

論理回路入門（9）

組合せ回路4：デコーダとエンコーダ

李 亜民

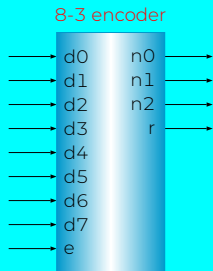
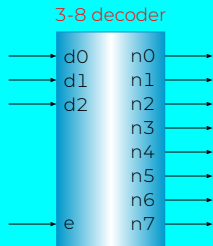
2024年11月19日(火)

デコーダとエンコーダ

ポイント

- デコーダ ($n \Rightarrow 2^n$)
- イネーブル付きデコーダ
- デマルチプレクサ

- エンコーダ ($2^n \Rightarrow n$)
- イネーブル付きエンコーダ
- プライオリティエンコーダ



3 入力 8 出力デコーダ

3 入力 8 出力 デコーダ の設計

3 入力 8 出力デコーダ (真理値表)

3 入力 8 出力デコーダ: 3 入力 ($d[2:0]$) を 2 進数について、その 10 進数に対応する 8 個の出力 ($n[7:0]$) を生成する回路。

真理値表:

入力			出力							
$d[2]$	$d[1]$	$d[0]$	$n[7]$	$n[6]$	$n[5]$	$n[4]$	$n[3]$	$n[2]$	$n[1]$	$n[0]$
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

3 入力 8 出力デコーダ (論理式)

入力			出力							
d[2]	d[1]	d[0]	n[7]	n[6]	n[5]	n[4]	n[3]	n[2]	n[1]	n[0]
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$$n[0] = \overline{d[2]} \cdot \overline{d[1]} \cdot \overline{d[0]}$$

$$n[1] = \overline{d[2]} \cdot \overline{d[1]} \cdot d[0]$$

$$n[2] = \overline{d[2]} \cdot d[1] \cdot \overline{d[0]}$$

$$n[3] = \overline{d[2]} \cdot d[1] \cdot d[0]$$

$$n[4] = d[2] \cdot \overline{d[1]} \cdot \overline{d[0]}$$

$$n[5] = d[2] \cdot \overline{d[1]} \cdot d[0]$$

$$n[6] = d[2] \cdot d[1] \cdot \overline{d[0]}$$

$$n[7] = d[2] \cdot d[1] \cdot d[0]$$

3 入力 8 出力デコーダ (回路)

```
'timescale 1ns/1ns
module dec38 (d, n);
    input  [2:0] d;
    output [7:0] n;

    assign n[0] = ~d[2] & ~d[1] & ~d[0]; // 000
    assign n[1] = ~d[2] & ~d[1] &  d[0]; // 001
    assign n[2] = ~d[2] &  d[1] & ~d[0]; // 010
    assign n[3] = ~d[2] &  d[1] &  d[0]; // 011
    assign n[4] =  d[2] & ~d[1] & ~d[0]; // 100
    assign n[5] =  d[2] & ~d[1] &  d[0]; // 101
    assign n[6] =  d[2] &  d[1] & ~d[0]; // 110
    assign n[7] =  d[2] &  d[1] &  d[0]; // 111

endmodule
```

[dec38.v](#)

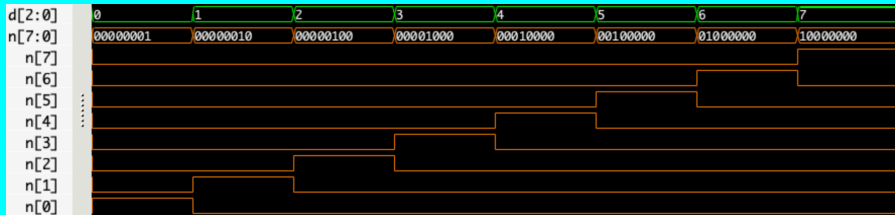
3 入力 8 出力デコーダ (テストベンチ)

```
'timescale 1ns/1ns
module dec38_tb;
    reg [2:0] d;
    wire [7:0] n;
    dec38 i0 (d, n);
    initial begin
        #0 d = 0;
        #8 $finish;
    end
    always #1 d = d + 1;
    initial begin
        $dumpfile ("dec38.vcd");
        $dumpvars;
    end
endmodule
```

[dec38_tb.v](#)

3 入力 8 出力デコーダ (波形)

```
% iverilog -Wall -o dec38 dec38_tb.v dec38.v  
% vvp dec38  
% gtkwave dec38.vcd
```



入力: d = 0 d = 1 d = 2 d = 3 d = 4 d = 5 d = 6 d = 7
出力: n[0]=1 n[1]=1 n[2]=1 n[3]=1 n[4]=1 n[5]=1 n[6]=1 n[7]=1

イネーブル付き 3-8 デコーダ

入力				出力							
e	d[2]	d[1]	d[0]	n[7]	n[6]	n[5]	n[4]	n[3]	n[2]	n[1]	n[0]
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
0	x	x	x	0	0	0	0	0	0	0	0

e: enable; x: ドントケア (Don't care)

$$n[0] = e \cdot \overline{d[2]} \cdot \overline{d[1]} \cdot \overline{d[0]}$$

$$n[1] = e \cdot \overline{d[2]} \cdot \overline{d[1]} \cdot d[0]$$

$$n[2] = e \cdot \overline{d[2]} \cdot d[1] \cdot \overline{d[0]}$$

$$n[3] = e \cdot \overline{d[2]} \cdot d[1] \cdot d[0]$$

$$n[4] = e \cdot d[2] \cdot \overline{d[1]} \cdot \overline{d[0]}$$

$$n[5] = e \cdot d[2] \cdot \overline{d[1]} \cdot d[0]$$

$$n[6] = e \cdot d[2] \cdot d[1] \cdot \overline{d[0]}$$

$$n[7] = e \cdot d[2] \cdot d[1] \cdot d[0]$$

イネーブル付き 3-8 デコーダ (回路)

```
'timescale 1ns/1ns
module dec38e (d, e, n);
    input  [2:0] d;
    input          e;
    output [7:0] n;

    assign n[0] = e & ~d[2] & ~d[1] & ~d[0]; // 1_000
    assign n[1] = e & ~d[2] & ~d[1] &  d[0]; // 1_001
    assign n[2] = e & ~d[2] &  d[1] & ~d[0]; // 1_010
    assign n[3] = e & ~d[2] &  d[1] &  d[0]; // 1_011
    assign n[4] = e &  d[2] & ~d[1] & ~d[0]; // 1_100
    assign n[5] = e &  d[2] & ~d[1] &  d[0]; // 1_101
    assign n[6] = e &  d[2] &  d[1] & ~d[0]; // 1_110
    assign n[7] = e &  d[2] &  d[1] &  d[0]; // 1_111

endmodule
```

[dec38e.v](#)

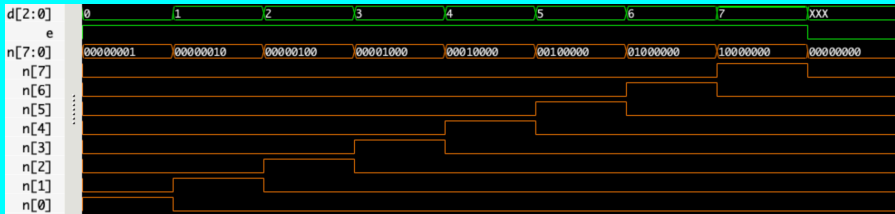
イネーブル付き 3-8 デコーダ (テストベンチ)

```
'timescale 1ns/1ns
module dec38e_tb;
    reg [2:0] d;
    reg      e;
    wire [7:0] n;
    dec38e i0 (d, e, n);
    initial begin
        #0 d = 3'b000; e = 1'b1;
        #1 d = 3'b001;
        #1 d = 3'b010;
        #1 d = 3'b011;
        #1 d = 3'b100;
        #1 d = 3'b101;
        #1 d = 3'b110;
        #1 d = 3'b111;
        #1 d = 3'bxxx; e = 1'b0;
        #1 $finish;
    end
    initial begin
        $dumpfile ("dec38e.vcd");
        $dumpvars;
    end
endmodule
```

[dec38e_tb.v](#)

イネーブル付き 3-8 デコーダ (波形)

```
% iverilog -Wall -o dec38e dec38e_tb.v dec38e.v  
% vvp dec38e  
% gtkwave dec38e.vcd
```



e = 0
n = 00000000

デコーダの応用 — RegFile

コンピューターシステム

1. コンピューター

(1) メモリ

(2) 入出カインターフェース

(3) CPU (プロセッサ)

ALU

...

レジスタファイル

2. ソフトウェア

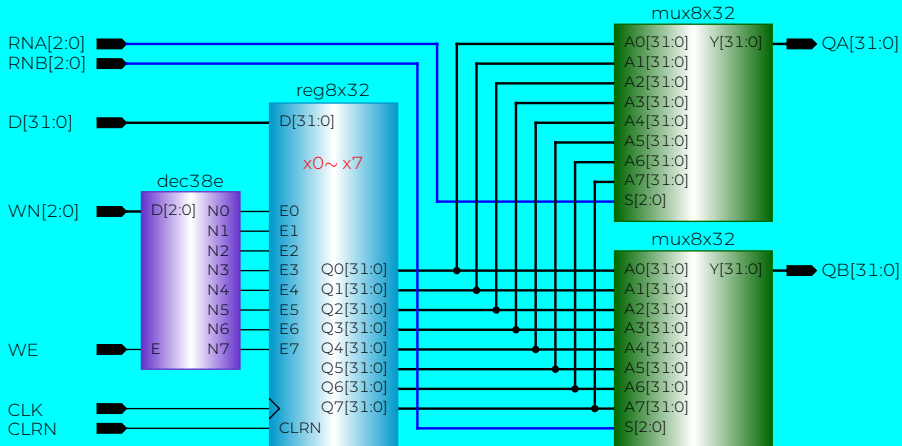
(OS やコンパイラなど)

3. 入出力デバイス

(キーボードやディスプレイなど)

デコーダの応用 — RegFile

8 × 32 ビットレジスタ・ファイル



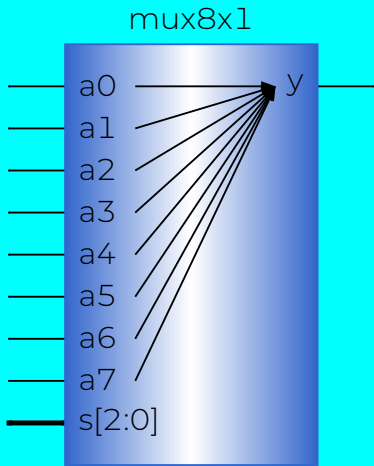
イネーブル付き
3入力8出力デコーダ

レジスタアレイ
(8個 × 32ビット)

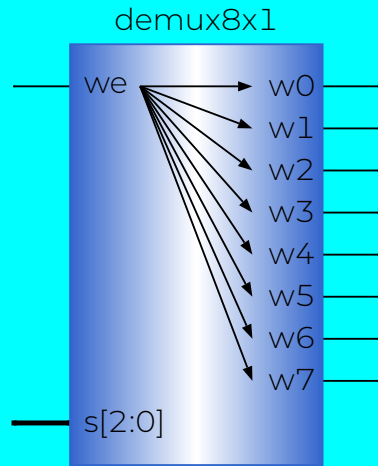
マルチプレクサ
(8入力 × 32ビット)

デコーダの応用 — デマルチプレクサ

マルチプレクサ



デマルチプレクサ



デコーダの応用 — デマルチプレクサ

マルチプレクサ

真理値表

s[2:0]	Y
0 0 0	a0
0 0 1	a1
0 1 0	a2
0 1 1	a3
1 0 0	a4
1 0 1	a5
1 1 0	a6
1 1 1	a7

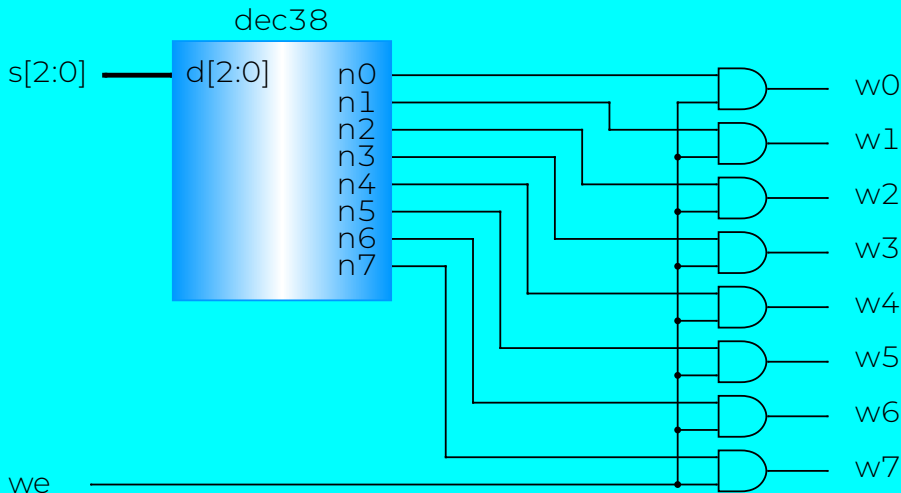
デマルチプレクサ

真理値表

s[2:0]	w7	w6	w5	w4	w3	w2	w1	w0
0 0 0	0	0	0	0	0	0	0	we
0 0 1	0	0	0	0	0	0	we	0
0 1 0	0	0	0	0	0	we	0	0
0 1 1	0	0	0	0	we	0	0	0
1 0 0	0	0	0	we	0	0	0	0
1 0 1	0	0	we	0	0	0	0	0
1 1 0	0	we	0	0	0	0	0	0
1 1 1	we	0	0	0	0	0	0	0

デコーダの応用 — デマルチプレクサ

デマルチプレクサの回路



デマルチプレクサ (回路)

```
'timescale 1ns/1ns
module demux8x1 (we, s, w);
    input  [2:0] s;
    input          we;
    output [7:0] w;

    wire  [7:0] n;
    dec38 i0 (s, n);

    assign w = n & {8{we}};

endmodule
```

[demux8x1.v](#)

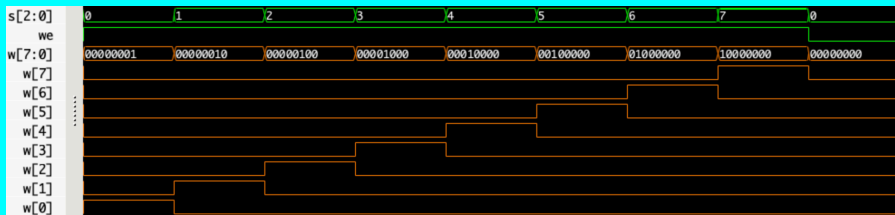
デマルチプレクサ (テストベンチ)

```
'timescale 1ns/1ns
module demux8x1_tb;
    reg [2:0] s;
    reg      we;
    wire [7:0] w;
    demux8x1 i0 (we, s, w);
    initial begin
        #0 s = 0; we = 1;
        #8 we = 0;
        #1 $finish;
    end
    always #1 s = s + 1;
    initial begin
        $dumpfile ("demux8x1.vcd");
        $dumpvars;
    end
endmodule
```

[demux8x1_tb.v](#)

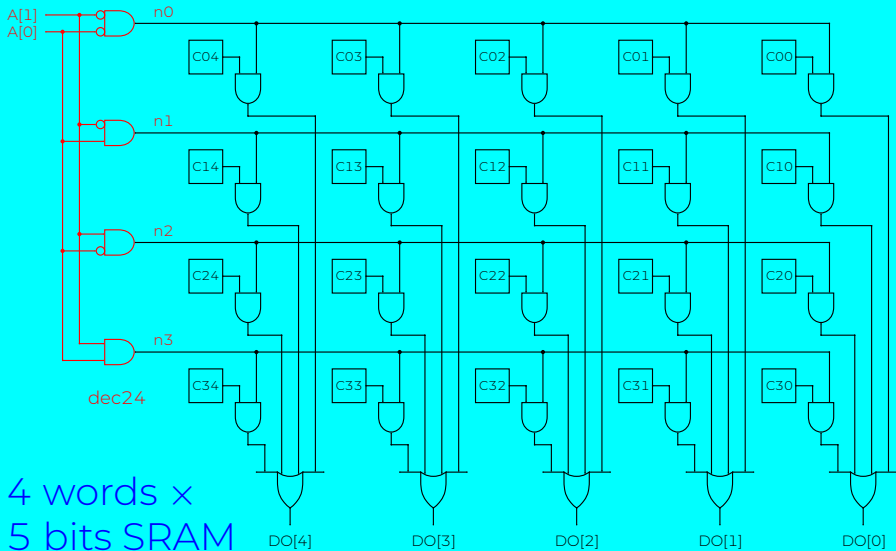
デマルチプレクサ (波形)

```
% iverilog -Wall -o demux8x1 \  
demux8x1_tb.v dec38.v  
% vvp demux8x1  
% gtkwave demux8x1.vcd
```



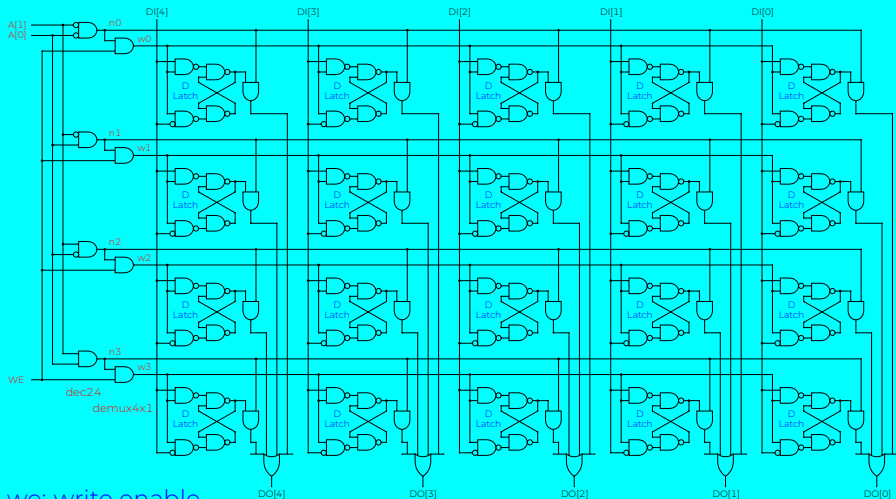
入力: s = 0 s = 1 s = 2 s = 3 s = 4 s = 5 s = 6 s = 7 s = 0
出力: w[0]=we w[1]=we w[2]=we w[3]=we w[4]=we w[5]=we w[6]=we w[7]=we w[0]=we

デコーダの応用 — SRAM デコーダ



4 words x
5 bits SRAM

デマルチプレクサの応用 — SRAM 書き込む



we: write enable

4 words: 2-bit address A[1:0]: $2^2 = 4$

5 bits/word: data-in DI[4:0]; data-out DO[4:0]

8 入力 3 出力エンコーダ

8 入力 3 出力 エンコーダ の設計

8 入力 3 出力エンコーダ (真理値表)

8 入力 3 出力エンコーダ: 8 入力 ($d[7:0]$) のいずれか一つが値 1 をとるものとして、それらの 2 進数に対応する 3 ビットの入力 ($n[2:0]$) を生成する回路。真理値表:

入力 (制限あり)								出力		
$d[7]$	$d[6]$	$d[5]$	$d[4]$	$d[3]$	$d[2]$	$d[1]$	$d[0]$	$n[2]$	$n[1]$	$n[0]$
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

入力制限: 一つの入力のみがアクティブである

8 入力 3 出力エンコーダ (論理式)

入力 (制限あり)								出力		
d[7]	d[6]	d[5]	d[4]	d[3]	d[2]	d[1]	d[0]	n[2]	n[1]	n[0]
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$n[0] = d[1] + d[3] + d[5] + d[7]$$

$$n[1] = d[2] + d[3] + d[6] + d[7]$$

$$n[2] = d[4] + d[5] + d[6] + d[7]$$

入力制限:

一つの入力のみが
アクティブである

8 入力 3 出力エンコーダ (回路)

$$n[0] = d[1] + d[3] + d[5] + d[7]$$

$$n[1] = d[2] + d[3] + d[6] + d[7]$$

$$n[2] = d[4] + d[5] + d[6] + d[7]$$

```
'timescale 1ns/1ns
module enc83 (d, n);
    input  [7:0] d;
    output [2:0] n;

    assign n[0] = d[1] | d[3] | d[5] | d[7];
    assign n[1] = d[2] | d[3] | d[6] | d[7];
    assign n[2] = d[4] | d[5] | d[6] | d[7];

endmodule
```

[enc83.v](#)

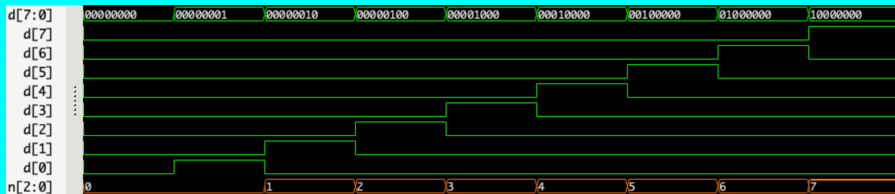
8 入力 3 出力エンコーダ (テストベンチ)

```
'timescale 1ns/1ns
module enc83_tb;
  reg [7:0] d;
  wire [2:0] n;
  enc83 i0 (d, n);
  initial begin
    #0 d = 8'b00000000;
    #1 d = 8'b00000001;
    #1 d = 8'b00000010;
    #1 d = 8'b00000100;
    #1 d = 8'b00001000;
    #1 d = 8'b00010000;
    #1 d = 8'b00100000;
    #1 d = 8'b01000000;
    #1 d = 8'b10000000;
    #1 $finish;
  end
  initial begin
    $dumpfile ("enc83.vcd");
    $dumpvars;
  end
endmodule
```

[enc83_tb.v](#)

8 入力 3 出力エンコーダ (波形)

```
% iverilog -Wall -o enc83 enc83_tb.v enc83.v  
% vvp enc83  
% gtkwave enc83.vcd
```



問題: $d = 00000000$, $n = 000$;
 $d = 00000001$, $n = 000$ 。区別できない。

エンコーダ (論理式)

$$n[0] = d[1] + d[3] + d[5] + d[7]$$

$$n[1] = d[2] + d[3] + d[6] + d[7]$$

$$n[2] = d[4] + d[5] + d[6] + d[7]$$

$$r = d[0] + d[1] + d[2] + d[3] + d[4] + d[5] + d[6] + d[7]$$

入力 (制限あり)								出力			
d[7]	d[6]	d[5]	d[4]	d[3]	d[2]	d[1]	d[0]	n[2]	n[1]	n[0]	r
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	1	0	0	0	1	0	1
0	0	0	0	1	0	0	0	0	1	1	1
0	0	0	1	0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0	1	0	1	1
0	1	0	0	0	0	0	0	1	1	0	1
1	0	0	0	0	0	0	0	1	1	1	1

エンコーダ (回路)

$$n[0] = d[1] + d[3] + d[5] + d[7]$$

$$n[1] = d[2] + d[3] + d[6] + d[7]$$

$$n[2] = d[4] + d[5] + d[6] + d[7]$$

$$r = d[0] + d[1] + d[2] + d[3] + d[4] + d[5] + d[6] + d[7]$$

```
'timescale 1ns/1ns
module enc83r (d, n, r);
    input  [7:0] d;
    output [2:0] n;
    output          r;
    assign n[0] = d[1] | d[3] | d[5] | d[7];
    assign n[1] = d[2] | d[3] | d[6] | d[7];
    assign n[2] = d[4] | d[5] | d[6] | d[7];
    assign r    = d[0] | d[1] | d[2] | d[3] |
                  d[4] | d[5] | d[6] | d[7];
endmodule
```

[enc83r.v](#)

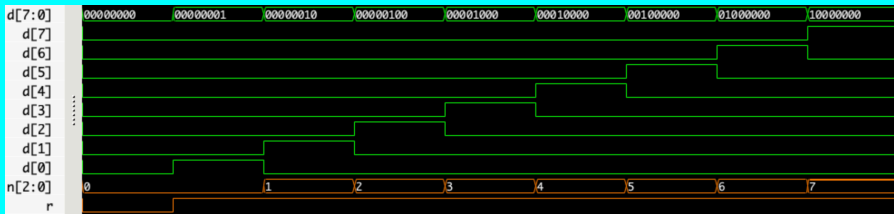
エンコーダ (テストベンチ)

```
'timescale 1ns/1ns
module enc83r_tb;
    reg [7:0] d;
    wire [2:0] n;
    wire      r;
    enc83r i0 (d, n, r);
    initial begin
        #0 d = 8'b00000000;
        #1 d = 8'b00000001;
        #1 d = 8'b00000010;
        #1 d = 8'b00000100;
        #1 d = 8'b00001000;
        #1 d = 8'b00010000;
        #1 d = 8'b00100000;
        #1 d = 8'b01000000;
        #1 d = 8'b10000000;
        #1 $finish;
    end
    initial begin
        $dumpfile ("enc83r.vcd");
        $dumpvars;
    end
endmodule
```

[enc83r_tb.v](#)

エンコーダ (波形)

```
% iverilog -Wall -o enc83r enc83r_tb.v enc83r.v  
% vvp enc83r  
% gtkwave enc83r.vcd
```



$d = 00000000$, $n = 000$, $r = 0$;

$d = 00000001$, $n = 000$, $r = 1$ 。 区別できる。

イネーブル付きエンコーダ

$$n[0] = e \cdot (d[1] + d[3] + d[5] + d[7])$$

$$n[1] = e \cdot (d[2] + d[3] + d[6] + d[7])$$

$$n[2] = e \cdot (d[4] + d[5] + d[6] + d[7])$$

$$r = e \cdot (d[0] + d[1] + d[2] + d[3] + d[4] + d[5] + d[6] + d[7])$$

入力 (制限あり)									出力			
e	d[7]	d[6]	d[5]	d[4]	d[3]	d[2]	d[1]	d[0]	n[2]	n[1]	n[0]	r
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	1	1
1	0	0	0	0	1	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0	0	1	1	0	1
1	1	0	0	0	0	0	0	0	1	1	1	1
0	x	x	x	x	x	x	x	x	0	0	0	0

イネーブル付きエンコーダ (回路)

$$n[0] = e \cdot (d[1] + d[3] + d[5] + d[7])$$

$$n[1] = e \cdot (d[2] + d[3] + d[6] + d[7])$$

$$n[2] = e \cdot (d[4] + d[5] + d[6] + d[7])$$

$$r = e \cdot (d[0] + d[1] + d[2] + d[3] + d[4] + d[5] + d[6] + d[7])$$

```
'timescale 1ns/1ns
module enc83e (d, e, n, r);
    input  [7:0] d;
    input          e;
    output [2:0] n;
    output          r;
    assign n[0] = e & (d[1] | d[3] | d[5] | d[7]);
    assign n[1] = e & (d[2] | d[3] | d[6] | d[7]);
    assign n[2] = e & (d[4] | d[5] | d[6] | d[7]);
    assign r    = e & (d[0] | d[1] | d[2] | d[3] |
                      d[4] | d[5] | d[6] | d[7]);
endmodule
```

[enc83e.v](#)

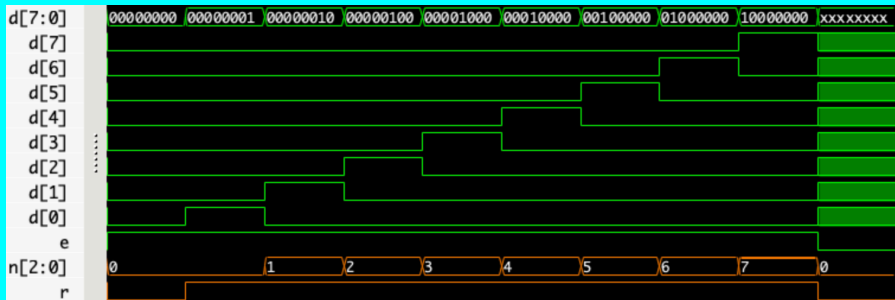
イネーブル付きエンコーダ (テスト)

```
'timescale 1ns/1ns
module enc83e_tb;
    reg [7:0] d;
    reg      e;
    wire [2:0] n;
    wire      r;
    enc83e i0 (d, e, n, r);
    initial begin
        #0 d = 8'b00000000; e = 1;
        #1 d = 8'b00000001;
        #1 d = 8'b00000010;
        #1 d = 8'b00000100;
        #1 d = 8'b00001000;
        #1 d = 8'b00010000;
        #1 d = 8'b00100000;
        #1 d = 8'b01000000;
        #1 d = 8'b10000000;
        #1 d = 8'bxxxxxxx; e = 0;
        #1 $finish;
    end
    initial begin
        $dumpfile ("enc83e.vcd");
        $dumpvars;
    end
endmodule
```

[enc83e_tb.v](#)

イネーブル付きエンコーダ (波形)

```
% iverilog -Wall -o enc83e enc83e_tb.v enc83e.v  
% vvp enc83e  
% gtkwave enc83e.vcd
```



プライオリティエンコーダ

入力 (制限なし)									出力			
e	d[7]	d[6]	d[5]	d[4]	d[3]	d[2]	d[1]	d[0]	n[2]	n[1]	n[0]	r
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	0	0	0	1	x	0	0	1	1
1	0	0	0	0	0	1	x	x	0	1	0	1
1	0	0	0	0	1	x	x	x	0	1	1	1
1	0	0	0	1	x	x	x	x	1	0	0	1
1	0	0	1	x	x	x	x	x	1	0	1	1
1	0	1	x	x	x	x	x	x	1	1	0	1
1	1	x	x	x	x	x	x	x	1	1	1	1
0	x	x	x	x	x	x	x	x	0	0	0	0

d[7] の優先順位 (プライオリティ) は一番高い、d[0] の優先順位は一番低い

$$\begin{aligned}
 n[2] &= e \cdot (d[7] + \overline{d[7]} \cdot d[6] + \overline{d[7]} \cdot \overline{d[6]} \cdot d[5] + \overline{d[7]} \cdot \overline{d[6]} \cdot \overline{d[5]} \cdot d[4]) \\
 &= e \cdot (d[7] + d[6] + \overline{d[6]} \cdot d[5] + \overline{d[6]} \cdot \overline{d[5]} \cdot d[4]) \\
 &= e \cdot (d[7] + d[6] + d[5] + \overline{d[5]} \cdot d[4]) \\
 &= e \cdot (d[7] + d[6] + d[5] + d[4])
 \end{aligned}$$

第三回講義資料 P27 を参照

プライオリティエンコーダ (回路)

$$n[2] = e \cdot (d[7] + d[6] + d[5] + d[4])$$

$$n[1] = e \cdot (\square + \square + \square \cdot \square \cdot \square + \square \cdot \square \cdot \square)$$

$$n[0] = e \cdot (\square + \square \cdot \square + \square \cdot \square \cdot \square + \square \cdot \square \cdot \square \cdot \square)$$

$$r = e \cdot (d[7] + d[6] + d[5] + d[4] + d[3] + d[2] + d[1] + d[0])$$

```
'timescale 1ns/1ns
module enc83p (d, e, n, r);
    input  [7:0] d;
    input          e;
    output [2:0] n;
    output          r;
    assign n[2] = e & (d[7] | d[6] | d[5] | d[4]);
    assign n[1] = e & ( | | & & | & & );
    assign n[0] = e & ( | & | & & | & & & );
    assign r    = e & (d[7] | d[6] | d[5] | d[4] |
                      d[3] | d[2] | d[1] | d[0]);
endmodule
```

[enc83p.v](#)

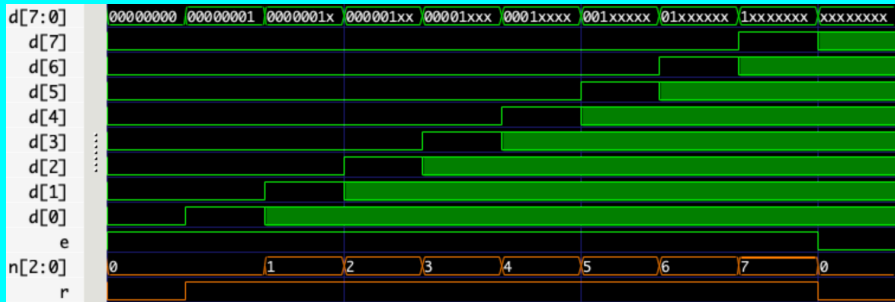
プライオリティエンコーダ (テスト)

```
'timescale 1ns/1ns
module enc83p_tb;
    reg [7:0] d;
    reg      e;
    wire [2:0] n;
    wire      r;
    enc83p i0 (d, e, n, r);
    initial begin
        #0 d = 8'b00000000; e = 1'b1;
        #1 d = 8'b00000001;
        #1 d = 8'b0000001x;
        #1 d = 8'b000001xx;
        #1 d = 8'b00001xxx;
        #1 d = 8'b0001xxxx;
        #1 d = 8'b001xxxxx;
        #1 d = 8'b01xxxxxx;
        #1 d = 8'b1xxxxxxx;
        #1 d = 8'bxxxxxxxx; e = 1'b0;
        #1 $finish;
    end
    initial begin
        $dumpfile ("enc83p.vcd");
        $dumpvars;
    end
endmodule
```

[enc83p_tb.v](#)

プライオリティエンコーダ (波形)

```
% iverilog -Wall -o enc83p enc83p_tb.v enc83p.v  
% vvp enc83p  
% gtkwave enc83p.vcd
```



プライオリティエンコーダ

入力制限なしの意味：任意の入力パターン

入力信号の数：9（1本のeと8本のD（d[7:0]））

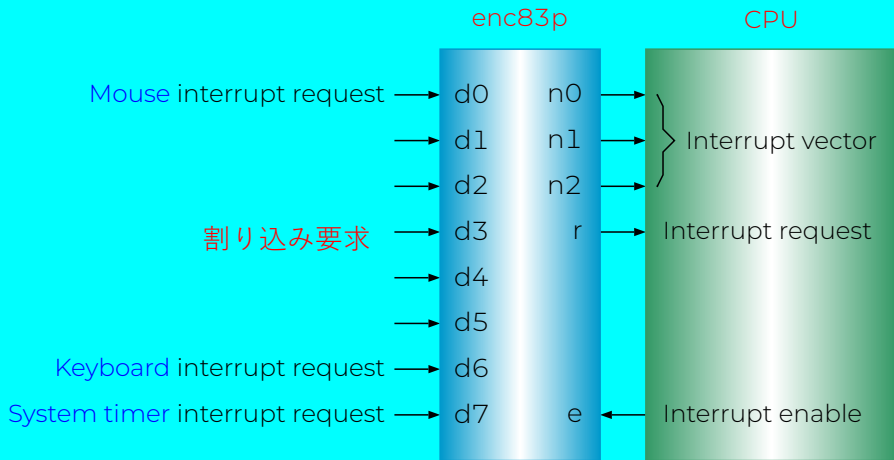
入力信号パターン数： $2^9 = 512$

入力（制限なし）									パターン数	
e	d[7]	d[6]	d[5]	d[4]	d[3]	d[2]	d[1]	d[0]		
1	0	0	0	0	0	0	0	0		1
1	0	0	0	0	0	0	0	1		1
1	0	0	0	0	0	0	1	X	Xが1個	2
1	0	0	0	0	0	1	X	X	Xが2個	4
1	0	0	0	0	1	X	X	X	Xが3個	8
1	0	0	0	1	X	X	X	X	Xが4個	16
1	0	0	1	X	X	X	X	X	Xが5個	32
1	0	1	X	X	X	X	X	X	Xが6個	64
1	1	X	X	X	X	X	X	X	Xが7個	128
0	X	X	X	X	X	X	X	X	Xが8個	256

合計： $1 + 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 = 512 = 2^9$

エンコーダの応用 — 割り込みコントローラ

割り込み (Interrupt) コントローラ (Controller)

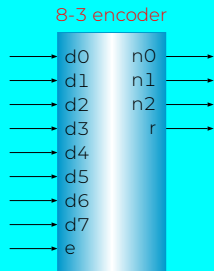
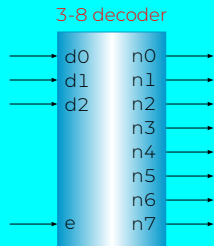


デコーダとエンコーダ

まとめ

- デコーダ ($n \Rightarrow 2^n$)
- イネーブル付きデコーダ
- デマルチプレクサ

- エンコーダ ($2^n \Rightarrow n$)
- イネーブル付きエンコーダ
- プライオリティエンコーダ



課題 X (100 点)

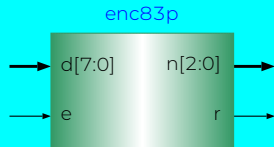
8 入力 3 出力プライオリティエンコーダを設計し動作検証シミュレーションして下さい (P37-P41 を参照)。

$$n[2] = e \cdot (d[7] + d[6] + d[5] + d[4])$$

$$n[1] = e \cdot (\square + \square + \square \cdot \square \cdot \square + \square \cdot \square \cdot \square)$$

$$n[0] = e \cdot (\square + \square \cdot \square + \square \cdot \square \cdot \square + \square \cdot \square \cdot \square \cdot \square)$$

$$r = e \cdot (d[7] + d[6] + d[5] + d[4] + d[3] + d[2] + d[1] + d[0])$$



モジュール名は [enc83p](#) にすること。

テストベンチ [enc83p_tb.v](#) を使って下さい。

真理値表から $n[2]$ 、 $n[1]$ 、 $n[0]$ の導出過程を書いて下さい。

課題 X (100 点)

$$n[2] = e \cdot (d[7] + d[6] + d[5] + d[4])$$

$$n[1] = e \cdot (\square + \square + \square \cdot \square \cdot \square + \square \cdot \square \cdot \square)$$

$$n[0] = e \cdot (\square + \square \cdot \square + \square \cdot \square \cdot \square + \square \cdot \square \cdot \square \cdot \square)$$

$$r = e \cdot (d[7] + d[6] + d[5] + d[4] + d[3] + d[2] + d[1] + d[0])$$

```
'timescale 1ns/1ns
module enc83p (d, e, n, r);
    input  [7:0] d;
    input          e;
    output [2:0] n;
    output          r;
    assign n[2] = e & (d[7] | d[6] | d[5] | d[4]);
    assign n[1] = e & ( | | & & | & & );
    assign n[0] = e & ( | & | & & | & & & );
    assign r    = e & (d[7] | d[6] | d[5] | d[4] |
                      d[3] | d[2] | d[1] | d[0]);
endmodule
```

[enc83p.v](#)

課題 X (100 点)

```
% iverilog -Wall -o enc83p enc83p_tb.v enc83p.v  
% vvp enc83p  
% gtkwave enc83p.vcd
```

