برای انجام این تمرین روش پیشنهادی من به این گونه است که برای تمام سیم ها یک وکتور از تغییرات ذخیره داشته باشیم به این گونه که در هر یک از ساختار های تغییرات مقدار سیم و لحظه ای که به این مقدار رسیده را نگه میداریم. برای هر گیت نیز برای انجام عملیات منطقی یک ساختار تعریف میکنیم که مقدار ورودی و زمان گرفتن آن مقدار را به گیت بدهد.

در ابتدا ساختار مربوط به سیم ها را که با nodes نامگذاری شده را با نام و تغییرات مقادیرشان در هر زمان که بستگی به test case های ورودی دارد پر میکنیم. از طرفی تمام گیت های موجود را به همراه مشخصاتشان که در ساختار gate آمده است در وکتور ذخیره میکنیم. حال زمان مرتب کردن node ها رسیده است به گونه ای که سیم ای باید در وکتور زود تر آمده باشد که در ابتدا کار گیت اش زود تر شروع میشود. برای این مرتب سازی از وکتور gates کمک میگیریم به این گونه که در وکتور nodes که در حال حاضر تنها input ها را دارد به دنبال ورودی های گیت اول میگردیم، اگر هر دو اینپوت موجود بود که خروجی آن گیت را به انتهای وکتور نود اضافه میکنیم و آن گیت را از وکتور گیت ها (این وکتور یک کپی از وکتور اصلی است) پاک میکنیم. اگر اینپوت های آن گیت بین node ها تا این لحظه موجود نبود؛ ابتدا یک کپی از آن گیت را به انتهای وکتور گیت ها اضافه میکنیم و از ابتدای وکتور گیت ها آن را حذف میکنیم. این کار تا آنجایی انجام میشود که اندازه وکتور گیت ها صفر شود.

الان که سیم ها را به شکل مرتب شده داریم، با تمرکز بر روی خروجی گیت ها که به ترتیب در وکتور nodes آمده اند اینپوت های گیت ها و تغییراتشان را پیدا میکنیم و برای این کار باید تغییرات هر کدام از اینپوت ها و زمان آن ها را در نظر داشته باشیم و وکتوری از تغییرات ورودی های گیت ها و زمان آن تشکیل دهیم و برای این کار در تابع generateGateTransitions با در نظر گرفتن زمان ها و مقادیر قبلی ای که بر روی هر اینپوت بوده است کل تغییرات ورودی ها و زمانشان را در خروجی تابع در یک وکتور میدهیم.

برای بخش محاسبات منطقی هم از همین وکتور تغییرات ورودی ها و زمان تغییر استفاده میکنیم و از آنجا که این تغییرات به ترتیب هستند به راحتی خروجی هر یک را محاسبه میکنیم و در صورت متفاوت بودن مقدار جدید با قبلی و دیر تر بودن زمان تولید آن به انتهای وکتور تغییرات خروجی گیت اضافه میشود. گاهی اوقات برخی مقادیر ورودی به گونه ای اند که باعث میشود مقدار خروجی بسیار دیر به وجود بیاید در صورتی که قبل از تولید خروجی ورودی ها تغییر میکنند و این نتیجه دوم سریع تر بر خروجی می نشیند، برای رفع کردن این مشکل هم در صورتی که نتیجه جدید زمان تولید کمتری از نتیجه قبل داشته باشد نتیجه قبل را از وکتور حذف میکنیم و در صورت متفاومت بودن این نتیجه جدید با نتیجه قبل تر آن را به انتهای وکتور اضافه میکنیم. تمام این تغییرات بر روی وکتور nodes پیاده میشود و در نتیجه این وکتور در مراحل بعد اطلاعات کامل ورودی های گیت های بعدی را خواهد داشت.

در نهایت وکتور nodes ما تمام تغییرات سیم ها را با زمانشان خواهد داشت که در فایل تکست نتیجه قابل مشاهده است.

یک نمونه از ورودی:

```
#00 111
#21 110
#31 101
#19 011
```

```
module mux (a, b, s, y, w);
input a;
input b;
input s;
output y;
output w;
wire sbar;
wire aa;
wire bb;

nand #(3,5) U1 (sbar, s, s);
xor #(3,5) U2 (aa, a, sbar);
nor #(3,5) U3 (bb, b, s);
or #(3,5) U4 (y, aa, bb);
and #(3,5) U4 (w, aa, bb);



endmodule
```

خروجی:

```
name:a  ->   value:1  time:0  /    value:0  time:71
name:b  ->   value:1  time:0  /    value:0  time:52  /    value:1  time:71
name:s  ->   value:1  time:0  /    value:0  time:21  /    value:1  time:52
name:sbar  ->   value:0  time:5  /    value:1  time:24  /    value:0  time:57
name:aa  ->   value:1  time:8  /    value:0  time:29  /    value:1  time:60  /    value:0  time:76
name:bb  ->   value:0  time:5
name:y  ->   value:1  time:11  /    value:0  time:34  /    value:1  time:63  /    value:0  time:81
name:w  ->   value:0  time:10
```