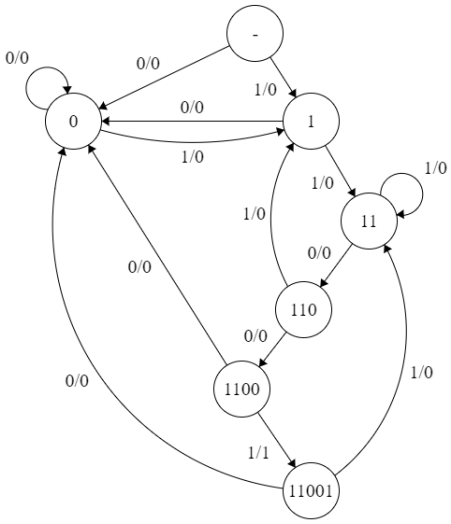
به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال	آرین احدی نیا
تمرین ۴، سوال ۱	۹۸۱۰۳۸۷۸

نمودار ماشین **Mealy** مورد نظر، به شکل زیر می‌شود.



برای پیاده‌سازی، یک مدار ترکیبی برای مشخص کردن حالت بعدی در نظر می‌گیریم.

```
always @(*) begin
    next = STT_START;
    if (in || !in) begin    // To avoid `x` and `z`
        case (state)
            STT_START:
                if (in)
                    next = STT_1;
                else
                    next = STT_0;
            STT_0:
                if (in)
                    next = STT_1;
                else
                    next = STT_0;
            STT_1:
                if (in)
                    next = STT_11;
                else
                    next = STT_0;
            STT_11:
                if (in)
                    next = STT_11;
                else
                    next = STT_110;
            STT_110:
                if (in)
                    next = STT_1;
                else
```

```
                next = STT_1100;
            STT_1100:
                if (in)
                    next = STT_11001;
                else
                    next = STT_0;
            STT_11001:
                if (in)
                    next = STT_11;
                else
                    next = STT_0;
        endcase
    end
end
```

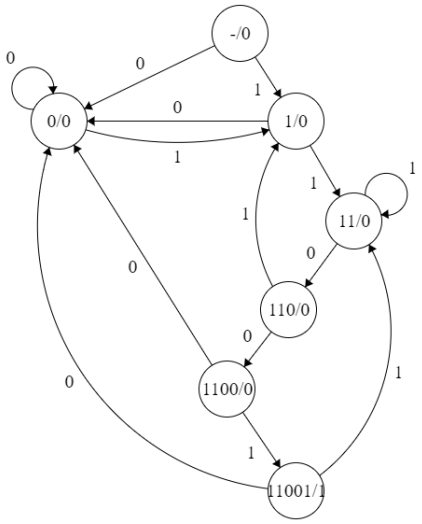
همچنین بر سرضرب بالارونده کلاک، مقدار بعدی را در **state** قرار می‌دهیم.

```
always @(posedge clk, posedge reset) begin
    if (reset)
        state <= STT_START;
    else
        state <= next;
end
```

همچنین یک مدار ترکیبی برای مشخص کردن خروجی از روی **state** و ورودی فعلی می‌سازیم. توجه کنید که خروجی تنها در صورتی برابر 1 است که در حالت 1100 باشیم و ورودی 1 باشد.

```
always @(*) begin
    if (in && state == STT_1100)
        out = 1;
    else
        out = 0;
end
```

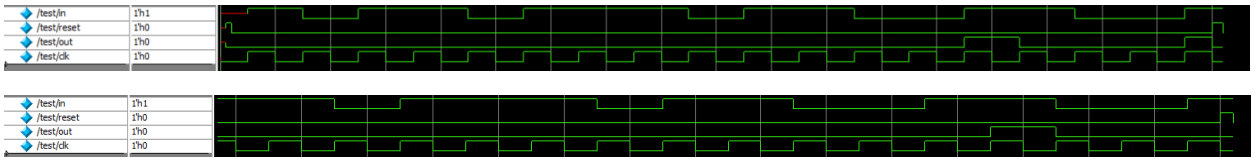
در ماشین **Moore**، خروجی به **state** وابسته است. بنابرین در **FSM** این ماشین، خروجی را در هر **state** مشخص می‌کنیم. **FSM** مورد نظر، به شکل زیر خواهد شد.



مشابها می‌توانیم مدار مورد نظر را پیاده‌سازی کنیم. توجه کنید که در این پیاده‌سازی، **state** تنها در لبه بالارونده کلاک تغییر می‌کند. از آنجایی که خروجی نیز تابع **state** است، خروجی نیز تنها در لبه بالارونده کلاک تغییر میکند. در ادامه بخشی از کد را مشاهده می‌کنید که مربوط به موارد مذکور در پیاده‌سازی **Moore** می‌شود.

```
always @(posedge clk, posedge reset) begin
    if (reset)
        state <= STT_START;
    else
        state <= next;
end

assign out = (state == STT_11001);
```

توجه بفرمایید که در فایل ارسالی، پیاده‌سازی **Moore** به صورت **comment** نوشته شده است.



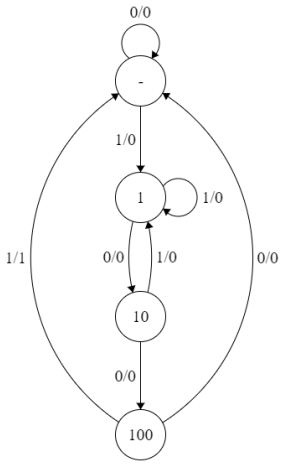
در تصاویر بالا میتوانید شکل موج به ترتیب ماشین **Mealy** و **Moore** پیاده‌سازی شده را مشاهده فرمایید. توجه کنید که این تفاوت در **Output** کاملا طبیعی است چرا که ماشین **Mealy** در لحظه خروجی خود را تغییر می‌دهد در صورتی که ماشین **Moore**، تابع **state** است.

به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال	آرین احدی نیا
تمرین ۴، سوال ۲	۹۸۱۰۳۸۷۸

فرض معقول آن است که حالت اولیه را **S0** در نظر بگیریم. توجه بفرمایید که این نمودار قطعا مربوط به ماشین **Mealy** است. اگر که فلش‌های متصل از **S3** به **S0** را در نظر بگیرید، خواهید دید که خروجی تنها تابعی از **state** نیست.

توجه بفرمایید که پس از مشاهده **100** در حالی که ورودی **1** است، خروجی برابر **1** میشود. بنابرین معقول است که در نظر بگیریم که مدار **1001** را مشاهده می‌کند. حال فرضیه خود را اثبات می‌کنیم. کافی است که به جای حالت‌ها، مقدار مشاهده شده در آن لحظه را قرار دهیم.



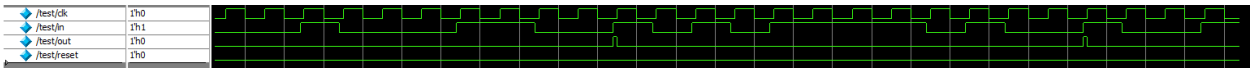
همانگونه که مشاهده می‌کنید، با جایگذاری فوق، مدار در صدد یافتن **غیر همپوشان 1001** است. توجه کنید که تنها حالت همپوشان به این صورت است که یک انتهایی با یک ابتدایی مشترک باشد. در این حالت ورودی باید **1001001** باشد که مدار این حالت را به عنوان الگوی مورد نظر شناسایی نمی‌کند.

مدار را مشابه سوال قبل پیاده‌سازی می‌کنیم.

```
module sd_mealy (out, in, clk, reset);
    output  reg out;
    input       in, clk, reset;

    parameter S0 = 2'b00;
    parameter S1 = 2'b01;
    parameter S2 = 2'b10;
    parameter S3 = 2'b11;

    reg [1:0] state;
    reg [1:0] next;

    always @(*) begin
        next = S0;
        if (in || !in) begin    // To avoid `x` and `z`
            case (state)
```

```
                S0: if (!in)
                        next = S0;
                    else
                        next = S1;
                S1: if (!in)
                        next = S2;
                    else
                        next = S1;
                S2: if (!in)
                        next = S3;
                    else
                        next = S1;
                S3: next = S0;
            endcase
        end
    end

    always @(posedge clk, posedge reset) begin
        if (reset)
            state <= S0;
        else
            state <= next;
    end

    always @(*) begin
        if (in && state == S3)
            out = 1;
        else
            out = 0;
    end
endmodule
```

Testbench را به گونه‌ای نوشته‌ایم که تمام ترکیبات مختلف از ۴ بیت در ورودی ظاهر شوند. مشاهده می‌کنیم که مدار تنها به **1001** واکنش نشان می‌دهد. همچنان در انتهای تست دو **1001** را به صورت همپوشان به مدار می‌دهیم و مشاهده می‌کنیم که مدار به ورودی دوم واکنشی نشان نمی‌دهد. این اثباتی بر فرضیه مطرح شده است.



توجه کنید که علت کوتاه بودن فعال شدن **out**، به دلیل زمان‌بندی تغییر **in** است.

به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال	آرین احدی نیا
تمرین ۴، سوال ۳	۹۸۱۰۳۸۷۸

توضیحات تکمیلی:

دو ماژول **arithmetic**و **logic** به ترتیب بخش محاسباتی و منطقی پردازنده را می‌سازند و **processor** وظیفه کنترل را بر عهده دارد. این پردازنده هر دو ماتریس را دریافت و در حافظه قرار میدهد سپس خروجی را بر میگرداند. برای دادن ماتریس به عنوان **port** به این دو ماژول، آن را اصطلاحا **flatten** می‌کنیم. به این صورت که سطر ها را کنار هم قرار می‌دهیم.

برای هر عملیات حسب نیاز، یک یا دو ماتریس را از ورودی سطر به سطر دریافت می‌کنیم. به جز عملیات **NOP** که نیازی به ورودی ندارد.

برای تست، دو ماتریس تصادفی به کمک پایتون می‌سازیم و عملیات‌های مورد نظر را روی آنها اجرا می‌کنیم.