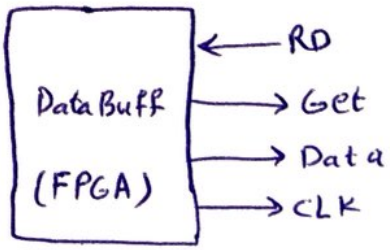
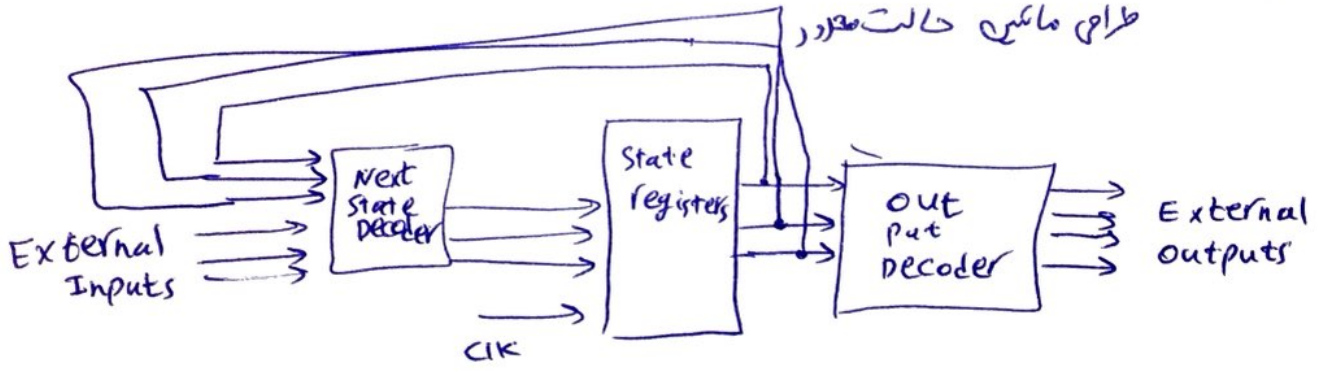
۱)



سیگنال CLK دارای 100MHz است

با به توسط FSM کنترل شود با دستورات و آیند (مدل سازی رفتاری)

طراحی ماشین حالت مدور



(2

یک تابع در بسته Package برای عملگر (-) تفریق با نوع STD_LOGIC_VECTOR

از مدل مثال های سربارگذاری است با استفاده از فانکشن (Function) منفی

پس ابتدا یک پکیج می آوریم. اسم تابع را عملگر فانکشن منفی می گذاریم

```
------Package:--------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

در ابتدا کتابخانه ها را معرفی می کنیم و یک پکیج درست می کنیم.

```
---------------------------
PACK my_package IS
  FUNCTION "-" (a,b : STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR IS
    ALIAS aa: STD_LOGIC_VECTOR(1 To a 'LENGTH) IS a;
    ALIAS bb: STD_LOGIC_VECTOR(1 To b 'LENGTH) IS b;
    VARIABLE result: STD_LOGIC_VECTOR(1 To a 'LENGTH);
    VARIABLE carry: STD_LOGIC := '0';
  BEGIN
    FoR i IN result'REVERSE_RANGE Loop
      result(i) := aa(i) XOR bb(i) XOR carry;
```

از بخش فانکشن تا بگین (Function to begin) بخش اعلان است. دو تا ALIAS اعلان می شود

دو ورودی a و b که STD_LOGIC_VECTOR هستند هر اندیس گذاری می توانند داشته باشند. پس با aa و bb بگذار

که از 1 تا طولی که دارد باشد. یعنی نرمالیزه کرده است. دو متغیر result و carry هم داریم

```
      carry := (aa AND bb(i)) oR (aa(i) AND carry) oR (bb(i) AND carry);
         oR(bb(i) AND carry);
    END Loop;
    RETURN result;
  END FUNCTION "-";
END PACKAGE BODY;
---------------------------
```

ادامه در صفحه بعدی

ادامه سوال 2)

چون ما پکیج داریم درست است اصلی داخل کتابخانه ها باید work.my_package.all را بنویسیم

main code: -

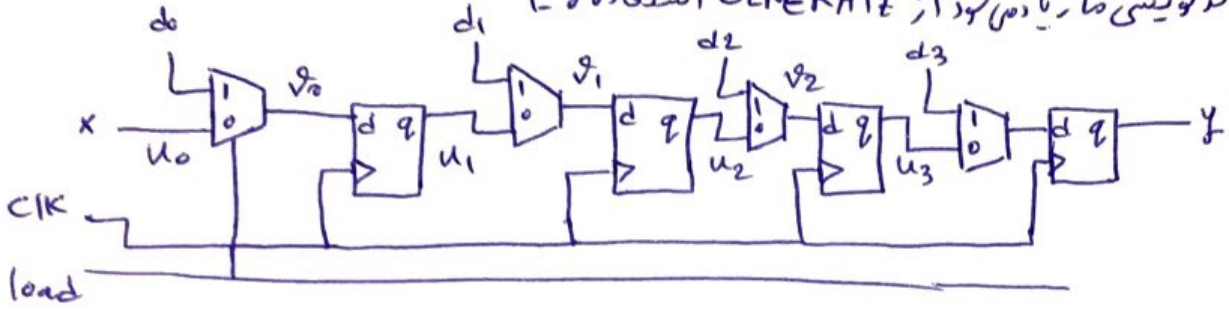
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.my_package.all;

ENTITY add_stdlogic IS
  PORT(x: IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
       y: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
END ENTITY;

ARCHITECTURE adder OF add_stdlogic IS
   CONSTANT const: STD_LOGIC_VECTOR(7 DOWNTO 0) := "00001111";
BEGIN
   Y <= X + const + "01111111"; --overloaded "+" operator
END ARCHITECTURE;
```

خروجی و ورودی باید هم طول و هم اندیس خروجی باشد. داخل بدنه آرشیتکچر باید قبلی یک مقدار بدهیم (00001111_2) در داخل بدنه معماری هم باید Y تخصیص داده شود به پورت ورودی و کانستنت و عدد داده شده که این مقدار باید سربارگذاری یا overload بشود

یک شیفت رجیستر 4-بیتی در نظر می‌گیریم که بتواند بارگذاری موازی انجام دهد.
کد عام Generic در حالت دلخواه N بیتی.
مواقعی که کدنویسی ما زیاد می‌شود از GENERATE استفاده می‌کنیم




```
------ Package: ------------
PACKAGE my_declarations IS
   TYPE twoD IS ARRAY(NATURAL RANGE <>, NATURAL RANGE <>) OF BIT;

-------------------------------
COMPONENT mux IS
   PORT (a, b, sel: IN BIT; x: out BIT);
END COMPONENT;

-------------------------------
COMPONENT flipflop IS
   PORT (d, clk: IN BIT; q: OUT BIT);
END COMPONENT;

-------------------------------
END PACKAGE;

-------------------------------
--- Main code: ----------
USE work.my_declarations.all;

-------------------------------
ENTITY shift_register IS
   GENERIC (M: POSITIVE := 4;
            N: POSITIVE := 8);
   PORT (clk, load: IN BIT;
     x: IN BIT_VECTOR(N-1 DOWNTO 0);
     d: IN twoD(0 TO M-1, N-1 DOWNTO 0);
     y: OUT BIT_VECTOR(N-1 DOWNTO 0));
END ENTITY;
```

ادامه در صفحه بعد

13.

ادامه سوال 3)

```
ARCHITECTURE structural OF shift_register IS
    SIGNAL u: twoD (0 TO M, N-1 DOWNTO 0);
    SIGNAL v: twoD (0 TO M-1, N-1 DOWNTO 0);
BEGIN
    ---- Transfer x→u and u→y: ----
    gen1: FOR i IN N-1 DOWNTO 0 GENERATE
        u(0,i) <= x(i);
        y(i) <= u(M,i);
    END GENERATE gen1;
    ---- update internal arry: ----
    gen2: FOR i IN 0 TO M-1 GENERATE
        gen3: FOR j IN N-1 DOWNTO 0 GENERATE
            mux1: mux PORT MAP (u(i,j), d(i,j), load, v(i,j));
            dff1: flipflop PORT MAP (v(i,j), clk, u(i+1,j));
        END GENERATE gen3;
    END GENERATE gen2;
END ARCHITECTURE;
```

توضیحات:

فرض شده است که فایلی به نام mux و flipflop قبلا نوشته شده است. در پکیج ما اعلان ها را قرار دادیم تا کد اصلی درگیر اعلان ها نشود. 3 عملیات اعلان داریم. یک اعلان twoD، آرایه. در واقع حدود اندیس ها را مشخص کردیم و در این قسمت حدود آزاد است و نوع کلی و آرایه دوبعدی تعریف کردیم. از نوع بیت است در کامپوننت و فلیپ فلاپ داخل پکیج آوردیم. کد اصلی ابتدا که پکیج را می نویسیم. M تعداد طبقات است که 4 تا است و عرض بیت N است که 8 بیتی است و D ها دو بعدی ما است. چون do not داریم و هر ورودی یک عرض بیت دارد. پس twoD تعریف می شود. در بخش اعلان معماری، سیگنال میانی برای چسباندن کامپوننت ها می آید. و باید دیکلر کنیم. بعد نمونه سازی انجام دادیم پس قدم اول وصل کردن u به x است و آخر u به خروجی y وصل می شود. با دستور for generate، u و y وصل کردیم. هرکدام N عرض بیت داشتند. بعد اول طبقه است و شماره دوم عرض بیت است. در حلقه for نیز نمونه سازی می کنیم.