گزارش تمرین سری 4 همطراحی سخت افزار–نرم افزار

عرفان رفیعی اسکویی – 98243027

فربد فولادی – 98243045

مقدمه :

هدف در این تمرین پیاده سازی یک سیستم codesign با استفاده از محیط GEZEL و استفاده از پردازنده ARM است.

الگوریتم داده شده CORDIC است که با کد C زده شده و ما باید بخشی از کد را که خطوط داده شده هستند به صورت سخت افزاری پیاده کنیم.

بخش نرم افزاری(کد C) :

ابتدا متغییر هایی که قرار است در بخش سخت افزاری استفاده شوند را با آدرس هرکدام تعریف میکنیم :

```
5
6   volatile unsigned int *starter = (int *) 0x80000000;
7   volatile unsigned int *theta_input = (int *) 0x80000004;
8   volatile unsigned int *iteration_input = (int *) 0x80000008;
9   volatile unsigned int *shift_input = (int *) 0x8000000C;
10  volatile unsigned int *cos_output = (int *) 0x80000010;
11  volatile unsigned int *sin_output = (int *) 0x8000001C;
12
13
```

حال در مرحله بعدی ما نیازمند یک connection یا به عبارتی handshake بین سخت افزار و نرم افزار هستیم که متغییر هارا به HW پاس بدهیم و مقادیر تغییر یافته را در SW بگیریم.

این کار را با تابع زیر انجام میدهیم:

```
void HW_Connection(int theta, int iterations,int shift, int *sin , int *cos) {
```

در ادامه ما یک مقدار starter برای شروع کار HW داریم و با یک حلقه ی while تا زمانی که sin و cos مقدار بگیرند صبر میکنیم، و بعد از آن مقدار starter صفر شده و به ادامه ی کار در نرم افزار میپردازیم :

```
    *iteration_input = iterations;
    *theta_input = theta;
    *shift_input = shift;
    *starter = 1;


    while (*sin_output == 0 && *cos_output == 0);



    *sin = *sin_output;
    *cos = *cos_output;

    *starter = 0;
}
```

ادامه کد مانند کد اصلی است و مقادیری برای theta و iteration میدهیم و خطوط 77 تا 106 را حذف میکنیم و در محیط GEZEL پیاده سازی میکنیم، و برای ارسال مقادیر تابع مورد نظر را صدا میزنیم :

```
HW_Connection(theta, iterations, shift, &sin, &cos);
```

بخش سخت افزاری (ARM) :

ابتدا اولین ipblock برای پردازنده را تعریف میکنیم با نام armcore و نام تایپ و فایلی که قرار است خوانده شود را میدهیم:

```
ipblock armcore
{
        iptype "armsystem";
        ipparm "exec = cordic";
}
```

حال در ادامه مقادیر ورودی را که 5 تای آنها input و 2 تای آنها output هستند را با ادرسی که در کد C داشتیم تعریف میکنیم :

```
ipblock starter(out data : ns(1))
{
        iptype "armsystemsource";
        ipparm "core = armcore";
        ipparm "address = 0x80000000";
}

ipblock theta_input(out data : tc(32))
{
    iptype "armsystemsource";
    ipparm "core = armcore";
    ipparm "address=0x80000004";
}

ipblock iteration_input(out data : tc(32))
{
    iptype "armsystemsource";
    ipparm "core = armcore";
    ipparm "address = 0x80000008";
}

ipblock shift_input(out data : tc(32))
{
    iptype "armsystemsink";
    ipparm "core = armcore";
    ipparm "address = 0x8000000C";
}

ipblock cos_output(in data : tc(32))
{
    iptype "armsystemsink";
    ipparm "core = armcore";
    ipparm "address = 0x80000010";
}

ipblock sin_output(in data : tc(32))
{
    iptype "armsystemsink";
    ipparm "core = armcore";
    ipparm "address = 0x80000014";
}
```

در ادامه DataPath مورد نظر با ورودی و خروجی و lookup table خود را تعریف میکنیم و سپس register و سیگنال های خود را تعریف میکنیم:

```
dp Cordic ( in in_start : ns(1);
                        in in_theta : tc(32);
                        in in_itr : tc(32);
                        in in_shift : tc(32);
                        out out_data_x : tc(32);
                        out out_data_y : tc(32))
                        {

        lookup atantable : tc(32) = {
                                                                0x4000,
                                                                0x25C8,
                                                                0x13F6,
                                                                0x0A22,
                                                                0x0516,
                                                                0x028B,
                                                                0x0145,
                                                                0x00A2,
                                                                0x0051,
                                                                0x0029,
                                                                0x0014,
                                                                0x000A,
                                                                0x0005,
                                                                0x0003,
                                                                0x0002,
                                                                0x0001
                                                                };

        reg X, Y, theta, shift, angle, step, iteration, tmp_x, tmp_y: tc(32);
        reg start : ns(1);
        sig cmp, cmp2, cmp3 : ns(1);
```

در ادامه state machine های خود را با sfg ها تعریق میکنیم که هرکدام بخشی از دستورات را که خودمان تعریف کرده ایم را انجام میدهند :

```
always
{
        start = in_start;
        out_data_x = tmp_x;
        out_data_y = tmp_y;
}

sfg shift_gt
{
        Y = X;
        X = -1 * Y;


}
```

```
        sfg shift_lt
        {
                Y = -1 * X;
                X = Y;
        }

        sfg shift_eq
        {
                Y = Y;
                X = X;
        }

        sfg get_data
        {
                shift = in_shift;
                theta = in_theta;
                iteration = in_itr;
                X = 0x4DBA;
                Y = 0;
                step = 0;
                angle = 0;
        }

        sfg loop
        {
                cmp = (theta < angle);
                X = cmp ? X + (Y >> step): X - (Y >> step);
                Y = cmp ?  Y - (X >> step): Y + (X >> step);
                angle = cmp ? angle - atantable(step) : angle + atantable(step);
                step = step + 1;
        }

        sfg output
        {
                tmp_x = X;
                tmp_y = Y;
                $display("Hardware : sin = ", out_data_y, ", cos = ", out_data_x);

        }

        sfg negative
        {
                X = -1 * X;
        }

        sfg skip
        {
                tmp_x = 0;
                tmp_y = 0;

        }
        sfg do_nothing
        {

        }
}
```

حال میاییم در fsm controller خود با استفاده از شروط استیت هارا بهم وصل میکنیم :

```
fsm controller(Cordic)
{
        initial s0;
    state s1, s2, s3, s4;

    @s0 if (start == 1) then (get_data) -> s1;
        else (do_nothing) -> s0;

    @s1 if (step < iteration) then (loop) -> s1;
        else (do_nothing) -> s2;

    @s2 if (X < 0) then (negative) -> s3;
        else (do_nothing) -> s3;

    @s3 if (shift > 0) then (shift_gt) -> s4;
        else if (shift < 0) then (shift_lt) -> s4;
        else (shift_eq) -> s4;

    @s4 if (start == 1) then (output) -> s4;
        else (do_nothing) -> s0;
 }
```

در نهایت نیز یک test bench مینویسیم و کار تمام میشود :

```
dp TestBench
{
        sig in_start: ns(1);
        sig in_theta, in_itr, in_shift, out_data_x, out_data_y : tc(32);

        use Cordic(in_start, in_theta, in_itr, in_shift, out_data_x, out_data_y);

        use armcore;
        use starter(in_start);
        use theta_input(in_theta);
        use iteration_input(in_itr);
        use shift_input(in_shift);
        use cos_output(out_data_x);
        use sin_output(out_data_y);
}

system S
{
        TestBench;
}
```

Plain Text ▾ Tab Width: 8 ▾ Ln 183, Col

بخش شبیه سازی :

در این بخش خروجی را برای 3 مقدار theta به اندازه های 30 و 45 و 110 درجه هرکدام با iteration 15 نشان میدهیم.

1- 30 درجه :

```
rfosk@ubuntu:~/Codesign/HW4$ gplatform cordic.fdl 15
core arm
armsystem: loading executable [cordic]
Warning: ipblock shift_input terminal mismatch for output data (0)
armsystemsink: set address 2147483660
armsystemsink: set address 2147483664
armsystemsink: set address 2147483668
Hardware : sin = 0, cos = 0
Hardware : sin = 3ff2, cos = 6ee2
Hardware : sin = 3ff2, cos = 6ee2
Software : sin = 0, cos = 6ee2
Total Cycles: 18359
rfosk@ubuntu:~/Codesign/HW4$
```

2- 45 درجه :

```
rfosk@ubuntu:~/Codesign/HW4$ gplatform cordic.fdl 15
core arm
armsystem: loading executable [cordic]
Warning: ipblock shift_input terminal mismatch for output data (0)
armsystemsink: set address 2147483660
armsystemsink: set address 2147483664
armsystemsink: set address 2147483668
Hardware : sin = 0, cos = 0
Hardware : sin = 5a82, cos = 5a83
Hardware : sin = 5a82, cos = 5a83
Software : sin = 0, cos = 5a83
Total Cycles: 18386
rfosk@ubuntu:~/Codesign/HW4$
```

3- 110 درجه :

```
rfosk@ubuntu:~/Codesign/HW4$ gplatform cordic.fdl 15
core arm
armsystem: loading executable [cordic]
Warning: ipblock shift_input terminal mismatch for output data (0)
armsystemsink: set address 2147483660
armsystemsink: set address 2147483664
armsystemsink: set address 2147483668
Hardware : sin = 0, cos = 0
Hardware : sin = 2bc3, cos = 7846
Hardware : sin = 2bc3, cos = 7846
Software : sin = 0, cos = ffff87ba
Total Cycles: 18474
rfosk@ubuntu:~/Codesign/HW4$
```