به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال — پاییز ۱۴۰۰

تمرین ۳، سوال ۱ — آرین احدی نیا، ۹۸۱۰۳۸۷۸

برای حل این سوال ابتدا لازم است که به رابطه‌ای برای Increament کردن یک عدد که در gray code بیان شده است، پی ببریم. برای این منظور، جدول کارنو را برای هر یک از بیت‌ها رسم می‌کنیم و رابطه منطقی میان ورودی که خود یک عدد چهار بیتی در gray code است و بیت iام خروجی را به دست می‌آوریم.

در جدول مقابل میتوانید حالت بعد از Increament شدن هر عدد در نمایش gray code را مشاهده بفرمایید.

AB / CD	00	01	11	10
00	0001	1100	1101	0000
01	0011	0100	1111	1000
11	0010	0101	1110	1001
10	0110	0111	1010	1011

اگر پس از Increament حاصل $A'B'C'D'$ باشد، برای A' خواهیم داشت.

$$A' = BC'D' + AC + AD$$

AB / CD	00	01	11	10
00	0	1	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

اگر پس از Increament حاصل $A'B'C'D'$ باشد، برای B' خواهیم داشت.

$$B' = A'CD' + BC' + BD$$

AB / CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	1	1	0	0

اگر پس از Increament حاصل $A'B'C'D'$ باشد، برای C' خواهیم داشت.

$$C' = A'B'D + ABD + CD'$$

AB / CD	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	0	1	0
10	1	1	1	1

اگر پس از Increament حاصل $A'B'C'D'$ باشد، برای D' خواهیم داشت.

$$D' = A'B'C' + A'BC + ABC' + AB'C$$

AB / CD	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	0	1	0	1
10	0	1	0	1

با توجه به روابط فوق، میتوانیم هم به صورت gray code نیز عمل Increament را انجام دهیم.

پیاده‌سازی را به این صورت انجام می‌دهیم که reset به صورت active high باشد و زمانی که تبدیل به ۱ شود، مقدار شمارنده به صفر بازگردد.

این شمارنده را به صورت rising edge triggered طراحی میکنیم. به این صورت که در کلاک بالارونده، مقدار جدید محاسبه و در خروجی قرار داده میشود. دقت بفرمایید که این در صورتی است که ورودی reset فعال نباشد.

توجه بفرمایید که مقداردهی اولیه به شمارنده با مقدار صفر انجام میشود.

```
module counter (reset, gray, clock, out);
    input reset, gray, clock;
    output reg [3:0] out;

    initial begin
        out = 4'b0000;
    end

    always @(posedge reset) begin
        out = 4'b0000;
    end

    always @(posedge clock) begin
        if (reset == 1'b0) begin
            if (gray == 1'b0) begin
                out = out + 1;
            end else begin
                out[3] <= (out[2] & ~out[1] & ~out[0]) | (out[3] & out[1]) | (out[3] & out[0]);
                out[2] <= (~out[3] & out[1] & ~out[0]) | (out[2] & ~out[1]) | (out[2] & out[0]);
                out[1] <= (~out[3] & ~out[2] & out[0]) | (out[3] & out[2] & out[0]) | (out[1] & ~out[0]);
                out[0] <= (~out[3] & out[2] & out[1]) | (out[3] & ~out[2] & out[1]) | (out[3] & out[2] & ~out[1]) |
(~out[3] & ~out[2] & ~out[1]);
            end
        end
    end
endmodule
```

برای تست یک ماژول به نام clock generator می‌سازیم که کلاک را در مدار شبیه سازی کند. از این ماژول در سوالات دیگر نیز استفاده خواهیم کرد.

```
module clock_generator(output reg clock);
    parameter HALF_T = 1;

    initial begin
        clock = 0;
    end

    always begin
        #HALF_T clock = ~clock;
    end
endmodule
```

سپس یک نمونه از این ماژول میگیریم و آن را به ازای ورودی های مختلف تست می‌کنیم. شکل زیر بخشی از شکل موج این مدار به ازای تست نوشته شده را نشان می‌دهد.