**نگین سفاری ۸۱۰۱۹۷۵۲۵**
**گزارش کار تمرین کامپیوتری ۳**

برای پیاده سازی الگریتم کویین مک کلاسکی بر روی کد ورودی، از کلاس QM استفاده شده است.

```
class QM {
    list<string> minterm;
    list<string> PIs;
    list<string> EPIs;
    list <list <string>> cubes;
    list <list <string>> num_of_ones;
```

برای انجام این کار ابتدا assign هایی تشخیص داده می شوند که حالت شرطی ندارند و بر روی آن ها کویین مک کلاسکی پیاده سازی می شود. برای انجام این کار ابتدا minterm ها تشکیل داده می شوند. به اینگونه که اگر علامت ~ پشت ورودی ای باشد به جای آن 0 و در غیر این صورت یک در نظر گرفته میشود. Minterm ها را در قالب string هایی از صفر و یک نگهداری میکنیم. این minterm ها در اصل همان 0 cube می باشند.

تعداد یک های موجود در هر minterm را در num_of_ones نگه داری میکنیم. برای تشکیل 1cube و دیگر cube ها از محاسبه تعداد یک ها استفاده می شود. به این گونه که مثلا برای محاسبه 1cube یک minterm انتخاب میشود و با باقی minterm ها از لحاظ داشتن تعداد ۱ هایی به اندازه ی یکی کمتر و یا بیشتر مقایسه می شود اگر minterm دیگری باشد که در تعداد ۱ تنها یکی اختلاف داشته باشند، از لحاظ جایگاه اعداد ۱ و صفر مقایسه می شوند، اگر تنها در یک جا این تفاوت باشد 1cube مربوطه به همراه X در محل تفاوت تشکیل میشود. پس از آنکه این minterm با بقیه minterm ها چک شد و 1cube های مربوطه اضافه شدند، این minterm از لیست حذف میشود و اگر هیچ 1cube ای تولید نکرده باشد، نگه داشته می شود. به همین ترتیب همین عملیات را بر روی 1cube ها برای تولید 2cube ها و باقی cube ها پیش میگیرم. در انتها در cubes تنها PI ها باقی می مانند که ممکن است PI تکراری هم بین آن ها باشد. آن تکراری ها را پاک میکنیم و در لیست PIs تمامی آن ها نگه داشته میشوند.

**نگین سفاری ۸۱۰۱۹۷۵۲۵**
**گزارش کار تمرین کامپیوتری ۳**

حال برای تشکیل EPI ها باید جدولی تشکیل دهیم که در آن مشخص شود که هر PI کدام minterm ها را پوشش دهی میکند. به ازای هر پوشش دهی عدد ۱ و در غیر این صورت ۰ را قرار میدهیم. سپس minterm هایی را پیدا میکنیم که تنها یک PI آن ها را پوشش دهی میکند و پس از انتخاب آن PI به عنوان یک EPI تمامی minterm هایی که به کمک آن PI پوشش دهی میشدند را حذف میکنیم (هم از جدول و هم از لیست minterm). و همچنین آن PI از لیست PIs هم حذف میشود. سپس به دنبال آن PI ای میگردیم که تا این مرحله بیشترین minterm را پوشش دهی کند و عملیات حذف و اضافه را باز تکرار میکنیم. سپس دوباره عملیات را از پیدا کردن minterm هایی که تنها یک PI آن ها را پوشش دهی میکند شروع و تکرار میکنیم تا آنجایی که جدول کاملا خالی شده باشد.

حال که EPI ها را داریم باید گیت های and و or و not آن ها را تشکیل دهیم. ازانجا که در تمرین کامپیوتری ۱ گیت ها ماکسیمم ۲ ورودی بودند، باید گیت های دو ورودی و یک ورودی داشته باشیم. برای نگهداری ورودی و خروجی و تاخیر گیت ها از struct استفاده کرده ام. برای تشکیل گیت های چند ورودی از گیت های دو ورودی ابتدا دو تا دو تا ورودی ها را به گیت ها می دهیم و برایشان گیت تشکیل می دهیم و خروجی شان را در وکتوری ذخیره میکنیم و اگر تعداد ورودی ها زوج نبود آن ورودی آخر را با اولین خروجی وارد گیت میکنیم و آن ورودی دیگر را از وکتور خروجی ها پاک میکنیم و آنقدر این کار را تکرار میکنیم که وکتور خروجی ها تنها یک عضو داشته باشد. و بدین ترتیب تمامی گیت ها را با ورودی ها و خروجی ها در وکتور ها ذخیره میکنیم.

در انتها تمامی wire های میانی و ورودی های اصلی و خروجی های اصلی را به همراه گیت ها در فایل مینویسیم.

**نگین سفاری ۸۱۰۱۹۷۵۲۵**
**گزارش کار تمرین کامپیوتری ۳**

ورودی داده شده به برنامه به صورت زیر است:

```
module main(input a,b,c,d,s, output o);
  wire w,f,g;

  assign #10 w = (~a & ~b & ~c & ~d )| (~a & ~b & ~c & d )| (~a & ~b & c
& ~d )| (~a & b & ~c & d )| (~a & b & c & ~d )| (~a & b & c & d )| (a &
~b & ~c & ~d )| (a & ~b & ~c & d )| (a & ~b & c & ~d )| (a & b & c & ~d
);
  assign #12 f = (~w & ~b & c & ~d) | (~w & b & c & ~d) | (w & ~b & ~c &
~d) | (w & ~b & ~c & d) | (w & ~b & c & ~d) | (w & ~b & c & d) | (w & b &
c & ~d) | (w & b & c & d);
  assign #14 g = (~a & ~b & ~c & ~d )| (~a & ~b & c & d )| (~a & b & ~c &
d )| (~a & b & c & ~d )| (~a & b & c & d )| (a & ~b & c & ~d )| (a & ~b &
c & d )| (a & b & ~c & d );
  assign #5 o = s ? g : f ;

endmodule
```

EPI های تشخیص داده برای هر assign ای که QM بر روی آن پیاده شده به صورت زیر است:

```
EPIs : X00X  XX10  01X1
EPIs : XX10  10XX  1X1X
EPIs : 0000  011X  101X  X101  0X11
```

کد خروجی به صورت زیر است:

```
module main ( a , b , c , d , s , o );

input a ;

input b ;

input c ;

input d ;

input s ;

output o ;

wire aBAR ;

wire bBAR ;

wire cBAR ;

wire dBAR ;

wire wBAR ;
```

```
wire sBAR ;
wire w0 ;
wire w1 ;
wire w2 ;
wire w3 ;
wire f0 ;
wire f1 ;
wire f2 ;
wire g0 ;
wire g1 ;
wire g2 ;
wire g3 ;
wire g4 ;
wire g5 ;
wire g6 ;
wire g7 ;
wire g8 ;
wire g9 ;
wire g10 ;
wire o0 ;
wire o1 ;
wire w4 ;
wire w ;
wire f3 ;
wire f ;
wire g11 ;
wire g12 ;
wire g13 ;
wire g ;
nand #( 3 , 3 ) GaBAR ( aBAR , a , a );
```

```
nand #( 3 , 3 ) GbBAR ( bBAR , b , b );
nand #( 3 , 3 ) GcBAR ( cBAR , c , c );
nand #( 3 , 3 ) GdBAR ( dBAR , d , d );
nand #( 3 , 3 ) GwBAR ( wBAR , w , w );
nand #( 3 , 3 ) GsBAR ( sBAR , s , s );
and #( 5 , 5 ) Gw0 ( w0 , bBAR , cBAR );
and #( 5 , 5 ) Gw1 ( w1 , c , dBAR );
and #( 5 , 5 ) Gw2 ( w2 , aBAR , b );
and #( 5 , 5 ) Gw3 ( w3 , d , w2 );
and #( 5 , 5 ) Gf0 ( f0 , c , dBAR );
and #( 5 , 5 ) Gf1 ( f1 , w , bBAR );
and #( 5 , 5 ) Gf2 ( f2 , w , c );
and #( 5 , 5 ) Gg0 ( g0 , aBAR , bBAR );
and #( 5 , 5 ) Gg1 ( g1 , cBAR , dBAR );
and #( 5 , 5 ) Gg2 ( g2 , g0 , g1 );
and #( 5 , 5 ) Gg3 ( g3 , aBAR , b );
and #( 5 , 5 ) Gg4 ( g4 , c , g3 );
and #( 5 , 5 ) Gg5 ( g5 , a , bBAR );
and #( 5 , 5 ) Gg6 ( g6 , c , g5 );
and #( 5 , 5 ) Gg7 ( g7 , b , cBAR );
and #( 5 , 5 ) Gg8 ( g8 , d , g7 );
and #( 5 , 5 ) Gg9 ( g9 , aBAR , c );
and #( 5 , 5 ) Gg10 ( g10 , d , g9 );
and #( 5 , 5 ) Go0 ( o0 , s , g );
and #( 5 , 5 ) Go1 ( o1 , sBAR , f );
or #( 5 , 5 ) Gw4 ( w4 , w0 , w1 );
or #( 5 , 5 ) Gw ( w , w3 , w4 );
or #( 5 , 5 ) Gf3 ( f3 , f0 , f1 );
or #( 5 , 5 ) Gf ( f , f2 , f3 );
or #( 5 , 5 ) Gg11 ( g11 , g2 , g4 );
```

```
or #( 5 , 5 ) Gg12 ( g12 , g6 , g8 );

or #( 5 , 5 ) Gg13 ( g13 , g10 , g11 );

or #( 5 , 5 ) Gg ( g , g12 , g13 );

or #( 5 , 5 ) Go ( o , o0 , o1 );

endmodule
```

تست وکتور های داده شده به این کد به صورت زیر است:

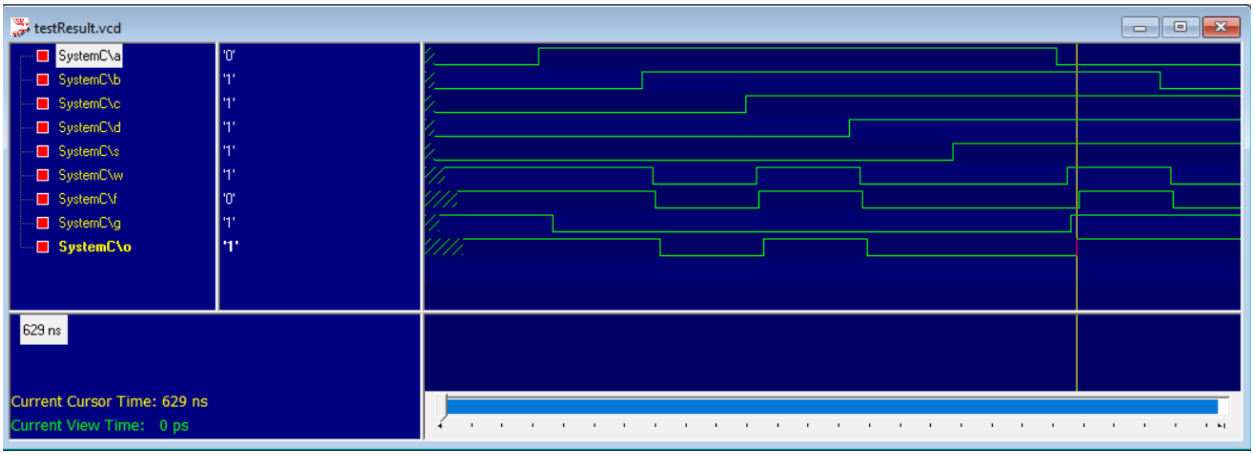
```
#10 00000
#100 10000
#100 11000
#100 11100
#100 11110
#100 11111
#100 01111
#100 00111
```

خروجی تولید شده در کد تمرین کامپیوتری ۱ به صورت زیر است:

```
name:c  ->   value:0  time:10  /    value:1  time:310
name:d  ->   value:0  time:10  /    value:1  time:410
name:s  ->   value:0  time:10  /    value:1  time:510
name:aBAR  ->   value:1  time:13  /    value:0  time:113  /    value:1  time:613
name:bBAR  ->   value:1  time:13  /    value:0  time:213  /    value:1  time:713
name:cBAR  ->   value:1  time:13  /    value:0  time:313
name:dBAR  ->   value:1  time:13  /    value:0  time:413
name:sBAR  ->   value:1  time:13  /    value:0  time:513
name:w0  ->   value:1  time:18  /    value:0  time:218
name:w1  ->   value:0  time:15  /    value:1  time:315  /    value:0  time:418
name:w2  ->   value:0  time:15  /    value:1  time:618  /    value:0  time:715
name:w3  ->   value:0  time:15  /    value:1  time:623  /    value:0  time:720
name:f0  ->   value:0  time:15  /    value:1  time:315  /    value:0  time:418
name:g0  ->   value:1  time:18  /    value:0  time:118  /    value:1  time:718
name:g1  ->   value:1  time:18  /    value:0  time:318
name:g2  ->   value:1  time:23  /    value:0  time:123
name:g3  ->   value:0  time:15  /    value:1  time:618  /    value:0  time:715
name:g4  ->   value:0  time:15  /    value:1  time:623  /    value:0  time:720
name:g5  ->   value:0  time:15  /    value:1  time:115  /    value:0  time:218
name:g6  ->   value:0  time:15
name:g7  ->   value:0  time:15  /    value:1  time:215  /    value:0  time:318
name:g8  ->   value:0  time:15
name:g9  ->   value:0  time:15  /    value:1  time:618
name:g10  ->   value:0  time:15  /    value:1  time:623
name:w4  ->   value:1  time:23  /    value:0  time:223  /    value:1  time:320  /    value:0  time:423
name:w  ->   value:1  time:28  /    value:0  time:228  /    value:1  time:325  /    value:0  time:428  /    value:1  time:628  /    value:0  time:725
name:g11  ->   value:1  time:28  /    value:0  time:128  /    value:1  time:628  /    value:0  time:725
name:g12  ->   value:0  time:20
name:g13  ->   value:1  time:33  /    value:0  time:133  /    value:1  time:628
name:g  ->   value:1  time:38  /    value:0  time:138  /    value:1  time:633
name:wBAR  ->   value:0  time:31  /    value:1  time:231  /    value:0  time:328  /    value:1  time:431  /    value:0  time:631  /    value:1  time:728
name:f1  ->   value:1  time:33  /    value:0  time:218  /    value:1  time:718  /    value:0  time:730
name:f2  ->   value:0  time:15  /    value:1  time:330  /    value:0  time:433  /    value:1  time:633  /    value:0  time:730
name:o0  ->   value:0  time:15  /    value:1  time:638
name:f3  ->   value:1  time:38  /    value:0  time:223  /    value:1  time:320  /    value:0  time:423  /    value:1  time:723  /    value:0  time:735
name:f  ->   value:1  time:43  /    value:0  time:228  /    value:1  time:325  /    value:0  time:438  /    value:1  time:638  /    value:0  time:740
name:o1  ->   value:1  time:48  /    value:0  time:233  /    value:1  time:330  /    value:0  time:443
name:o  ->   value:1  time:53  /    value:0  time:238  /    value:1  time:335  /    value:0  time:448  /    value:1  time:643
```

**نگین سفاری ۸۱۰۱۹۷۵۲۵**
**گزارش کار تمرین کامپیوتری ۳**

خروجی مربوط به systemC به ازای همین تست وکتور ها:




همانطور که انتظار می رفت تنها تفاوت این دو سیمیولیشن در تاخیر است. برای نوشتن کد sv در بخش cpp باید از گیت های ماکسیمم ۲ ورودی با تاخیر های داده شده ی ۵ برای and و or و ۳ برای not استفاده شود که در نتیجه ی ترکیب و سری شدن آن ها تاخیر بیشتر شده است. اما در systemC دقیقا با همان دیلی داده شده در assign ها خروجی ها تولید شده اند.