DETAILED REPORT ON MULTIPLEXER

Trương Quang Huy
Email: quanghuytruong2003@gmail.com

1. Introduction

A 16:1 multiplexer (MUX) is a combinational digital circuit that select one input out of 16 available inputs and routes it to a single output based on the binary value of 4 selection lines (S3, S2, S1, S0). Additionally, the inclusion of an enable (E) line provides a critical control mechanism to activate or deactivate the multiplexer. When the enable line is active (commonly logic 1), the MUX operates as expected, routing the selected input to the output, when the enable line is inactive (logic 0), the MUX output is disabled (typically held at logic 0 or high-impedance state, depending on implementation).

The addition of the enable signal makes the 16:1 MUX more versatile , as it can be controlled externally to enable or disable functionality as required. This is especially useful in large-scale systems with multiplexers operating in parallel.

2. Structure and functionality

2.1 Number of inputs, selection lines and enable line

The 16:1 MUX consits of:

- 16 input lines (I0, I1, …., I15): The represent the data or signals from which one is selected.
- 4 Selection line (S3, S2, S1, S0): these are used to select one input out of the 16 based on the binary address formed by their values.
- 1 enable line (E): This activates or deactivates the MUX operation.
- 1 output line (Y): this represents the routed output of the selected input.

2.2 Truth table

The truth table of the 16:1 MUX with an enable line describes its behaviour for various input combinations of selection lines and enable signal.

Enable (E)	S_3	S_2	S_1	S_0	Selected input (Y)
0	X	X	X	X	0
1	0	0	0	0	I_0
1	0	0	0	1	I_1
1	0	0	1	0	I_2

1	0	0	1	1	I_3
1	0	1	0	0	I_4
1	0	1	0	1	I_5
1	0	1	1	0	I_6
1	0	1	1	1	I_7
1	1	0	0	0	I_8
1	1	0	0	1	I_9
1	1	0	1	0	I_{10}
1	1	0	1	1	I_{11}
1	1	1	0	0	I_{12}
1	1	1	0	1	I_{13}
1	1	1	1	0	I_{14}
1	1	1	1	1	I_{15}

Key note:

- ✓ When E = 0: the output Y is disabled (logic 0 or high impedance)
- ✓ When E = 1; the multiplexer behaves as a standard 16:1 MUX

3. Boolean logic representation.

The boolean equation for the output (Y) of a 16:1 multiplexer with enable is:

$$Y = E[(S'_3.S'_2.S'_1.S'_0.I_0) + (S'_3.S'_2.S'_1.S_0.I_1) + \ldots + (S_3.S_2.S_1.S_0.I_{15})]$$

This expression ensures that the output is only active when E = 1.

4. Circuit design and implementation

4.1 Logic gate design

The circuit is constructed using:

- ➢ 16 AND gates: each gate corresponds to one input line and ensures that the input signals is passed only when the selection line combination matches.
- ➢ 4 Not gates: these generate the inverted signals of the selection lines for inputs requiring a logic 0.
- ➢ 1 OR gate: Combines the outputs of all AND gate to produce the final output
- ➢ Enable logic: the enable logic line is connected to each AND gate to control whether the MUX is active or not.

4.2 Cascading for implementation

A 16:1 MUX can also be implemented by cascading smaller multiplexers:

- ➢ Use two 8:1 MUXes to handle the 16 inputs.
- ➢ Combine their outputs using a 2:1 MUX controlled by the most significant line (S3).
- ➢ The enable line can be fed into Both 8:1 MUXes and the final 2:1 MUX to ensure consistent control.

5. Application of 16:1 Multiplexer with enable line.

a) Data routing in digital systems:

Used to select one multiple data sources to route to a shared resource like a bus or processing unit

b) Control units in microprocessors.

Microprocessors use 16:1 MUXes to select one control signal out of many, based on the current operation or instruction.

c) Address selection in memory systems:

A 16:1 MUX can select one memory addresses based on the address bit provided as selection lines.

d) Communication Networks

MUXes enable the efficient transmission of multiple signals over a single channel using time-division multiplexing.

e) Signal processing:

Audio and video systems use multiplexers to switch between different input sources or processing units.

f) Test and Debugging systems:

MUXes can route signals from different parts of a system to a single output for testing or debugging purposes.

6. Advantage and limitations

6.1 Advantage:

- ✓ Versatility: the enable line allows external control over the circuit's operation.
- ✓ Scalability: Handles a large number of inputs minimal control lines.
- ✓ Simplicity: Reduces the need for multiple signal lines, simplifying wiring in complex systems.

6.2 Limitations:

- ✓ Propagation Delay: large MUXes introduce delays due to multiple logic gates.
- ✓ Power consumption: the use of more gates increases power requirements.
- ✓ Hardware cost: the inclusion of additional gates and enable lofic increases implementation cost.

7. Design multiplexer 16:1

7.1 Block diagram

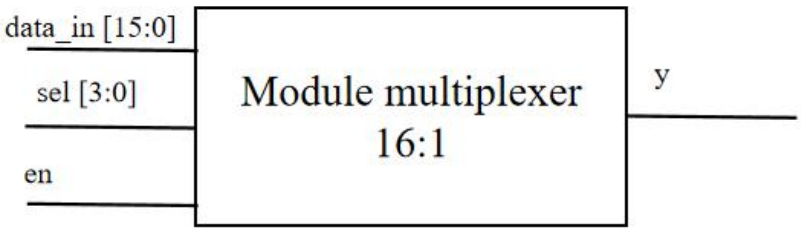

Image: Block diagram of MUX 16:1

Describe IO Table

Signal	Direction	Bit-Width	Description
Data_in	Input	16	Input data of binary signal
Sel	Input	4	The value of the binary signal on the select pins determines which input will be passed to the output.
En	Input	1	Enable operation when en = 1 and opposite
Y	Output	1	Output data

Waveform expected:

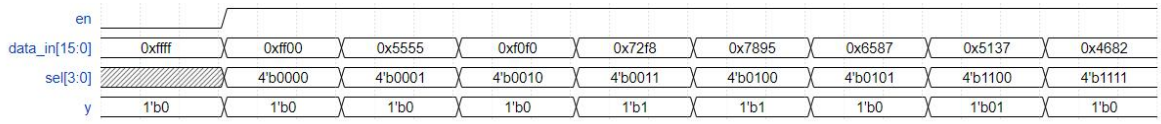

Image: The expected waveform

Code wavedrom:

```
{signal: [
{name: 'en', wave: '0..1.....................'},
{name: 'data_in[15:0]', wave:
'=..=..=..=..=..=..=..=..=..',data:["0xffff","0xff00","0x5555","0xf0f0","0x72f8","0x7895","0x6587","0x5137","0x4682"]},
{name: 'sel[3:0]', wave:
'x..=..=..=..=..=..=..=..=..',data:["4'b0000","4'b0001","4'b0010","4'b0011","4'b0100","4'b0101","4'b1100","4'b1111" ]},
{name: 'y', wave:
'=..=..=..=..=..=..=..=..=..',data:["1'b0","1'b0","1'b0","1'b0","1'b1","1'b1","1'b0","1'b01","1'b0"]},
]}
```

RTL code for MUX 16:1.

```
module mux16to1 (
        input wire [15:0] data_in,
        input wire [3:0] sel,
        input wire en,
        output reg y
);
```

```
always @(*) begin
        y = (!en) ? 1'b0 : data_in[sel];
end

endmodule
```

Testbench for decoder 3 to 8.

To verify the functionality of the MUX 16:1 circuit, a testbench is used. The testbench simulates the input signals with different test cases and checks if the output matches the expected result. The testbench also prints out whether each test case passed or failed.

Truth table:

The truth table below show the expected result for each test case:

Testcase	en	Data_in	sel	Y
1	0	0xffff	4'b1110	0
2	1	0xff00	4'b0000	0
3	1	0x5555	4'b0001	0
4	1	0xf0f0	4'b0010	0
5	1	0x72f8	4'b0011	1
6	1	0x7895	4'b0100	1
7	1	0x6587	4'b0101	0
8	1	0x5137	4'b1100	1
9	1	0x4682	4'b1111	0

TestBench code:

```
`timescale 1ns/1ps
module test_bench;
        reg [15:0] data_in;
        reg [3:0] sel;
        reg en;
        wire y;

        mux16to1 uut(
                .data_in(data_in),
                .sel(sel),
                .en(en),
                .y(y)
        );
```

```
	integer i = 0;

	initial begin
		$dumpfile("test_bench.vcd");
		$dumpvars(0, test_bench);

		$display("------------------------------------------------------------
-----------------");
		$display("----------------------TESTBENCH                FOR
MULTIPLEXER 16:1-----------------------");
		$display("------------------------------------------------------------
-----------------");

		for (i = 0; i < 16; i = i + 1) begin
			en = 0;
			data_in = $random % 65535;
			sel = i;
			#1;
			check_result(1'b0);
			#10;
		end

		for (i = 0; i < 16; i = i + 1) begin
			en = 1;
			data_in = $random % 65535;
			sel = i;
			#1;
			check_result(calc_expected(data_in,sel,en));
			#10;
		end

		for (i = 0; i < 20; i = i+ 1) begin
			en = $random % 2;
			data_in = $random % 65535;
			sel = $random % 16;
			#1;
			check_result(calc_expected(data_in,sel,en));
			#10;
		end

		#100;
		$finish;
```

```
        end

        function calc_expected (input [15:0] in, input [3:0] sel_val, input
en_val);
                begin
                        calc_expected = (!en_val) ? 0: in[sel_val];
                end
        endfunction


        task check_result;
                input expected_result;
                begin
                        $display("At time: %t, en = 1'b%b, sel = 4'b%b,
data_in = 16'b%b", $time, en, sel, data_in);
                        if ( y == expected_result) begin
                                $display("----------------------------------------
---------------------------------");
                                $display("PASSED: expected: 1'b%b, Got:
1'b%b", expected_result, y);
                                $display("----------------------------------------
---------------------------------");
                        end else begin
                                $display("----------------------------------------
---------------------------------");
                                $display("FAILED: expected: 1'b%b, Got:
1'b%b", expected_result, y);
                                $display("----------------------------------------
---------------------------------");
                        end
                end
        endtask
endmodule
```

Simulation:

```
vp test_bench
VCD info: dumpfile test_bench.vcd opened for output.
--------------------------------------------------------------------------------
----------------------TESTBENCH FOR MULTIPLEXER 16:1----------------
------------
--------------------------------------------------------------------------------
```

```
At time:                1000, en = 1'b0, sel = 4'b0000, data_in = 16'b0100011100111001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                12000, en = 1'b0, sel = 4'b0001, data_in = 16'b0001111100001011
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                23000, en = 1'b0, sel = 4'b0010, data_in = 16'b0101101010001110
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                34000, en = 1'b0, sel = 4'b0011, data_in = 16'b0000100001010100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                45000, en = 1'b0, sel = 4'b0100, data_in = 16'b1000000111000110
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                56000, en = 1'b0, sel = 4'b0101, data_in = 16'b1110000001101100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                67000, en = 1'b0, sel = 4'b0110, data_in = 16'b0011011100101000
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                78000, en = 1'b0, sel = 4'b0111, data_in = 16'b1101101101001001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                89000, en = 1'b0, sel = 4'b1000, data_in = 16'b1110001111110100
--------------------------------------------------------------------------------
```

```
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 100000, en = 1'b0, sel = 4'b1001, data_in = 16'b1101001111100100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 111000, en = 1'b0, sel = 4'b1010, data_in = 16'b0010110010011010
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 122000, en = 1'b0, sel = 4'b1011, data_in = 16'b1110101111001010
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 133000, en = 1'b0, sel = 4'b1100, data_in = 16'b1100111011000001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 144000, en = 1'b0, sel = 4'b1101, data_in = 16'b0011110110111010
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 155000, en = 1'b0, sel = 4'b1110, data_in = 16'b0110011011110111
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 166000, en = 1'b0, sel = 4'b1111, data_in = 16'b0000011111111110
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 177000, en = 1'b1, sel = 4'b0000, data_in = 16'b0110011110111101
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                 188000, en = 1'b1, sel = 4'b0001, data_in =
```

16'b1010011110111110

PASSED: expected: 1'b1, Got: 1'b1

At time: 199000, en = 1'b1, sel = 4'b0010, data_in = 16'b0110101010010101

PASSED: expected: 1'b1, Got: 1'b1

At time: 210000, en = 1'b1, sel = 4'b0011, data_in = 16'b0010111001001010

PASSED: expected: 1'b1, Got: 1'b1

At time: 221000, en = 1'b1, sel = 4'b0100, data_in = 16'b0101111101000101

PASSED: expected: 1'b0, Got: 1'b0

At time: 232000, en = 1'b1, sel = 4'b0101, data_in = 16'b0010001101111100

PASSED: expected: 1'b1, Got: 1'b1

At time: 243000, en = 1'b1, sel = 4'b0110, data_in = 16'b1110001100100010

PASSED: expected: 1'b0, Got: 1'b0

At time: 254000, en = 1'b1, sel = 4'b0111, data_in = 16'b0111111001000101

PASSED: expected: 1'b0, Got: 1'b0

At time: 265000, en = 1'b1, sel = 4'b1000, data_in = 16'b0110111011101001

PASSED: expected: 1'b0, Got: 1'b0

At time: 276000, en = 1'b1, sel = 4'b1001, data_in = 16'b0011000110010000

PASSED: expected: 1'b0, Got: 1'b0

```
--------------------------------------------------------------------------------
At time:                 287000, en = 1'b1, sel = 4'b1010, data_in =
16'b0111011110110100
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                 298000, en = 1'b1, sel = 4'b1011, data_in =
16'b0000011101001100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 309000, en = 1'b1, sel = 4'b1100, data_in =
16'b1110111011011000
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 320000, en = 1'b1, sel = 4'b1101, data_in =
16'b1101100100001100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 331000, en = 1'b1, sel = 4'b1110, data_in =
16'b0001010001010011
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 342000, en = 1'b1, sel = 4'b1111, data_in =
16'b1000110001111101
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                 353000, en = 1'b0, sel = 4'b1010, data_in =
16'b0011000110000100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 364000, en = 1'b1, sel = 4'b0011, data_in =
16'b0000100110110111
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                 375000, en = 1'b1, sel = 4'b1011, data_in =
16'b0111111111111011
```

```
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                386000, en = 1'b1, sel = 4'b1110, data_in =
16'b1100101100011001
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                397000, en = 1'b1, sel = 4'b0011, data_in =
16'b0101101010010101
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                408000, en = 1'b0, sel = 4'b1100, data_in =
16'b1111000000111101
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                419000, en = 1'b0, sel = 4'b0001, data_in =
16'b1010011010111000
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                430000, en = 1'b0, sel = 4'b1001, data_in =
16'b0010111110011001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                441000, en = 1'b1, sel = 4'b0110, data_in =
16'b1100000010111000
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                452000, en = 1'b0, sel = 4'b1010, data_in =
16'b1110000100110001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                463000, en = 1'b1, sel = 4'b0101, data_in =
16'b0111011100001001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
```

```
At time:                474000, en = 1'b1, sel = 4'b1010, data_in = 16'b1111111000000111
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                485000, en = 1'b0, sel = 4'b0001, data_in = 16'b0101010110111111
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                496000, en = 1'b1, sel = 4'b1100, data_in = 16'b0011100010000101
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                507000, en = 1'b1, sel = 4'b1000, data_in = 16'b1110100101001000
--------------------------------------------------------------------------------
PASSED: expected: 1'b1, Got: 1'b1
--------------------------------------------------------------------------------
At time:                518000, en = 1'b1, sel = 4'b1100, data_in = 16'b0010011001100100
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                529000, en = 1'b1, sel = 4'b1001, data_in = 16'b0111110001100111
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                540000, en = 1'b0, sel = 4'b0001, data_in = 16'b1010110011010110
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                551000, en = 1'b0, sel = 4'b0010, data_in = 16'b1000010100101001
--------------------------------------------------------------------------------
PASSED: expected: 1'b0, Got: 1'b0
--------------------------------------------------------------------------------
At time:                562000, en = 1'b0, sel = 4'b1101, data_in = 16'b0001010111111010
--------------------------------------------------------------------------------
```

PASSED: expected: 1'b0, Got: 1'b0

Waveform: