



گزارش کار آزمایشگاه DSD

آزمایش شماره 5



26 فروردین 1400

عرشیا اخوان

محمدحسین عبدی

علیرضا ایلامی

شماره آزمایش: 5	موضوع: طراحی ضرب کننده	تاریخ آزمایش: 26 فروردین 1400
عرشیا اخوان 97110422	محمدحسین عبدی 97110285	علیرضا ایلامی 97101286

مقدمه:

در این آزمایش ما باید ضرب به روش booth را پیاده سازی کنیم.

شرح آزمایش:

ابتدا یک سیگنال start داریم که باید حتما 1 شود تا الگوریتم شروع شود
تعدادی رجیستر داریم به نام های a, b, x که قرار است رجیستر های a, b در یکدیگر ضرب شوند و در b نوشته شوند. X هم رجیستر کمکی است. و خروجی نهایی حاصل ضرب xb است.
یک کانتر n داریم که تعدد مراحل ضرب را نشان میدهد
یک سیگنال end flag داریم که وقتی 1 میشود یعنی خروجی ما آماده است.
یک رجیستر $1-b$ هم هست که موقع شیفت دادن xb بیرون انداخته میشود. (بیت سمت راست عدد xb)

در ابتدا اعداد را لود میکنیم در a, b
به جای الگوریتم ضرب عادی، ما در روش بوث می آییم در b ، ابتدا index اولین بیت یک، ایندکس آن را i میگیریم. تا وقتی بیت های بعدی 1 هستند ادامه میدهیم و کاری نمیکنیم، تا زمانی که دوباره به یک بیت 0 برسیم و اندیس آن نیز j است.

چیزی که برای ما مهم میشود، دو به توان j منهای دو به توان i است. یعنی

$$2^j - 2^i$$

علت آن هم این است که مثلا 111 که برابر با 7 است، برابر با دو به توان 3 منهای دو به توان 0 است.
بنابراین الگوریتم بوث تعداد operation های ما را به شدت کاهش میدهد و نتیجه را نیز کاملا درست محاسبه میکند.

کار ما بطور خلاصه این است:

بیت های $0b$ و $1-b$ را چک میکنیم.

اگر که b $1-b0$ برابر با 00 یا 11 بود:

یعنی ما وسط دنباله متوالی در الگوریتم بوث هستیم و نیازی نیست کاری انجام دهیم. و به راحتی فقط شیفت میدهیم و فقط از کانتر یک واحد کم میکنیم.

اگر برابر با 10 بود: باید شروع کنیم به کم کردن دو عدد و سپس یک واحد از کانتر کم میکنیم

اگر برابر با 01 بود: باید شروع کنیم به جمع کردن دو عدد و مثل قبل کانتر را یک واحد کم میکنیم

واضح است که کل الگوریتم زمانی خاتمه پیدا میکند که کانتر ما برابر با 0 شده باشد

تعدادی مثال به روش booth آورده شده است:

Signed Binary Multiplication

The acting chary for the Booth's signed multiplication scheme is shown.

Example 1: Follow Booth's signed multiplication algorithm step by step to multiply 4-bit numbers 7 and -5. (A=0111)

X	B	b ₀	Counter	Operation
0000	1011	1	4	Subtract
1001	1011	1	4	Shift
1100	1101	1	3	Shift
1110	0110	0	2	Add
0101	0110	0	2	Shift
0010	1011	1	1	Subtract
1011	1011	1	1	Shift
1101	1101	1	0	End.

-00100011 -35 = 7 x (-5)

Signed Binary Multiplication

The acting chart for the Booth's signed multiplication scheme is shown.

Example 3: Follow Booth's signed multiplication algorithm step by step to multiply 4-bit numbers -7 and -5. ($A=1001$)

X	B	b_0	Counter	Operation
0000	1011	1	4	Subtract
0111	1011	1	4	Shift
0011	1101	1	3	Shift
0001	1110	0	2	Add
1010	1110	0	2	Shift
1101	0111	1	1	Subtract
0100	0110	0	1	Shift
0010	0011	1	0	End.


$$35 = -7 \times (-5)$$

Signed Binary Multiplication

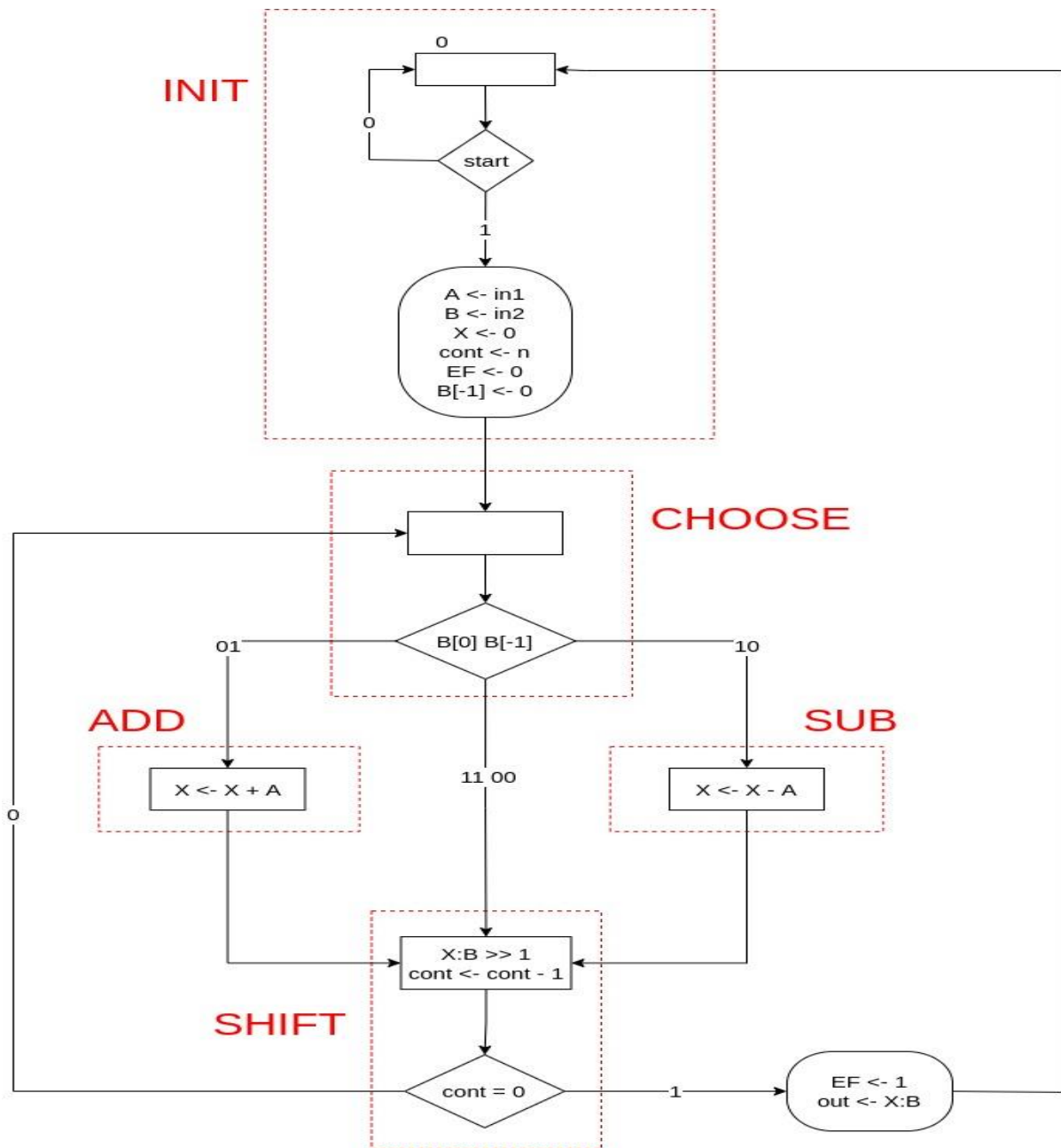
The acting chary for the Booth's signed multiplication scheme is shown.

Example 2: Follow Booth's signed multiplication algorithm step by step to multiply 4-bit numbers -7 and 6. ($A=1001$)

X	B	b_0	Counter	Operation
0000	0110		4	Shift
0000	0011		3	Subtract
0111	0011		3	Shift
0011	1001		2	Shift
0001	1100		1	Add
1010	1100		1	Shift
1101	0110		0	End.


-00101010 **-42 = -7 x 6**

تحليل نمودار ASM chart



در این نمودار ما 5 تا state داریم:

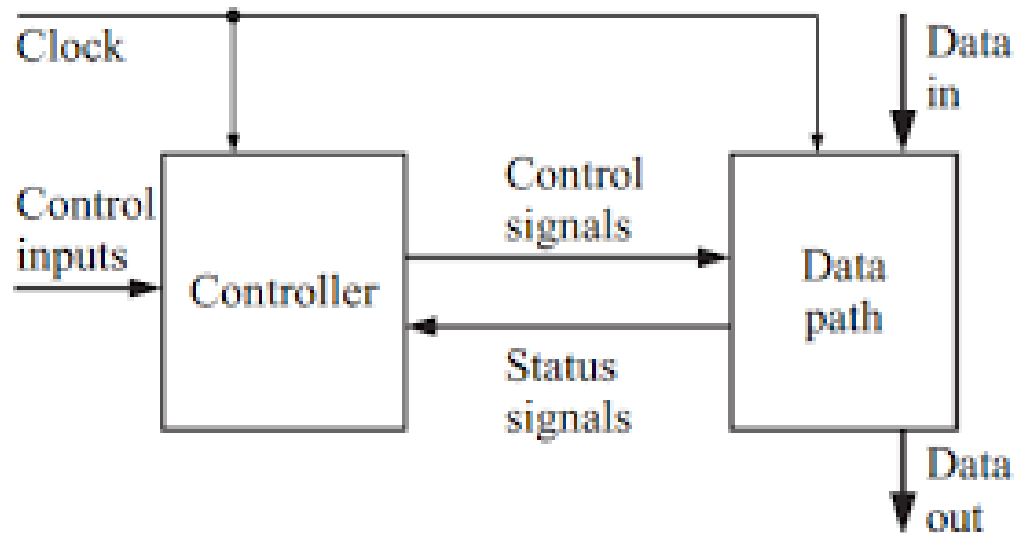
Init, Choose, Sub, Add, Shift

در init state که اگر سیگنال start = 0 باشد همینطور کلاک زده میشود تا وقتی که start = 1 شود و سیگنال های اولیه ست گردند و ما به استیت بعدی برویم.

در choose state ما یک باکس خالی داریم. علت آن در زیر شرح داده شده است:

میدانیم که تمام دستورات در یک **asm block** که با مستطیل نشان داده میشود، بصورت کاملاً موازی و همزمان انجام میشوند. و همچنین هر کدام از این بلاک‌ها نمایانگر یک کلاک واحد زمانی هستند. ضمن اینکه میدانیم تمام تغییرات ما در یک **asm block** در بلاک بعدی این تغییرات ما اعمال میشوند. بنابراین آن بلاک خالی باعث میشود شروط **b**, **b0-1** بعد از اجرای استیت های **ADD** و **SUB** و **SHIFT** به درستی مقدار دهی شوند.

تحلیل کد:



شمایی از ساختار کلی ماژول:

داده‌ها وارد دیتاپت میشوند، دیتاپت **signal** های وضعیت فعلی را به واحد **Controller** که در اینجا **Control Unit** است میدهد، و واحد کنترلر تصمیم میگیرد که **next state** چیست و آن را در **current state** بریزد. و این نتیجه را به دیتاپت اطلاع میدهد. هر دوی این بلاک‌ها با کلاک کار میکنند.

Mul booth

```
module MUL_BOOTH(in1,
                 in2,
                 out,
                 out_r,
                 clk,
                 start,
                 rstn);
    parameter BIT_LEN = 4;

    input [BIT_LEN-1:0] in1, in2;
    input clk, start, rstn;
    output [2*BIT_LEN-1:0] out;
    output out_r;

    wire [2:0] status, control;

    CU cu (
        .status(status),
        .control(control),
        .start(start),
        .rstn(rstn),
        .clk(clk)
    );

    DP #(.BIT_LEN(BIT_LEN)) dp(
        .control(control),
        .IN1(in1),
        .IN2(in2),
        .OUT(out),
        .status(status),
        .EF(out_r),
        .rstn(rstn),
        .clk(clk)
    );

endmodule //mul_booth
```

این ماژول صرفاً کنترل یونیت و دیتاپت را وصل میکند و حکم واسط را دارد. اصل کار در دو ماژول دیگر است.

Control unit

```
define ST_INIT      3'b000
`define ST_ADD      3'b001
`define ST_SUB      3'b010
`define ST_SHIFT    3'b011
`define ST_CHOOSE   3'b100

module CU(clk,
          status,
          control,
          rstn,
          start);

    input [2:0] status;
    input clk, rstn, start;
    output [2:0] control;

    wire B0, B1, FIN;
    reg [2:0] state;
    reg [2:0] next_state;

    assign B0      = status[2];
    assign B1      = status[1];
    assign FIN     = status[0];
    assign control = state;

    always @(posedge clk or negedge rstn) begin
        if (!rstn)
            state <= `ST_INIT;
        else
            state <= next_state;
    end
end
```

```

always @(state or B0 or B0 or FIN or start) begin
    case (state)
        `ST_INIT:
        begin
            if (start) begin
                next_state <= `ST_CHOOSE;
            end
        end
        `ST_ADD:
        begin
            next_state <= `ST_SHIFT;
        end
        `ST_SUB:
        begin
            next_state <= `ST_SHIFT;
        end
        `ST_SHIFT:
        begin
            if (FIN)
                next_state <= `ST_INIT;
            else
                next_state <= `ST_CHOOSE;
            end
        end
        `ST_CHOOSE:
        begin
            if (B0 && !B0)
                next_state <= `ST_SUB;
            else if (!B0 && B0)
                next_state <= `ST_ADD;
            else
                next_state <= `ST_SHIFT;
            end
        end
    endcase
end
endmodule

```

هر بار **next state** مشخص میشود و در استیت فعلی ریخته میشود.
نکست استیت از روی سیگنال های کنترلی که دیتاپت می فرستد آپدیت میشوند.

Datapath

```

`define ST_INIT      3'b000
`define ST_ADD       3'b001
`define ST_SUB       3'b010
`define ST_SHIFT     3'b011
`define ST_CHOOSE    3'b100

module DP (control,
           IN1,
           IN2,
           OUT,
           status,
           EF,
           rstn,
           clk);
    parameter BIT_LEN = 4;

    input [2:0]control;
    input [BIT_LEN-1:0]IN1;
    input [BIT_LEN-1:0]IN2;
    input clk, rstn;

    output [2*BIT_LEN-1:0]OUT;
    output [2:0]status;
    output EF;
    output reg B0;

    reg [BIT_LEN-1:0] A, B, X;
    reg[$clog2(BIT_LEN):0] cont;

    assign OUT      = (EF)? {X,B}: OUT;
    assign status   = {B[0], B0,EF};
    assign EF       = (cont == 0);

    always @(posedge clk, negedge rstn) begin
        if (!rstn) begin
            A      <= IN1;

```

```

        B    <= IN2;
        X    <= 0;
        cont <= BIT_LEN;
        B0   <= 0;
    end
    else begin
        case (control)
            `ST_INIT:
            begin
                cont <= BIT_LEN;
                A    <= IN1;
                B    <= IN2;
                X    <= 0;
                B0   <= 0;
            end
            `ST_ADD:
            begin
                X <= X + A;
            end
            `ST_SUB:
            begin
                X <= X - A;
            end
            `ST_SHIFT:
            begin
                B0    <= B[0];
                {X,B} <= {X[BIT_LEN-1], X, B[BIT_LEN-1:1]};
                cont <= cont - 1;
            end
            `ST_CHOOSE:
            begin
                //do nothing
            end
        endcase
    end
end
end

```

استیت ها را به دیتاپت می‌دهیم و دیتاپت متناسب با اینکه در چه استیتی قرار دارد، کارهای مرتبط با آن استیت را انجام میشود

تست پنچ:

```

define NULL 0
module testbench();

    parameter BIT_LEN = 4;
    parameter clk_c    = 5;
    wire out_r;
    reg clk, start, rstn;
    reg signed [BIT_LEN - 1:0] in1, in2;
    wire signed [2 * BIT_LEN - 1:0] out;

    MUL_BOOTH #(.BIT_LEN(BIT_LEN)) mb (
        .in1(in1),
        .in2(in2),
        .out(out),
        .out_r(out_r),
        .clk(clk),
        .start(start),
        .rstn(rstn)
    );

    initial begin
        $dumpfile("report/waveform.vcd");
        $dumpvars(0,mb);
    end

    integer data_file;
    integer scan_file;
    integer seed;
    initial begin
        data_file = $fopen("seed.dat", "r");
        if (data_file == `NULL) begin
            $display("data_file handle was NULL");
            $finish;
        end
        scan_file = $fscanf(data_file, "%d", seed);
    end
endmodule

```



```

        if (scan_file == `NULL) begin
            $display("integer read error");
            $finish;
        end
    end
end

initial begin
    clk      = 0;
    forever clk = #(clk_c/2) ~clk;
end

integer i,n;
initial begin
    rstn = 0;
    start = 0;
    n     = {$random(seed)}%15 + 10;
    for (i = 0; i < n; i = i+1) begin
        rstn = 0;
        #clk_c;
        in1 = {BIT_LEN{$random(seed)}};
        while (in1 == {1'b1,{BIT_LEN-1{1'b0}}})begin
            in1 = {BIT_LEN{$random(seed)}};
        end
        in2 = {BIT_LEN{$random(seed)}};
        rstn = 1;
        start = 1;
        while (!out_r)
            begin
                #clk_c;
            end
        $display("result is ready: \t%d * %d = %d",in1, in2, out);
    end
    $finish;
end
initial
    $monitor($time, "  A:%b, XB:%b%b, state = %d, next_state = %d, count = %d");

```

مثل آزمایش سوم ما از عدد رندوم استفاده میکنیم.
 در فایل seed.dat یک عدد رندوم قرار داده ایم.
 این عدد رندوم، همان \$random ماست.

برای اینکه با هر سیمولیت این رندوم بودن ما تفاوت کند، ما به کمک یک بش در فایل seed.dat عدد رندوم را نوشته ایم. (به علت محدودیت لینوکس اینکار را کرده ایم. در نسخه ویندوز برای مادلسیم random flag\$ وجود دارد.)

نتایج:

```
result.txt - Notepad
File Edit Format View Help
VCD info: dumpfile report/waveform.vcd opened for output.
0 A:xxxx, XB:0000xxxx, state = 0, next_state = x, count = 4, B0,B-1 = x0, out_ready = 0
5 A:xxxx, XB:0000xxxx, state = 0, next_state = 4, count = 4, B0,B-1 = x0, out_ready = 0
6 A:1010, XB:00000011, state = 4, next_state = 2, count = 4, B0,B-1 = 10, out_ready = 0
10 A:1010, XB:00000011, state = 2, next_state = 3, count = 4, B0,B-1 = 10, out_ready = 0
14 A:1010, XB:01100011, state = 3, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
18 A:1010, XB:00110001, state = 4, next_state = 3, count = 3, B0,B-1 = 11, out_ready = 0
22 A:1010, XB:00110001, state = 3, next_state = 4, count = 3, B0,B-1 = 11, out_ready = 0
26 A:1010, XB:00011000, state = 4, next_state = 1, count = 2, B0,B-1 = 01, out_ready = 0
30 A:1010, XB:00011000, state = 1, next_state = 3, count = 2, B0,B-1 = 01, out_ready = 0
34 A:1010, XB:10111000, state = 3, next_state = 4, count = 2, B0,B-1 = 01, out_ready = 0
38 A:1010, XB:11011100, state = 4, next_state = 3, count = 1, B0,B-1 = 00, out_ready = 0
42 A:1010, XB:11011100, state = 3, next_state = 4, count = 1, B0,B-1 = 00, out_ready = 0
46 A:1010, XB:11011110, state = 4, next_state = 3, count = 0, B0,B-1 = 00, out_ready = 1
result is ready:
-6 * 3 = -18
50 A:1010, XB:00000011, state = 0, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
58 A:1010, XB:00000011, state = 4, next_state = 2, count = 4, B0,B-1 = 10, out_ready = 0
62 A:1010, XB:00000011, state = 2, next_state = 3, count = 4, B0,B-1 = 10, out_ready = 0
66 A:1010, XB:01100011, state = 3, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
70 A:1010, XB:00110001, state = 4, next_state = 3, count = 3, B0,B-1 = 11, out_ready = 0
74 A:1010, XB:00110001, state = 3, next_state = 4, count = 3, B0,B-1 = 11, out_ready = 0
78 A:1010, XB:00011000, state = 4, next_state = 1, count = 2, B0,B-1 = 01, out_ready = 0
82 A:1010, XB:00011000, state = 1, next_state = 3, count = 2, B0,B-1 = 01, out_ready = 0
86 A:1010, XB:10111000, state = 3, next_state = 4, count = 2, B0,B-1 = 01, out_ready = 0
90 A:1010, XB:11011100, state = 4, next_state = 3, count = 1, B0,B-1 = 00, out_ready = 0
94 A:1010, XB:11011100, state = 3, next_state = 4, count = 1, B0,B-1 = 00, out_ready = 0
98 A:1010, XB:11011110, state = 4, next_state = 3, count = 0, B0,B-1 = 00, out_ready = 1
result is ready:
-6 * 3 = -18
100 A:1010, XB:00000011, state = 0, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
106 A:1011, XB:00000110, state = 4, next_state = 3, count = 4, B0,B-1 = 00, out_ready = 0
110 A:1011, XB:00000110, state = 3, next_state = 4, count = 4, B0,B-1 = 00, out_ready = 0
114 A:1011, XB:00000011, state = 4, next_state = 2, count = 3, B0,B-1 = 10, out_ready = 0
118 A:1011, XB:00000011, state = 2, next_state = 3, count = 3, B0,B-1 = 10, out_ready = 0
122 A:1011, XB:01010011, state = 3, next_state = 4, count = 3, B0,B-1 = 10, out_ready = 0
126 A:1011, XB:00101001, state = 4, next_state = 3, count = 2, B0,B-1 = 11, out_ready = 0
130 A:1011, XB:00101001, state = 3, next_state = 4, count = 2, B0,B-1 = 11, out_ready = 0
134 A:1011, XB:00010100, state = 4, next_state = 1, count = 1, B0,B-1 = 01, out_ready = 0
138 A:1011, XB:00010100, state = 1, next_state = 3, count = 1, B0,B-1 = 01, out_ready = 0
142 A:1011, XB:11000100, state = 3, next_state = 4, count = 1, B0,B-1 = 01, out_ready = 0
146 A:1011, XB:11100010, state = 4, next_state = 3, count = 0, B0,B-1 = 00, out_ready = 1
result is ready:
-5 * 6 = -30
150 A:1011, XB:00000110, state = 0, next_state = 4, count = 4, B0,B-1 = 00, out_ready = 0
```

```

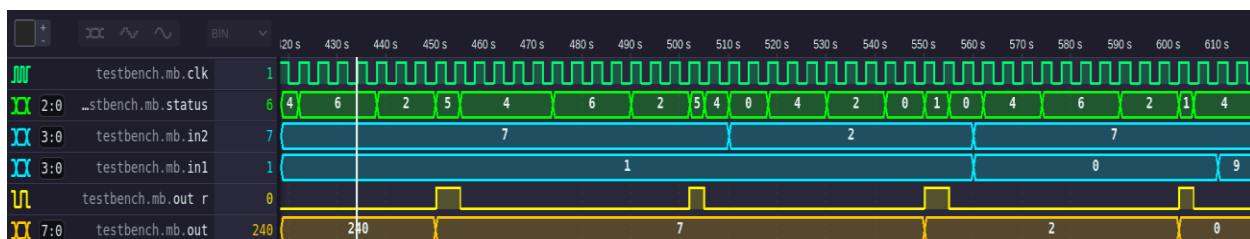
result.txt - Notepad
File Edit Format View Help

255 A:1010, XB:00001101, state = 0, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
262 A:0010, XB:00000011, state = 4, next_state = 2, count = 4, B0,B-1 = 10, out_ready = 0
266 A:0010, XB:00000011, state = 2, next_state = 3, count = 4, B0,B-1 = 10, out_ready = 0
270 A:0010, XB:11110001, state = 3, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
274 A:0010, XB:11110001, state = 4, next_state = 3, count = 3, B0,B-1 = 11, out_ready = 0
278 A:0010, XB:11110001, state = 3, next_state = 4, count = 3, B0,B-1 = 11, out_ready = 0
282 A:0010, XB:11110000, state = 4, next_state = 1, count = 2, B0,B-1 = 01, out_ready = 0
286 A:0010, XB:11110000, state = 1, next_state = 3, count = 2, B0,B-1 = 01, out_ready = 0
290 A:0010, XB:00011000, state = 3, next_state = 4, count = 2, B0,B-1 = 01, out_ready = 0
294 A:0010, XB:00011000, state = 4, next_state = 3, count = 1, B0,B-1 = 00, out_ready = 0
298 A:0010, XB:00011000, state = 3, next_state = 4, count = 1, B0,B-1 = 00, out_ready = 0
302 A:0010, XB:00000110, state = 4, next_state = 3, count = 0, B0,B-1 = 00, out_ready = 1
result is ready: 2 * 3 = 6
305 A:0010, XB:00000011, state = 0, next_state = 4, count = 4, B0,B-1 = 10, out_ready = 0
310 A:1011, XB:00000100, state = 4, next_state = 3, count = 4, B0,B-1 = 00, out_ready = 0
314 A:1011, XB:00000100, state = 3, next_state = 4, count = 4, B0,B-1 = 00, out_ready = 0
318 A:1011, XB:00000010, state = 4, next_state = 3, count = 3, B0,B-1 = 00, out_ready = 0
322 A:1011, XB:00000010, state = 3, next_state = 4, count = 3, B0,B-1 = 00, out_ready = 0
326 A:1011, XB:00000001, state = 4, next_state = 2, count = 2, B0,B-1 = 10, out_ready = 0
330 A:1011, XB:00000001, state = 2, next_state = 3, count = 2, B0,B-1 = 10, out_ready = 0
334 A:1011, XB:01010001, state = 3, next_state = 4, count = 2, B0,B-1 = 10, out_ready = 0
338 A:1011, XB:00101000, state = 4, next_state = 1, count = 1, B0,B-1 = 01, out_ready = 0
342 A:1011, XB:00101000, state = 1, next_state = 3, count = 1, B0,B-1 = 01, out_ready = 0
346 A:1011, XB:11011000, state = 3, next_state = 4, count = 1, B0,B-1 = 01, out_ready = 0
350 A:1011, XB:11101100, state = 4, next_state = 3, count = 0, B0,B-1 = 00, out_ready = 1
354 A:1011, XB:11101100, state = 3, next_state = 0, count = 0, B0,B-1 = 00, out_ready = 1
result is ready: -5 * 4 = -20
355 A:1011, XB:00000100, state = 0, next_state = 4, count = 4, B0,B-1 = 00, out_ready = 0
362 A:1100, XB:00000100, state = 4, next_state = 3, count = 4, B0,B-1 = 00, out_ready = 0
366 A:1100, XB:00000100, state = 3, next_state = 4, count = 4, B0,B-1 = 00, out_ready = 0
370 A:1100, XB:00000010, state = 4, next_state = 3, count = 3, B0,B-1 = 00, out_ready = 0
374 A:1100, XB:00000010, state = 3, next_state = 4, count = 3, B0,B-1 = 00, out_ready = 0
378 A:1100, XB:00000001, state = 4, next_state = 2, count = 2, B0,B-1 = 10, out_ready = 0
382 A:1100, XB:00000001, state = 2, next_state = 3, count = 2, B0,B-1 = 10, out_ready = 0
386 A:1100, XB:01000001, state = 3, next_state = 4, count = 2, B0,B-1 = 10, out_ready = 0
390 A:1100, XB:00100000, state = 4, next_state = 1, count = 1, B0,B-1 = 01, out_ready = 0
394 A:1100, XB:00100000, state = 1, next_state = 3, count = 1, B0,B-1 = 01, out_ready = 0
398 A:1100, XB:11100000, state = 3, next_state = 4, count = 1, B0,B-1 = 01, out_ready = 0
402 A:1100, XB:11110000, state = 4, next_state = 3, count = 0, B0,B-1 = 00, out_ready = 1
result is ready: -4 * 4 = -16
405 A:1100, XB:00000100, state = 0, next_state = 4, count = 4, B0,B-1 = 00, out_ready = 0
410 A:0001, XB:00000111, state = 4, next_state = 2, count = 4, B0,B-1 = 10, out_ready = 0

```

تعداد تست های گرفته شده زیاد است. تنها تصویر بعضی از آنها را در گزارش آورده ایم.
اطلاعات تکمیلی در فایل results.txt در پوشه اصلی موجود است.

نتایج Waveform:



پایان