این پروژه **معادلات ورودی دو فلیپ فلاپ نوع دی** را نمایش می دهد که با قطعه کد های زیر ساخته شده است :

در قسمت اول کد به طراحی مدل اولیه یک فلیپ فلاپ نوع دی پرداخته شده است که از ان در ادامه پروژه استفاده میگردد .

```
module D_Flip_Flop(
    input D,
    input clk,
    output reg Q,
    output reg Q_prime
);
    always @(*)
    begin
        Q = ~(~(D & clk) & Q_prime);
        Q_prime = ~(~(~D & clk) & Q);
    end
endmodule
```

این قطعه حافظه بوسیله یک ورودی و یک کلاک به همراه خروجی هایی از نوع reg طراحی میشود و به وسیله ساختار always میتوان به آن کارایی داد.

در قسمت بعدی بدنه اصلی کد را مشاهده میکنیم که ماژول های فلیپ فلاپ ما در ان فراخوانی میشوند . این بخش با تعریف ورودی های اصلی مدار و سیم های مورد نیاز که بوسیله reg طراحی شده اند شکل می گیرد.

```
module DFF_Input_Equation(
    input X,
    input clk,
    output Y
);
    reg A;
    reg A_prime;
    reg B;
    reg B_Prime;
    reg D1;
    reg D2;
```

در ادامه ، دو قطعه از reg ها که حاوی محاسبات منطقی هستند را قبل از فراخوانی توابع مشاهده میکنیم.

```
always @(*)
begin
    D1 = (X & A) | (X & B);
    D2 = A_prime & X;
end
```

در این قسمت به شکل روبرو ماژولی که در اول کد طراحی کرده بودیم ، دوبار فراخوانی شده و با معماری مد نظر سوال مقدار دهی شده است.

```
D_Flip_Flop D_Flip_Flop_1(
    .D(D1),
    .clk(clk),
    .Q(A),
    .Q_prime(A_prime)
);

D_Flip_Flop D_Flip_Flop_2(
    .D(D2),
    .clk(clk),
    .Q(B),
    .Q_prime(B_Prime)
);
```

```
    always @(*)
    begin
        Y = ((A & ~X) | (B & ~X));
    end

endmodule
```

در قسمت پایانی برنامه ما محاسبات مورد نظر برای بدست امدن خروجی اصلی کد (ایگرگ) را بوسیله ابزار آلویز پیاده سازی کرده و کد ما به پایان میرسد .

نمایی از خروجی شبیه سازی شده ی کد را میتوانید در شکل ذیل مشاهده فرمایید :