CSU22022 Computer Architecture I

Fourth Lecture

Michael Manzke

2023-2024

Trinity College Dublin

3-to-8-Line-Decoder

x	y	z	$\mathbf{D}_i$
0	0	0	$\mathbf{D}_0 = x'y'z'$
0	0	1	$\mathbf{D}_1 = x'y'z$
0	1	0	$\mathbf{D}_2 = x'yz'$
0	1	1	$\mathbf{D}_3 = x'yz$
1	0	0	$\mathbf{D}_4 = xy'z'$
1	0	1	$\mathbf{D}_5 = xy'z$
1	1	0	$\mathbf{D}_6 = xyz'$
1	1	1	$\mathbf{D}_7 = xyz$

Table 1: 3-to-8-Line-Decoder Truth Table

3-input Logic AND Gate

- The truth table 1 for the 3-to-8-Line-Decoder shows a need for a 3-input logic AND gate.

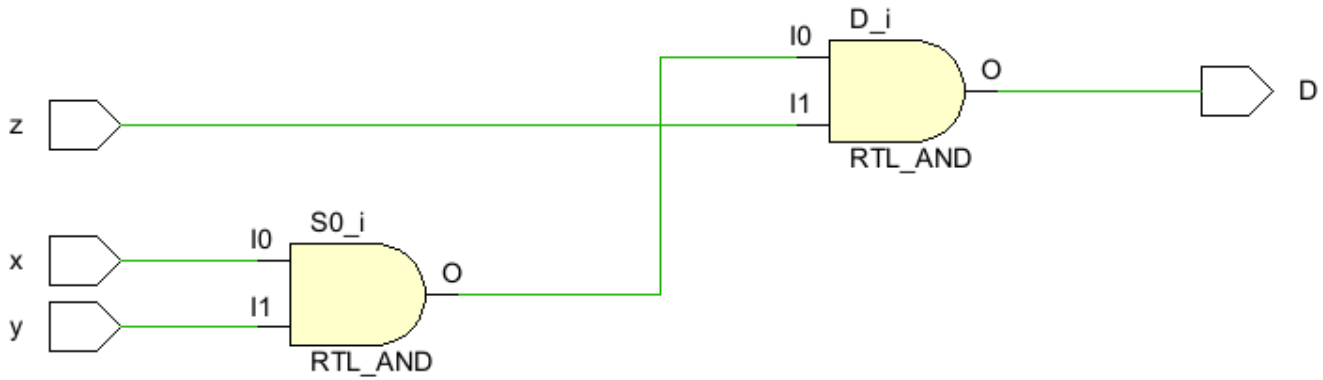


Figure 1: Two AND gates that implement a 3-input logic AND gate.

3-input Logic AND Gate VHDL Code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity AND3inputs is
5      Port ( z : in STD_LOGIC;
6             y : in STD_LOGIC;
7             x : in STD_LOGIC;
8             D : out STD_LOGIC);
9  end AND3inputs;
10
11 architecture Behavioral of AND3inputs is
12    signal S0 : std_logic;
13 begin
14     S0 <= x and y after 8ns;
15     D <= S0 and z after 8ns;
16 end Behavioral;
```

Listing 1: This code implements the 3-input logic AND gate

3-to-8-Line-Decoder Schematic

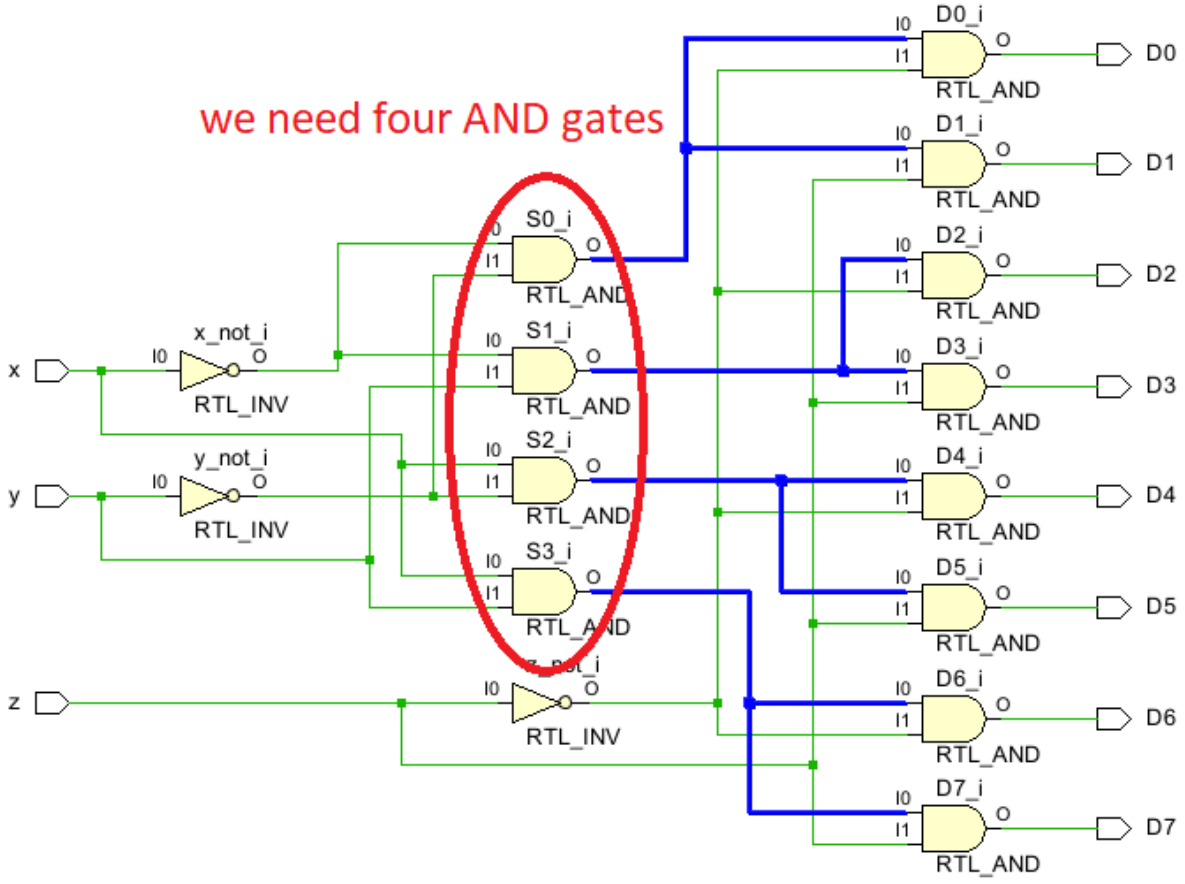


Figure 2: The truth table 1 shows that x and y are the same for $D_0 D_1$, $D_2 D_3$, $D_4 D_5$, and $D_6 D_7$. Therefore, we need four AND gates

3-to-8-Line-Decoder

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Decoder_3_to_8_Line is
5      Port ( x, y, z : in STD_LOGIC;
6             D0, D1, D2, D3, D4, D5, D6, D7 : out STD_LOGIC);
7  end Decoder_3_to_8_Line;
8
9  architecture Behavioral of Decoder_3_to_8_Line is
10
11     signal x_not, y_not, z_not : std_logic;
12     signal S0, S1, S2, S3 : std_logic;
```

Listing 2: entity

3-to-8-Line-Decoder

```
1  begin
2       x_not <= not x after 7ns;
3       y_not <= not y after 7ns;
4       z_not <= not z after 7ns;
5       S0 <= x_not and y_not after 8ns;
6       D0 <= S0 and z_not after 8ns;
7       D1 <= S0 and z after 8ns;
8       S1 <= x_not and y after 8ns;
9       D2 <= S1 and z_not after 8ns;
10      D3 <= S1 and z after 8ns;
11      S2 <= x and y_not after 8ns;
12      D4 <= S2 and z_not after 8ns;
13      D5 <= S2 and z after 8ns;
14      S3 <= x and y after 8ns;
15      D6 <= S3 and z_not after 8ns;
16      D7 <= S3 and z after 8ns;
17 end Behavioral;
```

Listing 3: Signal assignment statements

3-to-8-Line-Decoder Example

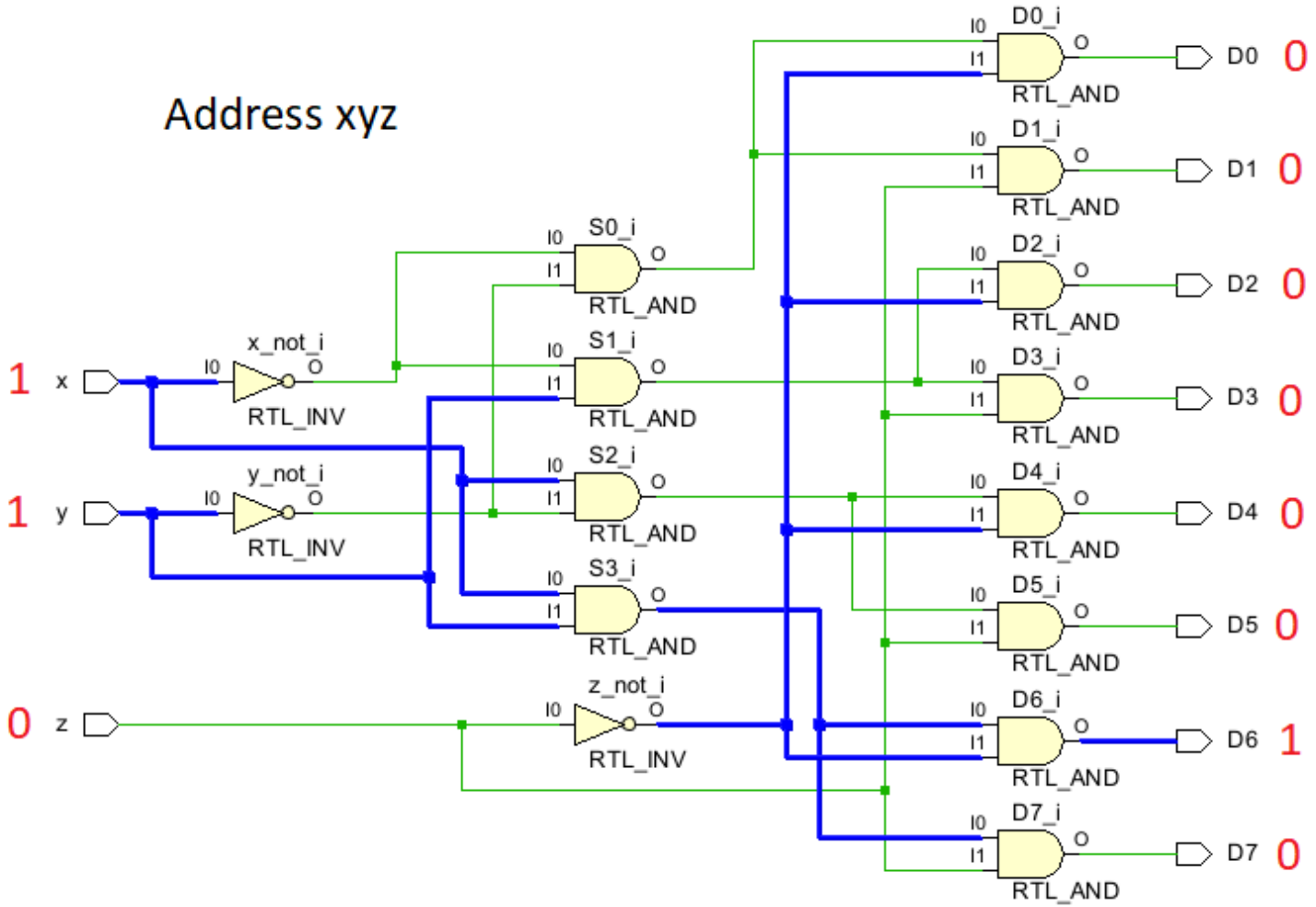


Figure 3: Address 110 selects D6

3-to-8-Line-Decoder another Example

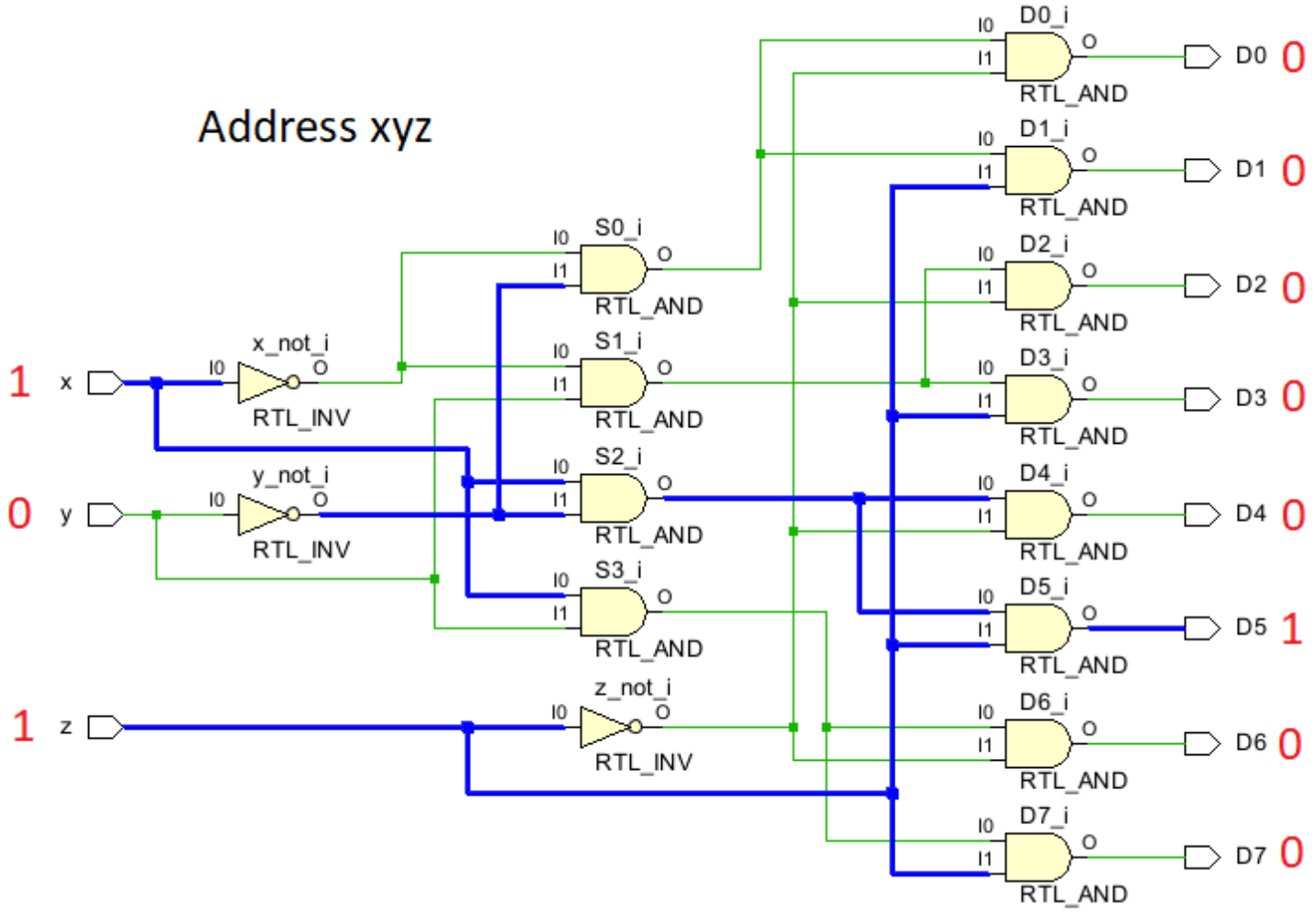


Figure 4: Address 101 selects D5

3-to-8-Line-Decoder Simulation Code One

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Decoder_3_to_8_Line_TB is
5  --  we don't need ports
6  end Decoder_3_to_8_Line_TB;
7
8  architecture Simulation of Decoder_3_to_8_Line_TB is
9
10    -- Component Declaration for the Unit Under Test (UUT)
11
12     COMPONENT Decoder_3_to_8_Line
13     Port ( x, y, z : in STD_LOGIC;
14            D0, D1, D2, D3, D4, D5, D6, D7 : out STD_LOGIC);
15     END COMPONENT;
```

Listing 4: 3-to-8-Line-Decoder_TB entity

3-to-8-Line-Decoder Simulation Code Two

```
1    --Inputs Signals
2    signal x_TB, y_TB, z_TB : STD_LOGIC := '0';
3
4    --Output Signal
5    signal D0_TB, D1_TB, D2_TB, D3_TB : STD_LOGIC := '0';
6    signal D4_TB, D5_TB, D6_TB, D7_TB : STD_LOGIC := '0';
7
8    -- StudentID e.g. 26 33 57 25(DEC) = 1 91 D9 ED(HEX)
9    constant StudentID : STD_LOGIC_VECTOR (27 downto 0) := x"
      191D9ED";
```

Listing 5: signal and constant

3-to-8-Line-Decoder Simulation Code Three

```
1  begin
2
3     -- Instantiate the Unit Under Test (UUT)
4     uut: Decoder_3_to_8_Line PORT MAP (
5             x => x_TB,
6             y => y_TB,
7             z => z_TB,
8             D0 => D0_TB,
9             D1 => D1_TB,
10            D2 => D2_TB,
11            D3 => D3_TB,
12            D4 => D4_TB,
13            D5 => D5_TB,
14            D6 => D6_TB,
15            D7 => D7_TB
16          );
```

Listing 6: Port map for the Decoder_3_to_8_Line entity

3-to-8-Line-Decoder Simulation Code Four

```
stim_proc: process
   begin
      x_TB <= '0'; y_TB <= '0'; z_TB <= '0'; -- case A
      wait for 60 ns;
      x_TB <= '0'; y_TB <= '0'; z_TB <= '1'; -- case B
      wait for 60 ns;
      x_TB <= '0'; y_TB <= '1'; z_TB <= '0'; -- case C
      wait for 60 ns;
      x_TB <= '0'; y_TB <= '1'; z_TB <= '1'; -- case D
      wait for 60 ns;
      x_TB <= '1'; y_TB <= '0'; z_TB <= '0'; -- case E
      wait for 60 ns;
      x_TB <= '1'; y_TB <= '0'; z_TB <= '1'; -- case F
      wait for 60 ns;
      x_TB <= '1'; y_TB <= '1'; z_TB <= '0'; -- case G
      wait for 60 ns;
      x_TB <= '1'; y_TB <= '1'; z_TB <= '1'; -- case H
      wait for 60 ns;
   end process;
end Simulation;
```

3-to-8-Line-Decoder Timing Diagram

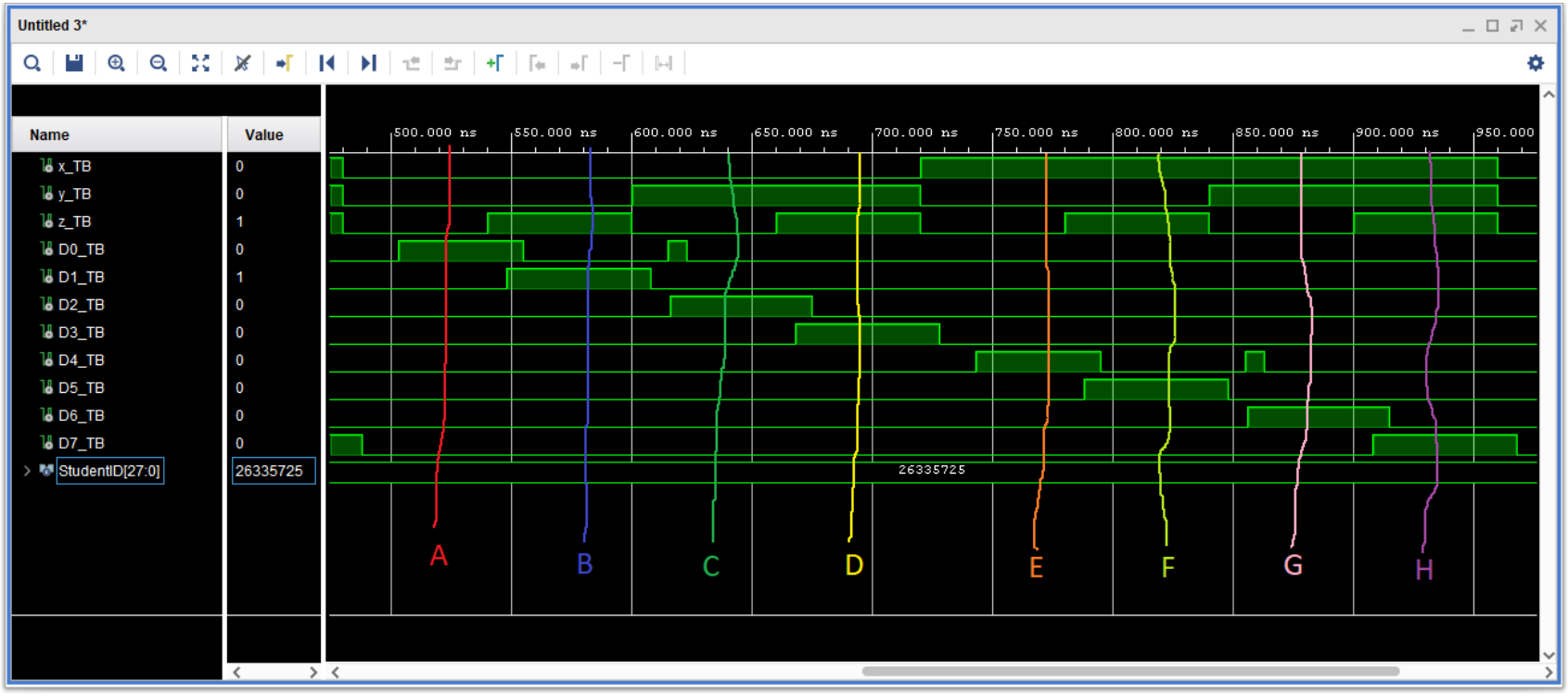


Figure 5: Cases A to H

3-to-8-Line-Decoder Propagation Delay

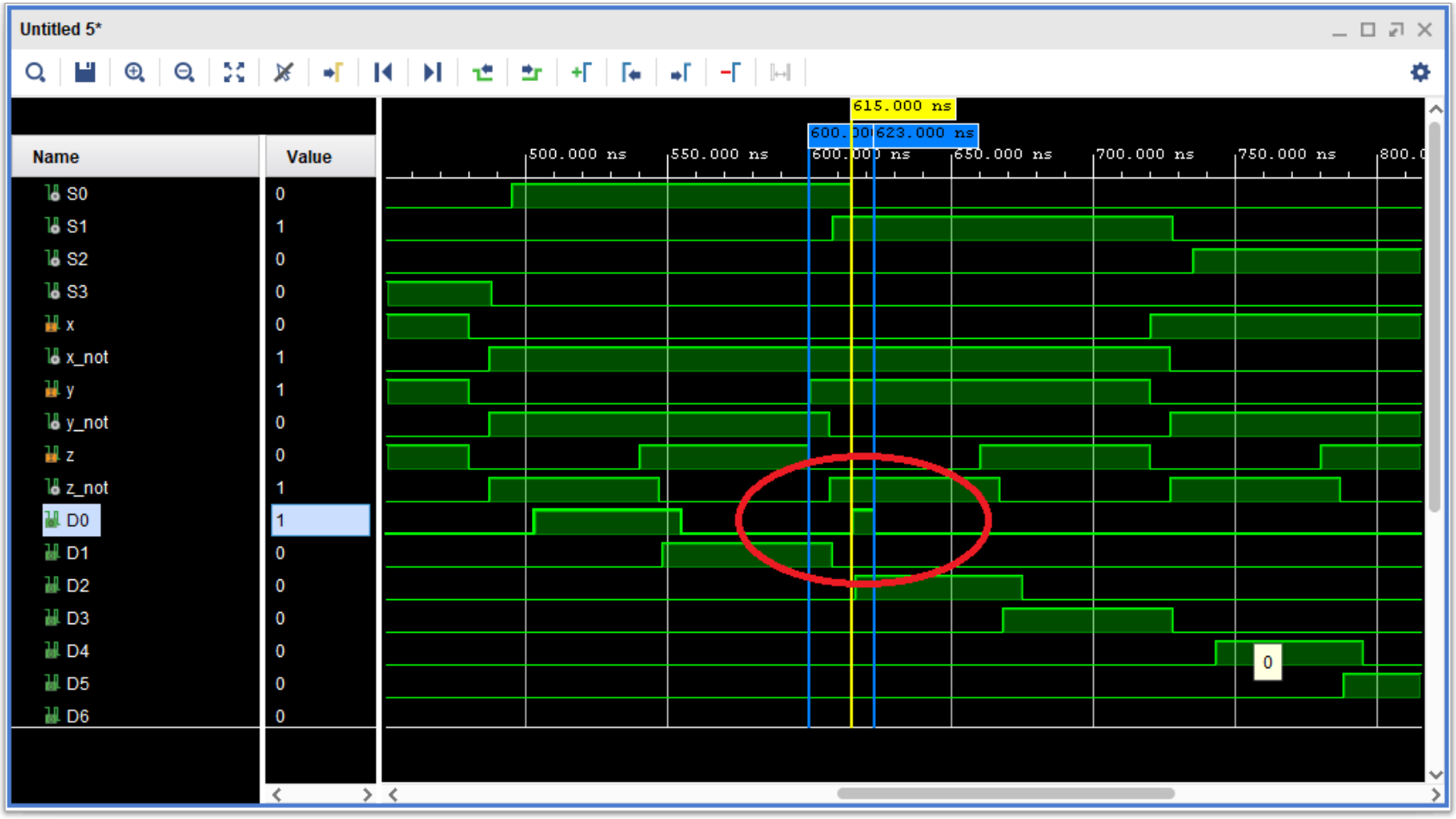
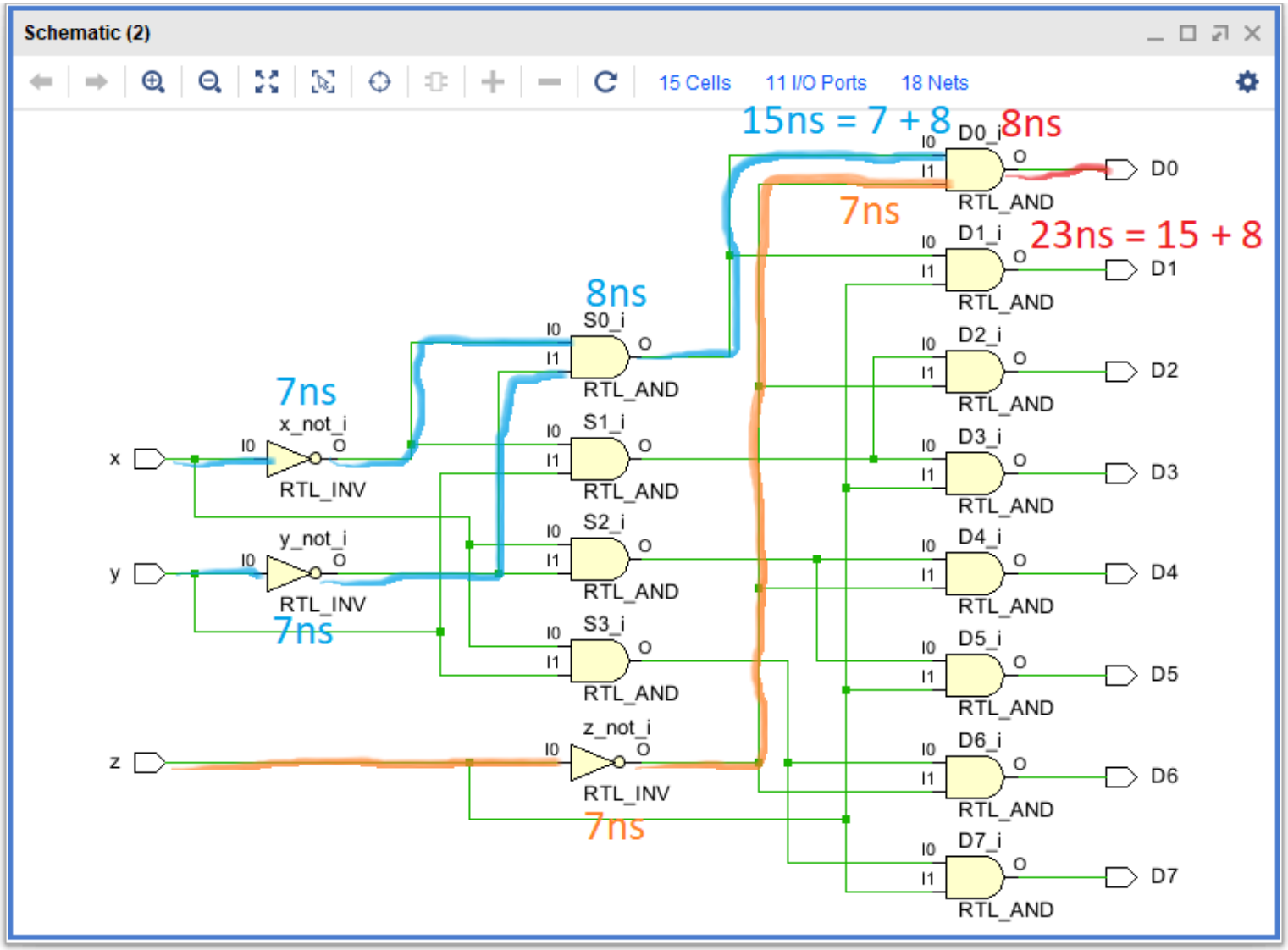


Figure 6: A result is only valid after all signals propagated through the network

Propagation Delay Example



Decoder to select Register in the Register File

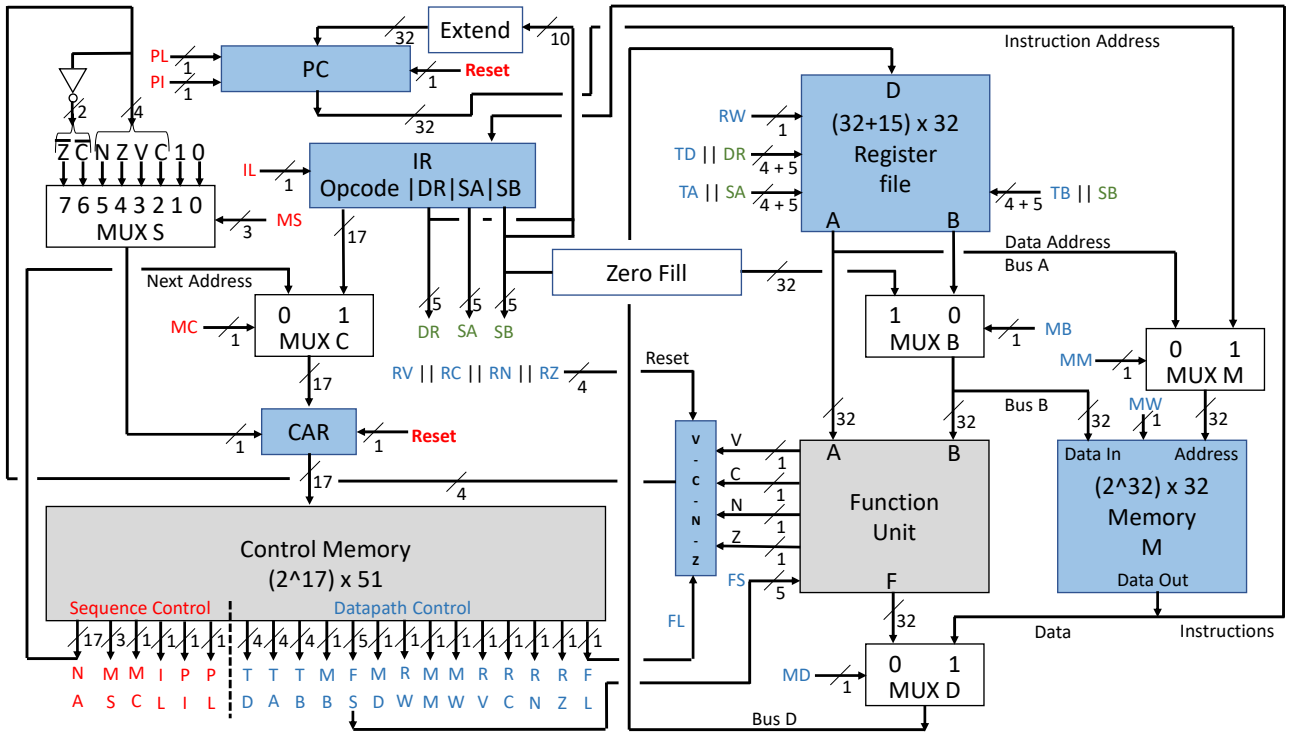


Figure 7: **DR** and **TD** provide the address for the decoder