CSU22022 Computer Architecture I

Fifth Lecture

Michael Manzke

2023-2024

Trinity College Dublin

2-to-1-Line Multiplexer

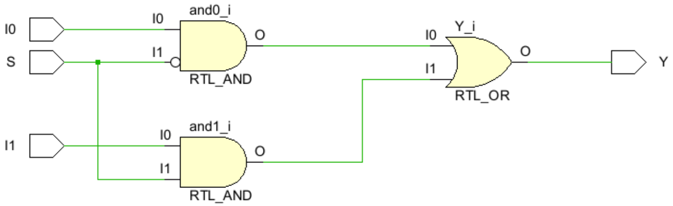


Figure 1: 2-to-1-Line Multiplexer Schematic

S	**Y**
0	$\mathbf{I}_0$
1	$\mathbf{I}_1$

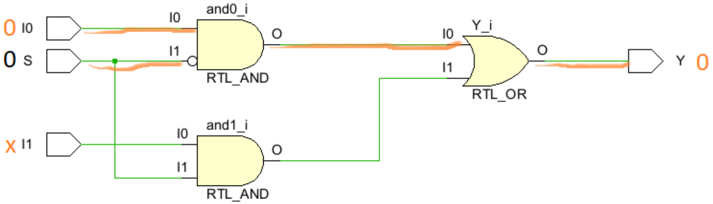
Table 1: 2-to-1-Line Multiplexer Truth Table

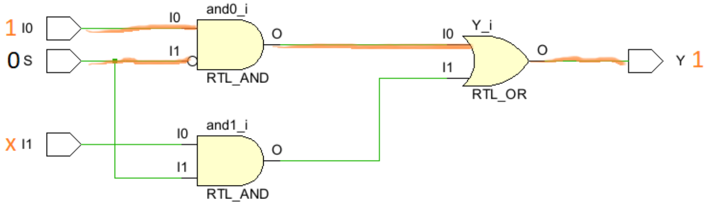
2-to-1-Line Multiplexer VHDL Code

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

entity Mux_2_to_1 is
    Port (I0, I1, S : in STD_LOGIC;
          Y : out STD_LOGIC);
end Mux_2_to_1;

architecture Behavioral of Mux_2_to_1 is
    signal S_not, and0, and1 : std_logic;
begin
    S_not <= not S after 3ns;
    and0 <= I0 and S_not after 4ns;
    and1 <= I1 and S after 4ns;
    Y <= and0 or and1 after 2ns;
end Behavioral;
```

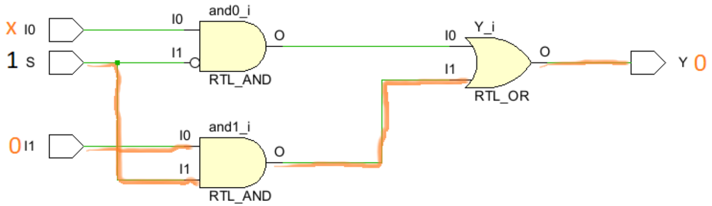
Listing 1: This code implements a 2-to-1-Line Multiplexer

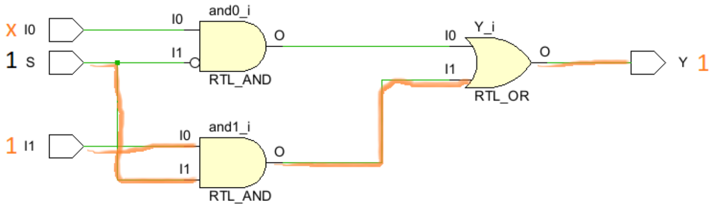
2-to-1-Line Multiplexer - S = 0





2-to-1-Line Multiplexer - S = 1





8-to-1-Line Multiplexer

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Table 2: 8-to-1-Line Multiplexer Truth Table

8-input Logic OR Gate

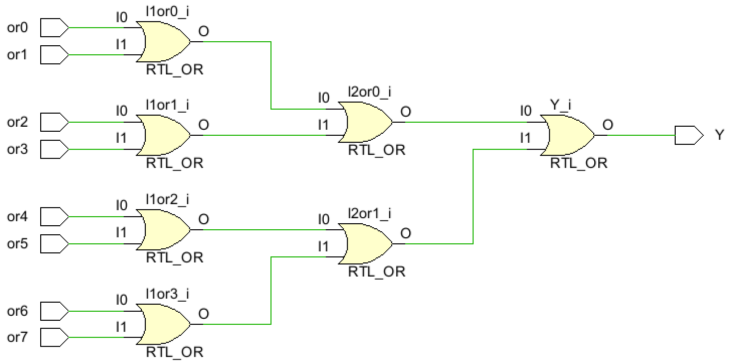


Figure 2: A 8-input logic OR gate requires seven OR gates

3-input Logic OR gate Example

A	**B**	**C**	$Y = A \vee B \vee C$
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Table 3: 3-input Logic OR gate Truth Table

8-input Logic OR Gate VHDL Code

```vhdl
entity OR8inputs is
    Port ( or0, or1, or2, or3  : in STD_LOGIC;
           or4, or5, or6, or7  : in STD_LOGIC;
           Y : out STD_LOGIC);
end OR8inputs;
architecture Behavioral of OR8inputs is
    signal l1or0, l1or1, l1or2, l1or3 : std_logic;
    signal l2or0, l2or1 : std_logic;
begin
    l1or0 <= or0 or or1 after 8ns;
    l1or1 <= or2 or or3 after 8ns;
    l1or2 <= or4 or or5 after 8ns;
    l1or3 <= or6 or or7 after 8ns;
    l2or0 <= l1or0 or l1or1 after 8ns;
    l2or1 <= l1or2 or l1or3 after 8ns;
    Y <= l2or0 or l2or1 after 8ns;
end Behavioral;
```

Listing 2: This code implements the 8-input logic OR gate

8-to-1-Line Multiplexer

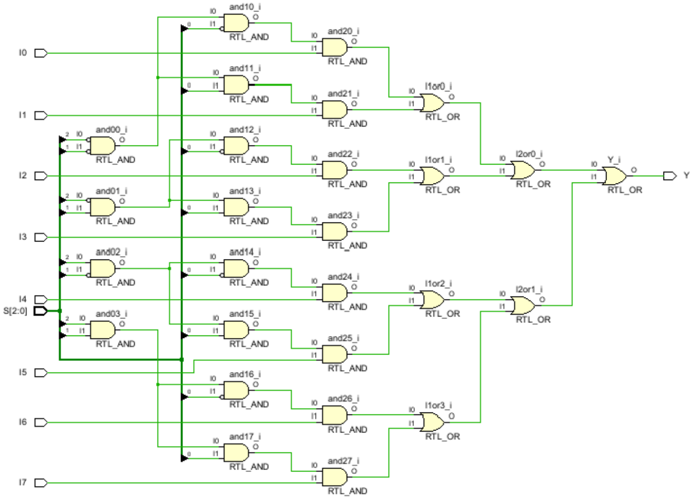


Figure 3: 8-to-1-line multiplexer schematic

8-to-1-Line Multiplexer - OR Logic

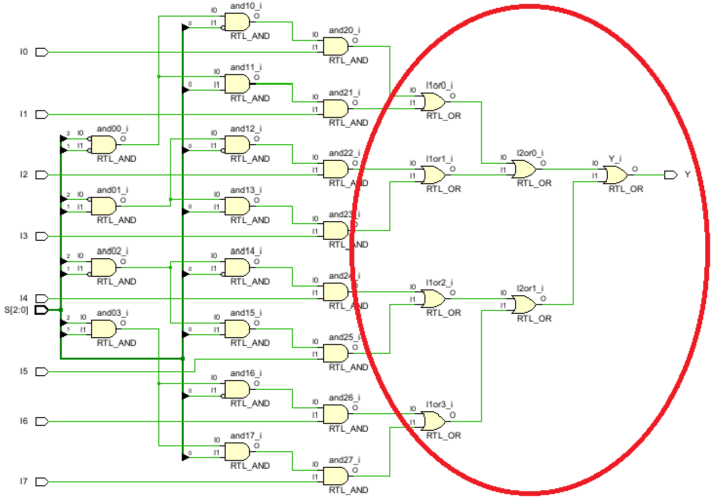


Figure 4: 8-input logic OR gate

8-to-1-Line Multiplexer - AND Logic

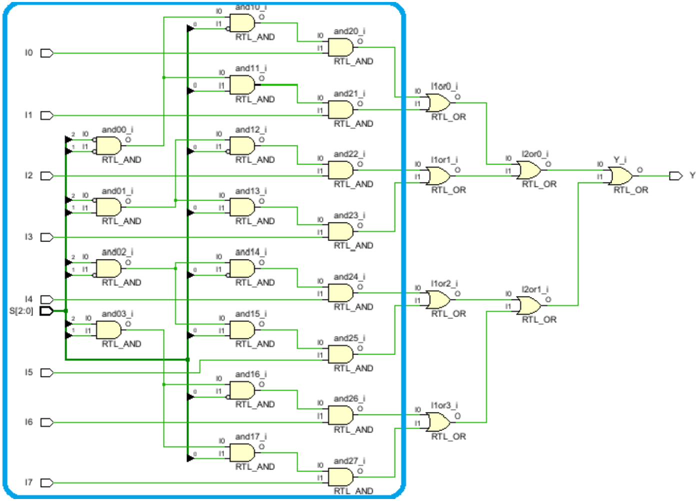


Figure 5: Implements truth table 2

AND and OR Logic

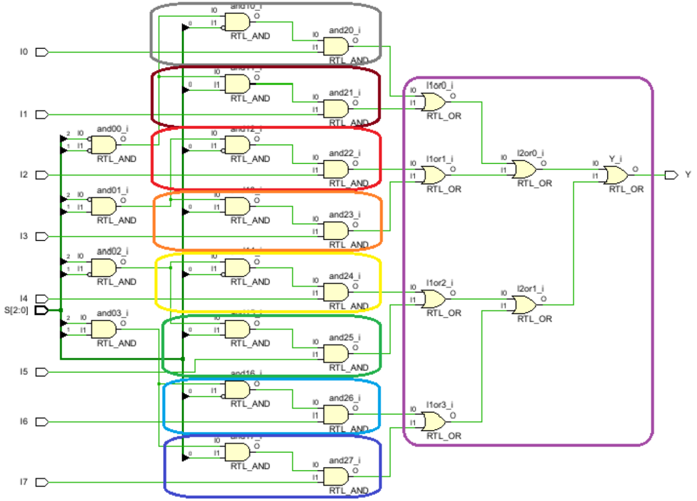


Figure 6: 3-input AND gates

8-to-1-Line Multiplexer - VHDL Code - One

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

entity Mux_8_to_1 is
    Port ( I0, I1, I2, I3 : in STD_LOGIC;
           I4, I5, I6, I7 : in STD_LOGIC;
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC);
end Mux_8_to_1;
```

Listing 3: 8-to-1-Line Multiplexer entity

8-to-1-Line Multiplexer - VHDL Code - Two

```
architecture Behavioral of Mux_8_to_1 is

    signal S0_not, S1_not, S2_not : std_logic;

    signal and00, and01, and02, and03 :std_logic;

    signal and10, and11, and12, and13 :std_logic;
    signal and14, and15, and16, and17 :std_logic;

    signal and20, and21, and22, and23 :std_logic;
    signal and24, and25, and26, and27 :std_logic;

    signal l1or0, l1or1, l1or2, l1or3 : std_logic;
    signal l2or0, l2or1 : std_logic;
```

Listing 4: 8-to-1-Line Multiplexer signal

8-to-1-Line Multiplexer - VHDL Code - Three

```vhdl
begin
    S0_not <= not S(0) after 3ns;
    S1_not <= not S(1) after 3ns;
    S2_not <= not S(2) after 3ns;

    and00 <= S2_not and S1_not after 2ns;
    and10 <= and00 and S0_not after 2ns;
    and20 <= and10 and I0 after 2ns;
    and11 <= and00 and S(0) after 2ns;
    and21 <= and11 and I1 after 2ns;

    and01 <= S2_not and S(1) after 2ns;
    and12 <= and01 and S0_not after 2ns;
    and22 <= and12 and I2 after 2ns;
    and13 <= and01 and S(0) after 2ns;
    and23 <= and13 and I3 after 2ns;
```

Listing 5: Concurrent Assignment Statements

8-to-1-Line Multiplexer - VHDL Code - Four

```
    and02 <= S(2) and S1_not after 2ns;
    and14 <= and02 and S0_not after 2ns;
    and24 <= and14 and I4 after 2ns;
    and15 <= and02 and S(0) after 2ns;
    and25 <= and15 and I5 after 2ns;

    and03 <= S(2) and S(1) after 2ns;
    and16 <= and03 and S0_not after 2ns;
    and26 <= and16 and I6 after 2ns;
    and17 <= and03 and S(0) after 2ns;
    and27 <= and17 and I7 after 2ns;
```

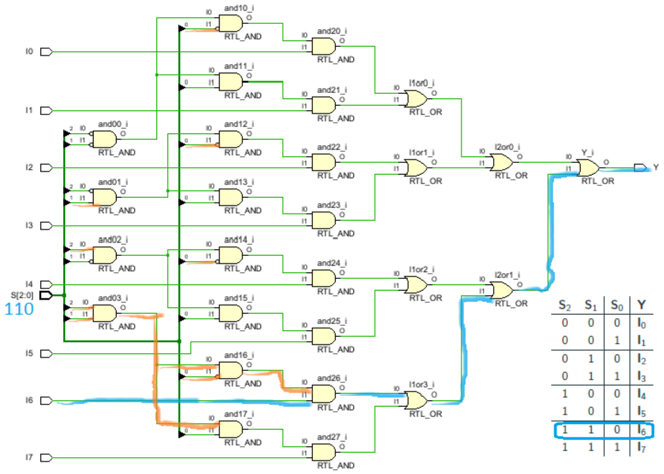
Listing 6: Concurrent Assignment Statements

8-to-1-Line Multiplexer - VHDL Code - Five

```
    l1or0 <= and20 or and21 after 4ns;
    l1or1 <= and22 or and23 after 4ns;
    l1or2 <= and24 or and25 after 4ns;
    l1or3 <= and26 or and27 after 4ns;

    l2or0 <= l1or0 or l1or1 after 4ns;
    l2or1 <= l1or2 or l1or3 after 4ns;

    Y <= l2or0 or l2or1 after 4ns;

end Behavioral;
```

Listing 7: Concurrent Assignment Statements

8-to-1-Line Multiplexer Example



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Figure 7: $S =' 110'$ selects I6 as input

Multiplexer to select Register in the Register File

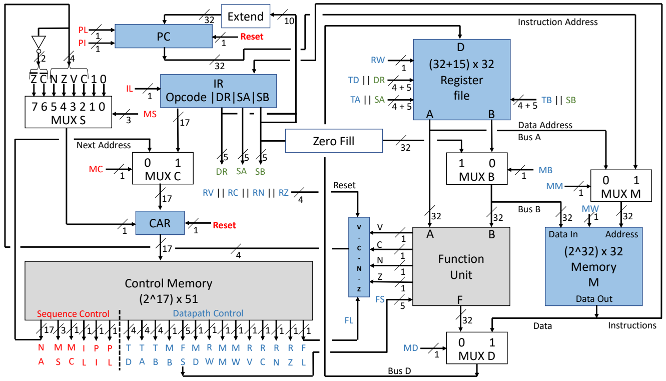


Figure 8: **SA**,**SB**,**TA** and **TB** provide the address for the Multiplexer