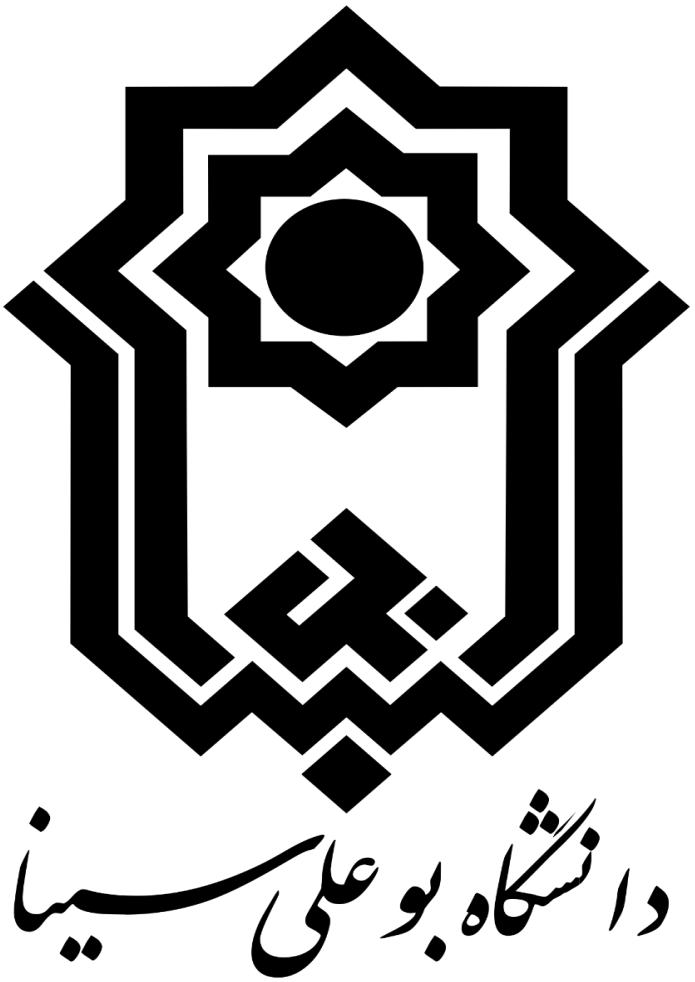
پروژه مدارهای منطقی

پروژه عملی:vending machine



نیمسال اول 1403-1402

استاد درس:دکتر حاتم عبدولی

تدریس یار:نوید پاکنژاد پناهی

دانشجو : دانیال کشاورز نژاد

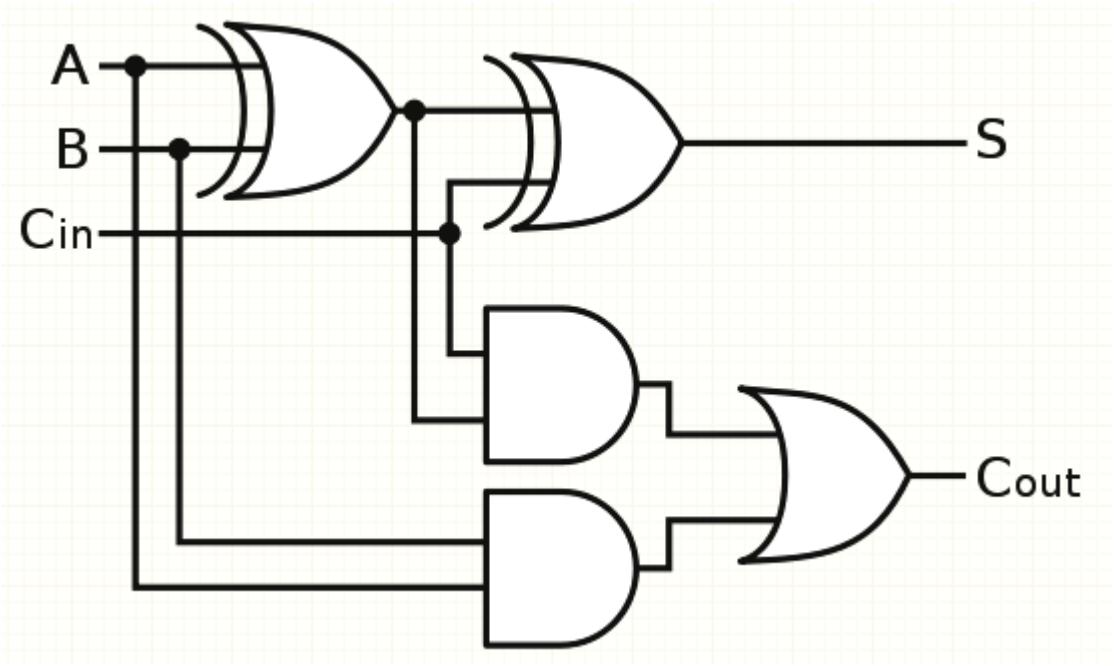
فهرست مطالب

مقدمه :

یکی از راه های تسهیل فرایند خرید و فروش استفاده از دستگاه vending machine است این دستگاه می تواند با گرفتن موجودی کاربر موجودی او را مرتباً بروز کند و در نهایت با فشار دادن دکمه بلیت ، بلیت دلخواه به کاربر در صورت داشتن موجودی به او تحویل داده می‌شود در پایان عملیات کاربر با فشار دادن دکمه پایان می تواند بقیه موجودی خود را تحویل بگیرد.

ماژول simple adder:

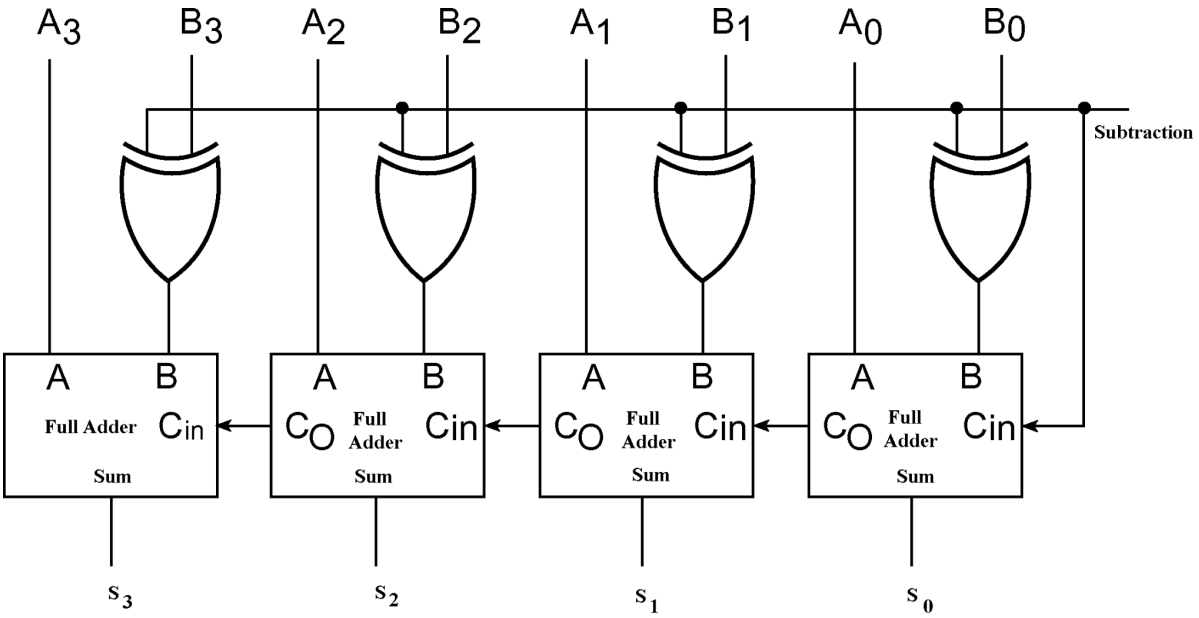
در این ماژول ما یک جمع کننده ساده داریم این ماژول دو بیت a و b را از کاربر گرفته و توانایی جمع کردن آنها را نیز دارد در پایان عملیات خروجی عملیات از بیت به نام sum خارج می شود و در صورت درست شدن carry out خروجی cout برابر با ۱ می‌شود



ماژول adder:

این ماژول توانایی جمع و تفریق دو عدد 6 بیت را دارا است نحوه این انجام این عملیات با کمک از ماژول simple adder می باشد مجموعا پنج simple adder در این ماژول وجود دارند و هر کدام وظیفه محاسبه یک بیت را دارند

برای رسیدن به جواب درست تمامی cin ماژول ها با بیت کم ارزش تر به cout ماژولی با بیت پر ارزش تر وصل می شود و در نهایت آخرین cout به cout ماژول بزرگتر وصل می شود
در این طراحی cin کم ارزش ترین بیت به cin ماژول بالاتر وصل شده است نقش این ورودی در add در عملیات تفریق کلیدی می شود زیرا هر بیت ورودی عدد پنج بیتی اول با cin اولیه ماژول xor گرفته می شود علت این عملیات تولید مکمل دوم عدد برای منفی سازی عدد است



طراحی 4 بیتی از ماژول adder

ماژول vending :

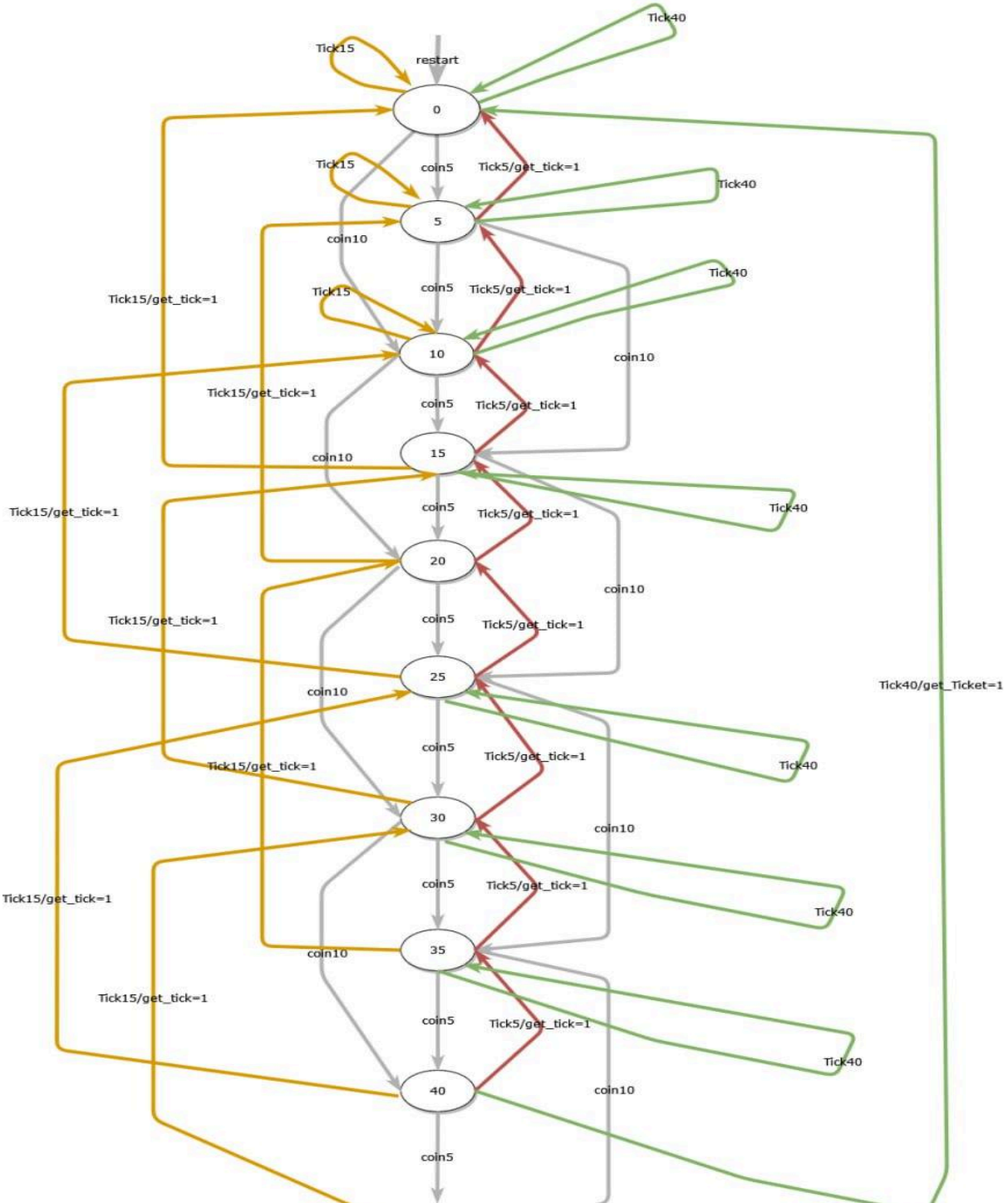
این ماژول ، ماژول اصلی و هدایتگر اصلی برنامه است در این ماژول با entity به شرح زیر است

```
entity vending is
    port
    (   ten , five , clk           : in  std_logic;
        tick_5 , tick_15 , tick_40 : in  std_logic;
        done , reset               : in  std_logic;
        get_tick                   : out std_logic;
        z                          : out std_logic_vector(5 downto 0)) ;
end entity;
```

اولین ورودی clk است که نماینده clock است ماشین ما حساس به لبه بالا رونده کلاک است دو ورودی ten و five به ترتیب نماینده سکه های 1000 و 500 است و خانواده tick هر کدام نماینده بلیت به ترتیب 500 و 1500 و 40000 هستند در نهایت کاربر با فشار دادن دکمه done می تواند مقدار باقیمانده پول خود را در z دریافت کند
خروجی get_tick در صورتی که کاربر از دستگاه بلیت بخواهد و موجودی کافی داشته باشد آنگاه این خروجی برابر 1 می شود و از موجودی کم می شود

شرح fsm:

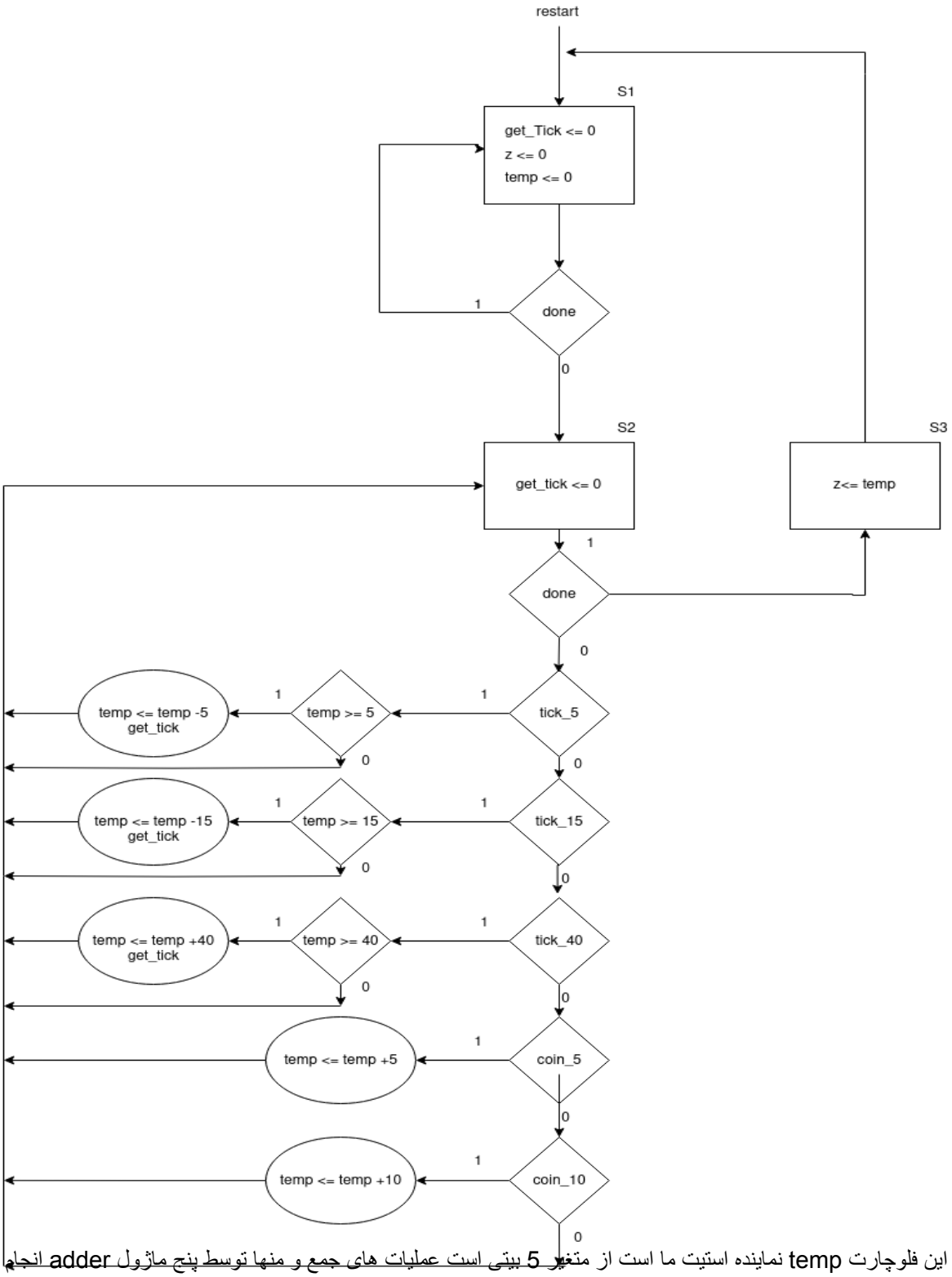
در این ماشین fsm با شرح زیر استفاده شده است



برای آسان تر شدن پیاده سازی هر state با مقدار موجودی کاربر نشان داده شده است این fsm به صورت moore پیاده شده است تا تعداد state های کمتری وجود داشته باشد و همچنین بلیت به محض فشار دادن دکمه صادر شود

توضیحات flowchart:

با توجه که هر state با موجودی کاربر نشان داده می شود می توان فلوچارت زیر را رسم کرد



در این فلوچارت temp نماینده استیت ما است از متغیر 5 بیتی است عملیات های جمع و منها توسط پنج ماژول adder انجام می شوند این 5 ماژول به یک مالتی پلکسر در temp جایگزین می شود از آنجا که temp دقیقا برابر موجودی کاربر است با فشار دادن دکمه done متغیر temp درون z ریخته می شود و به کاربر نشان داده می شود

پیاده سازی :

```
begin

    adder5  :adder port map(temp,"000101" , '0' , result5  , garbage);
    adder10 :adder port map(temp,"001010" , '0' , result10 , garbage);
    subber5 :adder port map("000101",temp , '1' , sub5     , garbage);
    subber15:adder port map("001111",temp , '1' , sub15    , garbage);
    subber40:adder port map("101000",temp , '1' , sub40    , garbage);

    process(clk)
    begin
        if(clk'event and clk = '1') then
            get_tick <= '0';
            z <= "000000";
            if   res = '1' then

                temp <= "000000";
                if done = '0' then
                    res <= '0';
                end if;
            elsif done = '1' then
                z <= temp;
                res <= '1';
            elsif    tick_5  = '1'  and temp > "000101" then
                temp <= sub5;
                get_tick <= '1';
            elsif tick_15  = '1' and temp > "001111" then
                temp <= sub15;
                get_tick <= '1';
            elsif tick_40  = '1' and temp > "101000" then
                temp <= sub40;
                get_tick <= '1';
            elsif five = '1' then
                temp <= result5;
            elsif ten  = '1' then
                temp <= result10;
            end if;
            --z <= temp;
        end if;
    end process;
end architecture;
```

در ادامه در process با ظاهر شدن clock بالا روند برنامه به دنبال یکی از سناریو های ممکنه می گردد مثلا در آخرین شرط چک می شود که سکه 1000 تومانی دیده شده است یا خیر در ادامه با کمک یکی از ماژول های adder مقدار temp را افزایش می دهد

بعد از فشار دادن دکمه done مقدار ذخیره شده در temp به کاربر داده می شود و سپس دستگاه reset می شود