```
 1: library STD;
 2: library IEEE;
 3: use IEEE.std_logic_1164.all;
 4:
 5: entity Cache_Cell_Data_Block is
 6:     port(
 7:         Data    :   in  std_logic_vector(7 downto 0);
 8:         W_En_r  :   in  std_logic; --row wr enable
 9:         W_En_c  :   in  std_logic; --column wr enable
10:         reset   :   in  std_logic;
11:         Gnd     :   in  std_logic;
12:         Output  :   out std_logic_vector(7 downto 0);
13:         Rd_En_r :   in  std_logic; --row rd enable
14:         Rd_En_c :   in  std_logic  --column rd enable
15:     );
16: end Cache_Cell_Data_Block;
17:
18: architecture structural of Cache_Cell_Data_Block is
19:
20:     component Cache_Cell
21:     port(
22:         Data    :   in  std_logic;
23:         W_En    :   in  std_logic;
24:         reset   :   in  std_logic;
25:         Gnd     :   in  std_logic;
26:         Output  :   out std_logic;
27:         Rd_En   :   in  std_logic;
28:         nRd_En  :   in  std_logic
29:     );
30:     end component;
31:
32:     component and2
33:     port(
34:         in1     :   in  std_logic;
35:         in2     :   in  std_logic;
36:         out1    :   out std_logic
37:     );
38:     end component;
39:
40:     component nand2
41:     port(
42:         in1     :   in  std_logic;
43:         in2     :   in  std_logic;
44:         out1    :   out std_logic
45:     );
46:     end component;
47:
48:     component invX1
49:     port(
50:         in1     :   in  std_logic;
51:         out1    :   out std_logic
52:     );
53:     end component;
54:
55:     signal Rd_En    :   std_logic;
56:     signal Wr_En    :   std_logic;
57:     signal nRd_En   :   std_logic;
58:
59:     for and_1   :   nand2 use entity work.nand2(structural);
60:     for and_2   :   and2 use entity work.and2(structural);
61:     for nrd     :   invX1 use entity work.invX1(structural);
62:     for cell0, cell1, cell2, cell3, cell4, cell5, cell6, cell7  :   Cache_Cell
 use entity work.Cache_Cell(structural);
63:
64: begin
65:
66:     and_1   :   nand2       port map(Rd_En_c, Rd_En_r, nRd_En);
67:     and_2   :   and2        port map(W_En_c, W_En_r, Wr_En);
68:     nrd     :   invX1       port map(nRd_En, Rd_En);
69:
70:     cell0   :   Cache_Cell  port map(Data(0), Wr_En, reset, Gnd, Output(0), Rd
_En, nRd_En);
71:     cell1   :   Cache_Cell  port map(Data(1), Wr_En, reset, Gnd, Output(1), Rd
_En, nRd_En);
72:     cell2   :   Cache_Cell  port map(Data(2), Wr_En, reset, Gnd, Output(2), Rd
_En, nRd_En);
73:     cell3   :   Cache_Cell  port map(Data(3), Wr_En, reset, Gnd, Output(3), Rd
_En, nRd_En);
74:     cell4   :   Cache_Cell  port map(Data(4), Wr_En, reset, Gnd, Output(4), Rd
_En, nRd_En);
75:     cell5   :   Cache_Cell  port map(Data(5), Wr_En, reset, Gnd, Output(5), Rd
_En, nRd_En);
76:     cell6   :   Cache_Cell  port map(Data(6), Wr_En, reset, Gnd, Output(6), Rd
_En, nRd_En);
77:     cell7   :   Cache_Cell  port map(Data(7), Wr_En, reset, Gnd, Output(7), Rd
_En, nRd_En);
78:
79: end structural;
```