

به نام خدا

پروژه پایانی

هم طراحی سخت افزار و نرم افزار

دکتر بیکی

زمستان و بهار ۱۴۰۰-۱۳۹۹

فهرست

۳	طراحی سخت افزاری
۱۳	طراحی نرم افزاری

طراحی سخت افزاری

برای طراحی سخت افزاری از ابزار Gezel استفاده شده است.

ورودی های برای محاسبه دترمینان های 2×2 و 3×3 به صورت سطر به سطر به برنامه داده می شود و اعداد در این پروژه به صورت علامت دار و ۸ بیتی در نظر گرفته شده اند و می توانند اعداد از بازه $[-128, 127]$ را پشتیبانی کنند.

• دترمینان ماتریس 2×2 با FSM :

برای طراحی سخت افزاری برای محاسبه دترمینان یک ماتریس 2×2 با fsm داریم:

```
1 dp det2 (in a_in , b_in, c_in, d_in : tc(8);
2   out result : tc(16))
3 {
4   reg a, b, c, d : tc(8);
5   reg multi1, multi2 : tc(16);
6   reg dif : tc(16);
7
8   sfg inint {a = a_in; b = b_in; c = c_in; d = d_in;
9     $display ("cycle=", $cycle, "|| a=", a_in, "|| b=", b_in, "|| c=", c_in, "|| d=", d_in);}
10  sfg multiplication {multi1 = a*d; multi2 = b*c;
11    $display ("cycle=", $cycle, "|| multi1=", multi1, "|| multi2=", multi2, "|| result=", result);}
12  sfg difference { dif = multi1 - multi2;
13    $display ("cycle=", $cycle, "|| multi1=", multi1, "|| multi2=", multi2, "|| dif=", dif, "|| result=", result);}
14  sfg output {result = dif;$display("result=", result, "|| cycle=", $cycle);}
15  sfg outidle {result = 0;}
16
17 }
18
19
20 fsm det2_ctl (det2){
21
22   initial s0;
23   state s1, s2, s3;
24
25   @s0 (inint, outidle) -> s1;
26
27   @s1 (multiplication, outidle) -> s2;
28
29   @s2 (difference, outidle) -> s3;
30
31   @s3 (output) -> s3;
32
33 }
```

شکل ۱ - کد برنامه gezel برای محاسبه دترمینان ماتریس 2×2

در ادامه تست کیس های مربوطه را مشاهده می کنیم

```

det2.fdl
30 @s2 (difference, outidle) -> s3;
31
32 @s3 (output) -> s3;
33
34 }
35
36 dp test_det2 (out a, b, c, d : tc (8)
37   sig run [
38     a = 7;
39     b = 8;
40     c = 6;
41     d = 5;
42   ]
43 )
44
45 hardware h_test_det2 (test_det2){run;
46
47 dp det2_sys {
48   sig a, b, c, d : tc (8);
49   sig result : tc(16);
50   use det2 (a, b, c, d, result);
51   use test_det2 (a, b, c, d);
52 }
53
54 system s {
55   det2_sys;
56 }
  
```

```

*** INFO: Cycle 2: eval
*** INFO: Cycle 2: eval ip
*** INFO: Cycle 2: disp
cycle=1||multi1=0/23||multi2=0/30||result=0
      0      23 det2.multi1
      0      30 det2.multi2
> Cycle 3
*** INFO: Cycle 3: RTCTL
det2_ctl: det2_ctl.s2 -> det2_ctl.s3
*** INFO: Cycle 3: eval_out ip
*** INFO: Cycle 3: eval
*** INFO: Cycle 3: eval ip
*** INFO: Cycle 3: disp
cycle=2||multi1=23/23||multi2=30/30||dif=0/fff3||result=0
      0      fff3 det2.dif
> Cycle 4
*** INFO: Cycle 4: RTCTL
det2_ctl: det2_ctl.s3 -> det2_ctl.s3
*** INFO: Cycle 4: eval_out ip
*** INFO: Cycle 4: eval
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
result=fff3||cycle=3
alireza@ubuntu:~/Desktop/finalproject$
  
```

شکل ۲ - تست کیس شماره ۱ محاسبه دترمینان 2x2

در این تست کیس، ورودی ها برابر هستند با (4, 6, 8, 7) و خروجی به صورت hex برابر است با 0xFFEC که به صورتی decimal در بازه اعداد علامت دار برابر است با -20 که نتیجه درستی می باشد. تعداد کلاک ها : خروجی در کلاک 4 آماده می باشد.

```

det2.fdl
30  @s2 (difference, outidle) -> s3;
31
32  @s3 (output) -> s3;
33
34  }
35
36  dp test_det2 (out a, b, c, d : tc (8)
37    sfg_run [
38      a = 3;
39      b = 5;
40      c = 1;
41      d = 8;
42    ]
43  )
44
45  hardware h_test_det2 (test_det2){run;
46
47  dp det2_sys {
48    sig a, b, c, d : tc (8);
49    sig result : tc(16);
50    use det2 (a, b, c, d, result);
51    use test_det2 (a, b, c, d);
52  }
53
54  system s {
55    det2_sys;
56  }

```

```

*** INFO: Cycle 2: eval
*** INFO: Cycle 2: eval ip
*** INFO: Cycle 2: disp
cycle=1||multi1=0/18||multi2=0/5||result=0
      0      18 det2.multi1
      0      5 det2.multi2
> Cycle 3
*** INFO: Cycle 3: RTCTL
det2_ctl: det2_ctl.s2 -> det2_ctl.s3
*** INFO: Cycle 3: eval_out ip
*** INFO: Cycle 3: eval
*** INFO: Cycle 3: eval ip
*** INFO: Cycle 3: disp
cycle=2||multi1=18/18||multi2=5/5||dif=0/13||result=0
      0      13 det2.dif
> Cycle 4
*** INFO: Cycle 4: RTCTL
det2_ctl: det2_ctl.s3 -> det2_ctl.s3
*** INFO: Cycle 4: eval_out ip
*** INFO: Cycle 4: eval
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
result=13||cycle=3
alireza@ubuntu:~/Desktop/finalproject$

```

شکل ۳ - تست کیس شماره ۲ محاسبه دترمینان 2x2

در این تست کیس، ورودی ها برابر هستند با (3, 5, 1, 8) که خروجی به صورت hex برابر است با 0x13 که به صورت decimal برابر است با 19 و مقدار بدست آمده صحیح می باشد.

تعداد کلاک ها : خروجی در کلاک 4 آماده می باشد.

```

det2.fdl
30 @s2 (difference, outidle) -> s3;
31
32 @s3 (output) -> s3;
33
34 }
35
36 dp test_det2 (out a, b, c, d : tc (8))
37   sfg run [f]
38     a = 117;
39     b = 110;
40     c = 120;
41     d = 126;
42   ]
43 }
44
45 hardwired h_test_det2 (test_det2){run;
46
47 dp det2_sys {
48   sig a, b, c, d : tc (8);
49   sig result : tc(16);
50   use det2 (a, b, c, d, result);
51   use test_det2 (a, b, c, d);
52 }
53
*** INFO: Cycle 2: eval
*** INFO: Cycle 2: eval ip
*** INFO: Cycle 2: disp
cycle=1|multi1=0/3996|multi2=0/3390||result=0
0 3996 det2.multi1
0 3390 det2.multi2
> Cycle 3
*** INFO: Cycle 3: RTCTL
det2_ctl: det2_ctl.s2 -> det2_ctl.s3
*** INFO: Cycle 3: eval_out ip
*** INFO: Cycle 3: eval
*** INFO: Cycle 3: eval ip
*** INFO: Cycle 3: disp
cycle=2|multi1=3996/3996|multi2=3390/3390||dif=0/606||result=0
0 606 det2.dif
> Cycle 4
*** INFO: Cycle 4: RTCTL
det2_ctl: det2_ctl.s3 -> det2_ctl.s3
*** INFO: Cycle 4: eval_out ip
*** INFO: Cycle 4: eval
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
result=606|cycle=3
alireza@ubuntu:~/Desktop/finalproject$

```

شکل ۴ - تست کیس شماره ۳ محاسبه دترمینان 2x2

در این تست کیس، ورودی ها (117, 110, 120, 126) هستند که خروجی به صورت hex برابر است با 0x606 می باشد، که در دستگاه decimal برابر است با 1542 و مقدار درستی می باشد.

تعداد کلاک ها : خروجی در کلاک 4 آماده می باشد.

• دترمینان ماتریس 3x3

برای محاسبه دترمینان ماتریس 3x3 به صورت سخت افزاری داریم:

```
det3_v1.fdl
1  dp calculate_det (in a_in, b_in, c_in, d_in : tc(8); out result : tc(16)){
2    | always{result = (a_in*d_in) - (b_in*c_in);}
3  }
4
5  dp calculate_multi (in a_in : tc(8); in multi : tc(16); out result : tc(32)){
6    | always{result = a_in * multi;}
7  }
8
9  dp calculate_sumsub (in a_in, b_in, c_in : tc(32); out result : tc(32)){
10   | always{result = a_in - b_in + c_in;}
11 }
12
13 dp calculate_det2 : calculate_det
14 dp calculate_det3 : calculate_det
15 dp calculate_multi2 : calculate_multi
16 dp calculate_multi3 : calculate_multi
17
18 dp det3 (in a_in, b_in, c_in, d_in, e_in, f_in, g_in, h_in, i_in : tc(8);
19   | out resultdet3 : tc(32))
20 {
21   sig temp1, temp2, temp3 : tc(16);
22   sig temp4, temp5, temp6 : tc(32);
23   sig result : tc(32);
24   use calculate_det (e_in, f_in, h_in, i_in, temp1);
25   use calculate_det2 (d_in, f_in, g_in, i_in, temp2);
26   use calculate_det3 (d_in, e_in, g_in, h_in, temp3);
27   use calculate_multi (a_in, temp1, temp4);
28   use calculate_multi2 (b_in, temp2, temp5);
29   use calculate_multi3 (c_in, temp3, temp6);
30   use calculate_sumsub (temp4, temp5, temp6, resultdet3);
31   always{$display("||a=", a_in, "||b=", b_in, "||c=", c_in, "||d=", d_in, "||e=", e_in, "||f=", f_in, "||g=", g_in, "||h=", h_in, "||i=", i_in,
32 }
33 }
```

شکل ۵ - کد *gezel* برای محاسبه دترمینان ماتریس 3x3

در ادامه تست های محاسبه دترمینان ماتریس 3x3 داریم:

```

det3_v1.fdl x
det3_v1.fdl
38 use calculate_mu
39 use calculate_mu
40 use calculate_mu
41 use calculate_mu
42
43 always{
44     $display("||");
45 }
46 }
47
48 dp test (out_a_in, b_in, c_in, d_in, e_in, f_in, g_in, h_in, i_in)
49
50 always{
51     a_in = 12;
52     b_in = 50;
53     c_in = 71;
54     d_in = 20;
55     e_in = 4;
56     f_in = 17;
57     g_in = 82;
58     h_in = 24;
59     i_in = 15;
60 }
61
62 }

```

```

alireza@ubuntu: ~/Desktop/finalproject
*** INFO: Cycle 5: RTCTL
det3_ctl: det3_ctl.s4 -> det3_ctl.s4
*** INFO: Cycle 5: eval_out ip
*** INFO: Cycle 5: eval
*** INFO: Cycle 5: eval ip
*** INFO: Cycle 5: disp
result=ffffff6e2|cycle=4
alireza@ubuntu:~/Desktop/finalproject$ fdl sim -d det3_v1.fdl 1
> Cycle 1
*** INFO: Cycle 1: RTCTL
*** INFO: Cycle 1: eval_out ip
*** INFO: Cycle 1: eval
*** INFO: Cycle 1: eval ip
*** INFO: Cycle 1: disp
||a=14||b=20||c=f||d=f||e=14||f=20||g=13||h=16||i=57||result=ffffff6e2
alireza@ubuntu:~/Desktop/finalproject$ fdl sim -d det3_v1.fdl 1
> Cycle 1
*** INFO: Cycle 1: RTCTL
*** INFO: Cycle 1: eval_out ip
*** INFO: Cycle 1: eval
*** INFO: Cycle 1: eval ip
*** INFO: Cycle 1: disp
||a=c||b=32||c=47||d=14||e=4||f=11||g=52||h=18||i=f||result=ef84
alireza@ubuntu:~/Desktop/finalproject$

```

شکل ۶- تست کیس شماره ۱ محاسبه دترمینان ماتریس 3x3

در این تست کیس، ورودی های برابر هستند با (12, 50, 71, 20, 4, 17, 82, 24, 15) که خروجی به صورت hex برابر است با 0xef84 که در دستگاه decimal برابر است با 61316 که مقدار درستی می باشد. تعداد کلاک ها : خروجی در آخر کلاک 1 آماده می باشد.


```

38 use calculate_mu
39 use calculate_mu
40 use calculate_mu
41 use calculate_mu
42
43 always{
44     $display("||i
45 }
46 }
47
48 dp test (out a_in, b
49
50 always[
51     a_in = 20;
52     b_in = 32;
53     c_in = 15;
54     d_in = 15;
55     e_in = 20;
56     f_in = 32;
57     g_in = 19;
58     h_in = 22;
59     i_in = 87;
60
61 ]
62 }
63
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
cycle=3||multidet3_1=0/50f0||multidet3_2=0/5720||multidet3_3=0/fffffd12||result=
0
0 50f0 det3.multidet3_1
0 5720 det3.multidet3_2
0 fffffd12 det3.multidet3_3
g_in, "||h=
> Cycle 5
*** INFO: Cycle 5: RTCTL
det3_ctl: det3_ctl.s4 -> det3_ctl.s4
*** INFO: Cycle 5: eval_out ip
*** INFO: Cycle 5: eval
*** INFO: Cycle 5: eval ip
*** INFO: Cycle 5: disp
result=fffff6e2||cycle=4
alireza@ubuntu:~/Desktop/finalproject$ fdlsim -d det3_v1.fdl 1
> Cycle 1
*** INFO: Cycle 1: RTCTL
*** INFO: Cycle 1: eval_out ip
*** INFO: Cycle 1: eval
*** INFO: Cycle 1: eval ip
*** INFO: Cycle 1: disp
||a=14||b=20||c=f||d=f||e=14||f=20||g=13||h=16||i=57||result=fffff6e2
alireza@ubuntu:~/Desktop/finalproject$

```

شکل ۷ - تست کیس شماره ۲ محاسبه دترمیان ماتریس 3x3

در این تست کیس ورودی ها برابر هستند با (20, 32, 15, 15, 20, 32, 19, 22, 87) که خروجی به صورت hex برابر است با 0xFFFFF6E2 که در دستگاه سیستم علامت دار و تبدیل به decimal برابر است با -2334 که مقدار درستی است.

تعداد کلاک ها : خروجی در آخر کلاک 1 آماده می باشد.

• دترمینان ماتریس 3x3 با FSM :

برای محاسبه سخت افزاری دترمینان ماتریس 3x3 با fsm داریم:

```
det3_v2.fdl
1  dp det3 (in a_in , b_in, c_in, d_in, e_in, f_in, g_in, h_in, i_in : tc(8); out result : tc(32))
2  {
3      reg a, b, c, d, e, f, g, h, i : tc(8);
4      reg multidet2_1, multidet2_2, multidet2_3, multidet2_4, multidet2_5, multidet2_6 : tc(16);
5      reg det21, det22, det23 : tc(16);
6      reg multidet3_1, multidet3_2, multidet3_3 : tc(32);
7      reg dif1, dif2, dif3 : tc(32);
8
9  >  sfg inint { ...
29  }
30  >  sfg multiplicationdet2 { ...
45  }
46  >  sfg differencedet2 { ...
55  }
56  >  sfg multiplicationdet3 { ...
65  }
66  >  sfg output { ...
70  }
71  >  sfg outidle { ...
73  }
74  }
75
76  fsm det3_ctl (det3)[]
77  initial s0;
78  state s1, s2, s3, s4;
79
80  @s0 (inint, outidle) -> s1;
81
82  @s1 (multiplicationdet2, outidle) -> s2;
83
84  @s2 (differencedet2, outidle) -> s3;
85
86  @s3 (multiplicationdet3, outidle) -> s4;
87
88  @s4 (output) -> s4;
89  }
```

شکل ۸- کد gezel محاسبه دترمینان ماتریس 3x3 با fsm

در ادامه تست کیس های مربوط به محاسبه دترمینان ماتریس 3×3 را خواهیم داشت:

```
@s0 (inint, outint)
@s1 (multiplication)
@s2 (difference)
@s3 (multiplication)
@s4 (output) ->
}

dp test_det3 (out a,
sfg run {
    a = 12;
    b = 50;
    c = 71;
    d = 20;
    e = 4;
    f = 17;
    g = 82;
    h = 24;
    i = 15;
}

*** INFO: Cycle 4: RTCTL
det3_ctl: det3_ctl.s3 -> det3_ctl.s4
*** INFO: Cycle 4: eval_out ip
*** INFO: Cycle 4: eval
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
cycle=3||multidet3_1=0/ffffefb0||multidet3_2=0/ffff2a54||multidet3_3=0/2a28||result=0
0 fea4 det3.det21
0 fbba det3.det22
0 98 det3.det23
> Cycle 5
*** INFO: Cycle 5: RTCTL
det3_ctl: det3_ctl.s4 -> det3_ctl.s4
*** INFO: Cycle 5: eval_out ip
*** INFO: Cycle 5: eval
*** INFO: Cycle 5: eval ip
*** INFO: Cycle 5: disp
result=ef84||cycle=4
alireza@ubuntu:~/Desktop/finalproject$
```

شکل ۹ - تست کیس شماره ۱ محاسبه دترمینان ماتریس 3×3 با fsm

در این تست کیس، ورودی های برابر هستند با (12, 50, 71, 20, 4, 17, 82, 24, 15) که خروجی به صورت hex برابر است با 0xef84 که در دستگاه decimal برابر است با 61316 که مقدار درستی می باشد. تعداد کلاک ها : خروجی در آخر کلاک 5 آماده می باشد.

```

@ s0 (inint, out)
@ s1 (multiplicat
@ s2 (differencec
@ s3 (multiplicat
@ s4 (output) ->
}

dp test_det3 (out a,
sfg run
a = 20;
b = 32;
c = 15;
d = 15;
e = 20;
f = 32;
g = 19;
h = 22;
i = 87;
}

0 40c det3.det21
0 2b9 det3.det22
0 ffce det3.det23

> Cycle 4
*** INFO: Cycle 4: RTCTL
det3_ctl: det3_ctl.s3 -> det3_ctl.s4
*** INFO: Cycle 4: eval_out ip
*** INFO: Cycle 4: eval
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
cycle=3||multidet3_1=0/50f0||multidet3_2=0/5720||multidet3_3=0/ffffffd12||result=
0
50f0 det3.multidet3_1
5720 det3.multidet3_2
ffffffd12 det3.multidet3_3

> Cycle 5
*** INFO: Cycle 5: RTCTL
det3_ctl: det3_ctl.s4 -> det3_ctl.s4
*** INFO: Cycle 5: eval_out ip
*** INFO: Cycle 5: eval
*** INFO: Cycle 5: eval ip
*** INFO: Cycle 5: disp
result=ffffff6e2||cycle=4
alireza@ubuntu:~/Desktop/finalproject$

```

شکل ۱۰ - تست کیس شماره ۲ محاسبه دترمینان ماتریس 3x3 با fsm

در این تست کیس ورودی ها برابر هستند با (20, 32, 15, 15, 20, 32, 19, 22, 87) که خروجی به صورت hex برابر است با 0xFFFFF6e2 که در دستگاه سیستم علامت دار و تبدیل به decimal برابر است با -2334 که مقدار درستی است.

تعداد کلاک ها : خروجی در آخر کلاک 5 آماده می باشد.

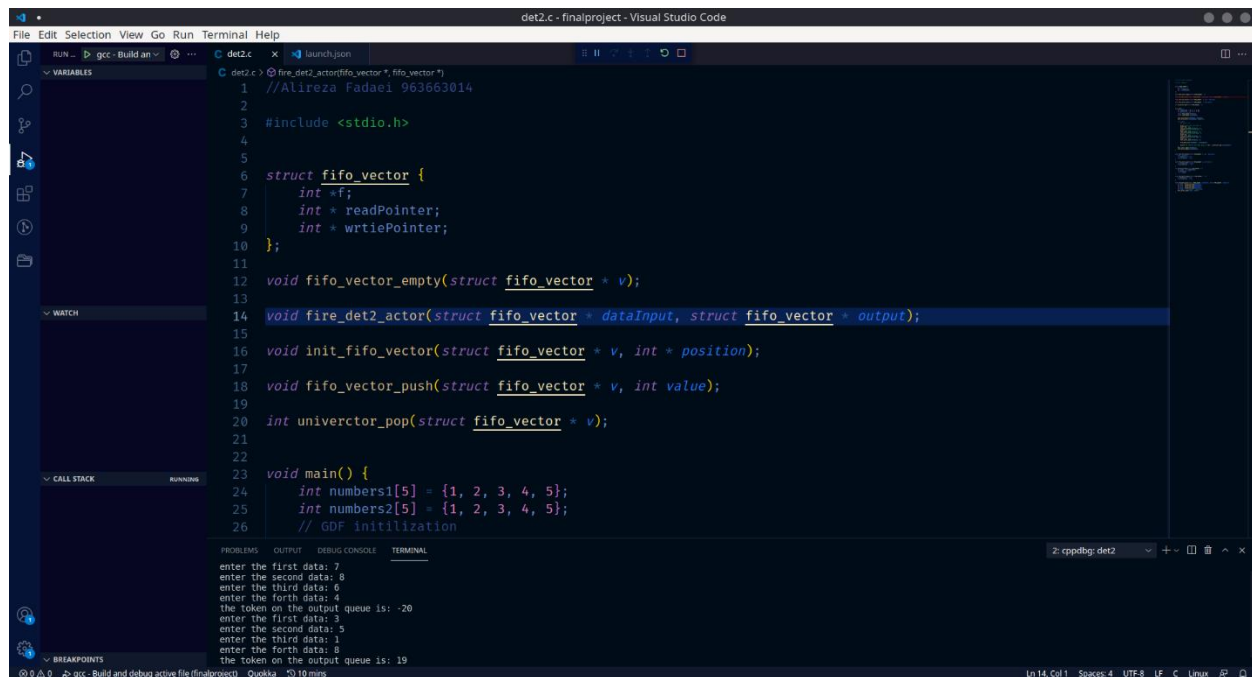
طراحی نرم افزاری

در طراحی نرم افزاری برای محاسبه دترمینان 2×2 به صورت نرم افزاری داریم:

```
det2.c > ...
1  #include <stdio.h>
2
3  struct fifo_vector {
4      int *f;
5      int * readPointer;
6      int * wrtiePointer;
7  };
8
9  void fifo_vector_empty(struct fifo_vector * v);
10
11 void fire_det2_actor(struct fifo_vector * dataInput, struct fifo_vector * output);
12
13 void init_fifo_vector(struct fifo_vector * v, int * position);
14
15 void fifo_vector_push(struct fifo_vector * v, int value);
16
17 int univector_pop(struct fifo_vector * v);
18
19
20 > void main() {...
53
54
55 void init_fifo_vector(struct fifo_vector *v, int * position) {
```

شکل ۱۱ - کد نرم افزاری محاسبه دترمینان ماتریس 2×2

در ادامه تست کیس برنامه را خواهیم داشت، تست کیس ها مشابه، تست کیس های سخت افزاری می باشند.



```
det2.c - finalproject - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN < g++ -std=c++11 -g -c det2.c -o det2.o
det2.c x bash:python
det2.c > fire_det2_actor(fifo_vector *, fifo_vector *)
1 //Alireza Fadaei 963663014
2
3 #include <stdio.h>
4
5
6 struct fifo_vector {
7     int *f;
8     int * readPointer;
9     int * writePointer;
10 };
11
12 void fifo_vector_empty(struct fifo_vector * v);
13
14 void fire_det2_actor(struct fifo_vector * dataInput, struct fifo_vector * output);
15
16 void init_fifo_vector(struct fifo_vector * v, int * position);
17
18 void fifo_vector_push(struct fifo_vector * v, int value);
19
20 int univector_pop(struct fifo_vector * v);
21
22
23 void main() {
24     int numbers1[5] = {1, 2, 3, 4, 5};
25     int numbers2[5] = {1, 2, 3, 4, 5};
26     // GDF initialization
27
28     enter the first data: 7
29     enter the second data: 8
30     enter the third data: 6
31     enter the forth data: 4
32     the token on the output queue is: -20
33     enter the first data: 3
34     enter the second data: 5
35     enter the third data: 1
36     enter the forth data: 8
37     the token on the output queue is: 19
```

شکل ۱۲ - تست کیس شماره ۱ محاسبه نرم افزاری ماتریس 2x2