به نام خداوند بخشنده مهربان

آز سیستم های دیجیتال

دکتر اجلالی

مبینا حیدری، عاطفه قندهاری، نیکا قادری
بهار ۱۴۰۳

تاریخ گزارش: ۲ اردیبهشت ۱۴۰۳ | **طراحی ضرب کننده** | **آزمایش پنجم**

۱. شرح آزمایش

هدف این آزمایش طراحی ضرب کنندهایست که به کمک روش Booth ، حاصلضرب دو ورودی را محاسبه کند.

ورودی ها:

in_A :· ورودی اول ۴ بیتی یا مضروب.

in_B :· ورودی دوم ۴ بیتی یا مضروب فیه.

- rstN
- clk

خروجی ها:

res :· حاصلضرب خروجی ۸ بیتی.

- Done: با پایان محاسبه، برابر با ۱ می شود.

۲. راه حل کلی

به طور کلی در این الگوریتم، با رسیدن به ۱ در مضروب فیه، مقدار مضروب را از حاصل کم می کنیم و با رسیدن به ۰ در مضروب فیه، مقدار مضروب را به حاصل اضافه میکنیم و سپس حاصل را به راست شیفت میدهیم و در صورتی که رقم فعلی و پیشین مضروب فیه یکسان بود، حاصل را فقط شیفت میدهیم.

اما ما در این پیاده سازی، اندکی متفاوت عمل می کنیم. با برخورد به هر رقم، محاسبات آن)جمع یا تفریق(در مرحله قبلی انجام شده است. پس فقط شیفت میدهیم و محاسبات مربوط به رقم بعدی را انجام میدهیم .برای انجام چندین شیفت در یک پالس ساعت، تنها کافیست در هر مرحله ای که باید محاسبات انجام شود)یعنی در مضروب فیه بیت قبلی ۰ و بیت فعلی ۱ باشد، یا برعکس(، اندیس اولین بیت ۰ یا ۱ پس از آن را بیابیم.

۳. ماژول ها

first_one : در این ماژول اندیس اولین)کم ارزشترین(بیت از عدد که برابر با ۱ است را می یابیم. محاسبه آن به کمک جدول کارنو انجام میگیرد.کد وریالگ آن به شرح زیر است:

```
module first_one (
    input   [3:0]   in,
    output  [1:0]   index
);

assign index = {~(in[1] | in[0]), ~in[0] & (in[1] | ~in[2])};

endmodule
```

first_zero : در این ماژول اندیس اولین)کم ارزشترین(بیت از عدد که برابر با ۰ است را می یابیم. محاسبه آن به کمک جدول کارنو انجام میگیرد.کد وریالگ آن به شرح زیر است :

```
module first_zero (
    input   [3:0]   in,
    output  [2:0]   index
);

assign index[2] = in[3] & in[2] & in[1] & in[0];
assign index[1] = in[1] & in[0] & (~(in[3] & in[2]));
assign index[0] = in[0] & (~in[1] | in[2]);

endmodule
```

Control_unit : در این ماژول مقادیر الزم برای شیفت دادن دو عدد A و B و هم چنین پایان یافتن ضرب مشخص می شود. در op مقدار کم ارزشترین بیت B قرار می گیرد و با توجه به ۰ یا ۱ بودن مقدار آن، به ترتیب اندیس کم ارزشترین بیت ۱ یا ۰ پس از آن در amt_shft_Bقرار میگیرد. مقدارshifted که برابر با مقدار شیفت خوردن B تا کالک قبلیست نیز با amt_shft_B جمع زده میشود تا amt_shft_A مشخص شود، یعنی مقداری که Aبرای جمع یا منها شدن با حاصل نهایی باید به چپ شیفت بخورد. البته اگر در کلاک اول باشیم، مقدار op بدون توجه به کم ارزشترین بیت B برابر با ۰ می شود و در حاصل نهایی منها رخ می دهد .اگر مقدار shifted + amt_shft_Bبرابر با ۴ یا بیشتر شود، یعنی با شیفت به راست دادن B به این مقدار، عمالً تمامی بیت های B بررسی شده است. پس محاسبات به اتمام رسیده است .

برای بخش ترتیبی مدار، در صورت ریست شدن، مقدار shifted برابر با صفر شده و در کالک اول پس از شروع محاسبات قرار میگیریم. در غیر اینصورت و با باال رفته لبه کالک، مقدار shifted با amt_shft_B جمع شده و دیگر در اولین کالک قرار نداریم .کد وریالگ آن به شرح زیر است:

```
module control_unit (
    input   [3:0]   B,
    input           rstN,
    input           clk,
    output  [2:0]   A_shft_amt,
    output  [2:0]   B_shft_amt,
    output          op,     // 0: subtract, 1: add
    output          done
);

reg     [2:0]   shifted;
reg             first_clock;
wire    [1:0]   one_index;
wire    [2:0]   zero_index;

first_one   FO (B, one_index);
first_zero  FZ (B, zero_index);

assign op = B[0] & (~first_clock);
assign B_shft_amt = op ? zero_index : {1'b0, one_index};
assign A_shft_amt = shifted + B_shft_amt;
assign done = shifted + B_shft_amt > 3;

always @(posedge clk or negedge rstN) begin
    if (~rstN) begin
        shifted <= 0;
        first_clock <= 1;
    end else begin
        first_clock <= 0;
        shifted <= shifted + B_shft_amt;
    end
end

endmodule
```

Datapath: چون مدار ترتیبیست، از block always استفاده می کنیم که به لبه ی مثبت کالک یا لبهی منفی rstNحساس است.

در صورت ریست شدن، مقدار ورودیهای ۴ بیتی in_A و in_B به ترتیب در دو رجیستر ۸ بیتی A وB که از سمت بیت پرارزش extendشده اند، منتقل میشوند، چون تمامی شیفت به راست ها arithmetic انجام میگیرند. مقدار خروجی نیز برابر با ۰ می شود .

در غیر این صورت و با بالا رفتن لبه کالک و پایان نپذیرفتن محاسبات ، مقدار B به اندازه ی amt_shft_B به راست شیفت می خورد و مقدار خروجی متناسب با ۰ یا ۱ بودن op با مدارA که به اندازه amt_shtf_A به چپ شیفت خورده، منها یا جمع می شود:.

```
module datapath (
    input       [3:0]   A_in,
    input       [3:0]   B_in,
    input               rstN,
    input               clk,
    input       [2:0]   A_shft_amt,
    input       [2:0]   B_shft_amt,
    input               op,
    input               done,
    output reg  [7:0]   ACC,
    output reg  [3:0]   B
);

reg [7:0]   A;

always @(posedge clk or negedge rstN) begin
    if (~rstN) begin
        A <= {{4{A_in[3]}}, A_in};
        B <= {{4{B_in[3]}}, B_in};
        ACC <= 0;
    end else if (~done) begin
        B <= B >> B_shft_amt;
        ACC <= ACC + (op ? 1 : -1) * (A << A_shft_amt);
    end
end

endmodule
```

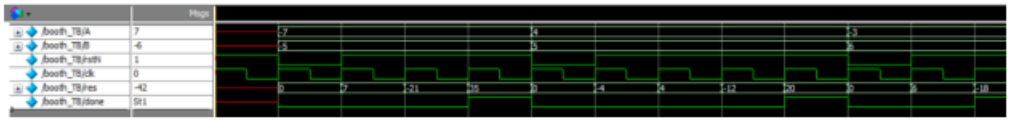
٤ . ماژول اصلی :

Both : این ماژول، ماژول اصلی آزمایش است. یک instance از ماژول unit_control در آن ساخته می شود که مقادیر amt_shft_A و amt_shft_B و done و op خروجی های آن هستند. یک instance هم از ماژول datapath ساخته میشود که مقادیر res و B را خروجی می دهد. این دو ماژول به هم وابستهاند. کد وریالگ آن به شرح زیر است:

```
module booth (
    input   [3:0]   A_in,
    input   [3:0]   B_in,
    input           rstN,
    input           clk,
    output  [7:0]   res,
    output          done
);

wire [2:0]  A_shft_amt;
wire [2:0]  B_shft_amt;
wire [3:0]  B;

control_unit CU (B, rstN, clk, A_shft_amt, B_shft_amt, op, done);
datapath DP (A_in, B_in, rstN, clk, A_shft_amt, B_shft_amt, op, done, res, B);

endmodule
```

Waveform

حاصل شبیه‌سازی test bench این آزمایش به شکل زیر است:



```
# res: xxxxxxxx
# res: 00000000
# res: 00000111
# res: 11101011
#  -7 *  -5 =   35
# res: 00100011
# res: 00000000
# res: 11111100
# res: 00000100
# res: 11110100
#   4 *   5 =   20
# res: 00010100
# res: 00000000
# res: 00000110
#  -3 *   6 =  -18
# res: 11101110
# res: 00000000
# res: 11110010
# res: 00001110
#   7 *  -6 =  -42
# res: 11010110
```