```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RAM_chip_32_word is
    Port ( Row : in  STD_LOGIC_VECTOR (3 downto 0);
           Col : in  STD_LOGIC;
           D_in : in  STD_LOGIC_VECTOR (15 downto 0);
           D_out : out  STD_LOGIC_VECTOR (15 downto 0);
           Chip_sel : in  STD_LOGIC;
           RW : in  STD_LOGIC);
end RAM_chip_32_word;

architecture Behavioral of RAM_chip_32_word is

--RAM slice
        COMPONENT RAM_slice_16_bit
        PORT(
                D_in : IN std_logic;
                Bit_sel : IN std_logic;
                Row_sel : IN std_logic_vector(15 downto 0);
                RW : IN std_logic;
                D_out : OUT std_logic
                );
        END COMPONENT;

--row decoder
        COMPONENT decoder_4_to_16
        PORT(
                A : IN std_logic_vector(3 downto 0);
                Q0 : OUT std_logic;
                Q1 : OUT std_logic;
                Q2 : OUT std_logic;
                Q3 : OUT std_logic;
                Q4 : OUT std_logic;
                Q5 : OUT std_logic;
                Q6 : OUT std_logic;
                Q7 : OUT std_logic;
                Q8 : OUT std_logic;
                Q9 : OUT std_logic;
                Q10 : OUT std_logic;
                Q11 : OUT std_logic;
                Q12 : OUT std_logic;
                Q13 : OUT std_logic;
                Q14 : OUT std_logic;
                Q15 : OUT std_logic
                );
        END COMPONENT;

--column decoder
        COMPONENT decoder_1_2
        PORT(
                A : IN std_logic;
                S : OUT std_logic_vector(1 downto 0)
                );
        END COMPONENT;

--three state buffer for output
        COMPONENT three_state_buffer_16_bit
        PORT(
                A : IN std_logic_vector(15 downto 0);
                enable : IN std_logic;
                Y : OUT std_logic_vector(15 downto 0)
                );
        END COMPONENT;


signal row_select, pre_buffer, post_buffer : STD_LOGIC_VECTOR(15 downto 0);
signal col_select : STD_LOGIC_VECTOR(1 downto 0);
signal enable_0, enable_1 : STD_LOGIC;
signal out_0, out_1, out_2, out_3, out_4, out_5, out_6, out_7, out_8, out_9, out_10, out_11, out_12, out_13,
out_14, out_15 : STD_LOGIC_VECTOR(1 downto 0);

begin
--port maps
        --row decoder
        Inst_decoder_4_16: decoder_4_to_16 PORT MAP(
                A => Row,
```

```
        Q0 => row_select(0),
        Q1 => row_select(1),
        Q2 => row_select(2),
        Q3 => row_select(3),
        Q4 => row_select(4),
        Q5 => row_select(5),
        Q6 => row_select(6),
        Q7 => row_select(7),
        Q8 => row_select(8),
        Q9 => row_select(9),
        Q10 => row_select(10),
        Q11 => row_select(11),
        Q12 => row_select(12),
        Q13 => row_select(13),
        Q14 => row_select(14),
        Q15 => row_select(15)
);

--column decoder
Inst_decoder_1_2: decoder_1_2 PORT MAP(
        A => Col,
        S => col_select
);

--three state buffer for output
Inst_three_state_buffer_16_bit: three_state_buffer_16_bit PORT MAP(
        A => pre_buffer,
        enable => Chip_sel,
        Y => post_buffer
);


--RAM slice 0
RAM_slice_00: RAM_slice_16_bit PORT MAP(
        D_in => D_in(0),
        D_out => out_0(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 1
RAM_slice_01: RAM_slice_16_bit PORT MAP(
        D_in => D_in(1),
        D_out => out_1(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 2
RAM_slice_02: RAM_slice_16_bit PORT MAP(
        D_in => D_in(2),
        D_out => out_2(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 3
RAM_slice_03: RAM_slice_16_bit PORT MAP(
        D_in => D_in(3),
        D_out => out_3(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 4
RAM_slice_04: RAM_slice_16_bit PORT MAP(
        D_in => D_in(4),
        D_out => out_4(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);
```

```
--RAM slice 5
RAM_slice_05: RAM_slice_16_bit PORT MAP(
        D_in => D_in(5),
        D_out => out_5(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 6
RAM_slice_06: RAM_slice_16_bit PORT MAP(
        D_in => D_in(6),
        D_out => out_6(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 7
RAM_slice_07: RAM_slice_16_bit PORT MAP(
        D_in => D_in(7),
        D_out => out_7(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 8
RAM_slice_08: RAM_slice_16_bit PORT MAP(
        D_in => D_in(8),
        D_out => out_8(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 9
RAM_slice_09: RAM_slice_16_bit PORT MAP(
        D_in => D_in(9),
        D_out => out_9(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 10
RAM_slice_10: RAM_slice_16_bit PORT MAP(
        D_in => D_in(10),
        D_out => out_10(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 11
RAM_slice_11: RAM_slice_16_bit PORT MAP(
        D_in => D_in(11),
        D_out => out_11(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 12
RAM_slice_12: RAM_slice_16_bit PORT MAP(
        D_in => D_in(12),
        D_out => out_12(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 13
RAM_slice_13: RAM_slice_16_bit PORT MAP(
        D_in => D_in(13),
        D_out => out_13(0),
        Bit_sel => enable_0,
```

```
        Row_sel => row_select,
        RW => RW
);

--RAM slice 14
RAM_slice_14: RAM_slice_16_bit PORT MAP(
        D_in => D_in(14),
        D_out => out_14(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 15
RAM_slice_15: RAM_slice_16_bit PORT MAP(
        D_in => D_in(15),
        D_out => out_15(0),
        Bit_sel => enable_0,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 16
RAM_slice_16: RAM_slice_16_bit PORT MAP(
        D_in => D_in(0),
        D_out => out_0(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 17
RAM_slice_17: RAM_slice_16_bit PORT MAP(
        D_in => D_in(1),
        D_out => out_1(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 18
RAM_slice_18: RAM_slice_16_bit PORT MAP(
        D_in => D_in(2),
        D_out => out_2(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 19
RAM_slice_19: RAM_slice_16_bit PORT MAP(
        D_in => D_in(3),
        D_out => out_3(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 20
RAM_slice_20: RAM_slice_16_bit PORT MAP(
        D_in => D_in(4),
        D_out => out_4(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 21
RAM_slice_21: RAM_slice_16_bit PORT MAP(
        D_in => D_in(5),
        D_out => out_5(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 22
```

```
RAM_slice_22: RAM_slice_16_bit PORT MAP(
        D_in => D_in(6),
        D_out => out_6(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 23
RAM_slice_23: RAM_slice_16_bit PORT MAP(
        D_in => D_in(7),
        D_out => out_7(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 24
RAM_slice_24: RAM_slice_16_bit PORT MAP(
        D_in => D_in(8),
        D_out => out_8(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 25
RAM_slice_25: RAM_slice_16_bit PORT MAP(
        D_in => D_in(9),
        D_out => out_9(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 26
RAM_slice_26: RAM_slice_16_bit PORT MAP(
        D_in => D_in(10),
        D_out => out_10(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 27
RAM_slice_27: RAM_slice_16_bit PORT MAP(
        D_in => D_in(11),
        D_out => out_11(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 28
RAM_slice_28: RAM_slice_16_bit PORT MAP(
        D_in => D_in(12),
        D_out => out_12(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 29
RAM_slice_29: RAM_slice_16_bit PORT MAP(
        D_in => D_in(13),
        D_out => out_13(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
        RW => RW
);

--RAM slice 30
RAM_slice_30: RAM_slice_16_bit PORT MAP(
        D_in => D_in(14),
        D_out => out_14(1),
        Bit_sel => enable_1,
        Row_sel => row_select,
```

```
                RW => RW
        );

        --RAM slice 31
        RAM_slice_31: RAM_slice_16_bit PORT MAP(
                D_in => D_in(15),
                D_out => out_15(1),
                Bit_sel => enable_1,
                Row_sel => row_select,
                RW => RW
        );

        enable_0 <= col_select(0) and Chip_sel;
        enable_1 <= col_select(1) and Chip_sel;

        pre_buffer(0) <= out_0(0) or out_0(1);
        pre_buffer(1) <= out_1(0) or out_1(1);
        pre_buffer(2) <= out_2(0) or out_2(1);
        pre_buffer(3) <= out_3(0) or out_3(1);
        pre_buffer(4) <= out_4(0) or out_4(1);
        pre_buffer(5) <= out_5(0) or out_5(1);
        pre_buffer(6) <= out_6(0) or out_6(1);
        pre_buffer(7) <= out_7(0) or out_7(1);
        pre_buffer(8) <= out_8(0) or out_8(1);
        pre_buffer(9) <= out_9(0) or out_9(1);
        pre_buffer(10) <= out_10(0) or out_10(1);
        pre_buffer(11) <= out_11(0) or out_11(1);
        pre_buffer(12) <= out_12(0) or out_12(1);
        pre_buffer(13) <= out_13(0) or out_13(1);
        pre_buffer(14) <= out_14(0) or out_14(1);
        pre_buffer(15) <= out_15(0) or out_15(1);

        D_out <= post_buffer;

end Behavioral;
```