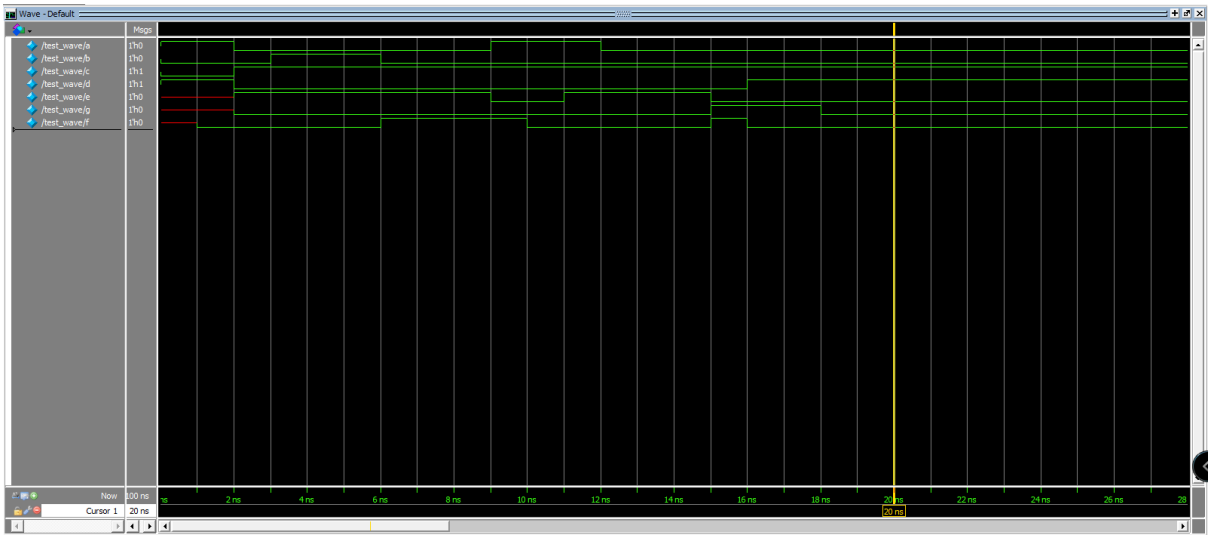
به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال — پاییز ۱۴۰۰

تمرین ۲، سوال ۱ — آرین احدی نیا، ۹۸۱۰۳۸۷۸



شکل امواج در modelsim به صورت فوق در می‌آیند (متاسفانه علی‌الرغم تلاش‌های بنده، نرم‌افزار modelsim حین export کردن عکس به بنده خطا میداد و مجبور شدم که به صورت فوق شکل امواج را نمایش دهم.)

دقت بفرمایید که حین تعریف گیت‌ها، برای آنها به صورت #(<rise>, <fall>) تاخیر تعریف شده. به این صورت که رفتن از هر مقداری غیر از ۱ به مقدار ۱ برای یک گیت به مقدار <rise> تاخیر دارد. برای رفتن به مقدار صفر و <fall> نیز همین مورد صادق است. رفتن به حالت خاموشی و نامشخص نیز به اندازه min(<rise>, <fall>) تاخیر دارد.

توجه بفرمایید که اگر یکی از ورودی‌های گیت AND برابر صفر باشد، حاصل صفر خواهد شد و اگر یکی از ورودی‌های گیت OR برابر ۱ باشد، حاصل ۱ خواهد بود. نوعا زمانی که بتوان خروجی گیت را بر حسب یک سری از ورودی‌ها تشخیص داد، مقدار آن گیت معین میشود. مثلا زمانی که یکی از ورودی‌های NOR برابر ۱ باشد مقدار آن برابر صفر خواهد شد.

در شروع کار، مقادیر اولیه a, b, c, d مقداردهی می‌شوند اما بقیه مقادیر به دلیل اینکه تحت تاثیر تاخیر گیت‌ها قرار دارند، مقداردهی نشده‌اند و مقدار x میگیرند.

بنابرین در شروع کار، حاصل e برابر ۱ خواهد بود. اما نمایان شدن این اثر با تاخیر rise گیت OR همراه خواهد بود که برابر ۲ است. بنابرین در زمان ۲ حاصل e برابر ۱ میشود.

در شروع کار چون مقدار c صفر است، حتی با اینکه e مقدار دهی نشده، حاصل گیت AND که برابر f است صفر خواهد بود که نمایان شدن آن با تاخیر fall این گیت که برابر ۱ است همراه است برای همین در زمان ۱ مقدار این گیت برابر صفر میشود.

در مورد g نیز همین‌گونه است. در ابتدای امر با اینکه f مقداردهی نشده، چون d برابر ۱ است حاصل برابر صفر خواهد شد و با تاخیر fall که برابر ۲ است در خروجی نمایان خواهد شد.

در ادامه ماجرا نیز روند به همین صورت طی خواهد شد.

یک مورد قابل توجه وجود دارد. به عنوان مثال در زمان ۲ مقدار f و g برابر صفر است. بنابرین انتظار داریم پس از گذشت تاخیر rise یعنی در زمان ۷ مقدار g به ۱ تغییر یابد. اما این اتفاق نمی‌افتد. علت این امر این است که در زمان ۶، مقدار f به یک تغییر می‌کند و حاصل g برابر صفر میشود. این در حالتی است که g هنوز rise نکرده. بنابرین rising مقدار g در زمان ۷ لغو میگردد و مقدار g برابر صفر می‌ماند. موارد دیگری از این دست نیز وجود دارد. (Delays are INERTIAL)

به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال — پاییز ۱۴۰۰

تمرین ۲، سوال ۲ — آرین احدی نیا، ۹۸۱۰۳۸۷۸

توجه کنید که حین تغییر مقدار ابتدا تاخیری صرف assign کردن مقدار جدید میشود سپس تاخیری صرف قرار گرفتن مقدار جدید روی سراسر wire میشود (این تاخیر را میتوان زمانی در نظر گرفته که طول میکشد سیگنال از این سر سیم به آن سر سیم برسد!). بنابرین در نهایت تاخیر برای قرار گیری مقدار جدید، برابر جمع تاخیر قرار گرفته روی assign و wire است. این مجموع برای هر دو سیم برابر ۷ است. بنابرین در حالت عادی اگر به C مقدار دهیم کنیم، هر دوی آنها پس از ۷ ثانیه به مقدار C خواهند رسید.

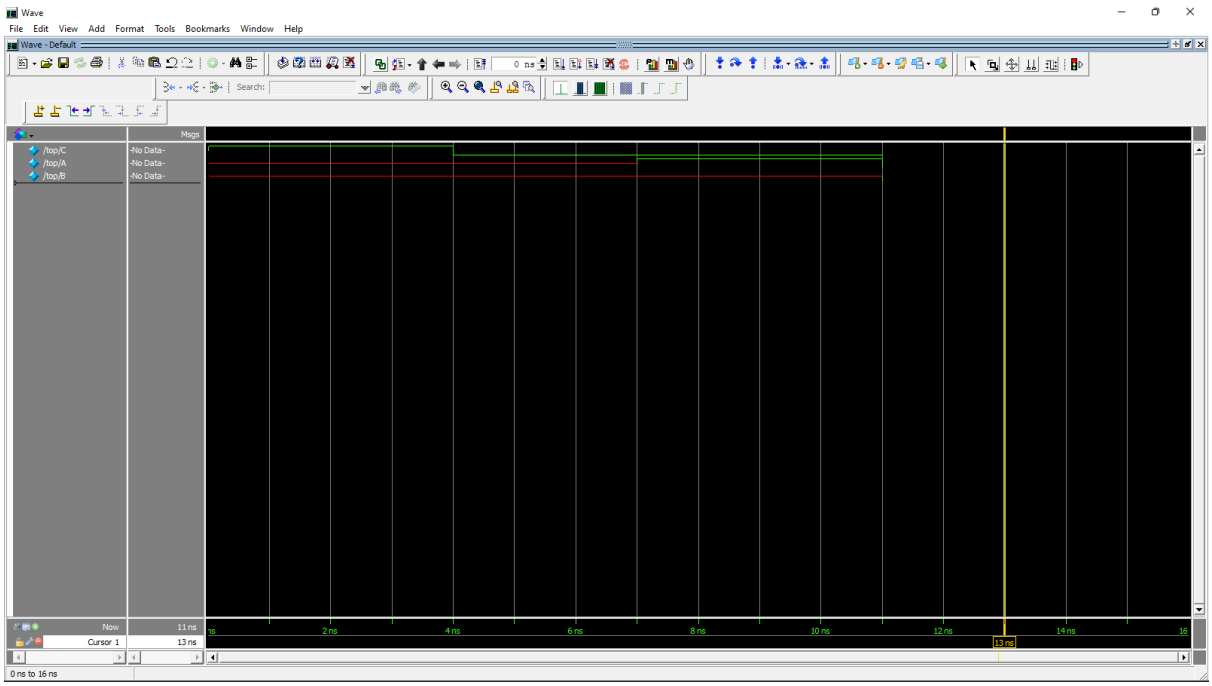
نکته‌ای که در رابطه با این تاخیرها وجود دارد، Inertial بودن آنها است. به این معنی که اگر در حین زمان تغییر، شرایطی رخ دهد که آن تغییر حالت نهایی را نمایان نکند، تغییر به طور کلی اتفاق نمی‌افتد. ما با استفاده از این موضوع می‌توانیم C را به گونه‌ای مقداردهی کنیم که شکل موج A و B متفاوت باشد. C را به شکل زیر مقداردهی میکنیم.

```
module top;
    reg C;

    wire #4 A;
    wire #2 B;

    assign #3 A = C;
    assign #5 B = C;

    initial begin
        assign C = 1'b1;
        #4
        assign C = 1'b0;
    end
endmodule
```



در ابتدا مقدار C برابر ۱ است. اتفاقی که می‌افتد این است که پس از ۳ واحد زمانی مقدار C به A نسبت (assign) داده می‌شود. و پس از ۴ واحد زمانی این assignment در wire نمایان می‌شود.

اما برای B شرایط متفاوت است. چون در زمان صفر مقدار C برابر یک است، در زمان ۵ مقدار ۱ به B نسبت داده خواهد شد. اما چون در زمان ۴ مقدار C تغییر می‌کند و به دلیل خاصیت Inertial بودن، این تغییر لغو خواهد شد و اگر شرایط مساعد باشد، در ۵ واحد زمانی بعد مقدار جدید C به B نسبت داده و ۲ واحد زمانی بعد این تغییر در wire نمایان میشود. همانگونه که در تصویر مشاهده میکنید. این اتفاق نیز می‌افتد در زمان ۱۱، مقدار صفر به assign C میشود.

به نام خداوند بخشنده مهربان

طراحی سیستم‌های دیجیتال — پاییز ۱۴۰۰

تمرین ۲، سوال ۳ — آرین احدی نیا، ۹۸۱۰۳۸۷۸

ابتدا توجه بفرمایید که ضرب ۴ بیتی را می‌توانیم به صورت حاصل جمع تعداد ضرب ۲ بیتی محاسبه کنیم

$$a_3a_2a_2a_1 \times b_3b_2b_1b_0 = (2^2(a_3a_2) + a_1a_0) \times (2^2(b_3b_2) + b_1b_0)$$
$$= 2^4(a_3a_2 \times b_3b_2) + 2^2(a_3a_2 \times b_1b_0) + 2^2(a_1a_0 \times b_3b_2) + (a_1a_0 \times b_1b_0)$$

توجه کنید که حاصل ضرب دو عدد n بیتی بدون علامت حداکثر برابر $(2^n - 1)(2^n - 1)$ خواهد بود. بنابرین این حاصل کوچکتر از $(2^n)^2$ خواهد بود بنابرین برای نمایش این حاصل 2^n بیت کافی است.

بنابرین حاصل هر یک از $a_{i+1}a_i \times b_{j+1}b_j$ یک عدد ۴ بیتی خواهد بود. توجه کنید که $2^n(a_{i+1}a_i \times b_{j+1}b_j)$ با چسبیدن n صفر در سمت راست عدد قابل تولید است.

با استفاده از Full Adder می‌توانیم یک جمع کننده n بیتی بسازیم. این Adder دو عدد n بیتی دریافت و یک عدد n بیتی خروجی می‌دهد که حاصل جمع دو ورودی است. در ساخت ماژول Adder، تعداد بیت ورودی، N، را به صورت پارامتر در نظر میگیریم تا در صورت لزوم بتوانیم جمع کننده‌هایی با طول‌های متفاوت بسازیم. برای نگارش این ماژول از generate استفاده می‌کنیم و n عدد Full Adder را به هم متصل می‌کنیم. برای این منظور از تعدادی wire کمکی نیز برای وصل کردن carry استفاده می‌کنیم.

در ماژول اصلی ابتدا با استفاده از ضرب‌کننده دو بیتی عبارات $a_{i+1}a_i \times b_{j+1}b_j$ تولید می‌کنیم. سپاس با رعایت ارزش‌گذاری مکانی ، حاصل‌ها را با یک دیگر جمع میکنیم. جدول زیر ارزش های مکانی را نشان می‌دهد.

Term #	7	6	5	4	3	2	1	0
1					$a_1a_0 \times b_1b_0$			
2			$a_1a_0 \times b_3b_2$					
3			$a_3a_2 \times b_1b_0$					
4	$a_3a_2 \times b_3b_2$							

توجه کنید که دو بیت کم‌ارزش $a_1a_0 \times b_1b_0$ با هیچ مقداری جمع نمی‌شوند بنابرین میتوانیم آنها را مستقیما با کمک یک buffer به دو بیت کم‌ارزش خروجی وصل کنیم.

در مرحله اول، می‌توانیم عبارات ۲ و ۳ را با یکدیگر جمع کنیم. توجه کنید که حاصل جمع دو عدد ۴ بیتی برابر یک عدد ۵ بیتی خواهد بود بنابرین برای جمع کردن این دو عبارت از ماژول Adder با پارامتر $N = 5$ استفاده می‌کنیم و یک صفر به سمت چپ این دو عبارت اضافه می‌کنیم تا پنج بیتی شوند. در نتیجه عبارات جدید به صورت زیر خواهند بود.

1					$a_1a_0 \times b_1b_0$			
(2 + 3)		$a_1a_0 \times b_3b_2 + a_3a_2 \times b_1b_0$						
4	$a_3a_2 \times b_3b_2$							

حال دو بیت پرارزش $a_1a_0 \times b_1b_0$ را در سمت راست $a_3a_2 \times b_3b_2$ قرار می‌دهیم تا یک عدد ۶ بیتی به دست آید. سپس یک صفر در سمت چپ عبارت ۵ بیتی $(3 + 2)$ قرار می‌دهیم تا ۶ بیتی شود. سپس این دو عدد را به کمک یک Adder با پارامتر $N = 6$ جمع می‌کنیم و به ۶ بیت پرارزش خروجی وصل می‌کنیم.

```
module voter(in1, in2, in3, out, error);
    input [1:0] in1, in2, in3;
    output [1:0] out;
    output error;

    wire [1:0] in_bits [0:3];
    wire [1:0] inequal [0:2];
    wire e [0:2];
    wire ne [0:2];
    wire sel [0:3];

    buf buff_1 [1:0] (in_bits[0], in1);
    buf buff_2 [1:0] (in_bits[1], in2);
    buf buff_3 [1:0] (in_bits[2], in3);
    buf buff_4 [1:0] (in_bits[3], 2'bxx);

    genvar i;
    generate for (i = 0; i < 3; i = i + 1) begin: xor_loop
        xor ieq [1:0] (inequal[i], in_bits[i], in_bits[(i + 1) % 3]);
        end
    endgenerate

    genvar j;
    generate for (j = 0; j < 3; j = j + 1) begin: error_loop
        or (e[j], inequal[j][0], inequal[j][1]);
        not (ne[j], e[j]);
        end
    endgenerate

    nor (error, ne[0], ne[1], ne[2]);

    buf (sel[0], ne[0]);
    and (sel[1], ne[1], e[0]);
    and (sel[2], ne[2], e[1], e[0]);
    and (sel[3], e[2],  e[1], e[0]);

    genvar k;
    generate for (k = 0; k < 4; k = k + 1) begin: out_loop
        bufif1 tribuff [1:0] (out, in_bits[k], sel[k]);
        end
    endgenerate
endmodule
```

برای حل این سوال، می‌خواهیم از generate استفاده کنیم. بنابرین در ابتدای کار سه ورودی in را در آرایه in_bits قرار می‌دهیم تا بتوانیم از طریق نمایه به آنها دسترسی داشته باشیم. همچنین عضو چهارم آرایه را نیز xx در نظر میگیریم تا در صورت وقوع error آن را خروجی دهیم.

توجه کنید که XOR دو بیت اگر آن دو برابر باشند برابر صفر و در غیر این صورت برابر ۱ خواهد بود. ابتدا به صورت بیت به بیت XOR هر دو ورودی را محاسبه میکنیم و در inequal قرار می‌دهیم. سپس بیت‌های آن را or می‌کنیم. به این صورت سه عدد یک بیتی متناظر با هر ترکیب دوتایی در e بدست خواهد آمد. هر یک از این اعداد اگر برابر ۱ باشد به این معنی است که آن دو عدد حداقل در یک بیت با یکدیگر اختلاف دارند و اگر برابر صفر باشد به معنی این است که آن دو عدد کاملا یکسان هستند. بنابرین nor نقیض این سه بیت مقدار error را مشخص می‌کنند. (ne نقیض e است و همچنین e[i] نمایان گر این است که آیا عدد iام و i+1ام نابرابر هستند یا خیر)

حال با استفاده از tri state buffer می‌توانیم خروجی out را مشخص کنیم. ابتدا باید مشخص کنیم که کدام عدد با خروجی متصل شود. اگر عدد اول با عدد دوم برابر باشد، عدد iام با i+1ام برابر باشد، عدد iام را به خروجی وصل می‌کنیم.