```
library STD;
library IEEE;
use IEEE.std_logic_1164.all;

entity Cache_Cell_Tag is
    port(
        Data    :   in  std_logic_vector(2 downto 0);
        Tag_Wr  :   in  std_logic; --tag wr enable
        reset   :   in  std_logic;
        Gnd     :   in  std_logic;
        Output  :   out std_logic_vector(2 downto 0);
        Row_En  :   in  std_logic
    );
end Cache_Cell_Tag;

architecture structural of Cache_Cell_Tag is

    component Cache_Cell
    port(
        Data    :   in  std_logic;
        W_En    :   in  std_logic;
        reset   :   in  std_logic;
        Gnd     :   in  std_logic;
        Output  :   out std_logic;
        Rd_En   :   in  std_logic;
        nRd_En  :   in  std_logic
    );
    end component;

    component and2
    port(
        in1     :   in  std_logic;
        in2     :   in  std_logic;
        out1    :   out std_logic
    );
    end component;

    component or2
    port(
        in1     :   in  std_logic;
        in2     :   in  std_logic;
        out1    :   out std_logic
    );
    end component;

    component invX1
    port(
        in1     :   in  std_logic;
        out1    :   out std_logic
    );
    end component;

    signal Wr_En    :   std_logic;
    signal nRd_En   :   std_logic;

    for nrd :   invX1 use entity work.invX1(structural);
    for cell0, cell1, cell2 :   Cache_Cell use entity work.Cache_Cell(structural);

begin

    nrd     :   invX1       port map(Row_En, nRd_En);

    cell0   :   Cache_Cell  port map(Data(0), Tag_Wr, reset, Gnd, Output(0), Row_En, nRd_En);
    cell1   :   Cache_Cell  port map(Data(1), Tag_Wr, reset, Gnd, Output(1), Row_En, nRd_En);
    cell2   :   Cache_Cell  port map(Data(2), Tag_Wr, reset, Gnd, Output(2), Row_En, nRd_En);

end structural;
```