به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

پروژه پایان ترم آزمایشگاه مدار منطقی

نام و نام خانوادگی گردآورندگان:

حسنا اویارحسینی – ۹۸۲۳۰۱۰

صبا فضلعلی – ۹۸۳۱۱۱۶

استاد آزمایشگاه: جناب آقای امامیان وردی

تاریخ پروژه: ۱۴۰۰/۴

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.

جدول ۱-۴ تصمیم‌گیری مقدار بیت افزونگی

نوع بیت توازن	زوج		فرد	
تعداد یک‌های داده	زوج	فرد	زوج	فرد
مقدار بیت توازن	۰	۱	۱	۰

در قسمت دوم نیز ابتدا تابع مورد نظر(تابعی که به ازای مینترم های ۰تا ۷ و ۲۳ تا ۳۱ 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	۱	A’B’C’D’E’
1	0	0	0	0	1	۱	A’B’C’D’E
2	0	0	0	1	0	۱	A’B’C’DE’
3	0	0	0	1	1	۱	A’B’C’DE
4	0	0	1	0	0	۱	A’B’CD’E’
5	0	0	1	0	1	۱	A’B’CD’E
6	0	0	1	1	0	۱	A’B’CDE’
7	0	0	1	1	1	۱	A’B’CDE
8	0	1	0	0	0	۰	-
9	0	1	0	0	1	۰	-
10	0	1	0	1	0	۰	-
11	0	1	0	1	1	۰	-
12	0	1	1	0	0	۰	-
13	0	1	1	0	1	۰	-

14	0	1	1	1	0	۰	-
15	0	1	1	1	1	۰	-
16	1	0	0	0	0	۰	-
17	1	0	0	0	1	۰	-
18	1	0	0	1	0	۰	-
19	1	0	0	1	1	۰	-
20	1	0	1	0	0	۰	-
21	1	0	1	0	1	۰	-
22	1	0	1	1	0	۰	-
23	1	0	1	1	1	۱	AB’CDE
24	1	1	0	0	0	۱	ABC’D’E’
25	1	1	0	0	1	۱	ABC’D’E
26	1	1	0	1	0	۱	ABC’DE’
27	1	1	0	1	1	۱	ABC’DE
28	1	1	1	0	0	۱	ABCD’E’
29	1	1	1	0	1	۱	ABCD’E
30	1	1	1	1	0	۱	ABCDE’
31	1	1	1	1	1	۱	ABCDE

جدول کارنو مربوط به تابع:

AB \ CDE	000	001	011	010	110	111	101	100
00	1	1	1	1	1	1	1	1
01	0	0	0	0	0	0	0	0
11	1	1	1	1	1	1	1	1
10	0	0	0	0	0	1	0	0

A'B'

AB

B’CDE

Map

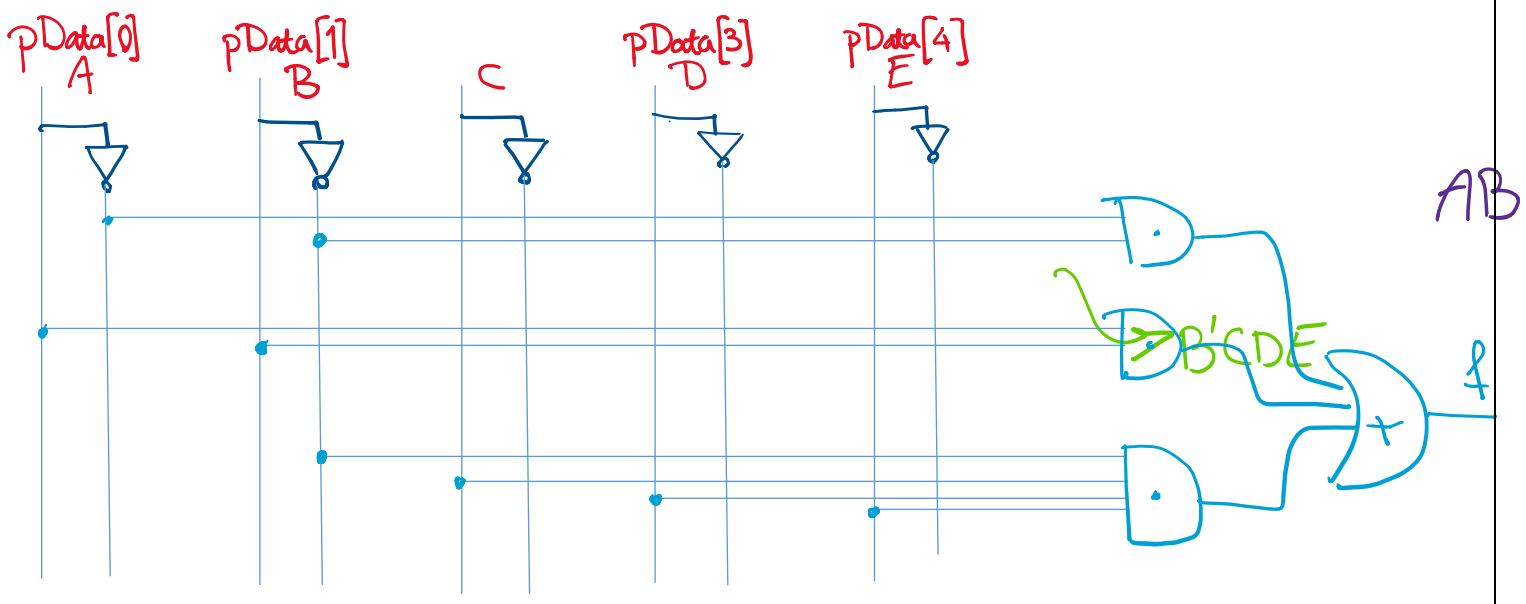
	$\overline{D}.\overline{E}$	$\overline{D}.E$	D.E	D.$\overline{E}$
$\overline{A}.\overline{B}.C$	1	1	1	1
$\overline{A}.\overline{B}.C$	1	1	1	1
$\overline{A}.B.C$	0	0	0	0
$\overline{A}.B.\overline{C}$	0	0	0	0
A.$\overline{B}$.$\overline{C}$	0	0	0	0
A.$\overline{B}$.C	0	0	1	0
A.B.C	1	1	1	1
A.B.$\overline{C}$	1	1	1	1

با توجه به جدول کارنو خواهیم داشت:

$F = A'B' + AB + B'CDE$

$$\sum_{i=0}^{7} m_i + \sum_{i=23}^{31} m_i$$ جمع مینترم

و مدار پیاده سازی شده به شکل زیر خواهد بود:



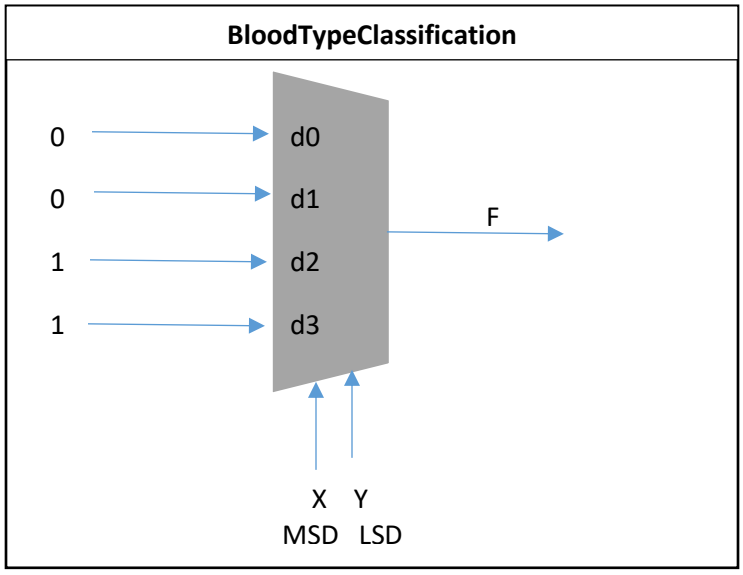
توضیحات مربوط به ماژول شمار ۲)

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 ۴ در ۱ 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.

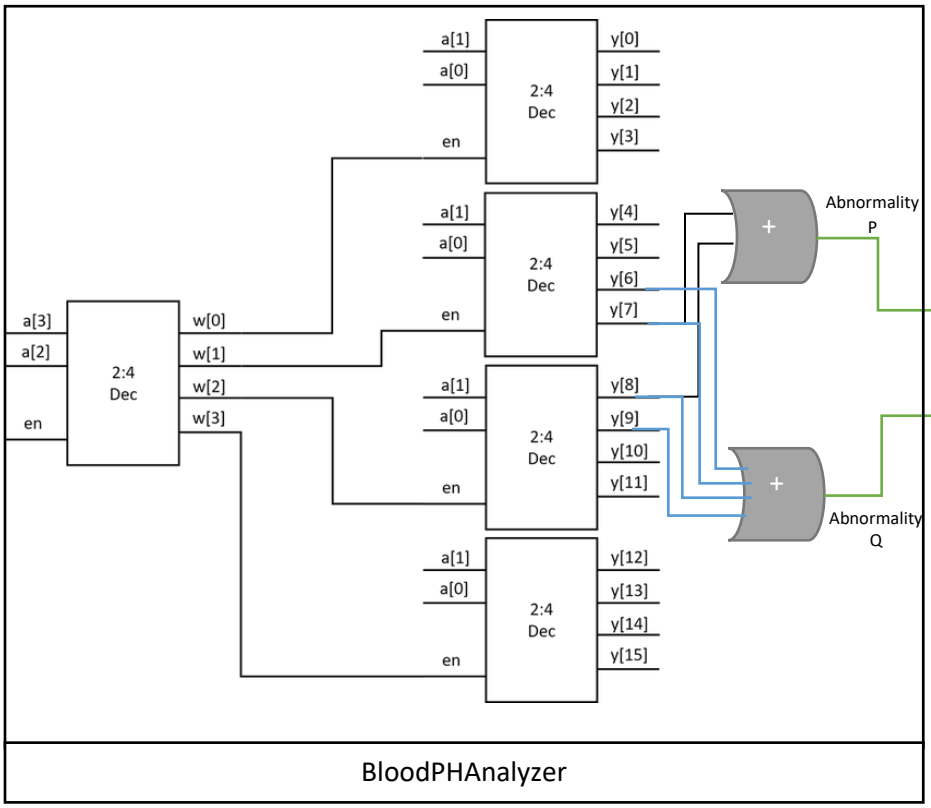
کد مربوطه			گروه خونی	خروجی	ورودی ماکس با سلکتور X,Y	شماره ورودی ماکس
X	Y	Z				
0	0	0	AB+	0	0	D0
0	0	1	AB-	0		
0	1	0	A+	0	0	D1
0	1	1	A-	0		
1	0	0	B+	1	1	D2
1	0	1	B-	1		
1	1	0	O+	1	1	D3
1	1	1	O-	1		

تابع خروجی:

F(X,Y, Z) = X



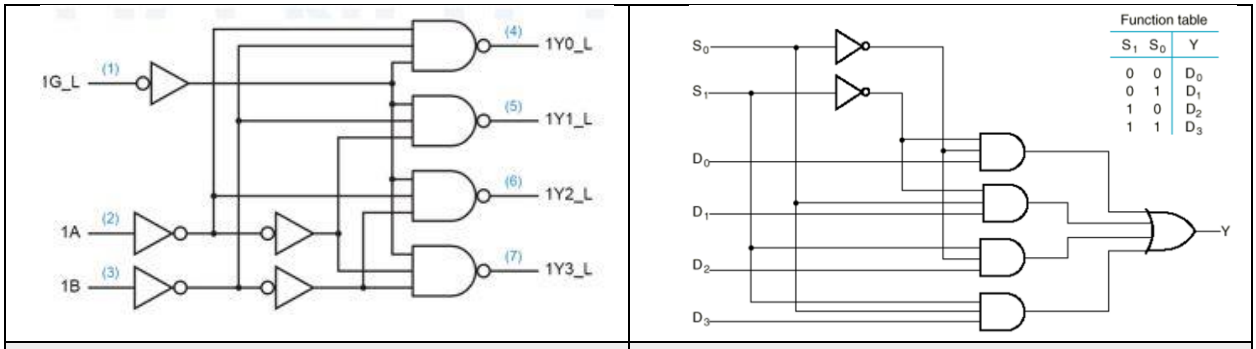
در مرحله دوم برای طراحی مدار تشخیص دهنده ی طبیعی بودن غلظت PHخون از یک دیکودر ۴ در ۱۶ (دیکودر ۴ در۱۶به کمک ۵ دیکودر ۲ در ۴ که هر یک به صورت گیت لول تعریف شده اند طراحی شده است)و یک ماکس ۲ در ۱ به صورت زیر استفاده کرده ایم:



BloodPHAnalyzer

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

مالتی پلکسر ۴ در ۱ و دیکودر ۲در ۴ به صورت گیت لول همانند طراحی زیر پیاده سازی شده اند:



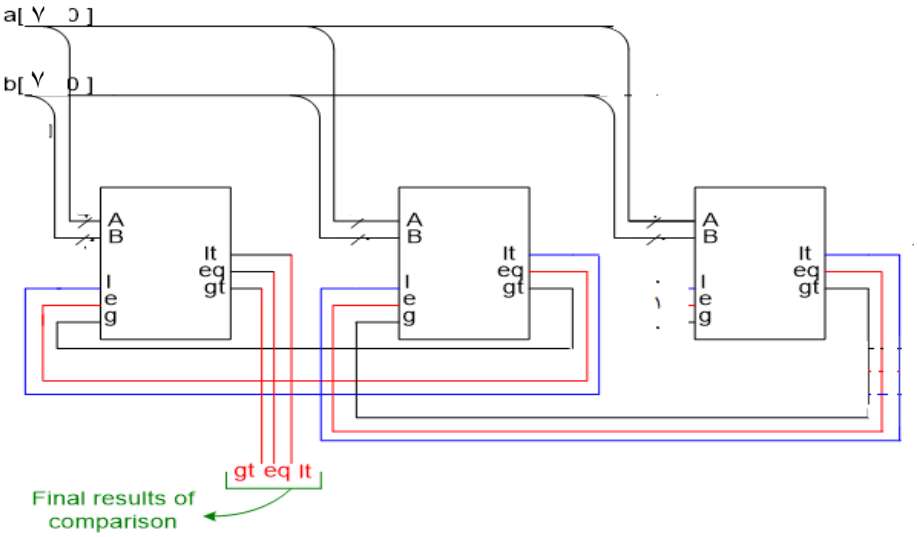
Decoder 2x4	Mux 4x1

و ماکس ۲در۱ بخش دوم نیز به صورت dataflow به کمک عملگر" ؟: "تعریف شده است:

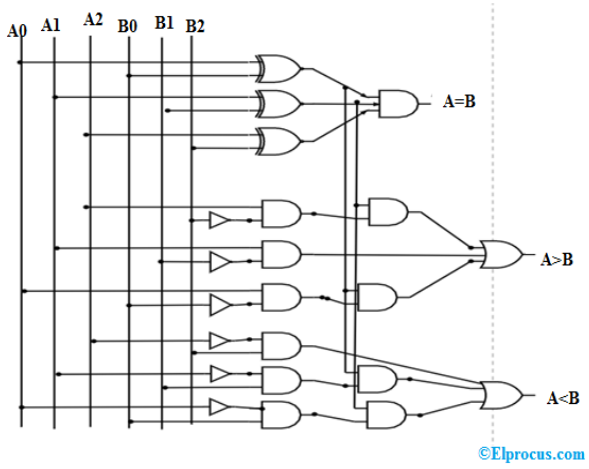
```
assign OUT = select ? A : B
```

توضیحات مربوط به ماژول شماره ۳)

در ماژول ۳ نیاز به یک مقایسه کننده ۸ بیت داریم، برای ساخت این مقایسه کننده از سه مقایسه کننده سه بیت به صورت زیر استفاده شده است، به صورتی که بیت ۰تا۲ 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 ۳تا۵ مقایسه شدند و خروجی متناسب با ورودیl,e,g میدهد و به همین صورت مقایسه کننده سوم بیت های ۶و۷ 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 ۸ بیت اولیه می باشد.(در اینجا با توجه به اینکه اعداد ما ۸ بیت هستند ولی ظرفیت مقایسه کننده ما ۹ بیت است بیت پر ارزش هر دو عدد صفر در نظر گرفته شده است).



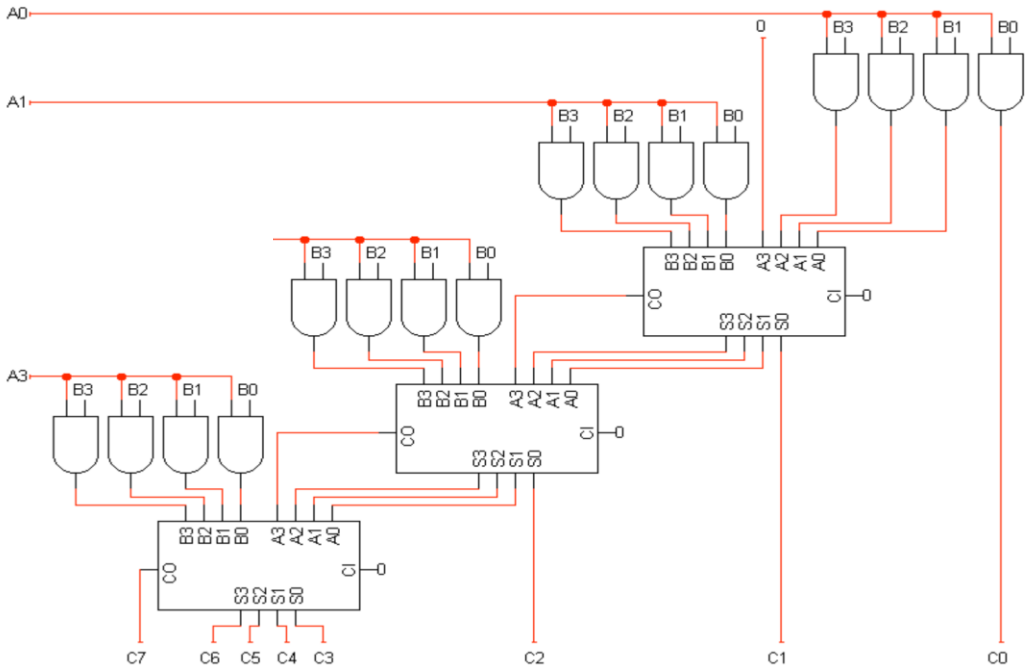
مقایسه کننده های سه بیتی به صورت گیت لول همانند شکل زیر پیاده سازی شده اند:



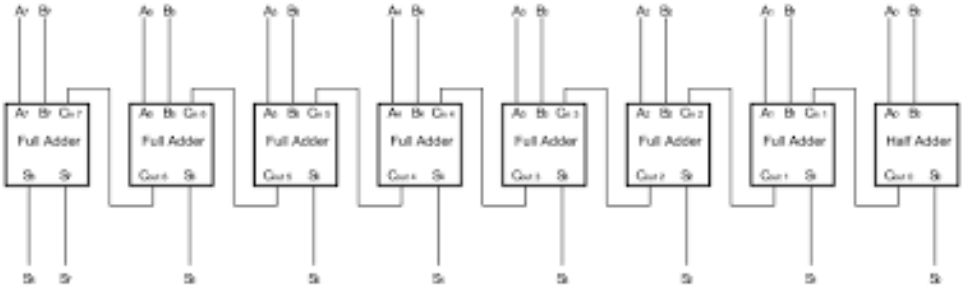
توضیحات مربوط به ماژول شماره۴)

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):

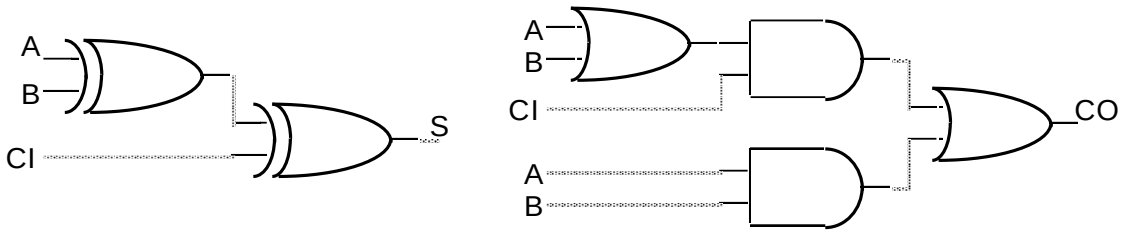
$$\begin{array}{cccccc} & a_3 & a_2 & a_1 & a_0 \\ x & b_3 & b_2 & b_1 & b_0 \\ \hline & \overline{a_3b_0} & a_2b_0 & a_1b_0 & a_0b_0 \\ & \overline{a_3b_1} & a_2b_1 & a_1b_1 & a_0b_0 \\ & \overline{a_3b_2} & a_2b_2 & a_1b_2 & a_0b_0 \\ a_3b_3 & \overline{a_2b_3} & \overline{a_1b_3} & \overline{a_0b_0} & \end{array}$$



و در بخش دوم این ماژول نیازمند یک ۸adder بیت هستیم، ما این جمع کننده را به وسیله ۸ full adder به صورت زیر پیاده سازی کرده ایم:



هر fulladderنیز به صورت گیت لول پیاده شده است:



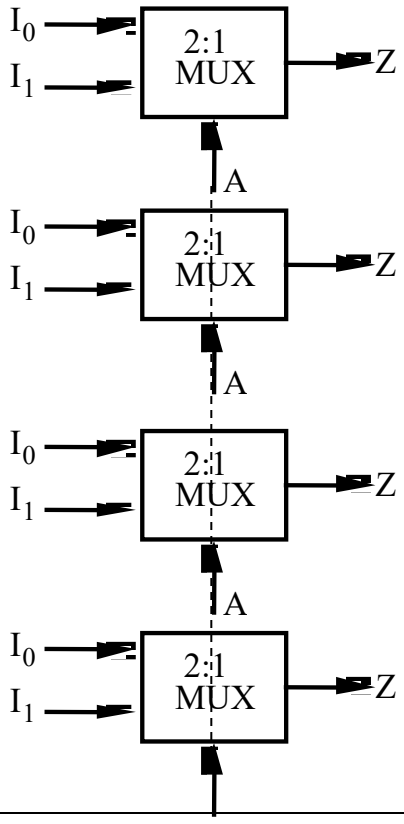
توضیحات مربوط به ماژول شمار ۵)

در این بخش ابتدا مکمل دو یک عدد را با توجه به رابطه آن برای یک عدد ۸ بیت به صورت زیر پیاده سازی کرده ایم:

100000000 – N = 2’s complement

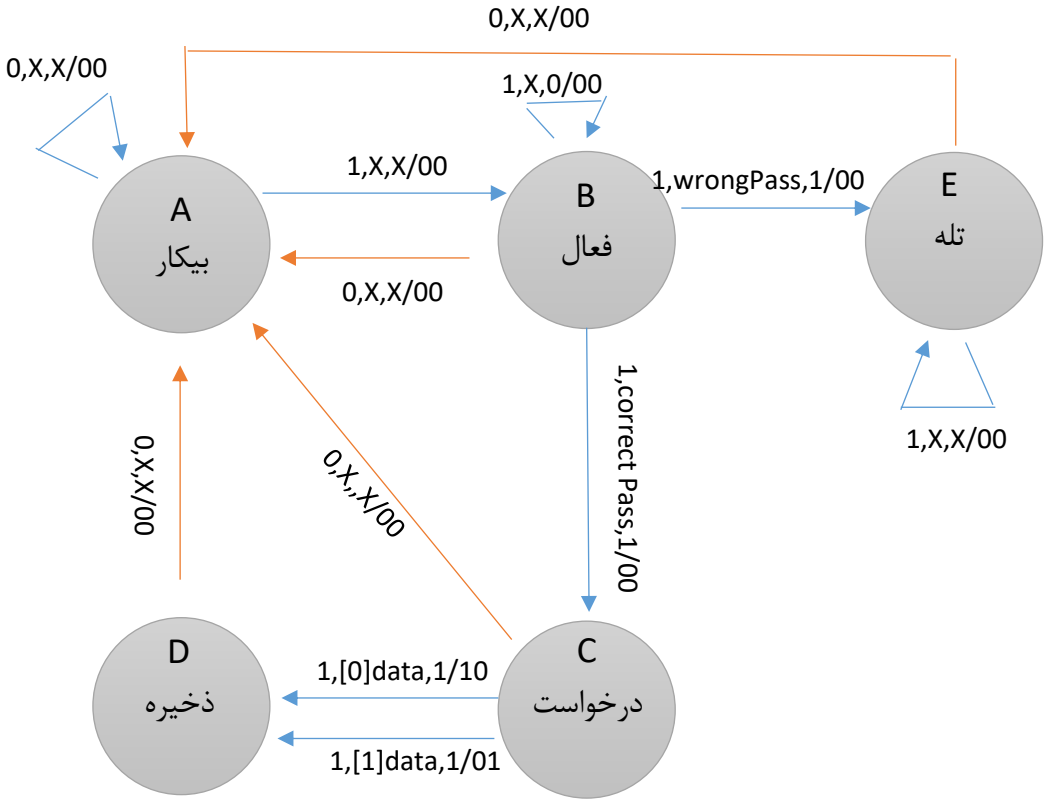
سپس با ماکس ۲ در ۱، ۸ بیت که به کمک ۸ ماکس ۲ در یک طراحی شده (مشابه شکل زیر که دوبرابر شده باشد)و سلکتور آن بیت پر ارزش عدد است(که نشان دهنده این است که عدد مثبت است یا منفی) تصمیم میگیریم که چه عددی را به عنوان قدر مطلق اعلام کنیم:

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



توضیحات مربوط به ماژول شماره ۷)

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



ترتیب ورودی: Request, password_data[8bit], confirm

ترتیب خروجی: EnableP,EnableQ