به نام خدا



**پروژه پایان ترم آزمایشگاه مدار منطقی**

نام و نام خانوادگی گردآورندگان:

حسنا اویارحسینی - 9823010

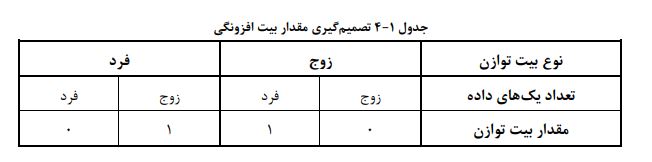
صبا فضلعلی - 9831116

استاد آزمایشگاه: جناب آقای امامیان وردی

تاریخ پروژه: 4/1400

**توضیحات مربوط به ماژول شمار 1)**

در بخش اول این ماژول برایی طراحی افزونگی بیت توازن از XOR استفاده شده است ، زیرا خروجی XOR زمانی که تعداد یک ها در بیت های XOR شده فرد باشد یک و در غیر اینصورت خروجی صفر میشود که این ویژگی به ما مدار مورد نیاز برای طراحی افزونگی بیت توازن فرد را میدهد.



در قسمت دوم نیز ابتدا تابع مورد نظر(تابعی که به ازای مینترم های 0تا 7 و 23 تا 31 خروجی یک را بدهد) به کمک کارنو محاسبه کرده و سپس این تابع را به کمک گیت های پایه and و or و not به صورت SOP پیاده سازی کرده ایم:

جدول ارزش :

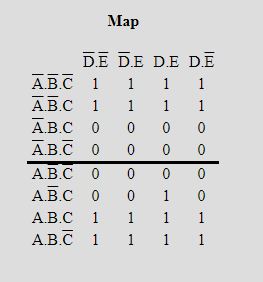
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| عدد معادل دسیمال | pData[0]  A | pData[1]  B | pData[2]  C | pData[3]  D | pData[4]  E | f | minterm |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | A’B’C’D’E’ |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | A’B’C’D’E |
| 2 | 0 | 0 | 0 | 1 | 0 | 1 | A’B’C’DE’ |
| 3 | 0 | 0 | 0 | 1 | 1 | 1 | A’B’C’DE |
| 4 | 0 | 0 | 1 | 0 | 0 | 1 | A’B’CD’E’ |
| 5 | 0 | 0 | 1 | 0 | 1 | 1 | A’B’CD’E |
| 6 | 0 | 0 | 1 | 1 | 0 | 1 | A’B’CDE’ |
| 7 | 0 | 0 | 1 | 1 | 1 | 1 | A’B’CDE |
| 8 | 0 | 1 | 0 | 0 | 0 | 0 | - |
| 9 | 0 | 1 | 0 | 0 | 1 | 0 | - |
| 10 | 0 | 1 | 0 | 1 | 0 | 0 | - |
| 11 | 0 | 1 | 0 | 1 | 1 | 0 | - |
| 12 | 0 | 1 | 1 | 0 | 0 | 0 | - |
| 13 | 0 | 1 | 1 | 0 | 1 | 0 | - |
| 14 | 0 | 1 | 1 | 1 | 0 | 0 | - |
| 15 | 0 | 1 | 1 | 1 | 1 | 0 | - |
| 16 | 1 | 0 | 0 | 0 | 0 | 0 | - |
| 17 | 1 | 0 | 0 | 0 | 1 | 0 | - |
| 18 | 1 | 0 | 0 | 1 | 0 | 0 | - |
| 19 | 1 | 0 | 0 | 1 | 1 | 0 | - |
| 20 | 1 | 0 | 1 | 0 | 0 | 0 | - |
| 21 | 1 | 0 | 1 | 0 | 1 | 0 | - |
| 22 | 1 | 0 | 1 | 1 | 0 | 0 | - |
| 23 | 1 | 0 | 1 | 1 | 1 | 1 | AB’CDE |
| 24 | 1 | 1 | 0 | 0 | 0 | 1 | ABC’D’E’ |
| 25 | 1 | 1 | 0 | 0 | 1 | 1 | ABC’D’E |
| 26 | 1 | 1 | 0 | 1 | 0 | 1 | ABC’DE’ |
| 27 | 1 | 1 | 0 | 1 | 1 | 1 | ABC’DE |
| 28 | 1 | 1 | 1 | 0 | 0 | 1 | ABCD’E’ |
| 29 | 1 | 1 | 1 | 0 | 1 | 1 | ABCD’E |
| 30 | 1 | 1 | 1 | 1 | 0 | 1 | ABCDE’ |
| 31 | 1 | 1 | 1 | 1 | 1 | 1 | ABCDE |

جدول کارنو مربوط به تابع:



AB

B’CDE



با توجه به جدول کارنو خواهیم داشت:

F = A’B’ + AB + B’CDE

و مدار پیاده سازی شده به شکل زیر خواهد بود:



**توضیحات مربوط به ماژول شمار 2)**

در طراحی ماژول دوم ابتدا با توجه به کد مربوط به هر نوع گروه خونی، گروه خونی های O , B را به کمک یک ماکس 4 در 1 که سلکتور های آن بیت های پر ارزش کد هستند جدا کرده ایم برای این منظور ابتدا رسم جدول درستی را رسم کرده و دو سطر دو سطر باهم درنظر میگیریم و مقدار دو سطر مربوطه را بر حسب بیت کم ارزش(در اینجا Z) محاسبه کرده ایم تا به عنوان ورودی های ماکس از آن استفاده کنیم.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| کد مربوطه | | | گروه خونی | خروجی | ورودی ماکس با سلکتور X,Y | شماره ورودی ماکس |
| X | Y | Z |  |  |  |  |
| 0 | 0 | 0 | AB+ | 0 | 0 | D0 |
| 0 | 0 | 1 | AB- | 0 |
| 0 | 1 | 0 | A+ | 0 | 0 | D1 |
| 0 | 1 | 1 | A- | 0 |
| 1 | 0 | 0 | B+ | 1 | 1 | D2 |
| 1 | 0 | 1 | B- | 1 |
| 1 | 1 | 0 | O+ | 1 | 1 | D3 |
| 1 | 1 | 1 | O- | 1 |

تابع خروجی:

**BloodTypeClassification**

F(X,Y, Z) = X

d0

d1

d2

d3

0

0

1

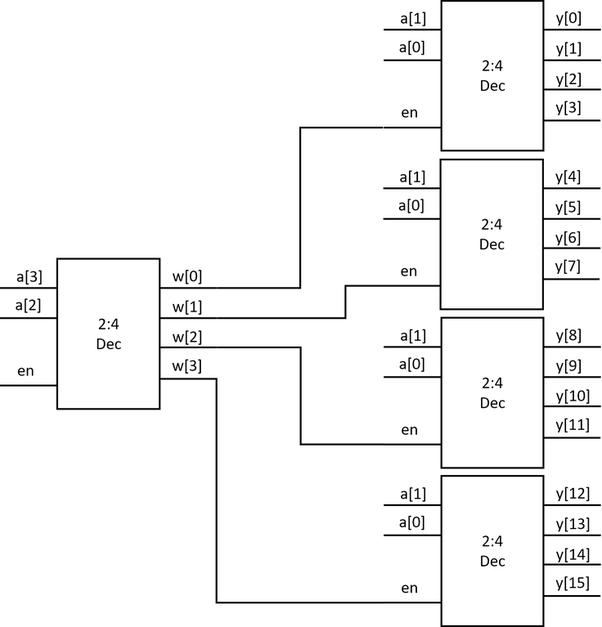
1

F

MSD LSD

X Y

در مرحله دوم برای طراحی مدار تشخیص دهنده ی طبیعی بودن غلظتPH خون از یک دیکودر 4 در 16 (دیکودر 4 در16به کمک 5 دیکودر 2 در 4 که هر یک به صورت گیت لول تعریف شده اند طراحی شده است)و یک ماکس 2 در 1 به صورت زیر استفاده کرده ایم:



Abnormality

P

Abnormality

Q

MUX 2X1

+

+

BloodPHAnalyzer

خروجی بخش اول ماژول دو به عنوان سلکتور در این مرحله استفاده شده است

مالتی پلکسر 4 در 1 و دیکودر 2در 4 به صورت گیت لول همانند طراحی زیر پیاده سازی شده اند:

|  |  |
| --- | --- |
| C:\Users\RAI\Desktop\Picture1.png | C:\Users\RAI\Desktop\de.JPG |
| Mux 4x1 | Decoder 2x4 |

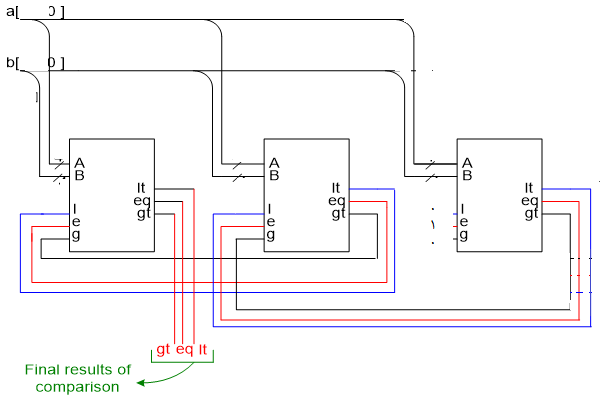
و ماکس 2در1 بخش دوم نیز به صورت dataflow به کمک عملگر" ؟: "تعریف شده است:

assign OUT = select ? A : B

**توضیحات مربوط به ماژول شمار3)**

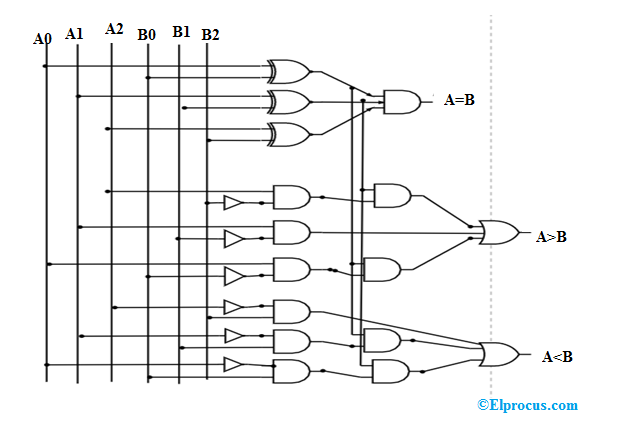
در ماژول 3 نیاز به یک مقایسه کننده 8 بیت داریم، برای ساخت این مقایسه کننده از سه مقایسه کننده سه بیت به صورت زیر استفاده شده است، یه صورتی که بیت 0تا2 دو عدد ابتدا در مقایسه کننده اول با ورودی های l,e,g باربرا با 0,1,0 (سمت راست) مقایسه میشوند و حاصل (اگر A>B باشد gt = 1 و اگر A<B باشد lt = 1 و در غیر اینصورت eq =1 ) به عنوان ورودی های l,e,g به مقایسه کننده بعدی فرستاده میشود و در مقایسه ککنده دوم بیت های 3تا5 مقایسه شدند و خروجی متناسب با ورودیl,e,g میدهد و به همین صورت مقایسه کننده سوم بیت های 6و7 ام را در هر دو عدد باهم مقایسه میکند و با توجه به تاثیری که مقایسه کننده های قبلی رویl,e,g گذاشته اند خروجی نهایی نشان دهنده وضعیت بزرگتر/کوچکتر/مساوی دو عدد 8 بیت اولیه می باشد.( در اینجا با توجه به اینکه اعداد ما 8 بیت هستند ولی ظرفیت مقایسه کننده ما 9 بیت است بیت پر ارزش هر دو عدد صفر در نظر گرفته شده است).

7

****

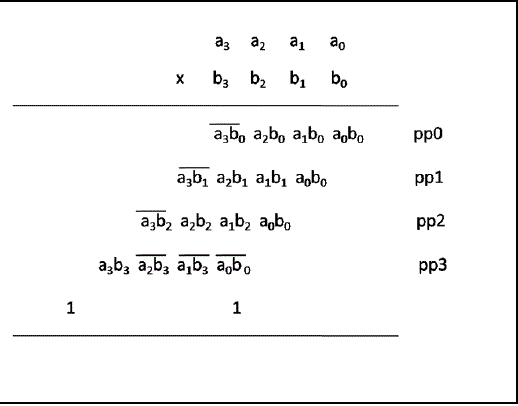
7

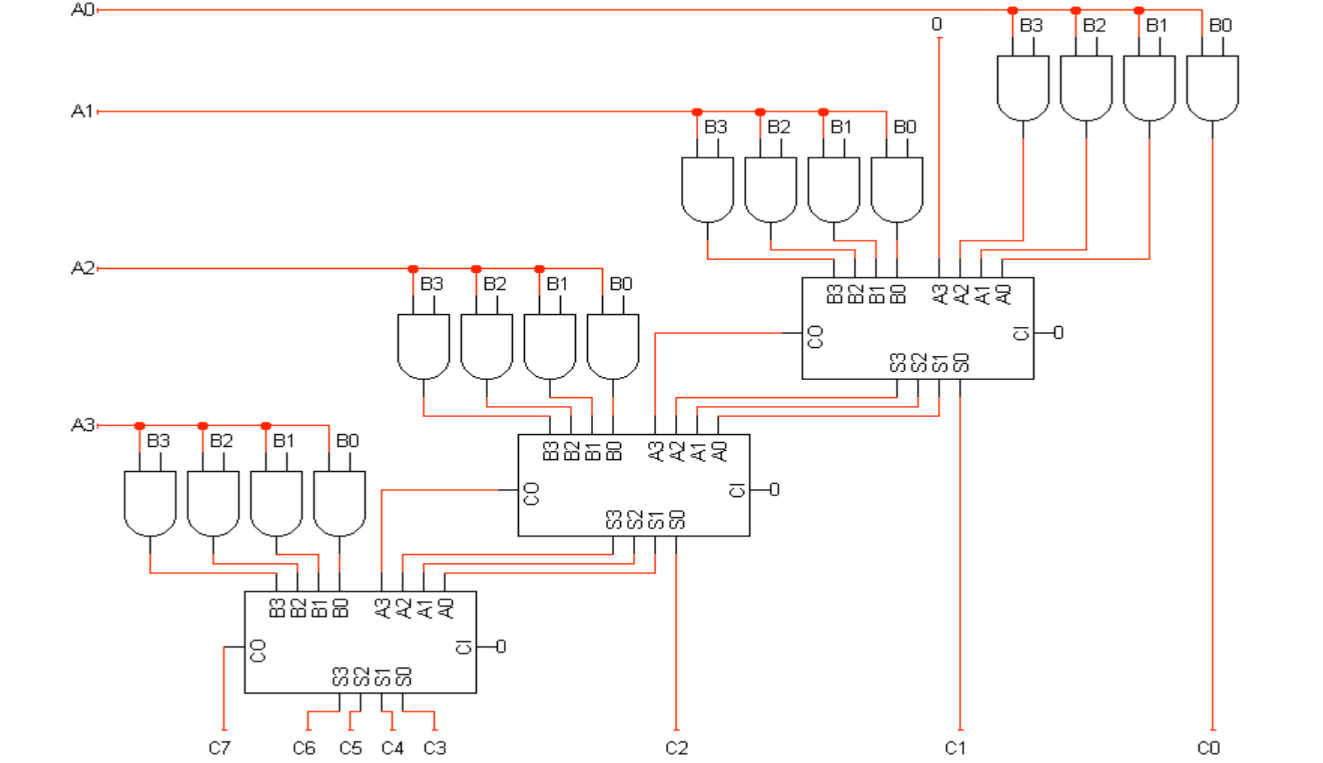
مقایسه کننده های سه بیتی به صورت گیت لول همانند شکل زیر پیاده سازی شده اند:



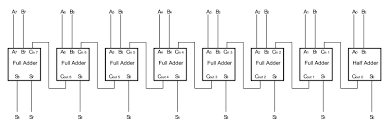
**توضیحات مربوط به ماژول شمار4)**

در این بخش نیاز به یک multiplier 4x4 داریم برای این منظور از سه adder 4 bit استفاده کرده ایم و با توجه به مراحل ضرب و این سه جمع کننده به صورت زیر مدار را پیاده سازی کرده ایم(در ضرب انجام شده در جمع انجام شده در هر مستطیل مشخص شده مربوط به یکی از سه جمع کننده می باشد):



****

و در بخش دوم این ماژول نیازمند یک adder 8 بیت هستیم، ما این جمع کننده را به وسیله 8 full adder به صورت زیر پیاده سازی کرده ایم:



هر fulladderنیز به صورت گیت لول پیاده شده است:

C:\Users\RAI\Desktop\Picture2.wmf

**توضیحات مربوط به ماژول شمار 5)**

در این بخش ابتدا مکمل دو یک عدد را با توجه به رابطه آن برای یک عدد 8 بیت به صورت زیر پیاده سازی کرده ایم:

100000000 – N = 2’s complement

سپس با ماکس 2 در 1، 8 بیت که به کمک 8 ماکس 2 در یک طراحی شده (مشابه شکل زیر که دوبرابر شده باشد)و سلکتور آن بیت پر ارزش عدد است(که نشان دهنده این است که عدد مثبت است یا منفی) تصمیم میگیریم که چه عددی را به عنوان قدر مطلق اعلام کنیم:

و سپس با جمع کردن بیت های عدد بدست امده تعداد یک های موجود در قدرمطلق اولین عدد بدست میاد.



2:1

MUX

I

0

I

1

A

Z



2:1

MUX

I

0

I

1

A

Z



2:1

MUX

I

0

I

1

A

Z



2:1

MUX

I

0

I

1

Z

**توضیحات مربوط به ماژول شمار7)**

در این ماژول در بخش اول کنترل را به صورت behavioral و باتوجهبه دیاگرام حالت زیر پیاده سازی میکنیم و سپس در controller unit به کمک کنترلر مقدار enableها رامشخص کرده و با توجه به آنها اطلاعات را در رجیستر مد نظر ذخیره میکنیم.

1,X,0/00

0,X,X/00

0,X,X/00

A

بیکار

1,wrongPass,1/00

E

تله

B

فعال

1,X,X/00

0,X,X/00

1,X,X/00

0,X,X/00

0,X,,X/00

1,correct Pass,1/00

1,[0]data,1/10

C

درخواست

D

ذخیره

1,[1]data,1/01

Request, password\_data[8bit], confirm ترتیب ورودی:

EnableP,EnableQ ترتیب خروجی: