DETAILED REPORT ON DECODER AND PRIORITY ENCODER

Trương Quang Huy
Email: quanghuytruong2003@gmail.com

1. Introduction

Decoder and priority encoder are fundamental digital circuits used in various applications such as memory addressing, control systems and data encoding. These circuits are designed to transform binary data into a more useful or manageable form, helping streamline operations within complex systems. A decoder takes a binary input and actives one or more output lines, corresponding to the binary input, while a priority encoder encodes multiple input lines into a smaller number of output lines, prioritizing the highest active input.

In this report, we will delve deeper into the operation, types and applications of both decoder and priority encoder, highlighting their significance in modern digital systems.

2. Decoder

A decoder is a combinational logic circuit the converts an n-bit binary input into one of 2^n possible output lines. The output of the decoder is typically used to activate one of several lines in systems, ensuring that a specific action is taken based on the input. In most systems, decoder are used for address decoding, where a given binary input is used to select one of many possible addresses.

2.1 Basic operation

The basic operation of a decoder is straightforward: it takes an n-bit binary number as input and activate the output line that corresponds to that binary value. For example, a 3-to-8 decoder takes 3 input bits and has 8 output lines. Each input combination from 000 to 111 will correspond to one active output.

In a 3-to-8 decoder, the outputs are typically connected to the binary values from 000 to 111, where only the output corresponding to the input binary code is active, while all others remain low. This can be understood as selecting a particular line or action from many options.

2.2 Types of Decoder

There are several types of decoders, each suited for different applications:

- **Binary Decoder:** This is the most common form of a decoder. It takes a binary input and generate a corresponding output .
- **Decimal-to-binary Decoder:** in this case, the input is a decimal number, which the decoder convert into its binary equivalent. This

form is used in various systems where decimal numbers are input but need to be processed or stored in binary format.

- **BCD (binary Coded Decimal) Decoder**: A special type of decimal-to-binary decoder, which converts BCD numbers (each decimal digit is represented by four binary digits) into their binary equivalents.
- **7-Segment display decoder**: Used in digital displays, this decoder takes a 4-bit BCD input and drives the seven-segment display accordingly. A 4-to 16 BCD decoder can drive a 7-segment display for all decimal digits (0-9).

2.3 Applications of Decoders

- Memory Addressing: in computer systems, decoders are crucial for selecting specific memory addresses. They take the address input and enable the corresponding memory location, allowing for efficient memory management.
- Display Drivers: Decoders are commonly used in driving 7-segment displays or other types of visual indicators, converting a binary or BCD input into the correct signal for each display segment.
- Multiplexing: Decoder can be used in multiplexers to select one of many possible data inputs. The decoder generates the appropriate select signals, allowing for multiple inputs to be transmitted over a single output.
- Data routing: Decoders can route data form one source to multiple destinations by selecting the correct output line based on the input address. This is often used in communication systems and large data networks.

3. Priority Encoder.

A priority encoder is a type of combinational circuit that converts multiple input lines into a binary code corresponding to the highest-priority active input. Priority encoders are used when there are multiple active inputs and only the one with the highest priority needs to be considered.

3.1 Basic Operation

In a priority encoder, each input is assigned a priority. When multiple inputs are active simultaneously, the encoder will output the binary value corresponding to the highest-priority encoder, there are 4 input lines and the encoder generates a 2-bit output that corresponds to the binary representation of the highest-priority input.

For instance:

- ✓ If input 1,2 and 3 are active, the output will be the binary representation of input 3, as it has the highest priority.
- ✓ If only input 2 is active, the output will represent input 2, and so on.

A key feature of priority encoders is that they also often include a valid signal that indicates whether any input is active. This prevents invalid outputs when no inputs are active.

3.2 Types of priority encoders.

Priority encoders can be classified based on the number of input lines and the corresponding output lines.

- 4-to-2 priority encoder: this type of encoder takes 4 input lines and produces a 2-bit binary output. It ensure that only the highest-priority active input is encoded.
- 8-to-3 priority encoder: This encoder takes 8 input lines and produce a 3-bit binary output. It works in the same way as a 4-to-2 encoder

3.3 Application of priority encoder.

- ✓ Interrupt systems: Priority encoders are used in interrupt systems where multiple interrupt sources may occur simultaneously, ensuring the highest priority interrupt is processed first.
- ✓ Digital communication systems: priority encoders are used in systems where data needs to transmitted with priority, ensuring critical data is transmitted first.
- ✓ Digital logic design: priority encoders are often employed in the design of digital systems where multiple conditions might trigger different actions and the system needs to ensure that the highest-priority condition is executed.
- ✓ Error handing: priority encoders are used in error detection systems to identify which error condition is the most critical, enabling immediate attention to the highest-priority error.

4. Comparison between Decoder and priority encoder.

Feature	Decoder	Priority encoder
Function	Converts binary input to a unique output	Encoder active input with the highest priority
Number of inputs	n inputs to 2^n outputs	2^n inputs to n outputs
Output	One active output for each input	Binary representation of the highest-priority active input
Application	Address decoding, display driver, multiplexers	Interrupt systems, data transmission, error handling

Both decoder and priority encoders are essential in modern electronics for routing signals and encoding data. While decoders handle address and display applications, priority encoders are critical for systems where multiple inputs need to be managed based on priority

5. Design

5.1 Design decoder binary circuit

Block diagram:

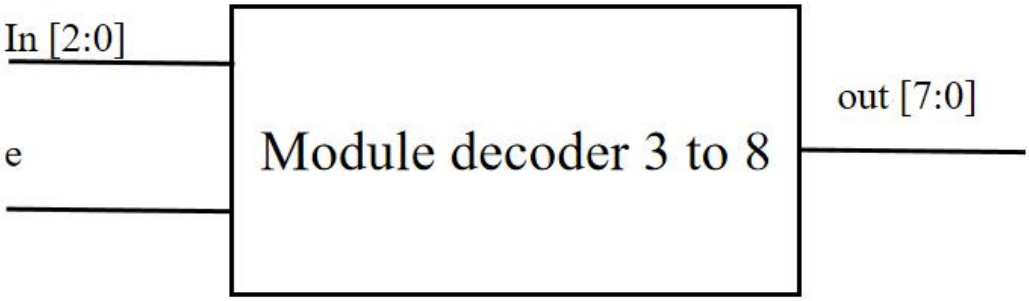


Image 01: Block diagram module 3 to 8

Describe IO Table

Signal	Direction	Bit-Width	Description
In	Input	3	Input data of binary signal
e	Input	1	Enable pin to control the circuit.
Out	Output	8	Output data

Waveform expected:

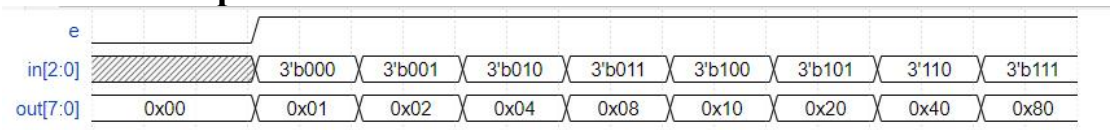


Image 02: Waveform expected for module decoder 3 to 8

Code wavedrom:

```
{signal: [
{name: 'e', wave: '0..1.....'},
{name: 'in[2:0]', wave:
'x..=.=.=.=.=.=.=.=.',data:["3'b000","3'b001","3'b010",
"3'b011","3'b100","3'b101","3'110", "3'b111" ]},
{name: 'out[7:0]', wave: '=..=.=.=.=.=.=.=.=.',data:["0x00","0x01","0x02",
"0x04", "0x08","0x10","0x20","0x40", "0x80"]},
]}
```

RTL code of decoder 3to8:

The following Verilog code implement the decoder 3 to 8 circuit. Which takes one 3-bit input signals and output a single 8-bit .

```
module binary_decoder_3to8 (
	input wire [2:0] in,
	input wire e,
	output reg [7:0] out
);

always @(*) begin
	if (e == 1'b0) begin
		out <= 8'b0000_0000;
	end else if (e == 1'b1) begin
		case (in)
			3'b000: out <= 8'b0000_0001;
			3'b001: out <= 8'b0000_0010;
			3'b010: out <= 8'b0000_0100;
			3'b011: out <= 8'b0000_1000;
			3'b100: out <= 8'b0001_0000;
			3'b101: out <= 8'b0010_0000;
			3'b110: out <= 8'b0100_0000;
			3'b111: out <= 8'b1000_0000;
			default: out <= 8'b0000_0000;
		endcase
	end else begin
		out <= 8'b0000_0000;
	end
end
endmodule
```

Testbench for decoder 3 to 8.

To verify the functionality of the decoder 3 to 8 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	e	In	Out
1	0	3’b101	8’b0000_0000
2	0	3’b100	8’b0000_0000
3	1	3’b000	8’b0000_0001
4	1	3’b001	8’b0000_0010
5	1	3’b010	8’b0000_0100
6	1	3’b011	8’b0000_1000
7	1	3’b100	8’b0001_0000

8	1	3’b101	8’b0010_0000
9	1	3’b110	8’b0100_0000
10	1	3’b111	8’b1000_0000

Testbench code:

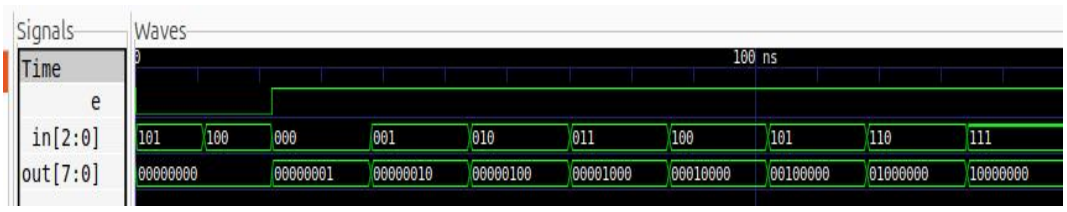
```
// testbench for binary decoder 3 to 8
`timescale 1ns/1ps

module test_bench;
	reg [2:0] in;
	reg e;
	wire [7:0] out;

binary_decoder_3to8 uut(
	.in(in),
	.e(e),
	.out(out)
);

integer i;

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

	$display("------------------------------------------------------------------
-");
	$display("---------- TESTBENCH FOR BINARY DECODER 3 TO
8----------------------");
	$display("------------------------------------------------------------------
-");

	e = 0;
	in = 3'b101;
	#1;
	check_result(8'h00);
	#10;

	e = 0;
	in = 3'b100;
	#1;
	check_result(8'h00);
```

```
        #10;

        for (i = 0; i < 8; i = i + 1) begin
                e = 1;
                in = i;
                #1;
                check_result(8'h01 << in);
                #15;
        end

        #100;
        $finish;
end

task check_result;
        input [7:0] expected_result;
        begin
        $display("At time: %t, in = %b, e = %b, out = %b",$time, in, e,
out);
        if (out == expected_result) begin
                $display("------------------------------------------------------------
--------");
                $display("PASSED:       Expected:       %b      Got:       %b",
expected_result, out);
                $display("------------------------------------------------------------
--------");
        end else begin
                $display("------------------------------------------------------------
--------");
                $display("FAILED:       Expected:       %b      Got:       %b",
expected_result, out);
                $display("------------------------------------------------------------
--------");
        end
end
endtask
endmodule
```

Simulation

```
----------------------------------------------------------------------
---------- TESTBENCH FOR BINARY DECODER 3 TO 8----------------------
----------------------------------------------------------------------
At time:                 1000, in = 101, e = 0, out = 00000000
----------------------------------------------------------------------
PASSED: Expected: 00000000 Got: 00000000
----------------------------------------------------------------------
At time:                12000, in = 100, e = 0, out = 00000000
----------------------------------------------------------------------
PASSED: Expected: 00000000 Got: 00000000
----------------------------------------------------------------------
At time:                23000, in = 000, e = 1, out = 00000001
----------------------------------------------------------------------
PASSED: Expected: 00000001 Got: 00000001
----------------------------------------------------------------------
At time:                39000, in = 001, e = 1, out = 00000010
----------------------------------------------------------------------
PASSED: Expected: 00000010 Got: 00000010
----------------------------------------------------------------------
At time:                55000, in = 010, e = 1, out = 00000100
----------------------------------------------------------------------
PASSED: Expected: 00000100 Got: 00000100
----------------------------------------------------------------------
At time:                71000, in = 011, e = 1, out = 00001000
----------------------------------------------------------------------
PASSED: Expected: 00001000 Got: 00001000
----------------------------------------------------------------------
At time:                87000, in = 100, e = 1, out = 00010000
----------------------------------------------------------------------
PASSED: Expected: 00010000 Got: 00010000
----------------------------------------------------------------------
At time:               103000, in = 101, e = 1, out = 00100000
----------------------------------------------------------------------
PASSED: Expected: 00100000 Got: 00100000
----------------------------------------------------------------------
At time:               119000, in = 110, e = 1, out = 01000000
----------------------------------------------------------------------
PASSED: Expected: 01000000 Got: 01000000
----------------------------------------------------------------------
At time:               135000, in = 111, e = 1, out = 10000000
----------------------------------------------------------------------
PASSED: Expected: 10000000 Got: 10000000
----------------------------------------------------------------------
binary_decoder_tb.v:46: $finish called at 250000 (1ps)
```

Waveform overview:



5.2 Design priority encoder circuit

Block diagram:

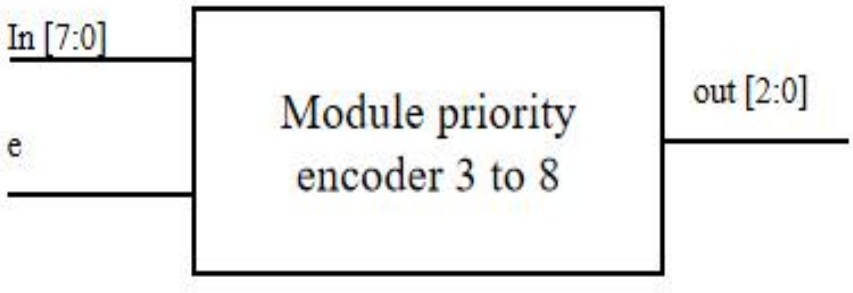


Image: Block diagram of module priority encoder 8 to 3

Describe IO Table

Signal	Direction	Bit-Width	Description
In	Input	8	Input data of binary signal
e	Input	1	Enable pin to control the circuit.
Out	Output	3	Output data

Waveform expected:

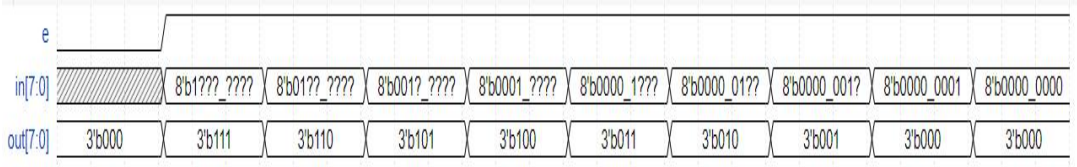


Image: waveform of module priority encoder 8 to 3

Code wavedrom:

```
{signal: [
{name: 'e', wave: '0..1.....................'},
{name: 'in[7:0]', wave:
'x..=..=..=..=..=..=..=..=..',data:["8'b1???_????","8'b01??_????","8'b001
?_????",
"8'b0001_????","8'b0000_1???","8'b0000_01??","8'b0000_001?",
"8'b0000_0001", "8'b0000_0000" ]},
{name: 'out[7:0]', wave:
'=..=..=..=..=..=..=..=..=..',data:["3'b000","3'b111","3'b110", "3'b101",
"3'b100","3'b011","3'b010","3'b001","3'b000","3'b000"]},
]}
```

RTL code of decoder 3to8:

The following Verilog code implement the priority encoder 8 to 3 circuit. Which takes one 8-bit input signals and output a single 3-bit.

```
module priority_encoder (
        input wire [7:0] in,
        input e,
        output reg [2:0] out
);

always @(*) begin
        if (!e) begin
                out = 3'b000;
        end else begin
                casez (in)
                        8'b1???_????: out <= 3'b111;
                        8'b01??_????: out <= 3'b110;
                        8'b001?_????: out <= 3'b101;
                        8'b0001_????: out <= 3'b100;
                        8'b0000_1???: out <= 3'b011;
                        8'b0000_01??: out <= 3'b010;
                        8'b0000_001?: out <= 3'b001;
                        8'b0000_0001: out <= 3'b000;
                        default: out <= 3'b000;
                endcase
        end
end
endmodule
```

Testbench for priority encoder 8 to 3.

To verify the functionality of the priority encoder 8 to 3 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	e	In	Out
0	0	8’b0000_0001	3’b000
1	0	8’b0000_0010	3’b000
3	1	8’b0000_0001	3’b000
4	1	8’b0000_0010	3’b001
…	…	…	…

Testbench code:

```
`timescale 1ns/1ps
module test_bench;
	reg [7:0] in;
	reg e;
	wire [2:0] out;

	priority_encoder uut(
		.in(in),
		.e(e),
		.out(out)
	);

	integer a, b, c;

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

		$display("-----------------------------------------------------");
		$display("--------TESTBENCH FOR PRIORITY
ENCODER 3 TO 8---------");
		$display("-----------------------------------------------------");

		for (a  = 0; a < 10; a = a + 1) begin
			e = 0;
			in = a;
			#1;
			check_result(3'b000);
			#10;
		end

		for (b = 0; b < 10; b = b + 1) begin
			e = 1;
			in = b;
			#1;
			check_result(calc_expected(in,e));
			#10;
		end

		for (c = 0; c < 15; c = c + 1) begin
			e = $urandom%2;
```

```
                    in = $urandom%10;
                    #1;
                    check_result(calc_expected(in,e));
                    #10;
                end

        end

        function [2:0] calc_expected (input [7:0] in_val, input e_val);
                begin
                    if (!e_val) begin
                        calc_expected = 3'b000;
                    end else begin
                        casez (in_val[7:0])
                            8'b1???????: calc_expected = 3'b111;
                            8'b01??????: calc_expected = 3'b110;
                            8'b001?????: calc_expected = 3'b101;
                            8'b0001????: calc_expected = 3'b100;
                            8'b00001???: calc_expected = 3'b011;
                            8'b000001??: calc_expected = 3'b010;
                            8'b0000001?: calc_expected = 3'b001;
                            8'b00000001: calc_expected = 3'b000;
                            default: calc_expected = 3'b000;
                        endcase
                    end
                end
        endfunction

        task check_result;
                input [2:0] expected_result;
                begin
                    $display("At time: %t, e = 1'b%b, in = 8'b%b, out =
3'b%b", $time, e, in, out);
                    if (out == expected_result) begin
                        $display("-----------------------------------------
---------");
                        $display("PASSED: expected: 3'b%b, Got:
3'b%b", expected_result, out);
                        $display("-----------------------------------------
---------");
                    end else begin
                        $display("-----------------------------------------
---------");
```

```
                $display("FAILED: expected: 3'b%b, Got: 3'b%b", expected_result, out);
                $display("-----------------------------------------");
            end
        end
    endtask

endmodule
```

Simulation

```
vvp test_bench
VCD info: dumpfile test_bench.vcd opened for output.
-------------------------------------------------
--------TESTBENCH FOR PRIORITY ENCODER 3 TO 8---------
-------------------------------------------------
At time:                 1000, e = 1'b0, in = 8'b00000000, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                12000, e = 1'b0, in = 8'b00000001, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                23000, e = 1'b0, in = 8'b00000010, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                34000, e = 1'b0, in = 8'b00000011, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                45000, e = 1'b0, in = 8'b00000100, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                56000, e = 1'b0, in = 8'b00000101, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                67000, e = 1'b0, in = 8'b00000110, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                78000, e = 1'b0, in = 8'b00000111, out = 3'b000
-------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------
At time:                89000, e = 1'b0, in = 8'b00001000, out = 3'b000
```

```
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               100000, e = 1'b0, in = 8'b00001001, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               111000, e = 1'b1, in = 8'b00000000, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               122000, e = 1'b1, in = 8'b00000001, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               133000, e = 1'b1, in = 8'b00000010, out = 3'b001
----------------------------------------------------
PASSED: expected: 3'b001, Got: 3'b001
----------------------------------------------------
At time:               144000, e = 1'b1, in = 8'b00000011, out = 3'b001
----------------------------------------------------
PASSED: expected: 3'b001, Got: 3'b001
----------------------------------------------------
At time:               155000, e = 1'b1, in = 8'b00000100, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               166000, e = 1'b1, in = 8'b00000101, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               177000, e = 1'b1, in = 8'b00000110, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               188000, e = 1'b1, in = 8'b00000111, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               199000, e = 1'b1, in = 8'b00001000, out = 3'b011
----------------------------------------------------
PASSED: expected: 3'b011, Got: 3'b011
----------------------------------------------------
At time:               210000, e = 1'b1, in = 8'b00001001, out = 3'b011
----------------------------------------------------
PASSED: expected: 3'b011, Got: 3'b011
----------------------------------------------------
At time:               221000, e = 1'b0, in = 8'b00001001, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               232000, e = 1'b1, in = 8'b00001001, out = 3'b011
----------------------------------------------------
```

```
----------------------------------------------------
PASSED: expected: 3'b011, Got: 3'b011
----------------------------------------------------
At time:               243000, e = 1'b1, in = 8'b00000101, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               254000, e = 1'b1, in = 8'b00000100, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               265000, e = 1'b1, in = 8'b00000111, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               276000, e = 1'b0, in = 8'b00000101, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               287000, e = 1'b1, in = 8'b00000000, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               298000, e = 1'b1, in = 8'b00001000, out = 3'b011
----------------------------------------------------
PASSED: expected: 3'b011, Got: 3'b011
----------------------------------------------------
At time:               309000, e = 1'b1, in = 8'b00000100, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               320000, e = 1'b1, in = 8'b00000111, out = 3'b010
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               331000, e = 1'b0, in = 8'b00000111, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b010, Got: 3'b010
----------------------------------------------------
At time:               331000, e = 1'b0, in = 8'b00000111, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               342000, e = 1'b0, in = 8'b00000010, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               353000, e = 1'b0, in = 8'b00000101, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
----------------------------------------------------
At time:               364000, e = 1'b0, in = 8'b00000101, out = 3'b000
----------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
```

```
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------------
At time:                375000, e = 1'b1, in = 8'b00000001, out = 3'b000
-------------------------------------------------------
PASSED: expected: 3'b000, Got: 3'b000
-------------------------------------------------------
```

Waveform: