📝 گزارش آزمایش ۵

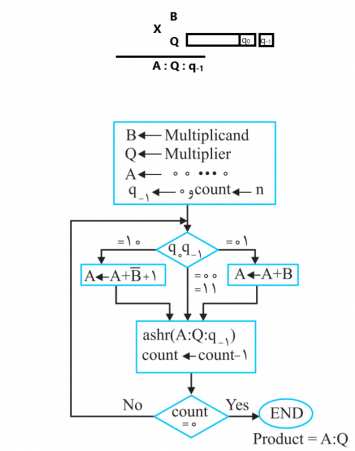
آزمایشگاه طراحی سیستم‌های دیجیتال

Alireza Habibzadeh `99109393`

May 3, 2022

الگوریتم Booth

در این الگوریتم مطابق فلوچارت روبه‌رو عمل می‌کنیم. هر بار به دو بیت ابتدایی نگاه می‌کنیم، اگر هر دو 0 یا هر دو 1 بودند یعنی هنوز در یک دنباله‌ی 0 یا 1 هستیم پس فعلا کاری نمی‌کنیم و شیفت را ادامه می‌دهیم. اما اگر به 01 یا 10 رسیدیم یعنی دنباله تغییر کرده و باید `B` را کم یا زیاد کنیم (مطابق فلوچارت).



booth algorithm flowchart

ماژول‌های کمکی

برای پیاده‌سازی الگوریتم از دو ماژول کمکی استفاده می‌کنیم. این ماژول‌ها مکان اولین پترن 01 و 10 از سمت راست را برای ما در عدد ۴ بیتی پیدا می‌کنند. برای سادگی این ماژول‌ها را با `assign` پیاده می‌کنیم. کد این ماژول‌ها در ادامه آمده است:

```
module next_zero(
    input [3:0] A,
    output [2:0] address
    );
        assign address[0] = ~A[1] & A[0] | ~A[3] & A[2] & A[1] & A[0];
    assign address[1] = ~A[2] & A[1] & A[0] | ~A[3] & A[2] & A[1] & A[0];
    assign address[2] = A[3] & A[2] & A[1] & A[0]; // no zero is found in the number (0)1111
endmodule
```

```
module next_one( // find the first 10 pattern in a 4 bit input
    input [3:0] A,
    output [1:0] address
    );
    assign address[0] = A[1] & ~A[0] | A[3] & ~A[2] & ~A[1] & ~A[0];
    assign address[1] = A[2] & ~A[1] & ~A[0] | A[3] & ~A[2] & ~A[1] & ~A[0];
endmodule

```

ماژول اصلی

ماژول اصلی را با کمک دو ماژول کمکی پیاده می‌کنیم. می‌توان برای این ماژول سیگنال `ready` در نظر گرفت اما چون در صورت سوال به چنین چیزی اشاره نشده فرض می‌کنیم کاربر ماژول به تعداد کافی کلاک می‌زند و صبر می‌کند تا نتیجه آماده شود.

```
module multiplier(
    input clk,
    input rst,
    input [3:0] B,
    input [3:0] Q,
    output reg [7:0] AQ
    );

    reg [3:0] shift;
    reg [7:0] Q_long, B_long;
    reg not_first;
    wire first_one = not_first & B_long[0];
    wire [2:0] next_zero_add;
    wire [1:0] next_one_add;

    next_zero nz (B_long, next_zero_add);
```

```
    next_one no (B_long, next_one_add);

    always @ (negedge rst) begin
        shift <= 4'b0;
        Q_long <= {4'b0, Q};
        B_long <= {4'b0, B};
        AQ <= 8'b0;
        not_first <= 1'b0;
    end

    always @ (posedge clk) begin
        if (B_long > 0) begin
            not_first <= 1'b1;
            if (first_one) begin // 01
                AQ <= AQ + (Q_long << (next_zero_add + shif
t));
                B_long <= B_long >> next_zero_add;
                shift <= shift + next_zero_add;
            end
            else begin // 10
                AQ <= AQ - (Q_long << (next_one_add + shift));
                B_long <= B_long >> next_one_add;
                shift <= shift + next_one_add;
            end
        end
    end
endmodule
```

تست

برای بررسی صحت عملکرد مدار از test bench زیر استفاده شده است. البته قسمت‌هایی از آن به صورت خودکار توسط نرم‌افزار تولید می‌شود و کافی است داخل بلاک `initial` را تغییر دهیم.

در این تست تعدادی عدد برای ضرب شدن در ماژول قرار گرفته‌اند. سعی شده اعداد متنوع باشند و حالات خاص را پوشش دهند. همچنین اعداد بزرگ ته بازه (9×9) نیز امتحان شده‌اند.

برای سیگنال ریست، باید دقت کنیم که با هر بار ورود عدد جدید باید ماژول را ریست کرد. همچنین ابتدای کار باید یک بار ماژول را ریست کنیم. و دقت کنیم که ریست آن active low است و در ابتدا در حالت ریست نیست و باید یک لبه‌ی پایین رونده‌ی ریست بزنیم.

```
`timescale 1ns / 1ps

module multiplier_tb;
    // Inputs
    reg clk;
    reg rst;
    reg [3:0] B;
    reg [3:0] Q;

    // Outputs
    wire [7:0] AQ;

    // Instantiate the Unit Under Test (UUT)
    multiplier uut (
            .clk(clk),
            .rst(rst),
            .B(B),
            .Q(Q),
            .AQ(AQ)
    );

    initial begin
            // Initialize Inputs
            clk = 0;
            rst = 1;
            B = 0;
```

```
27              Q = 0;
28
29              // Wait 100 ns for global reset to finish
30              #100;
31
32              // Add stimulus here
33              rst = 0;
34              $monitor("%d * %d = %d", B, Q, AQ);
35
36              B = 2;
37              Q = 2;
38              #5
39              rst = 1; #5 rst = 0;
40              clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5 cl
k = 0; #5 clk = 1; #5
41  clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5
42
43              B = 4;
44              Q = 3;
45              #5;
46              rst = 1; #5 rst = 0;
47              clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5 cl
k = 0; #5 clk = 1; #5
48  clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5
49
50              B = 7;
51              Q = 4;
52              #5
53              rst = 1; #5 rst = 0;
54              clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5 cl
k = 0; #5 clk = 1; #5
```

```
  clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5


            B = 8;
            Q = 0;
            #5
            rst = 1; #5 rst = 0;
            clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5
  clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5


            B = 0;
            Q = 9;
            #5
            rst = 1; #5 rst = 0;
            clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5
  clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5


            B = 9;
            Q = 9;
            #5
            rst = 1; #5 rst = 0;
            clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1; #5
  clk = 0; #5 clk = 1; #5 clk = 0; #5 clk = 1;
            $stop();
    end
endmodule
```

شبیه‌سازی

با اجرای شبیه‌سازی دستور `$monitor` با هر تغییر خروجی آن را چاپ می‌کند. نتیجه این است که مرحله به مرحله می‌توانیم پیشروی زیبای ضرب Booth را ببینیم. و میبینیم که در انتها حاصل به مقدار واقعی ضرب‌ها رسیده و پس از آن رجیستر `B` همه‌ی شیفت‌هایش را خورده و صفر شده است و از آنجا به بعد دیگر تغییری در حاصل نداریم. خروجی کامل کنسول در زیر آورده شده است:

```
ISim P.20131013 (signature 0x7708f090)
This is a Full version of ISim.
WARNING: File "C:/hw5-99109393/multiplier.v" Line 29.  For instance uut/no/, width 4 of formal port A is not equal to width 8 of actual variable B_long.
WARNING: File "C:/hw5-99109393/multiplier.v" Line 29.  For instance uut/nz/, width 4 of formal port A is not equal to width 8 of actual variable B_long.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
 2 *  2 =   0
 2 *  2 = 252
 2 *  2 =   4
 4 *  3 =   4
 4 *  3 =   0
 4 *  3 = 244
 4 *  3 =  12
 7 *  4 =  12
 7 *  4 =   0
 7 *  4 = 252
 7 *  4 =  28
 8 *  0 =  28
 8 *  0 =   0
 0 *  9 =   0
 9 *  9 =   0
 9 *  9 = 247
 9 *  9 =   9
 9 *  9 = 193
```

```
 9 *  9 =  81
Stopped at time : 455 ns :  in File "C:/hw5-99109393/multiplier_tb.v" Line 100
ISim>
```

پس می‌توان نتیجه گرفت که مدار ما برای این تست‌ها درست کار می‌کند.

آن Warningهای تولید شده به دلیل آن است که در قسمتی از یک رجیستر با طول کوتاه به عنوان رجیستری با طول بیشتر یا برعکس استفاده کردم. در این موارد نرم‌افزار به طور پیشفرض اضافات را صفر می‌گذارد یا در مواردی که بلندتر است از چپ قطع می‌کند. این استفاده فکر شده بوده و لذا می‌توانیم Warning را نادیده بگیریم.

Waveform

در ادامه Waveform مربوط به این تست آمده است.


0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
AQ[7:0]
clk
rst
B[3:0]
Q[3:0]
X
0
4
12
28
0
9
0
2
4
7
8
0
9
0
2
3
4
0
9