Logical Circuit Final Project Report
Mohammad Khoddam
97243029

Problem1

- نحوه حل مسئله

برای حل این سوال باید تابع F را به صورت مداری پیاده‌سازی کنیم.
می‌توانیم از طراحی CMOS کمک بگیریم ؛به این منظور توابع $\bar{F}$ و $D(\bar{F})$ را بدست آوریم.

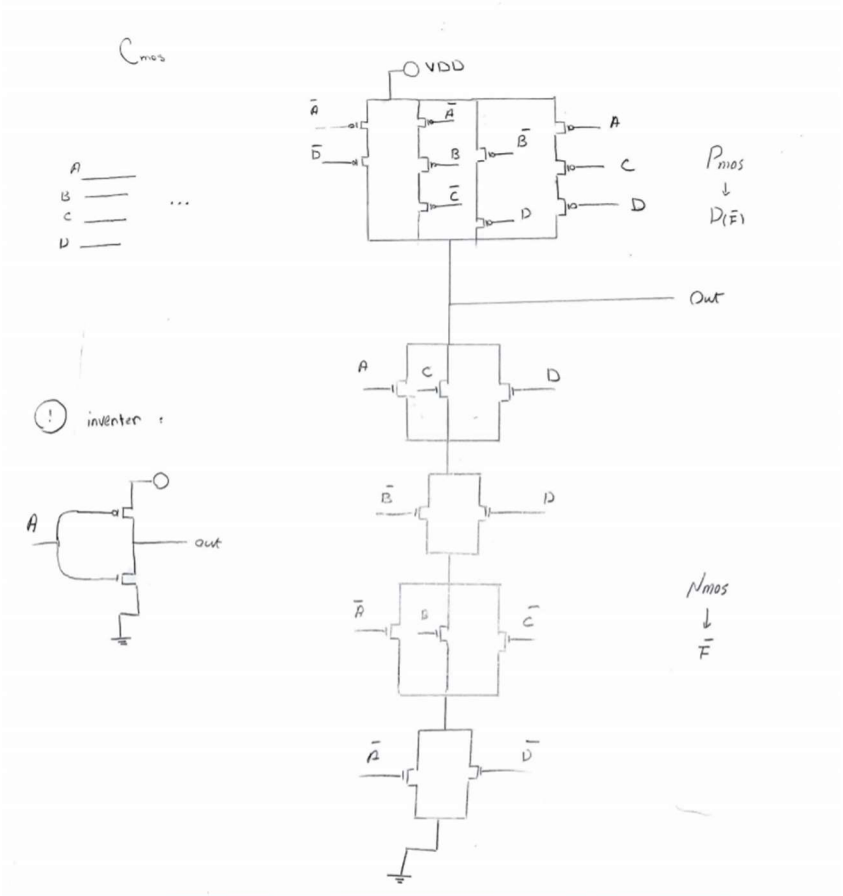
توابع به شکل زیر محاسبه می‌شوند:

$$F = A.D + A.\bar{B}.C + B.\bar{D} + \bar{A}.\bar{C}.\bar{D}$$

$$\bar{F} = (\bar{A} + \bar{D}).(\bar{A} + B + \bar{C}).(\bar{B} + D).(A + C + D)$$

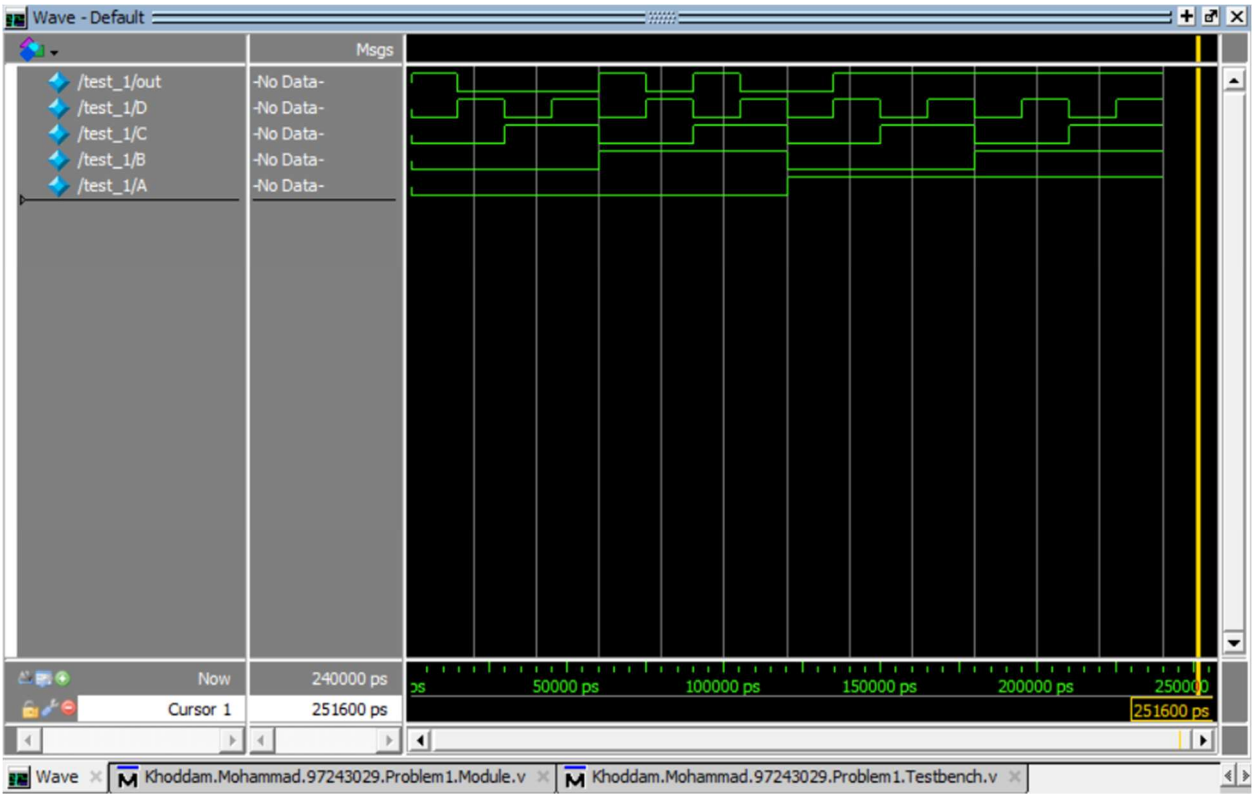
$$D(\bar{F}) = \bar{\mathrm{A}}.\bar{\mathrm{D}} + \bar{\mathrm{A}}.\mathrm{B}.\bar{\mathrm{C}} + \bar{\mathrm{B}}.\mathrm{D} + \mathrm{A}.\mathrm{C}.\mathrm{D}$$

مدار به شکل زیر پیاده‌سازی می‌شود: (inverter ها هم با cmos پیاده‌سازی شده)



Logical Circuit Final Project Report
Mohammad Khoddam
97243029

- سیگنال‌های شبیه‌سازی



Logical Circuit Final Project Report
Mohammad Khoddam
97243029

- اثبات صحت عملکرد

جدول درستی مدار به شکل زیر می‌باشد:

A	B	C	D	((A ∧ D) ∨ ((A ∧ (¬B ∧ C)) ∨ ((B ∧ ¬D) ∨ (¬A ∧ (¬C ∧ ¬D)))))
F	F	F	F	**T**
F	F	F	T	**F**
F	F	T	F	**F**
F	F	T	T	**F**
F	T	F	F	**T**
F	T	F	T	**F**
F	T	T	F	**T**
F	T	T	T	**F**
T	F	F	F	**F**
T	F	F	T	**T**
T	F	T	F	**T**
T	F	T	T	**T**
T	T	F	F	**T**
T	T	F	T	**T**
T	T	T	F	**T**
T	T	T	T	**T**

با مقایسه سیگنالها و جدول درستی به صحیح بودن آن پی می‌بریم.

$ABCD = 0000 \rightarrow Out = 1$

$ABCD = 0001 \rightarrow Out = 0$

$ABCD = 0010 \rightarrow Out = 0$

$ABCD = 0011 \rightarrow Out = 0$

$ABCD = 0100 \rightarrow Out = 1$

$ABCD = 0101 \rightarrow Out = 0$

$ABCD = 0110 \rightarrow Out = 1$

$ABCD = 0111 \rightarrow Out = 0$

$ABCD = 1000 \rightarrow Out = 0$

$ABCD = 1001 \rightarrow Out = 1$

$ABCD = 1010 \rightarrow Out = 1$

$ABCD = 1011 \rightarrow Out = 1$

$ABCD = 1100 \rightarrow Out = 1$

$ABCD = 1101 \rightarrow Out = 1$

$ABCD = 1110 \rightarrow Out = 1$

$ABCD = 1111 \rightarrow Out = 1$

Logical Circuit Final Project Report
Mohammad Khoddam
97243029

Problem2

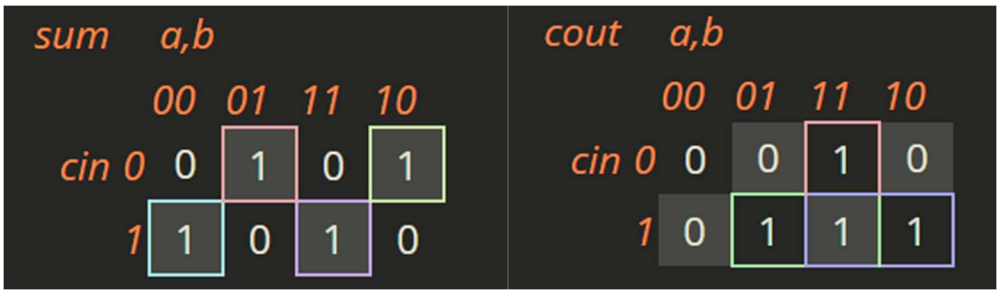
- نحوه حل مسئله

برای ساخت یک Full Adder به صورت زیر میتوان عمل کرد:

$$\{Cout\ , Sum\} = A + B + Cin$$

که جدول درستی آن به شکل زیر می‌باشد:

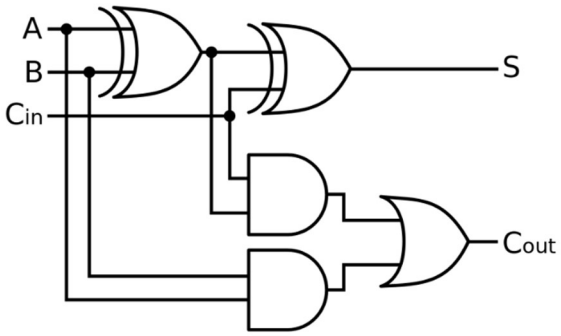
A	B	Cin	Sum	*Cout*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$Sum = Cin'a'b + Cin'AB' + CinA'B' + CinAB = Cin\,(A'B' + AB) + Cin'\,(A'B + AB')$$
$$= \text{Cin} \oplus (\text{A} \oplus \text{B})$$

$$Cout = AB + CinB + CinA$$

Full Adder *به شکل زیر پیاده‌سازی می‌شود:*



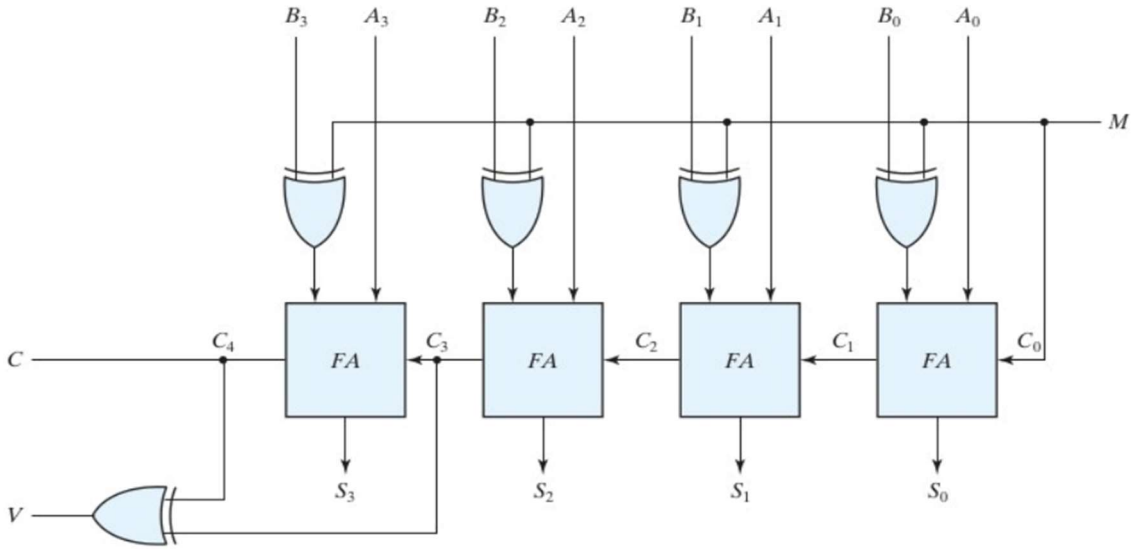
Logical Circuit Final Project Report
Mohammad Khoddam
97243029

نحوه پیاده‌سازی یک جمع‌کننده/تفریق کننده با استفاده از Full Adder به این شکل می‌باشد که حاصل جمع هر کدام از بیت‌ها حساب می‌شود و اگر عبارت بدست‌آمده دارای Cout بود این مقدار به عنوان Full Adder Cin بعدی قرار می‌گیرد.

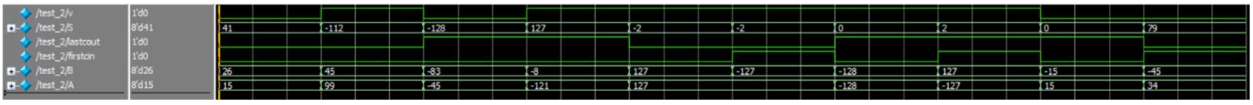
لازم به ذکر است که Cout Full Adder آخر Carry-Out اصلی ما می‌باشد.

همچنین برای تشخیص Overflow می‌توان با بهره‌گیری از شرط هم‌علامت نبودن دو بیت آخر و با استفاده از گیت XOR آن را محاسبه کرد.

برای کامل بودن مدار، تفریق‌کننده هم به آن اضافه شده و اگر Cin مدار اصلی برابر یک باشد این ساختار می‌تواند عمل تفریق را نیز انجام دهد. (قرینه‌کردن ورودی دوم: رقم‌های صفر آن به یک و رقم‌های یک آن به صفر تبدیل شود و یک واحد به آن اضافه‌شود.)



- سیگنال‌های شبیه‌سازی



- اثبات صحت عملکرد

در نظر داشته‌باشیم که برای جلوگیری از OverFlow اعداد باید در بازه‌ی ۱۲۸- تا ۱۲۷ باشند!
(TC8 = Two's Complement 8 bit Representation)

برای حالت جمع (cin = 0) داریم :

$A = -45\ , B = -83\ Sum = -128, V = 0$

$A = -121\ , B = -8\ Sum = -129 (b'1\_01111111) \rightarrow TC8 = 127, V = 1$

$A = 15\ , B = 26\ Sum = 41, V = 0$

$A = 99\ , B = 45\ Sum = 140 (b'10001100) \rightarrow TC8 = -112, V = 1$

Logical Circuit Final Project Report
Mohammad Khoddam
97243029

Problem3

- نحوه حل مسئله

برای این مسئله می‌بایست توابع گفته‌شده را به صورت تک تک در ماژول‌های جداگانه به صورت رفتاری!
پیاده‌سازی کرد و ورودی‌های خود را روی آن ۴ تابع اعمال کرد و نتیجه‌ها را به دست آورد.
در آخر هم بنا به ورودی selecter یکی از آنها را به عنوان خروجی کل انتخاب کرد.

شمای کلی یک ALU با ورودی یک بیتی و چهار بیت سلکتور :

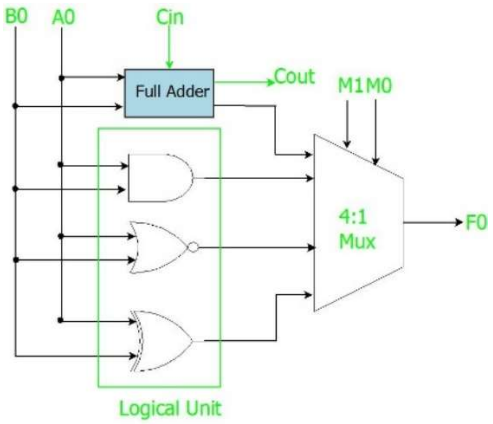
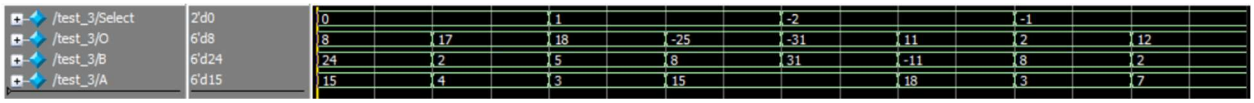


Figure: 1 bit ALU

- سیگنال‌های شبیه‌سازی



- اثبات صحت عملکرد

$Select = b'00\ (0)\ , A = 15, B = 24 \rightarrow Out = 15 \ll 2 + 24 \gg 1 = 60 + 12 = 72(b'01\_001000)$
$\rightarrow TC6 = 8$

$Select = b'00\ (0)\ , A = 4, B = 2 \rightarrow Out = 4 \ll 2 + 2 \gg 1 = 16 + 1 = 17$

$Select = b'01\ (1)\ , A = 3, B = 5 \rightarrow Out = 3 + 3 * 5 = 18$

$Select = b'01\ (1)\ , A = 15, B = 8 \rightarrow Out = 15 + 3 * 8 = 39(b'0\_100111) \rightarrow TC6 = -25$

$Select = b'10\ (-2)\ , A = 15, B = 31 \rightarrow Out = -31$

$Select = b'10\ (-2)\ , A = 18, B = -11 \rightarrow Out = 11$

$Select = b'11\ (-1)\ , A = 3, B = 8 \rightarrow Out = |2 * 3 - 8| = 2$

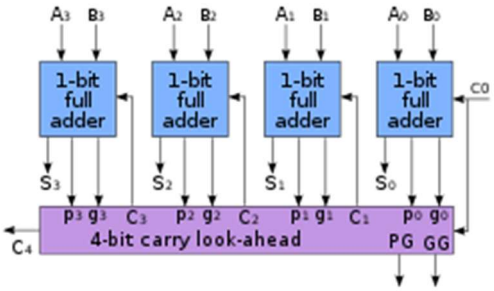
$Select = b'11\ (-1)\ , A = 7, B = 2 \rightarrow Out = |2 * 7 - 2| = 12$

Logical Circuit Final Project Report
Mohammad Khoddam
97243029

Problem4

- نحوه حل مسئله

برای حل این مسئله میتوان از ساختار Carry-Look-Ahead-4bit استفاده کرد و آنرا به 32bit تعمیم داد.
پس ساختار CLA-4bit را بررسی میکنیم:



هدف استفاده از CLA کم‌کردن تاخیر مدار می‌باشد و همه Carry ها رو هم‌زمان بدست می‌آوریم.

به این منظور دو متغیر جدید Propagate و Generate را در نظر می‌گیریم.

Propagate بیانگر این است که اگر یک بیت Carry داشته‌باشیم آیا جمع دو تا یک‌بیتی همچنان Carry تولید می‌کند یا خیر.

$AB = 00 \rightarrow P = 0$ $AB = 10 \rightarrow P = 1$

$AB = 01 \rightarrow P = 1$ $AB = 11 \rightarrow P = X$

در حالت آخر اگر بیت Carry ورودی هم نداشته باشیم همچنان Carry خواهیم‌داشت پس این حالت بستگی به Carry ورودی ندارد (Don't Care) در نتیجه این متغیر می‌تواند هم با گیت or هم xor پیاده‌سازی شود.
به دلیل ساختار ترانزیستوری or سریعتر از xor عمل می‌کند.پس or را انتخاب می‌کنیم.

Generate بیانگر تولیدشدن Carry برای جمع دو تا یک بیتی است.

$AB = 00 \rightarrow G = 0$ $AB = 10 \rightarrow G = 0$

$AB = 01 \rightarrow G = 0$ $AB = 11 \rightarrow G = 1$

واضح است که باید از گیت and استفاده شود.

حال به ادامه مسئله می‌پردازیم:
برای هر بخش می‌توان اینطوری فرض کرد که Carry Out در هر حالت Carry In حالت بعدی است.
Carry In : C_i
Carry Out : C_{i+1}

جدول درستی زیر را داریم :

Logical Circuit Final Project Report
Mohammad Khoddam
97243029

A	B	C_i	C_{i+1}	*Type*
0	0	0	0	None
0	0	1	0	None
0	1	0	0	None
0	1	1	1	Propagate
1	0	0	0	None
1	0	1	1	Propagate
1	1	0	1	Generate
1	1	1	1	Generate/Propagate

رابطه زیر برقرار است :

$$C_{i+1} = G_i + (P_i . C_i)$$

برای طراحی یک *CLA* ۳۲ بیتی باید دو ورودی ۳۲ بیتی داشته باشیم
می‌توانیم برای افزایش بهره‌وری و استفاده مجدد در کامپوننت‌های دیگر ،یک *Carry in* هم به ورودی‌ها اضافه کنیم.
خروجی ساختار نیز یک عدد ۳۲ بیتی به عنوان مجموع و یک بیت *CarryOut* خواهدبود.
برای جمع‌کردن بیت‌ها نیز می‌توان از *Full Adder* هایی که به صورت *Behavioral* تعریف کردیم استفاده‌کنیم.
مدار به شکلی کار خواهد کرد که همه *Carry* ها توسط توابع *P,G,C* محاسبه می‌شود و همه‌ی جمع‌ها با *FullAdder*ها و نتایج قبلی بدست می‌آید.

- سیگنال‌های شبیه‌سازی

- نحوه حل مسئله

$A = 546546\ , B = 123564, Cin = 0 \rightarrow Sum = 670110\ , Cout = b'0$

$A = -345957\ , B = 213568, Cin = 0 \rightarrow Sum = -132389, Cout = b'0$

$A = -654668\ , B = 889401, Cin = 0 \rightarrow Sum = 234733\ , Cout = b'1$

$A = -23132198\ , B = -46565454, Cin = 0 \rightarrow Sum = -69697652\ , Cout = b'1$

Logical Circuit Final Project Report
Mohammad Khoddam
97243029