گزارش پیاده سازی تکلیف 7 هم طراحی سخت افزار و نرم افزار

غزل کلهری 97243058

عرفان مشیری 97243066

هدف از انجام این تکلیف، شبیه سازی سیستم سخت افزار - نرم افزار با استفاده از محیط GEZEL برای پیاده سازی فیلتر FIR بود. بدین ترتیب که برنامه داده شده به زبان C را به دو بخش نرم افزاری و سخت افزاری تفکیک کرده و از طریق پردازنده ARM ارتباط این دو بخش را برقرار کردیم. در ادامه توضیحات هریک از بخش ها به تفکیک جزییات آورده شده است.

1. بخش نرم افزار:

ابتدا تعدادی پایه ارتباطی برای برقراری ارتباط با بخش سخت افزار تعریف میکنیم.

```
volatile int *in_ready  = (volatile int *) 0x80000000;
volatile int *out_ready = (volatile int *) 0x80000004;
volatile int *in_coef   = (volatile int *) 0x80000008;
volatile int *out_res   = (volatile int *) 0xA0000008;
volatile int *out_sign  = (volatile int *) 0xB0000008;
```

- In_ready برای فرستادن سیگنال آماده به کار از سمت نرم افزار به سخت افزار هنگام اتمام مقداردهی ورودی ها
- out_status برای فرستادن سیگنال آماده دریافت از سخت افزار به نرم افزار هنگام آماده شده نتایج بدست آمده
- in_coef برای ارسال مقادیر آرایه coeff به سخت افزار
- out_res برای دریافت اعداد خروجی بدست آمده از سخت افزار
- out_sign برای دریافت علامت خروجی های اعداد بدست آمده از سخت افزار

در ادامه تابع to_fixed را برای تبدیل اعداد اعشاری به fixed هنگام ارسال مقادیر ورودی به سخت افزار و تابع to_float را برای تبدیل اعداد fixed به اعشاری هنگام دریافت مقادیر خروجی از سخت افزار تعریف میکنیم.

```
// 32 bit: s(1), m(7), n(24)
int to_fixed(double float_num) {
    return (int)(float_num * conv_param);
}

double to_float(int int_num) {
    return 1.0 * int_num / conv_param;
}
```

فرمت عددی فرض شده برای اعداد اعشاری شامل 27 بیت اعشار، 7 بیت مقدار و 1 بیت علامت است.

تابع to_fixed عدد اعشاری را در پارامتر conv_param (2^{27}) ضرب کرده و بعد از دور ریختن اعشار، آن را برمیگرداند.
تابع to_float عدد fixed را بر conv_param تقسیم کرده تا عدد اعشاری تبدیل شده بدست بیاید.

تابع main، در دو مرحله، ورودی ها را به سخت افزار فرستاده و خروجی ها را از آن برمیگرداند. بدین ترتیب که ابتدا پایه in_ready را با مقدار صفر ست میکند که به معنای عدم آماده بودن ورودی ها و شروع کار سخت افزار میباشد. سپس در یک حلقه به تعداد عناصر آرایه coeff محتوای هر ایندکس از آرایه را بعد از تبدیل کردن به عدد fixed، در پایه in_coeff ست میکند. در ادامه پایه in_ready را با مقدار 1 ست میکند تا سخت افزار کار خود را شروع کند.

بعد از شروع کار سخت افزار، مادامیکه پایه out_ready برابر صفر است و نتایج آماده نشده، در یک حلقه صبر میکند. در نهایت با 1 شدن out_ready در یک حلقه 120 تایی، اعداد خروجی را تک به تک دریافت میکند. پس از تبدیل آنها به فرمت اعشاری، بنابر بیت علامت آنها که از پایه out_sign دریافت میشود، مثبت یا منفی بودن اعداد را تعیین کرده و سپس آنها را نمایش میدهد.

2. بخش سخت افزار:

ابتدا تعدادی ipblock برای برقراری ارتباط با بخش نرم افزار تعریف کرده و آدرس هرکدام را نظیر به نظیر با پایه ارتباطی مرتبط با آن در بخش نرم افزار ست میکنیم. ipblock های in_coef, out_res, out_sign که بیش از یک مقدار را ذخیره میکنند، همانند ساختار آرایه ای با IO های address برای تعیین خانه ای از حافظه که قصد خواندن یا نوشتن از آن را داریم، wr برای دسترسی نوشتن در خانه های حافظه، idata برای نوشتن در حافظه و odata برای خواندن از حافظه تعریف میکنیم. iptype آنها را armbuffer قرار داده تا معادل یک بافر اعداد را در قالب ساختاری آرایه مانند ذخیره کند، و range آن را به اندازه size بافر ضرب در طول مقادیر به بایت قرار میدهیم.

```
ipblock in_coef(in  idata   : tc(32);
                out odata   : tc(32);
                in  address : ns(10);
                in  wr      : ns(1)) {
    iptype "armbuffer";
    ipparm "core    = arm";
    ipparm "address = 0x80000008";
    ipparm "range   = 0x10C";   // 67 * 32bit -> 268(0x10C)
}
```

حال dp fir را تعریف میکنیم تا عملیات مد نظر را انجام دهد. ورودی/خروجی های آن مطابق با ipblock های تعریف شده است. تعدادی رجیستر کمکی نیز برای مقداردهی به آنها تعریف میکنیم. lookup pulse نیز مقادیر آرایه pulse را داراست. بلاک always نیز در هر لبه بالارونده کلاک، خروجی های dp را با مقادیر temp هریک از آنها که در sfg های آتی مقدار میگیرند، ست میکند.

در ادامه تعدادی sfg داریم که توسط fsm کنترلر fir فراخوانی میشوند:

- init تمامی مقادیر temp خروجی ها و شمارنده های حلقه را مقداردهی اولیه میکند.
- skip همانند اسمش کار خاصی انجام نمیدهد و تنها برای گذر از استیت های مختلف است.
- first_loop اولین خط از اولین حلقه کد اصلی را اجرا میکند یعنی ضرب مقدار pulse حال در coeff[0] و ست کردن آن در output[i]. سپس مقدار coeff را با افزودن آدرس آن به خانه بعدی میبرد.
- second_loop عبارت حلقه دوم از کد اصلی را محاسبه کرده و سپس j, coeff را یکی افزایش میکند.

- update_vals با تمام شدن حلقه دوم، مقدار شمارنده حلقه اول را یکی افزایش داده و شمارنده حلقه دوم را ریست میکند. در ادامه آدرس coeff را نیز ریست کرده و آدرس های مرتبط با خروجی ها را یک خانه افزایش میدهد.
- write_res دسترسی نوشتن خروجی ها (wr) را 1 کرده و سپس آنها را با مقدار و علامت output_i ست میکند.
- done سیگنال output_ready را 1 کرده تا نشان دهنده اتمام کار سخت افزار باشد.

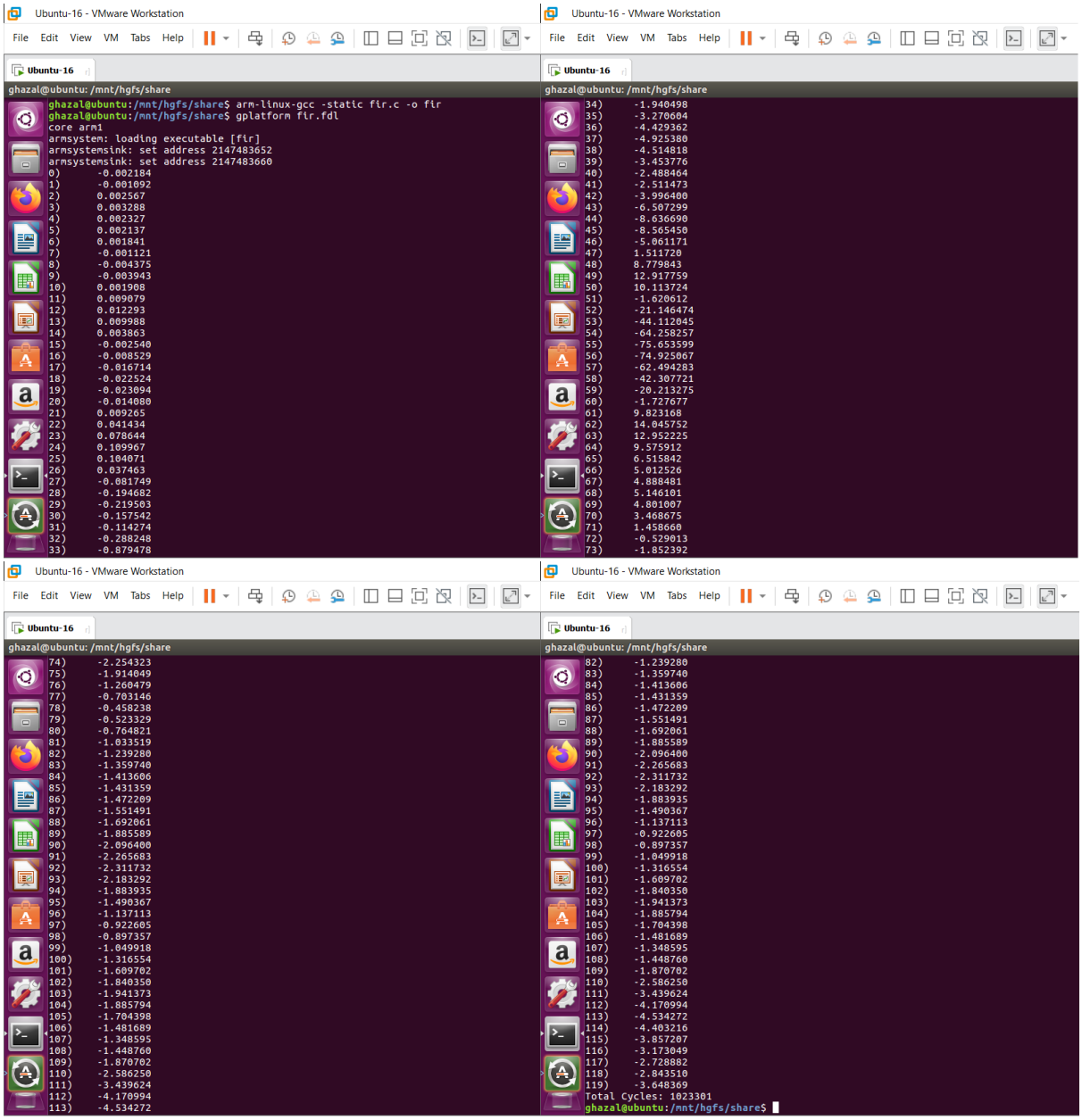
در ادامه fsm fir_ctl را تعریف میکنیم که کنترلر fir میباشد و فراخوانی sfg ها بر اساس استیت های مرتبط با هرکدام را بر عهده دارد. بطور کلی 5 استیت داریم که در ادامه توضیح داده میشوند:

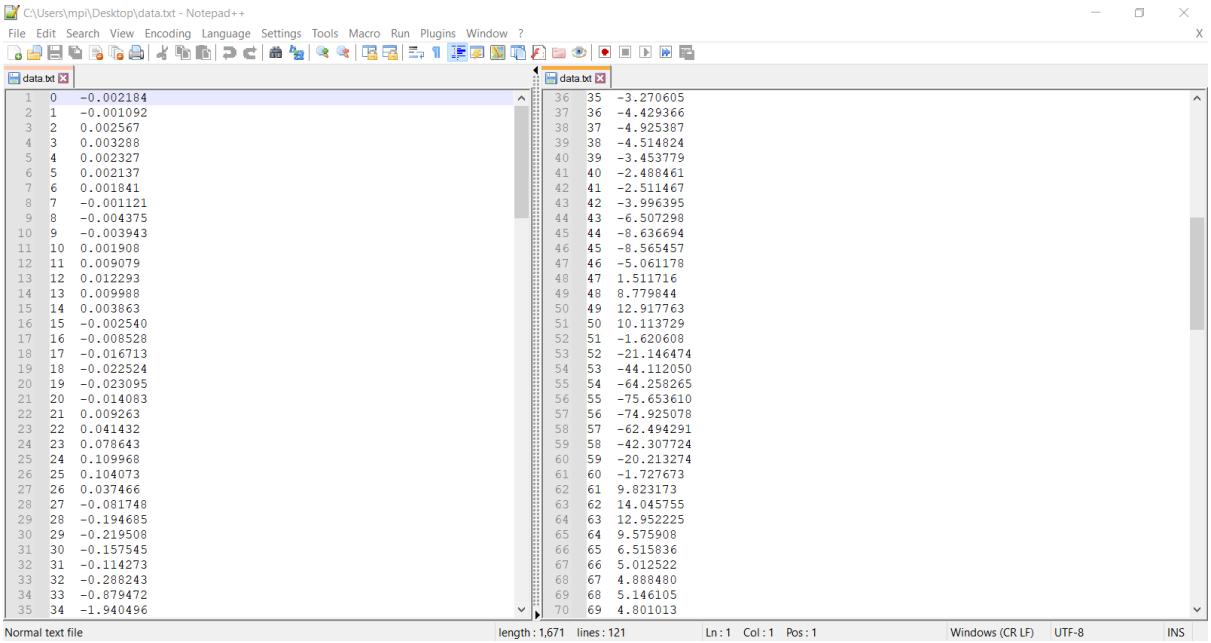
- s0 در این استیت، در صورت یک بودن سیگنال input_ready که به معنای انتقال کامل ورودی های از نرم افزار است، کار سخت افزار شروع شده و پس از اجرای input_ready به s1 میرویم. در غیر اینصورت با اجرای skip منتظر میمانیم تا input_ready فعال شود.
- s1 در این استیت در صورت برقرار بودن شرط ورود به حلقه اول، first_loop اجرا شده و به s2 میرویم. در غیر اینصورت به معنای اتمام کار حلقه است و به s4 میرویم.
- s2 در این استیت در صورت برقرار بودن شرط های ورود به حلقه دوم، second_loop اجرا شده و باز به همین استیت برمیگردیم تا زمانیکه کار حلقه دوم تمام شود. در غیر اینصورت به معنای اتمام کار حلقه دوم و آماده شدن یکی از خروجی ها است. لذا write_res را اجرا کرده و به s3 میرویم.
- s3 در این استیت update_vals را اجرا کرده تا مقادیر حلقه ها آپدیت شوند و سپس به s1 بازمیگردیم تا استپ بعدی از حلقه اول اجرا شود.
- s4 استیت نهایی ما که با اجرای done اتمام کار عملیاتی سخت افزار را اطلاع میدهد.

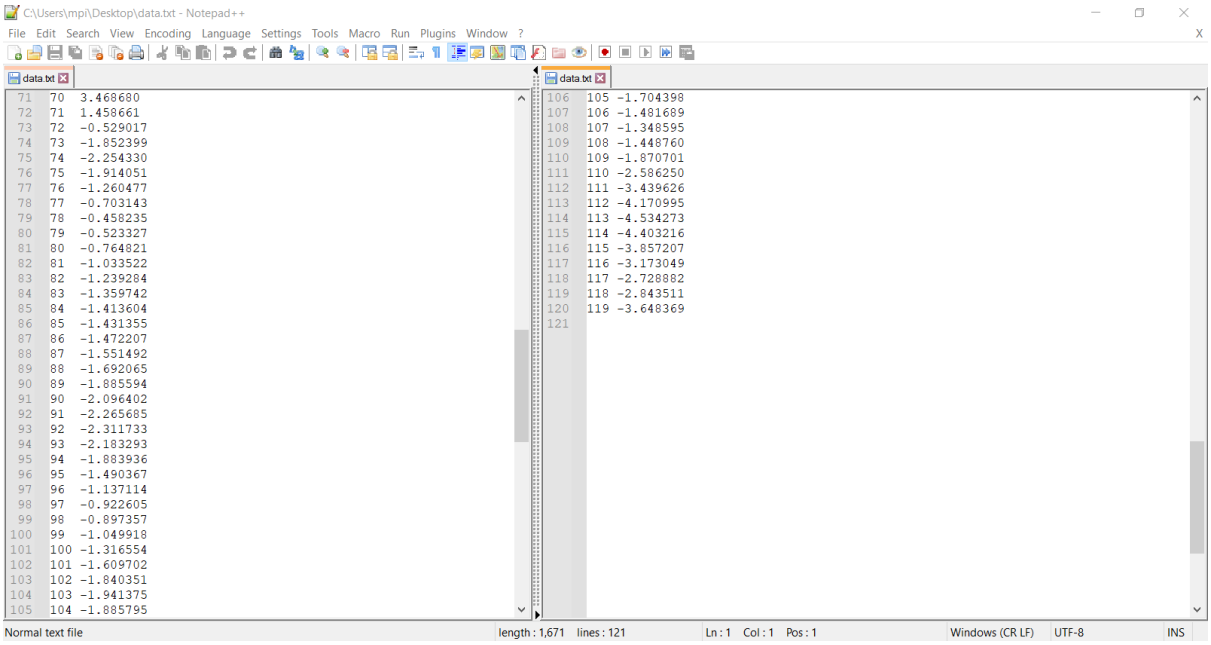
در نهایت dp tb را تعریف میکنیم تا با فراخوانی هریک از ipblock های تعریف شده و dp fir، برنامه اجرا شود.

3. نتایج:

تصاویر بخش اول، نتایج حاصل از اجرای برنامه در محیط شبیه سازی Gezel بوده و تصاویر بخش دوم حاصل از اجرای کد صورت سوال بطور جداگانه است. همانطور که مشاهده میشود، نتایج بدست آمده در دو بخش با تقریب بسیار نزدیک مشابه یکدیگرند.




C:\Users\mpi\Desktop\data.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
data.txt
0 -0.002184
1 -0.001092
2 0.002567
3 0.003288
4 0.002327
5 0.002137
6 0.001841
7 -0.001121
8 -0.004375
9 -0.003943
10 0.001908
11 0.009079
12 0.012293
13 0.009988
14 0.003863
15 -0.002540
16 -0.008528
17 -0.016713
18 -0.022524
19 -0.023095
20 -0.014083
21 0.009263
22 0.041432
23 0.078643
24 0.109968
25 0.104073
26 0.037466
27 -0.081748
28 -0.194685
29 -0.219508
30 -0.157545
31 -0.114273
32 -0.288243
33 -0.879472
34 -1.940496
35 -3.270605
36 -4.429366
37 -4.925387
38 -4.514824
39 -3.453779
40 -2.488461
41 -2.511467
42 -3.996395
43 -6.507298
44 -8.636694
45 -8.565457
46 -5.061178
47 1.511716
48 8.779844
49 12.917763
50 10.113729
51 -1.620608
52 -21.146474
53 -44.112050
54 -64.258265
55 -75.653610
56 -74.925078
57 -62.494291
58 -42.307724
59 -20.213274
60 -1.727673
61 9.823173
62 14.045755
63 12.952225
64 9.575908
65 6.515836
66 5.012522
67 4.888480
68 5.146105
69 4.801013
Normal text file length : 1,671 lines : 121 Ln : 1 Col : 1 Pos : 1 Windows (CR LF) UTF-8 INS


C:\Users\mpi\Desktop\data.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
data.txt
70 3.468680
71 1.458661
72 -0.529017
73 -1.852399
74 -2.254330
75 -1.914051
76 -1.260477
77 -0.703143
78 -0.458235
79 -0.523327
80 -0.764821
81 -1.033522
82 -1.239284
83 -1.359742
84 -1.413604
85 -1.431355
86 -1.472207
87 -1.551492
88 -1.692065
89 -1.885594
90 -2.096402
91 -2.265685
92 -2.311733
93 -2.183293
94 -1.883936
95 -1.490367
96 -1.137114
97 -0.922605
98 -0.897357
99 -1.049918
100 -1.316554
101 -1.609702
102 -1.840351
103 -1.941375
104 -1.885795
105 -1.704398
106 -1.481689
107 -1.348595
108 -1.448760
109 -1.870701
110 -2.586250
111 -3.439626
112 -4.170995
113 -4.534273
114 -4.403216
115 -3.857207
116 -3.173049
117 -2.728882
118 -2.843511
119 -3.648369
Normal text file length : 1,671 lines : 121 Ln : 1 Col : 1 Pos : 1 Windows (CR LF) UTF-8 INS