1 spu_lite.bde

1.1 spu_lite.vhd

```
-- Design unit header --
library IEEE;
use IEEE.std_logic_1164.all;

library spu_lite;
use spu_lite_pkg.all;

-- Included from components --
library spu_lite;
use spu_lite.spu_lite_pkg.all;

entity spu_lite is
  port(
       clk : in STD_LOGIC;
       even_RegWr_rf : in STD_LOGIC;
       odd_RegWr_rf : in STD_LOGIC;
       PC_rf : in STD_LOGIC_VECTOR(31 downto 0);
       RA_rf : in STD_LOGIC_VECTOR(6 downto 0);
       RB_rf : in STD_LOGIC_VECTOR(6 downto 0);
       RC_rf : in STD_LOGIC_VECTOR(6 downto 0);
       RD_rf : in STD_LOGIC_VECTOR(6 downto 0);
       RE_rf : in STD_LOGIC_VECTOR(6 downto 0);
       RF_rf : in STD_LOGIC_VECTOR(6 downto 0);
       even_Imm_rf : in STD_LOGIC_VECTOR(17 downto 0);
       even_Latency_rf : in STD_LOGIC_VECTOR(2 downto 0);
       even_RegDst_rf : in STD_LOGIC_VECTOR(6 downto 0);
       even_Unit_rf : in STD_LOGIC_VECTOR(2 downto 0);
       odd_Imm_rf : in STD_LOGIC_VECTOR(15 downto 0);
       odd_Latency_rf : in STD_LOGIC_VECTOR(2 downto 0);
       odd_RegDst_rf : in STD_LOGIC_VECTOR(6 downto 0);
       odd_Unit_rf : in STD_LOGIC_VECTOR(2 downto 0);
       op_BRU_rf : in STD_LOGIC_VECTOR(2 downto 0);
       op_BU_rf : in STD_LOGIC_VECTOR(1 downto 0);
       op_LSU_rf : in STD_LOGIC_VECTOR(2 downto 0);
       op_PU_rf : in STD_LOGIC_VECTOR(1 downto 0);
       op_SFU1_rf : in STD_LOGIC_VECTOR(4 downto 0);
       op_SFU2_rf : in STD_LOGIC_VECTOR(2 downto 0);
```

```
        op_SPU_rf : in STD_LOGIC_VECTOR(3 downto 0);
        PCWr : out STD_LOGIC;
        PC_Branch : out STD_LOGIC_VECTOR(31 downto 0)
    );
end spu_lite;

architecture rtl of spu_lite is

---- Component declarations -----

component branch_reg
    port (
        PCWr_d : in STD_LOGIC;
        PC_d : in STD_LOGIC_VECTOR(31 downto 0);
        clk : in STD_LOGIC;
        PCWr_q : out STD_LOGIC := '0';
        PC_q : out STD_LOGIC_VECTOR(31 downto 0) := (others => '0')
    );
end component;
component branch_unit
    port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        Imm : in STD_LOGIC_VECTOR(15 downto 0);
        PC : in STD_LOGIC_VECTOR(31 downto 0);
        T : in STD_LOGIC_VECTOR(127 downto 0);
        op_sel : in STD_LOGIC_VECTOR(2 downto 0);
        PCWr : out STD_LOGIC;
        Result : out STD_LOGIC_VECTOR(31 downto 0)
    );
end component;
component byte_unit
    port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        B : in STD_LOGIC_VECTOR(127 downto 0);
        op_sel : in STD_LOGIC_VECTOR(1 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
    );
end component;
component even_rf_reg
    port (
        Imm_d : in STD_LOGIC_VECTOR(17 downto 0);
        Latency_d : in STD_LOGIC_VECTOR(2 downto 0);
        RA_d : in STD_LOGIC_VECTOR(6 downto 0);
        RB_d : in STD_LOGIC_VECTOR(6 downto 0);
        RC_d : in STD_LOGIC_VECTOR(6 downto 0);
```

```
        RegDst_d : in STD_LOGIC_VECTOR(6 downto 0);
        RegWr_d : in STD_LOGIC;
        Unit_d : in STD_LOGIC_VECTOR(2 downto 0);
        clk : in STD_LOGIC;
        op_BU_d : in STD_LOGIC_VECTOR(1 downto 0);
        op_SFU1_d : in STD_LOGIC_VECTOR(4 downto 0);
        op_SFU2_d : in STD_LOGIC_VECTOR(2 downto 0);
        op_SPU_d : in STD_LOGIC_VECTOR(3 downto 0);
        Imm_q : out STD_LOGIC_VECTOR(17 downto 0) := (others => '0');
        Latency_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        RA_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RB_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RC_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RegDst_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RegWr_q : out STD_LOGIC := '0';
        Unit_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        op_BU_q : out STD_LOGIC_VECTOR(1 downto 0) := (others => '0');
        op_SFU1_q : out STD_LOGIC_VECTOR(4 downto 0) := (others => '0');
        op_SFU2_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        op_SPU_q : out STD_LOGIC_VECTOR(3 downto 0) := (others => '0')
  );
end component;
component forwarding_unit
  port (
        A_reg : in STD_LOGIC_VECTOR(127 downto 0);
        B_reg : in STD_LOGIC_VECTOR(127 downto 0);
        C_reg : in STD_LOGIC_VECTOR(127 downto 0);
        D_reg : in STD_LOGIC_VECTOR(127 downto 0);
        E_reg : in STD_LOGIC_VECTOR(127 downto 0);
        F_reg : in STD_LOGIC_VECTOR(127 downto 0);
        RA : in STD_LOGIC_VECTOR(6 downto 0);
        RB : in STD_LOGIC_VECTOR(6 downto 0);
        RC : in STD_LOGIC_VECTOR(6 downto 0);
        RD : in STD_LOGIC_VECTOR(6 downto 0);
        RE : in STD_LOGIC_VECTOR(6 downto 0);
        RF : in STD_LOGIC_VECTOR(6 downto 0);
        even2_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        even2_RegWr : in STD_LOGIC;
        even2_Result : in STD_LOGIC_VECTOR(127 downto 0);
        even3_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        even3_RegWr : in STD_LOGIC;
        even3_Result : in STD_LOGIC_VECTOR(127 downto 0);
        even4_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        even4_RegWr : in STD_LOGIC;
        even4_Result : in STD_LOGIC_VECTOR(127 downto 0);
```

```
        even5_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        even5_RegWr : in STD_LOGIC;
        even5_Result : in STD_LOGIC_VECTOR(127 downto 0);
        even6_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        even6_RegWr : in STD_LOGIC;
        even6_Result : in STD_LOGIC_VECTOR(127 downto 0);
        even7_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        even7_RegWr : in STD_LOGIC;
        even7_Result : in STD_LOGIC_VECTOR(127 downto 0);
        evenWB_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        evenWB_RegWr : in STD_LOGIC;
        evenWB_Result : in STD_LOGIC_VECTOR(127 downto 0);
        odd4_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        odd4_RegWr : in STD_LOGIC;
        odd4_Result : in STD_LOGIC_VECTOR(127 downto 0);
        odd5_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        odd5_RegWr : in STD_LOGIC;
        odd5_Result : in STD_LOGIC_VECTOR(127 downto 0);
        odd6_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        odd6_RegWr : in STD_LOGIC;
        odd6_Result : in STD_LOGIC_VECTOR(127 downto 0);
        odd7_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        odd7_RegWr : in STD_LOGIC;
        odd7_Result : in STD_LOGIC_VECTOR(127 downto 0);
        oddWB_RegDst : in STD_LOGIC_VECTOR(6 downto 0);
        oddWB_RegWr : in STD_LOGIC;
        oddWB_Result : in STD_LOGIC_VECTOR(127 downto 0);
        A : out STD_LOGIC_VECTOR(127 downto 0);
        B : out STD_LOGIC_VECTOR(127 downto 0);
        C : out STD_LOGIC_VECTOR(127 downto 0);
        D : out STD_LOGIC_VECTOR(127 downto 0);
        E : out STD_LOGIC_VECTOR(127 downto 0);
        F : out STD_LOGIC_VECTOR(127 downto 0)
  );
end component;
component local_store_unit
  port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        B : in STD_LOGIC_VECTOR(127 downto 0);
        Imm : in STD_LOGIC_VECTOR(15 downto 0);
        T : in STD_LOGIC_VECTOR(127 downto 0);
        clk : in STD_LOGIC;
        op_sel : in STD_LOGIC_VECTOR(2 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
  );
```

```
end component;
component odd_rf_reg
  port (
        Imm_d : in STD_LOGIC_VECTOR(15 downto 0);
        Latency_d : in STD_LOGIC_VECTOR(2 downto 0);
        PC_d : in STD_LOGIC_VECTOR(31 downto 0);
        RD_d : in STD_LOGIC_VECTOR(6 downto 0);
        RE_d : in STD_LOGIC_VECTOR(6 downto 0);
        RF_d : in STD_LOGIC_VECTOR(6 downto 0);
        RegDst_d : in STD_LOGIC_VECTOR(6 downto 0);
        RegWr_d : in STD_LOGIC;
        Unit_d : in STD_LOGIC_VECTOR(2 downto 0);
        clk : in STD_LOGIC;
        op_BRU_d : in STD_LOGIC_VECTOR(2 downto 0);
        op_LSU_d : in STD_LOGIC_VECTOR(2 downto 0);
        op_PU_d : in STD_LOGIC_VECTOR(1 downto 0);
        Imm_q : out STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
        Latency_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        PC_q : out STD_LOGIC_VECTOR(31 downto 0);
        RD_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RE_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RF_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RegDst_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RegWr_q : out STD_LOGIC := '0';
        Unit_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        op_BRU_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        op_LSU_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        op_PU_q : out STD_LOGIC_VECTOR(1 downto 0) := (others => '0')
  );
end component;
component permute_unit
  port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        op_sel : in STD_LOGIC_VECTOR(1 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
  );
end component;
component pipe_reg
  port (
        Latency_d : in STD_LOGIC_VECTOR(2 downto 0);
        RegDst_d : in STD_LOGIC_VECTOR(6 downto 0);
        RegWr_d : in STD_LOGIC;
        Result_d : in STD_LOGIC_VECTOR(127 downto 0);
        Unit_d : in STD_LOGIC_VECTOR(2 downto 0);
        clk : in STD_LOGIC;
```

```
        Latency_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
        RegDst_q : out STD_LOGIC_VECTOR(6 downto 0) := (others => '0');
        RegWr_q : out STD_LOGIC := '0';
        Result_q : out STD_LOGIC_VECTOR(127 downto 0) := (others => '0');
        Unit_q : out STD_LOGIC_VECTOR(2 downto 0) := (others => '0')
    );
end component;
component register_file
    port (
        A_rd_addr : in STD_LOGIC_VECTOR(6 downto 0);
        A_wr_addr : in STD_LOGIC_VECTOR(6 downto 0);
        A_wr_data : in STD_LOGIC_VECTOR(127 downto 0);
        A_wr_en : in STD_LOGIC;
        B_rd_addr : in STD_LOGIC_VECTOR(6 downto 0);
        B_wr_addr : in STD_LOGIC_VECTOR(6 downto 0);
        B_wr_data : in STD_LOGIC_VECTOR(127 downto 0);
        B_wr_en : in STD_LOGIC;
        C_rd_addr : in STD_LOGIC_VECTOR(6 downto 0);
        D_rd_addr : in STD_LOGIC_VECTOR(6 downto 0);
        E_rd_addr : in STD_LOGIC_VECTOR(6 downto 0);
        F_rd_addr : in STD_LOGIC_VECTOR(6 downto 0);
        clk : in STD_LOGIC;
        A_rd_data : out STD_LOGIC_VECTOR(127 downto 0);
        B_rd_data : out STD_LOGIC_VECTOR(127 downto 0);
        C_rd_data : out STD_LOGIC_VECTOR(127 downto 0);
        D_rd_data : out STD_LOGIC_VECTOR(127 downto 0);
        E_rd_data : out STD_LOGIC_VECTOR(127 downto 0);
        F_rd_data : out STD_LOGIC_VECTOR(127 downto 0)
    );
end component;
component result_mux
    generic(
        G_UNIT : unit_t;
        G_LATENCY : NATURAL
    );
    port (
        Latency : in STD_LOGIC_VECTOR(2 downto 0);
        Result0 : in STD_LOGIC_VECTOR(127 downto 0);
        Result1 : in STD_LOGIC_VECTOR(127 downto 0);
        Unit_sel : in STD_LOGIC_VECTOR(2 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
    );
end component;
component result_reg
    port (
```

```
        clk : in STD_LOGIC;
        d : in STD_LOGIC_VECTOR(127 downto 0);
        q : out STD_LOGIC_VECTOR(127 downto 0) := (others => '0')
  );
end component;
component simple_fixed_unit1
  port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        B : in STD_LOGIC_VECTOR(127 downto 0);
        Imm : in STD_LOGIC_VECTOR(17 downto 0);
        op_sel : in STD_LOGIC_VECTOR(4 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
  );
end component;
component simple_fixed_unit2
  port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        B : in STD_LOGIC_VECTOR(127 downto 0);
        Imm : in STD_LOGIC_VECTOR(17 downto 0);
        op_sel : in STD_LOGIC_VECTOR(2 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
  );
end component;
component single_precision_unit
  port (
        A : in STD_LOGIC_VECTOR(127 downto 0);
        B : in STD_LOGIC_VECTOR(127 downto 0);
        C : in STD_LOGIC_VECTOR(127 downto 0);
        Imm : in STD_LOGIC_VECTOR(17 downto 0);
        op_sel : in STD_LOGIC_VECTOR(3 downto 0);
        Result : out STD_LOGIC_VECTOR(127 downto 0)
  );
end component;

----     Constants     -----
constant GND_CONSTANT    : STD_LOGIC := '0';

---- Signal declarations used on the diagram ----

signal even0_RegWr : STD_LOGIC;
signal even1_RegWr : STD_LOGIC;
signal even2_RegWr : STD_LOGIC;
signal even3_RegWr : STD_LOGIC;
signal even4_RegWr : STD_LOGIC;
signal even5_RegWr : STD_LOGIC;
```

```
signal even6_RegWr : STD_LOGIC;
signal even7_RegWr : STD_LOGIC;
signal evenWB_RegWr : STD_LOGIC;
signal GND : STD_LOGIC;
signal odd0_RegWr : STD_LOGIC;
signal odd1_RegWr : STD_LOGIC;
signal odd2_RegWr : STD_LOGIC;
signal odd3_RegWr : STD_LOGIC;
signal odd4_RegWr : STD_LOGIC;
signal odd5_RegWr : STD_LOGIC;
signal odd6_RegWr : STD_LOGIC;
signal odd7_RegWr : STD_LOGIC;
signal oddWB_RegWr : STD_LOGIC;
signal PCWr1 : STD_LOGIC;
signal PCWr_BRU : STD_LOGIC;
signal A : STD_LOGIC_VECTOR(127 downto 0);
signal A_reg : STD_LOGIC_VECTOR(127 downto 0);
signal B : STD_LOGIC_VECTOR(127 downto 0);
signal BU_Result : STD_LOGIC_VECTOR(127 downto 0);
signal BU_Result1 : STD_LOGIC_VECTOR(127 downto 0);
signal BU_Result2 : STD_LOGIC_VECTOR(127 downto 0);
signal BU_Result3 : STD_LOGIC_VECTOR(127 downto 0);
signal B_reg : STD_LOGIC_VECTOR(127 downto 0);
signal C : STD_LOGIC_VECTOR(127 downto 0);
signal C_reg : STD_LOGIC_VECTOR(127 downto 0);
signal D : STD_LOGIC_VECTOR(127 downto 0);
signal D_reg : STD_LOGIC_VECTOR(127 downto 0);
signal E : STD_LOGIC_VECTOR(127 downto 0);
signal even0_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even0_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even0_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even1_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even1_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even1_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even1_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even2_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even2_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even2_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even2_Result_MUX : STD_LOGIC_VECTOR(127 downto 0);
signal even2_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even3_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even3_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even3_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even3_Result_MUX1 : STD_LOGIC_VECTOR(127 downto 0);
signal even3_Result_MUX2 : STD_LOGIC_VECTOR(127 downto 0);
```

```
signal even3_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even4_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even4_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even4_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even4_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even5_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even5_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even5_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even5_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even6_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even6_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even6_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even6_Result_MUX : STD_LOGIC_VECTOR(127 downto 0);
signal even6_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even7_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal even7_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal even7_Result : STD_LOGIC_VECTOR(127 downto 0);
signal even7_Result_MUX : STD_LOGIC_VECTOR(127 downto 0);
signal even7_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal evenWB_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal evenWB_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal evenWB_Result : STD_LOGIC_VECTOR(127 downto 0);
signal evenWB_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal even_Imm : STD_LOGIC_VECTOR(17 downto 0);
signal E_reg : STD_LOGIC_VECTOR(127 downto 0);
signal F : STD_LOGIC_VECTOR(127 downto 0);
signal F_reg : STD_LOGIC_VECTOR(127 downto 0);
signal GND_128 : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result1 : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result2 : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result3 : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result4 : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result5 : STD_LOGIC_VECTOR(127 downto 0);
signal LSU_Result6 : STD_LOGIC_VECTOR(127 downto 0);
signal odd0_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd0_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd0_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd1_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd1_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd1_Result : STD_LOGIC_VECTOR(127 downto 0);
signal odd1_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd2_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd2_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd2_Result : STD_LOGIC_VECTOR(127 downto 0);
```

```
signal odd2_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd3_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd3_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd3_Result : STD_LOGIC_VECTOR(127 downto 0);
signal odd3_Result_MUX : STD_LOGIC_VECTOR(127 downto 0);
signal odd3_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd4_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd4_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd4_Result : STD_LOGIC_VECTOR(127 downto 0);
signal odd4_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd5_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd5_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd5_Result : STD_LOGIC_VECTOR(127 downto 0);
signal odd5_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd6_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd6_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd6_Result : STD_LOGIC_VECTOR(127 downto 0);
signal odd6_Result_MUX : STD_LOGIC_VECTOR(127 downto 0);
signal odd6_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd7_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal odd7_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal odd7_Result : STD_LOGIC_VECTOR(127 downto 0);
signal odd7_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal oddWB_Latency : STD_LOGIC_VECTOR(2 downto 0);
signal oddWB_RegDst : STD_LOGIC_VECTOR(6 downto 0);
signal oddWB_Result : STD_LOGIC_VECTOR(127 downto 0);
signal oddWB_Unit : STD_LOGIC_VECTOR(2 downto 0);
signal odd_Imm : STD_LOGIC_VECTOR(15 downto 0);
signal op_BRU : STD_LOGIC_VECTOR(2 downto 0);
signal op_BU : STD_LOGIC_VECTOR(1 downto 0);
signal op_LSU : STD_LOGIC_VECTOR(2 downto 0);
signal op_PU : STD_LOGIC_VECTOR(1 downto 0);
signal op_SFU1 : STD_LOGIC_VECTOR(4 downto 0);
signal op_SFU2 : STD_LOGIC_VECTOR(2 downto 0);
signal op_SPU : STD_LOGIC_VECTOR(3 downto 0);
signal PC_BRU : STD_LOGIC_VECTOR(31 downto 0);
signal PC_BRU1 : STD_LOGIC_VECTOR(31 downto 0);
signal PC_fw : STD_LOGIC_VECTOR(31 downto 0);
signal PU_Result : STD_LOGIC_VECTOR(127 downto 0);
signal PU_Result1 : STD_LOGIC_VECTOR(127 downto 0);
signal PU_Result2 : STD_LOGIC_VECTOR(127 downto 0);
signal PU_Result3 : STD_LOGIC_VECTOR(127 downto 0);
signal RA_fw : STD_LOGIC_VECTOR(6 downto 0);
signal RB_fw : STD_LOGIC_VECTOR(6 downto 0);
signal RC_fw : STD_LOGIC_VECTOR(6 downto 0);
```

```
signal RD_fw : STD_LOGIC_VECTOR(6 downto 0);
signal RE_fw : STD_LOGIC_VECTOR(6 downto 0);
signal RF_fw : STD_LOGIC_VECTOR(6 downto 0);
signal SFU1_Result : STD_LOGIC_VECTOR(127 downto 0);
signal SFU1_Result1 : STD_LOGIC_VECTOR(127 downto 0);
signal SFU1_Result2 : STD_LOGIC_VECTOR(127 downto 0);
signal SFU2_Result : STD_LOGIC_VECTOR(127 downto 0);
signal SFU2_Result1 : STD_LOGIC_VECTOR(127 downto 0);
signal SFU2_Result2 : STD_LOGIC_VECTOR(127 downto 0);
signal SFU_Result3 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result1 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result2 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result3 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result4 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result5 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result6 : STD_LOGIC_VECTOR(127 downto 0);
signal SPU_Result7 : STD_LOGIC_VECTOR(127 downto 0);

begin

----  Component instantiations  ----

BRU : branch_unit
  port map(
       A => D,
       Imm => odd_Imm,
       PC => PC_fw,
       PCWr => PCWr_BRU,
       Result => PC_BRU,
       T => F,
       op_sel => op_BRU
  );

BRU_pipe1 : branch_reg
  port map(
       PCWr_d => PCWr_BRU,
       PCWr_q => PCWr1,
       PC_d => PC_BRU,
       PC_q => PC_BRU1,
       clk => clk
  );

BU : byte_unit
  port map(
```

```
        A => A,
        B => B,
        Result => BU_Result,
        op_sel => op_BU
  );

BU_MUX : result_mux
  generic map(
        G_UNIT => UNIT_BYTE,
        G_LATENCY => 4
  )
  port map(
        Latency => even3_Latency,
        Result => even3_Result_MUX2,
        Result0 => even3_Result_MUX1,
        Result1 => BU_Result3,
        Unit_sel => even3_Unit
  );

BU_pipe1 : result_reg
  port map(
        clk => clk,
        d => BU_Result,
        q => BU_Result1
  );

BU_pipe2 : result_reg
  port map(
        clk => clk,
        d => BU_Result1,
        q => BU_Result2
  );

BU_pipe3 : result_reg
  port map(
        clk => clk,
        d => BU_Result2,
        q => BU_Result3
  );

Even_RF : even_rf_reg
  port map(
        Imm_d => even_Imm_rf,
        Imm_q => even_Imm,
        Latency_d => even_Latency_rf,
```

```
        Latency_q => even0_Latency,
        RA_d => RA_rf,
        RA_q => RA_fw,
        RB_d => RB_rf,
        RB_q => RB_fw,
        RC_d => RC_rf,
        RC_q => RC_fw,
        RegDst_d => even_RegDst_rf,
        RegDst_q => even0_RegDst,
        RegWr_d => even_RegWr_rf,
        RegWr_q => even0_RegWr,
        Unit_d => even_Unit_rf,
        Unit_q => even0_Unit,
        clk => clk,
        op_BU_d => op_BU_rf,
        op_BU_q => op_BU,
        op_SFU1_d => op_SFU1_rf,
        op_SFU1_q => op_SFU1,
        op_SFU2_d => op_SFU2_rf,
        op_SFU2_q => op_SFU2,
        op_SPU_d => op_SPU_rf,
        op_SPU_q => op_SPU
  );

Even_WB : pipe_reg
  port map(
        Latency_d => even7_Latency,
        Latency_q => evenWB_Latency,
        RegDst_d => even7_RegDst,
        RegDst_q => evenWB_RegDst,
        RegWr_d => even7_RegWr,
        RegWr_q => evenWB_RegWr,
        Result_d => even7_Result_MUX,
        Result_q => evenWB_Result,
        Unit_d => even7_Unit,
        Unit_q => evenWB_Unit,
        clk => clk
  );

Even_pipe1 : pipe_reg
  port map(
        Latency_d => even0_Latency,
        Latency_q => even1_Latency,
        RegDst_d => even0_RegDst,
        RegDst_q => even1_RegDst,
```

```
        RegWr_d => even0_RegWr,
        RegWr_q => even1_RegWr,
        Result_d => GND_128,
        Result_q => even1_Result,
        Unit_d => even0_Unit,
        Unit_q => even1_Unit,
        clk => clk
  );

Even_pipe2 : pipe_reg
  port map(
        Latency_d => even1_Latency,
        Latency_q => even2_Latency,
        RegDst_d => even1_RegDst,
        RegDst_q => even2_RegDst,
        RegWr_d => even1_RegWr,
        RegWr_q => even2_RegWr,
        Result_d => even1_Result,
        Result_q => even2_Result,
        Unit_d => even1_Unit,
        Unit_q => even2_Unit,
        clk => clk
  );

Even_pipe3 : pipe_reg
  port map(
        Latency_d => even2_Latency,
        Latency_q => even3_Latency,
        RegDst_d => even2_RegDst,
        RegDst_q => even3_RegDst,
        RegWr_d => even2_RegWr,
        RegWr_q => even3_RegWr,
        Result_d => even2_Result_MUX,
        Result_q => even3_Result,
        Unit_d => even2_Unit,
        Unit_q => even3_Unit,
        clk => clk
  );

Even_pipe4 : pipe_reg
  port map(
        Latency_d => even3_Latency,
        Latency_q => even4_Latency,
        RegDst_d => even3_RegDst,
        RegDst_q => even4_RegDst,
```

```
        RegWr_d => even3_RegWr,
        RegWr_q => even4_RegWr,
        Result_d => even3_Result_MUX2,
        Result_q => even4_Result,
        Unit_d => even3_Unit,
        Unit_q => even4_Unit,
        clk => clk
  );

Even_pipe5 : pipe_reg
  port map(
        Latency_d => even4_Latency,
        Latency_q => even5_Latency,
        RegDst_d => even4_RegDst,
        RegDst_q => even5_RegDst,
        RegWr_d => even4_RegWr,
        RegWr_q => even5_RegWr,
        Result_d => even4_Result,
        Result_q => even5_Result,
        Unit_d => even4_Unit,
        Unit_q => even5_Unit,
        clk => clk
  );

Even_pipe6 : pipe_reg
  port map(
        Latency_d => even5_Latency,
        Latency_q => even6_Latency,
        RegDst_d => even5_RegDst,
        RegDst_q => even6_RegDst,
        RegWr_d => even5_RegWr,
        RegWr_q => even6_RegWr,
        Result_d => even5_Result,
        Result_q => even6_Result,
        Unit_d => even5_Unit,
        Unit_q => even6_Unit,
        clk => clk
  );

Even_pipe7 : pipe_reg
  port map(
        Latency_d => even6_Latency,
        Latency_q => even7_Latency,
        RegDst_d => even6_RegDst,
        RegDst_q => even7_RegDst,
```

```
        RegWr_d => even6_RegWr,
        RegWr_q => even7_RegWr,
        Result_d => even6_Result_MUX,
        Result_q => even7_Result,
        Unit_d => even6_Unit,
        Unit_q => even7_Unit,
        clk => clk
  );

FWD_Unit : forwarding_unit
  port map(
        A => A,
        A_reg => A_reg,
        B => B,
        B_reg => B_reg,
        C => C,
        C_reg => C_reg,
        D => D,
        D_reg => D_reg,
        E => E,
        E_reg => E_reg,
        F => F,
        F_reg => F_reg,
        RA => RA_fw,
        RB => RB_fw,
        RC => RC_fw,
        RD => RD_fw,
        RE => RE_fw,
        RF => RF_fw,
        even2_RegDst => even2_RegDst,
        even2_RegWr => even2_RegWr,
        even2_Result => even2_Result_MUX,
        even3_RegDst => even3_RegDst,
        even3_RegWr => even3_RegWr,
        even3_Result => even3_Result,
        even4_RegDst => even4_RegDst,
        even4_RegWr => even4_RegWr,
        even4_Result => even4_Result,
        even5_RegDst => even5_RegDst,
        even5_RegWr => even5_RegWr,
        even5_Result => even5_Result,
        even6_RegDst => even6_RegDst,
        even6_RegWr => even6_RegWr,
        even6_Result => even6_Result_MUX,
        even7_RegDst => even7_RegDst,
```

```
        even7_RegWr => even7_RegWr,
        even7_Result => even7_Result_MUX,
        evenWB_RegDst => evenWB_RegDst,
        evenWB_RegWr => evenWB_RegWr,
        evenWB_Result => evenWB_Result,
        odd4_RegDst => odd4_RegDst,
        odd4_RegWr => odd4_RegWr,
        odd4_Result => odd4_Result,
        odd5_RegDst => odd5_RegDst,
        odd5_RegWr => odd5_RegWr,
        odd5_Result => odd5_Result,
        odd6_RegDst => odd6_RegDst,
        odd6_RegWr => odd6_RegWr,
        odd6_Result => odd6_Result_MUX,
        odd7_RegDst => odd7_RegDst,
        odd7_RegWr => odd7_RegWr,
        odd7_Result => odd7_Result,
        oddWB_RegDst => oddWB_RegDst,
        oddWB_RegWr => oddWB_RegWr,
        oddWB_Result => oddWB_Result
  );

LSU : local_store_unit
  port map(
        A => D,
        B => E,
        Imm => odd_Imm,
        Result => LSU_Result,
        T => F,
        clk => clk,
        op_sel => op_LSU
  );

LSU_MUX : result_mux
  generic map(
        G_UNIT => UNIT_LOCAL_STORE,
        G_LATENCY => 6
  )
  port map(
        Latency => odd6_Latency,
        Result => odd6_Result_MUX,
        Result0 => odd6_Result,
        Result1 => LSU_Result6,
        Unit_sel => odd6_Unit
  );
```

```
LSU_pipe1 : result_reg
  port map(
       clk => clk,
       d => LSU_Result,
       q => LSU_Result1
  );

LSU_pipe2 : result_reg
  port map(
       clk => clk,
       d => LSU_Result1,
       q => LSU_Result2
  );

LSU_pipe3 : result_reg
  port map(
       clk => clk,
       d => LSU_Result2,
       q => LSU_Result3
  );

LSU_pipe4 : result_reg
  port map(
       clk => clk,
       d => LSU_Result3,
       q => LSU_Result4
  );

LSU_pipe5 : result_reg
  port map(
       clk => clk,
       d => LSU_Result4,
       q => LSU_Result5
  );

LSU_pipe6 : result_reg
  port map(
       clk => clk,
       d => LSU_Result5,
       q => LSU_Result6
  );

Odd_RF : odd_rf_reg
  port map(
       Imm_d => odd_Imm_rf,
```

```
        Imm_q => odd_Imm,
        Latency_d => odd_Latency_rf,
        Latency_q => odd0_Latency,
        PC_d => PC_rf,
        PC_q => PC_fw,
        RD_d => RD_rf,
        RD_q => RD_fw,
        RE_d => RE_rf,
        RE_q => RE_fw,
        RF_d => RF_rf,
        RF_q => RF_fw,
        RegDst_d => odd_RegDst_rf,
        RegDst_q => odd0_RegDst,
        RegWr_d => odd_RegWr_rf,
        RegWr_q => odd0_RegWr,
        Unit_d => odd_Unit_rf,
        Unit_q => odd0_Unit,
        clk => clk,
        op_BRU_d => op_BRU_rf,
        op_BRU_q => op_BRU,
        op_LSU_d => op_LSU_rf,
        op_LSU_q => op_LSU,
        op_PU_d => op_PU_rf,
        op_PU_q => op_PU
  );

Odd_WB : pipe_reg
  port map(
        Latency_d => odd7_Latency,
        Latency_q => oddWB_Latency,
        RegDst_d => odd7_RegDst,
        RegDst_q => oddWB_RegDst,
        RegWr_d => odd7_RegWr,
        RegWr_q => oddWB_RegWr,
        Result_d => odd7_Result,
        Result_q => oddWB_Result,
        Unit_d => odd7_Unit,
        Unit_q => oddWB_Unit,
        clk => clk
  );

Odd_pipe1 : pipe_reg
  port map(
        Latency_d => odd0_Latency,
        Latency_q => odd1_Latency,
```

```
        RegDst_d => odd0_RegDst,
        RegDst_q => odd1_RegDst,
        RegWr_d => odd0_RegWr,
        RegWr_q => odd1_RegWr,
        Result_d => GND_128,
        Result_q => odd1_Result,
        Unit_d => odd0_Unit,
        Unit_q => odd1_Unit,
        clk => clk
  );

Odd_pipe2 : pipe_reg
  port map(
        Latency_d => odd1_Latency,
        Latency_q => odd2_Latency,
        RegDst_d => odd1_RegDst,
        RegDst_q => odd2_RegDst,
        RegWr_d => odd1_RegWr,
        RegWr_q => odd2_RegWr,
        Result_d => odd1_Result,
        Result_q => odd2_Result,
        Unit_d => odd1_Unit,
        Unit_q => odd2_Unit,
        clk => clk
  );

Odd_pipe3 : pipe_reg
  port map(
        Latency_d => odd2_Latency,
        Latency_q => odd3_Latency,
        RegDst_d => odd2_RegDst,
        RegDst_q => odd3_RegDst,
        RegWr_d => odd2_RegWr,
        RegWr_q => odd3_RegWr,
        Result_d => odd2_Result,
        Result_q => odd3_Result,
        Unit_d => odd2_Unit,
        Unit_q => odd3_Unit,
        clk => clk
  );

Odd_pipe4 : pipe_reg
  port map(
        Latency_d => odd3_Latency,
        Latency_q => odd4_Latency,
```

```
        RegDst_d => odd3_RegDst,
        RegDst_q => odd4_RegDst,
        RegWr_d => odd3_RegWr,
        RegWr_q => odd4_RegWr,
        Result_d => odd3_Result_MUX,
        Result_q => odd4_Result,
        Unit_d => odd3_Unit,
        Unit_q => odd4_Unit,
        clk => clk
  );

Odd_pipe5 : pipe_reg
  port map(
        Latency_d => odd4_Latency,
        Latency_q => odd5_Latency,
        RegDst_d => odd4_RegDst,
        RegDst_q => odd5_RegDst,
        RegWr_d => odd4_RegWr,
        RegWr_q => odd5_RegWr,
        Result_d => odd4_Result,
        Result_q => odd5_Result,
        Unit_d => odd4_Unit,
        Unit_q => odd5_Unit,
        clk => clk
  );

Odd_pipe6 : pipe_reg
  port map(
        Latency_d => odd5_Latency,
        Latency_q => odd6_Latency,
        RegDst_d => odd5_RegDst,
        RegDst_q => odd6_RegDst,
        RegWr_d => odd5_RegWr,
        RegWr_q => odd6_RegWr,
        Result_d => odd5_Result,
        Result_q => odd6_Result,
        Unit_d => odd5_Unit,
        Unit_q => odd6_Unit,
        clk => clk
  );

Odd_pipe7 : pipe_reg
  port map(
        Latency_d => odd6_Latency,
        Latency_q => odd7_Latency,
```

```
        RegDst_d => odd6_RegDst,
        RegDst_q => odd7_RegDst,
        RegWr_d => odd6_RegWr,
        RegWr_q => odd7_RegWr,
        Result_d => odd6_Result_MUX,
        Result_q => odd7_Result,
        Unit_d => odd6_Unit,
        Unit_q => odd7_Unit,
        clk => clk
  );

PU : permute_unit
  port map(
        A => D,
        Result => PU_Result,
        op_sel => op_PU
  );

PU_MUX : result_mux
  generic map(
        G_UNIT => UNIT_PERMUTE,
        G_LATENCY => 4
  )
  port map(
        Latency => odd3_Latency,
        Result => odd3_Result_MUX,
        Result0 => odd3_Result,
        Result1 => PU_Result3,
        Unit_sel => odd3_Unit
  );

PU_pipe1 : result_reg
  port map(
        clk => clk,
        d => PU_Result,
        q => PU_Result1
  );

PU_pipe2 : result_reg
  port map(
        clk => clk,
        d => PU_Result1,
        q => PU_Result2
  );
```

```
PU_pipe3 : result_reg
  port map(
       clk => clk,
       d => PU_Result2,
       q => PU_Result3
  );

Registers : register_file
  port map(
       A_rd_addr => RA_rf,
       A_rd_data => A_reg,
       A_wr_addr => evenWB_RegDst,
       A_wr_data => evenWB_Result,
       A_wr_en => evenWB_RegWr,
       B_rd_addr => RB_rf,
       B_rd_data => B_reg,
       B_wr_addr => oddWB_RegDst,
       B_wr_data => oddWB_Result,
       B_wr_en => oddWB_RegWr,
       C_rd_addr => RC_rf,
       C_rd_data => C_reg,
       D_rd_addr => RD_rf,
       D_rd_data => D_reg,
       E_rd_addr => RE_rf,
       E_rd_data => E_reg,
       F_rd_addr => RF_rf,
       F_rd_data => F_reg,
       clk => clk
  );

SFU1 : simple_fixed_unit1
  port map(
       A => A,
       B => B,
       Imm => even_Imm,
       Result => SFU1_Result,
       op_sel => op_SFU1
  );

SFU1_MUX : result_mux
  generic map(
       G_UNIT => UNIT_SIMPLE_FIXED1,
       G_LATENCY => 2
  )
  port map(
```

```
        Latency => even2_Latency,
        Result => even2_Result_MUX,
        Result0 => even2_Result,
        Result1 => SFU1_Result2,
        Unit_sel => even2_Unit
  );

SFU1_pipe1 : result_reg
  port map(
        clk => clk,
        d => SFU1_Result,
        q => SFU1_Result1
  );

SFU1_pipe2 : result_reg
  port map(
        clk => clk,
        d => SFU1_Result1,
        q => SFU1_Result2
  );

SFU2 : simple_fixed_unit2
  port map(
        A => A,
        B => B,
        Imm => even_Imm,
        Result => SFU2_Result,
        op_sel => op_SFU2
  );

SFU2_MUX : result_mux
  generic map(
        G_UNIT => UNIT_SIMPLE_FIXED2,
        G_LATENCY => 4
  )
  port map(
        Latency => even3_Latency,
        Result => even3_Result_MUX1,
        Result0 => even3_Result,
        Result1 => SFU_Result3,
        Unit_sel => even3_Unit
  );

SFU2_pipe1 : result_reg
  port map(
```

```
        clk => clk,
        d => SFU2_Result,
        q => SFU2_Result1
  );

SFU2_pipe2 : result_reg
  port map(
        clk => clk,
        d => SFU2_Result1,
        q => SFU2_Result2
  );

SFU2_pipe3 : result_reg
  port map(
        clk => clk,
        d => SFU2_Result2,
        q => SFU_Result3
  );

SPU : single_precision_unit
  port map(
        A => A,
        B => B,
        C => C,
        Imm => even_Imm,
        Result => SPU_Result,
        op_sel => op_SPU
  );

SPU_MUX1 : result_mux
  generic map(
        G_UNIT => UNIT_SINGLE_PRECISION,
        G_LATENCY => 6
  )
  port map(
        Latency => even6_Latency,
        Result => even6_Result_MUX,
        Result0 => even6_Result,
        Result1 => SPU_Result6,
        Unit_sel => even6_Unit
  );

SPU_MUX2 : result_mux
  generic map(
        G_UNIT => UNIT_SINGLE_PRECISION,
```

```
        G_LATENCY => 7
    )
    port map(
        Latency => even7_Latency,
        Result => even7_Result_MUX,
        Result0 => even7_Result,
        Result1 => SPU_Result7,
        Unit_sel => even7_Unit
    );

SPU_pipe1 : result_reg
    port map(
        clk => clk,
        d => SPU_Result,
        q => SPU_Result1
    );

SPU_pipe2 : result_reg
    port map(
        clk => clk,
        d => SPU_Result1,
        q => SPU_Result2
    );

SPU_pipe3 : result_reg
    port map(
        clk => clk,
        d => SPU_Result2,
        q => SPU_Result3
    );

SPU_pipe4 : result_reg
    port map(
        clk => clk,
        d => SPU_Result3,
        q => SPU_Result4
    );

SPU_pipe5 : result_reg
    port map(
        clk => clk,
        d => SPU_Result4,
        q => SPU_Result5
    );
```

```
SPU_pipe6 : result_reg
  port map(
       clk => clk,
       d => SPU_Result5,
       q => SPU_Result6
  );

SPU_pipe7 : result_reg
  port map(
       clk => clk,
       d => SPU_Result6,
       q => SPU_Result7
  );


---- Power , ground assignment ----

GND <= GND_CONSTANT;
GND_128(127) <= GND;
GND_128(126) <= GND;
GND_128(125) <= GND;
GND_128(124) <= GND;
GND_128(123) <= GND;
GND_128(122) <= GND;
GND_128(121) <= GND;
GND_128(120) <= GND;
GND_128(119) <= GND;
GND_128(118) <= GND;
GND_128(117) <= GND;
GND_128(116) <= GND;
GND_128(115) <= GND;
GND_128(114) <= GND;
GND_128(113) <= GND;
GND_128(112) <= GND;
GND_128(111) <= GND;
GND_128(110) <= GND;
GND_128(109) <= GND;
GND_128(108) <= GND;
GND_128(107) <= GND;
GND_128(106) <= GND;
GND_128(105) <= GND;
GND_128(104) <= GND;
GND_128(103) <= GND;
GND_128(102) <= GND;
GND_128(101) <= GND;
```

```
GND_128(100) <= GND;
GND_128(99) <= GND;
GND_128(98) <= GND;
GND_128(97) <= GND;
GND_128(96) <= GND;
GND_128(95) <= GND;
GND_128(94) <= GND;
GND_128(93) <= GND;
GND_128(92) <= GND;
GND_128(91) <= GND;
GND_128(90) <= GND;
GND_128(89) <= GND;
GND_128(88) <= GND;
GND_128(87) <= GND;
GND_128(86) <= GND;
GND_128(85) <= GND;
GND_128(84) <= GND;
GND_128(83) <= GND;
GND_128(82) <= GND;
GND_128(81) <= GND;
GND_128(80) <= GND;
GND_128(79) <= GND;
GND_128(78) <= GND;
GND_128(77) <= GND;
GND_128(76) <= GND;
GND_128(75) <= GND;
GND_128(74) <= GND;
GND_128(73) <= GND;
GND_128(72) <= GND;
GND_128(71) <= GND;
GND_128(70) <= GND;
GND_128(69) <= GND;
GND_128(68) <= GND;
GND_128(67) <= GND;
GND_128(66) <= GND;
GND_128(65) <= GND;
GND_128(64) <= GND;
GND_128(63) <= GND;
GND_128(62) <= GND;
GND_128(61) <= GND;
GND_128(60) <= GND;
GND_128(59) <= GND;
GND_128(58) <= GND;
GND_128(57) <= GND;
GND_128(56) <= GND;
```

```
GND_128(55) <= GND;
GND_128(54) <= GND;
GND_128(53) <= GND;
GND_128(52) <= GND;
GND_128(51) <= GND;
GND_128(50) <= GND;
GND_128(49) <= GND;
GND_128(48) <= GND;
GND_128(47) <= GND;
GND_128(46) <= GND;
GND_128(45) <= GND;
GND_128(44) <= GND;
GND_128(43) <= GND;
GND_128(42) <= GND;
GND_128(41) <= GND;
GND_128(40) <= GND;
GND_128(39) <= GND;
GND_128(38) <= GND;
GND_128(37) <= GND;
GND_128(36) <= GND;
GND_128(35) <= GND;
GND_128(34) <= GND;
GND_128(33) <= GND;
GND_128(32) <= GND;
GND_128(31) <= GND;
GND_128(30) <= GND;
GND_128(29) <= GND;
GND_128(28) <= GND;
GND_128(27) <= GND;
GND_128(26) <= GND;
GND_128(25) <= GND;
GND_128(24) <= GND;
GND_128(23) <= GND;
GND_128(22) <= GND;
GND_128(21) <= GND;
GND_128(20) <= GND;
GND_128(19) <= GND;
GND_128(18) <= GND;
GND_128(17) <= GND;
GND_128(16) <= GND;
GND_128(15) <= GND;
GND_128(14) <= GND;
GND_128(13) <= GND;
GND_128(12) <= GND;
GND_128(11) <= GND;
```

```
GND_128(10) <= GND;
GND_128(9) <= GND;
GND_128(8) <= GND;
GND_128(7) <= GND;
GND_128(6) <= GND;
GND_128(5) <= GND;
GND_128(4) <= GND;
GND_128(3) <= GND;
GND_128(2) <= GND;
GND_128(1) <= GND;
GND_128(0) <= GND;

---- Terminal assignment ----

    -- Output\buffer terminals
    PCWr <= PCWr1;
    PC_Branch <= PC_BRU1;


end rtl;
```