=>q016,
din17=>q017, fifo0=>fifomask0, fifo1=>fifomask1, fifo2=>fifomask2, fifo3=>fifomask3, fifo_read_clk=>fpga_core_clk,
rd_en0=>ren0, rd_en1=>ren1, rd_en2=>ren2, rd_en3=>ren3, out_en0=>oen0, out_en1=>oen1, out_en2=>oen2,
out_en3=>oen3, data_enable=>enable_data, dout0=>d0, dout1=>d1, dout2=>d2, dout3=>d3, dout4=>d4, dout5=>d5,
dout6=>d6, dout7=>d7, dout8=>d8, dout9=>d9, dout10=>d10, dout11=>d11, dout12=>d12, dout13=>d13, dout14=>d14,
dout15=>d15, dout16=>d16, dout17=>d17, dout18=>d18, dout19=>d19, dout20=>d20, dout21=>d21, dout22=>d22);
  compa1 : ifdx18_comp PORT MAP ( rst=>reset, c_e=>enable_data, d0=>datain0, d1=>datain1, d2=>datain2, d3=>datain3,
d4=>datain4, d5=>datain5, d6=>datain6, d7=>datain7, d8=>datain8, d9=>datain9, d10=>datain10, d11=>datain11,
d12=>datain12, d13=>datain13, d14=>datain14, d15=>datain15, d16=>datain16, d17=>datain17, ifdx_clk=>fpga_core_clk,
q0=>q00, q1=>q01, q2=>q02, q3=>q03, q4=>q04, q5=>q05, q6=>q06, q7=>q07, q8=>q08, q9=>q09, q10=>q010, q11=>q011,
q12=>q012, q13=>q013, q14=>q014, q15=>q015, q16=>q016, q17=>q017);
  compa2 : ifd4_comp PORT MAP (rst=>reset, d0=>ef_0, d1=>ef_1, d2=>ef_2, d3=>ef_3, ifd_clk=>fpga_core_clk, q0=>e_f0,
q1=>e_f1, q2=>e_f2, q3=>e_f3);
  compa3 : ifd4_comp PORT MAP (rst=>reset, d0=>ren0, d1=>ren1, d2=>ren2, d3=>ren3, ifd_clk=>fpga_core_clk,
q0=>read_en0, q1=>read_en1, q2=>read_en2, q3=>read_en3);
  compa4 : ifd4_comp PORT MAP (rst=>reset, d0=>oen0, d1=>oen1, d2=>oen2, d3=>oen3, ifd_clk=>fpga_core_clk,
q0=>output_en0, q1=>output_en1, q2=>output_en2, q3=>output_en3);
  compa5 : afd_comp PORT MAP (rst=>reset, clk=>fpga_core_clk, d_in=>enable, q_out=>d23);

end;

 configuration fpga_core_comp_conf of fpga_core_comp is

   for fpga_core_comp

     for compa0: fifo_reader_comp use configuration work.fifo_reader_comp_conf;
    end for;

     for compa1: ifdx18_comp use configuration work.ifdx18_comp_conf;
    end for;

    for compa2: ifd4_comp use configuration work.ifd4_comp_conf;
    end for;

    for compa3: ifd4_comp use configuration work.ifd4_comp_conf;
    end for;

    for compa4: ifd4_comp use configuration work.ifd4_comp_conf;
    end for;

    for compa5: afd_comp use configuration work.afd_comp_conf;
    end for;


   end for;

 end fpga_core_comp_conf;
```