#100 DAYS OF RTL

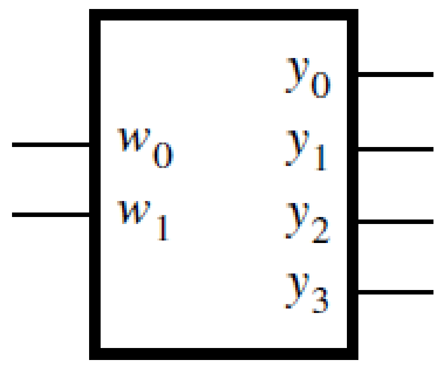
DAY 6/100 Insights

DECODER

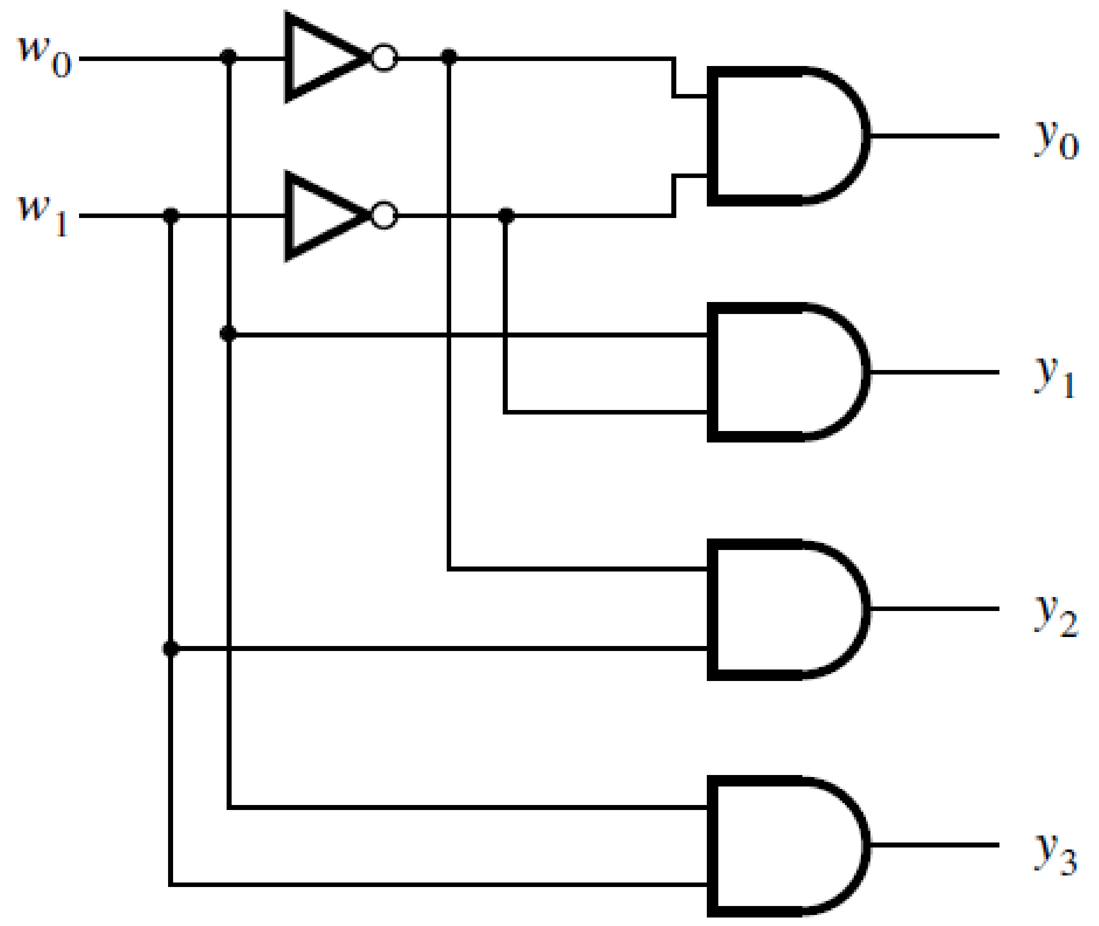
DECODER

w_1	w_0	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a) Truth table



(b) Graphical symbol



(c) Logic circuit

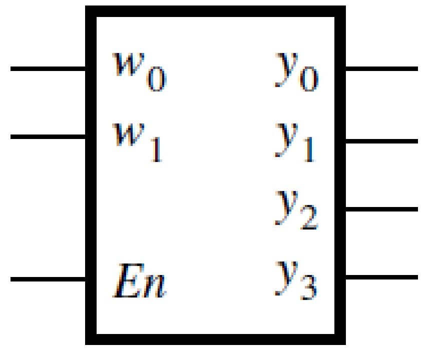
DECODER WITH ENABLE

Case statement

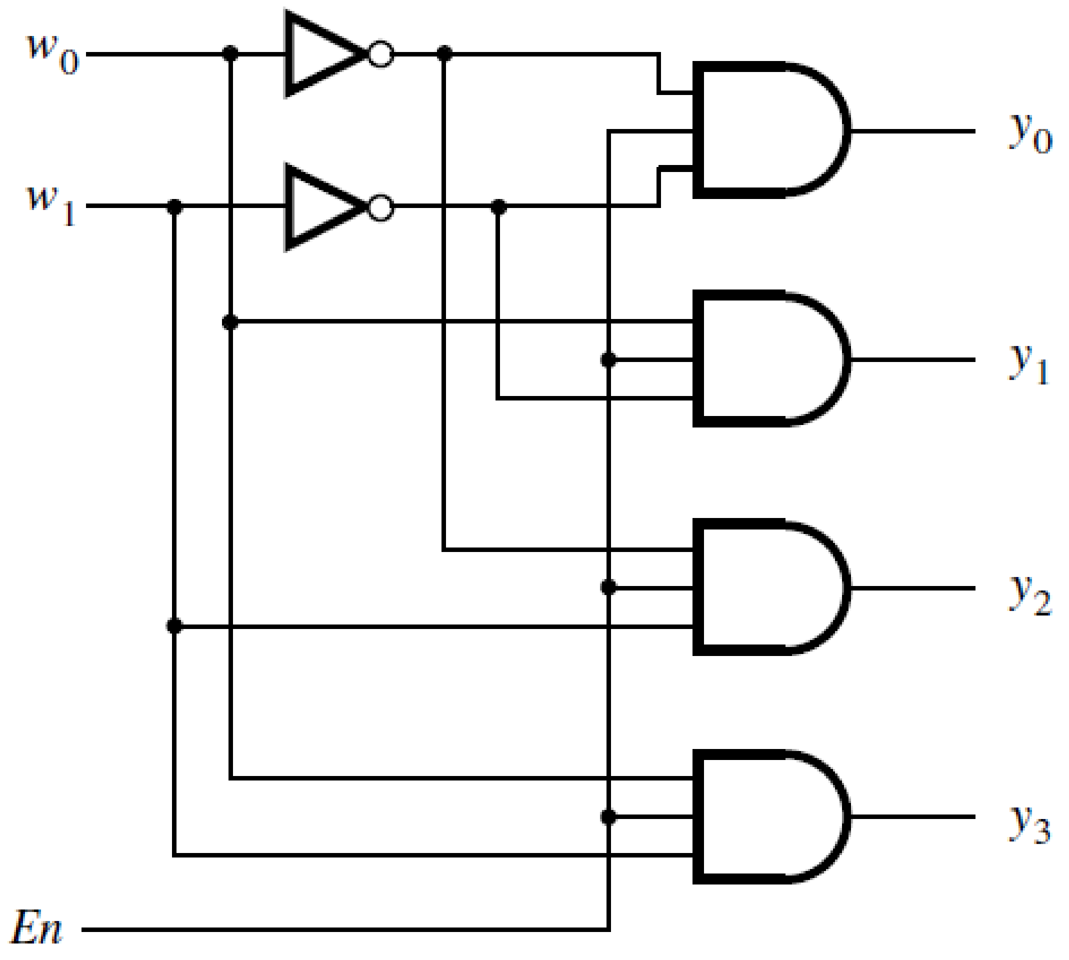
```
module dec2to4case (W, En, Y);
input [1:0]W;
input En;
output reg [0:3] Y;
always @(W, En)
case ({En,W})
3'b100: Y = 4'b1000;
3'b101: Y = 4'b0100;
3'b110: Y = 4'b0010;
3'b111: Y = 4'b0001;
default: Y = 4'b0000;
endcase
endmodule
```

En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table

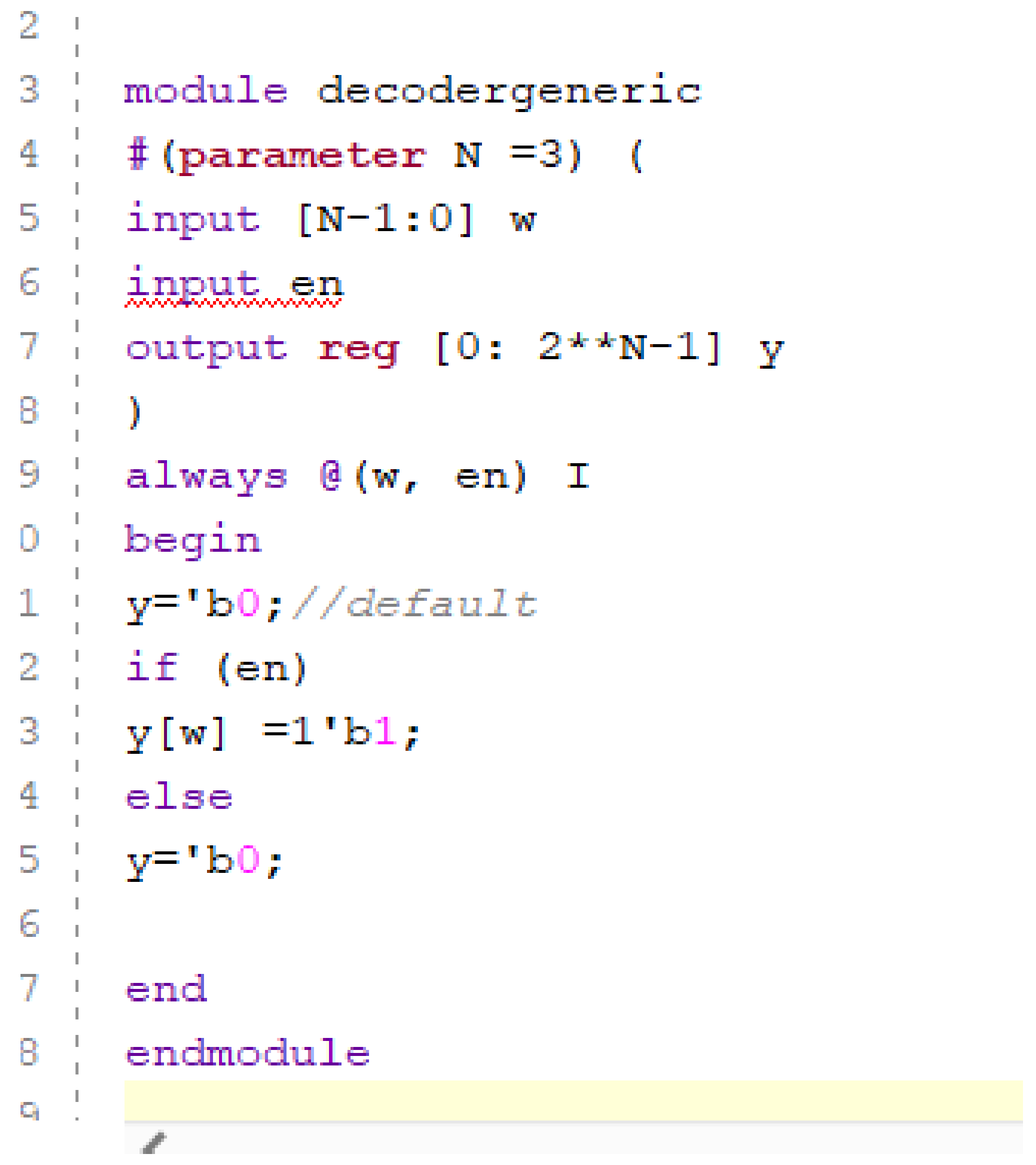


(b) Graphical symbol



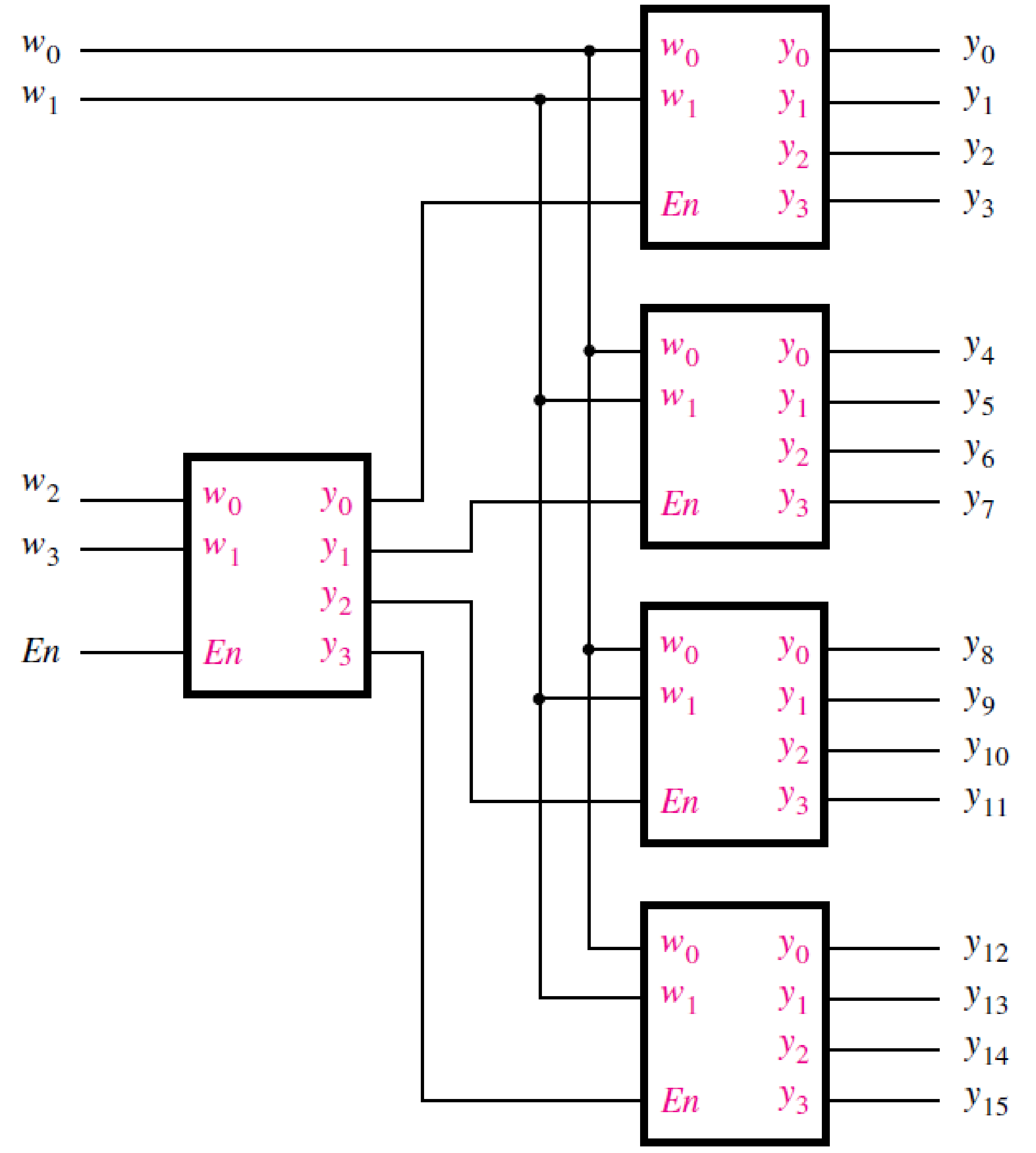
(c) Logic circuit

Decoder generic
Why is it helpful



```
module decodergeneric
#(parameter N =3) (
input [N-1:0] w
input en
output reg [0: 2**N-1] y
)
always @(w, en)
begin
y='b0;//default
if (en)
y[w] =1'b1;
else
y='b0;

end
endmodule
```



It can help implement such design easily