

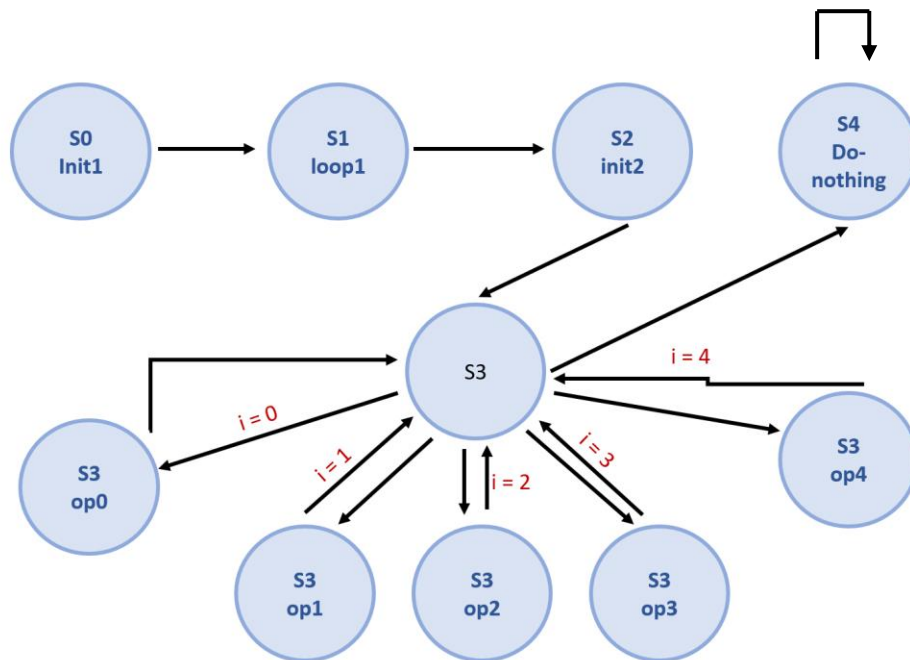


تمرین دوم هم طراحی سخت افزار - نرم افزار

نیلوفر مرادی جم 97243063

کیما صدیقی 97243046

## سوال اول



Init1:  $i \leftarrow 0$

loop1:  $\text{taps}[0 : 3] \leftarrow \text{taps}[1 : 4]$

init2:  $\text{taps}[4] \leftarrow a, r \leftarrow 0, i \leftarrow 0$

op[i]:  $r \leftarrow r + \text{taps}[i] * c[i], i \leftarrow i+1$

## سوال پنجم

### گزارش کار:

اولین چالش مطرح شده برای انجام این تمرین، نصب ابزار بود. با توجه به اینکه ما اوبونتوی 64 بیتی داشتیم و gezel تنها روی اوبونتوی 32 بیتی کار میکرد، ابتدا سعی کردیم راه حلی بیابیم که نسخه 64 بیتی ما، برنامه های 32 بیتی را نیز پشتیبانی کند. با وجود اینکه ظاهراً این مسئله حل شده بود همچنان gezel نصب نمیشد.

با راهنمایی یکی از بچه ها متوجه شدیم سابقاً آقای محمد صادق دهقان یک ایمپچ داکر برای gezel نوشته است و به کمک داکر موفق به دریافت آن و نصب gezel شدیم. در ادامه برای نوشتن کد، فلوچارت و استیت ماشین مناسبی مطابق با صورت سوال طراحی کردیم و سعی کردیم کد آن را بنویسیم.

اولین چالشی که در نوشتن کد با آن روبرو شدیم، اجبار مقداردهی اولیه برای nslookup که از آن برای نوشتن آرایه taps استفاده کرده بودیم، بود.

لذا برای جلوگیری از این مقداردهی اولیه، مجبور شدیم هر  $taps(i)$  را بصورت رجیستر تعریف کنیم.

چالش بعدی این بود که موفق نمیشدیم به خانه  $i$  از nslookup دسترسی پیدا کنیم.  $C[i]$  خطای سینتکسی میداد و تنها مرجع ما که کتاب و راهنمای جزل بود هم به همین روش به خانه های آرایه دسترسی داشت. با آزمون و خطا و بررسی فراوان فهمیدیم که باید از  $C(i)$  به جای  $C[i]$  استفاده کنیم.

## کدها و خروجی های سوال 7:

```
May 6 23:36 •
root@3e81cc5ea376: /opt/src

1 dp filter_dp(in a: ns(33); out r_out: ns(32)) {
2   reg taps0, taps1, taps2, taps3, taps4 : tc(32);
3   reg i, r : tc(32);
4   lookup c : tc(32) = {-1, 5, 10, 5, -1};
5
6   always{
7     r_out = r;
8   }
9
10  sfg init1 {
11    i = 0;
12  }
13
14  sfg loop1 {
15    taps0 = taps1;
16    taps1 = taps2;
17    taps2 = taps3;
18    taps3 = taps4;
19  }
20
21  sfg init2 {
22    taps4 = a;
23    r = 0;
24    i = 0;
25  }
26
27  sfg op0 {
28    r = r + taps0 * c(i);
29    i = i+1;
30  }
31
32  sfg op1 {
33    r = r + taps1 * c(i);
34    i = i+1;
35  }
36
37  sfg op2 {
38    r = r + taps2 * c(i);
39    i = i+1;
40  }
41
42  sfg op3 {
43    r = r + taps3 * c(i);
44    i = i+1;
45  }
46
47  sfg op4 {
48    r = r + taps4 * c(i);
49    i = i+1;
50  }
51
52  sfg do_nothing { }
53
54 }
```

1,1 Top

```
root@3e81cc5ea376: /opt/src

56
57 fsm filter_ctl (filter_dp) {
58     initial s0;
59     state s1, s2, s3, s4;
60
61     @s0 (init1) -> s1;
62     @s1 (loop1) -> s2;
63     @s2 (init2) -> s3;
64     @s3 if (i == 0) then (op0) -> s3;
65         else if (i == 1) then (op1) -> s3;
66         else if (i == 2) then (op2) -> s3;
67         else if (i == 3) then (op3) -> s3;
68         else if (i == 4) then (op4) -> s3;
69         else (do_nothing) -> s4;
70     @s4 (do_nothing) -> s4;
71 }
72
73 dp filter_test(out a : tc(32)) {
74
75     sfg run {
76         a = 10 ;
77     }
78 }
79
80 hardware h_filter_test(filter_test) {run;}
81
82 dp filter_sys {
83     sig a : tc(32);
84     sig finish : ns(1);
85     sig r : tc(32) ;
86     always {
87         $display("cycle = ", $cycle, "      r = ", r);
88     }
89     use filter_dp(a, r);
90     use filter_test(a);
91 }
92
93 system s {
94     filter_sys;
95 }
```

92,0-1 Bot

```
root@3e81cc5ea376: /opt/src

root@3e81cc5ea376:/opt/src# vim system.vhd
root@3e81cc5ea376:/opt/src# fdlvhd gcd.fdl
Pre-processing System ...
Output VHDL source ...
-----
Generate file: filter_dp.vhd
Generate file: filter_test.vhd
Generate file: filter_sys.vhd
Generate file: system.vhd
Generate file: std_logic_arithext.vhd
root@3e81cc5ea376:/opt/src# fdlsim gcd.fdl 10
cycle = 0      r = 0
cycle = 1      r = 0
cycle = 2      r = 0
cycle = 3      r = 0
cycle = 4      r = 0
cycle = 5      r = 0
cycle = 6      r = a
cycle = 7      r = f
cycle = 8      r = 18
cycle = 9      r = 18
root@3e81cc5ea376:/opt/src#
```



کدها و خروجی های سوال 8:

```
root@3e81cc5ea376: /opt/src
dp filter_dp(in a: ns(33); out r: ns(32)) {
    reg taps0, taps1, taps2, taps3, taps4 : tc(32);

    always {
        taps4 = a;
        taps3 = taps4;
        taps2 = taps3;
        taps1 = taps2;
        taps0 = taps1;
        r = -taps0 + 5*taps1 + 10*taps2 + 5*taps3 - taps4;
    }

    sfg do_nothing {}
}

fsm filter_ctl (filter_dp) {
    initial s0;

    @s0 (do_nothing) -> s0;
}

-- INSERT --
1,1 Top
```

```
root@3e81cc5ea376: /opt/src
Failed to parse gcd.fdl
root@3e81cc5ea376:/opt/src# vim gcd.fdl
root@3e81cc5ea376:/opt/src# fdlsim gcd.fdl 10
cycle = 0      r = 0
cycle = 1      r = ffffffff6
cycle = 2      r = 28
cycle = 3      r = 8c
cycle = 4      r = be
cycle = 5      r = b4
cycle = 6      r = b4
cycle = 7      r = b4
cycle = 8      r = b4
cycle = 9      r = b4
root@3e81cc5ea376:/opt/src# fdlvhd gcd.fdl
Pre-processing System ...
Output VHDL source ...
-----
Generate file: filter_dp.vhd
Generate file: filter_test.vhd
Generate file: filter_sys.vhd
Generate file: system.vhd
Generate file: std_logic_arithext.vhd
root@3e81cc5ea376:/opt/src#
```