وزارت علوم، تحقیقات و فناوری
دانشگاه جیرفت

سیستمهای دیجیتال ۱

جلسه ۱۰

مدرس: دکتر محمد هاشمی‌نژاد

محتوای درس

- حساب باینری
- نمایش مکمل دو
- حساب مکمل دو
- جمع هگزادسیمال
- جمع BCD
- مدارات حسابی

حساب باینری

- یکی از ویژگی های مهم سیستم های دیجیتال اجرای عملیات های حسابی است.
- با استفاده از گیت های منطقی می توان عملیات های حسابی را به صورت باینری انجام داد.
- به دلیل کاربرد گسترده عملیات های حسابی، آی سی های مخصوصی نیز در بازار وجود دارند.
- عملیات های حسابی پایه: جمع، تفریق، ضرب و تقسیم
- جمع باینری همانند دسیمال است با این تفاوت که حاصل جمع نمی تواند بزرگتر از ۱ شود.
- اگر حاصل جمع بزرگتر از ۱ شود، حامل یا رقم نقلی ایجاد می شود .
- حامل یا carrier به بیت پرارزش بعدی اضافه می شود .

$$0 + 0 = 0 \text{ carry } 0$$
$$0 + 1 = 1 \text{ carry } 0$$
$$1 + 0 = 1 \text{ carry } 0$$
$$1 + 1 = 0 \text{ carry } 1$$

حساب باینری

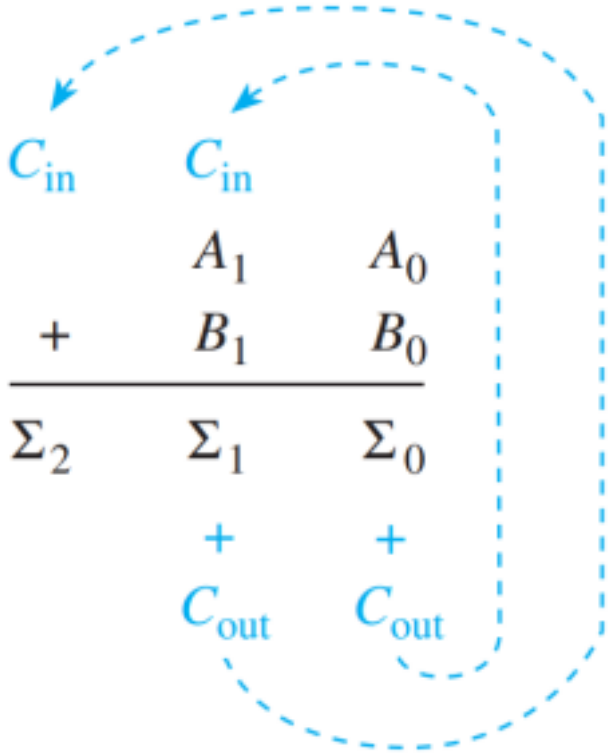
- جمع کننده باینری تک بیتی دارای دو خروجی حاصل جمع و حامل است.

$$A_0 + B_0 = \Sigma_0 + C_{out}$$

A_0	B_0	Σ_0	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

حساب باینری

در جمع کننده باینری چند بیتی حامل ایجادشده در جمع بیت های قبلی را باید درنظر گرفت.



A_1	B_1	C_{in}	Σ_1	C_{out}
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

حساب باینری

- چند مثال

	Decimal	Binary
(a)	5	0000 010
	+ 2	+ 0000 0010
	7	0000 0111 = 7_{10} ✓
(b)	8	0000 1000
	+ 3	+ 0000 0011
	11	0000 1011 = 11_{10} ✓
(c)	18	0001 0010
	+ 2	+ 0000 0010
	20	0001 0100 = 20_{10} ✓

حساب باینری

- چند مثال

(d)

$$\begin{array}{r} 147 \\ +\ 75 \\ \hline 222 \end{array} \qquad \begin{array}{r} 1001\ \ 0011 \\ +\ 0100\ \ 1011 \\ \hline 1101\ \ 1110 \end{array} = 222_{10} \checkmark$$

(e)

$$\begin{array}{r} 31 \\ +\ 7 \\ \hline 38 \end{array} \qquad \begin{array}{r} 0001\ \ 1111 \\ +\ 0000\ \ 0111 \\ \hline 0010\ \ 0110 \end{array} = 38_{10} \checkmark$$

حساب باینری

- تفریق تک بیتی
- اگر عدد اول کوچکتر از عدد دوم باشد، از رقم پر ارزش بعدی یک واحد قرض گرفته می شود .
- خروجی ها :باقی مانده و رقم قرضی

$$A_0 - B_0 = R_0 + B_{out}$$

$0 - 0 = 0$ borrow 0

$0 - 1 = 1$ borrow 1

$1 - 0 = 1$ borrow 0

$1 - 1 = 0$ borrow 0

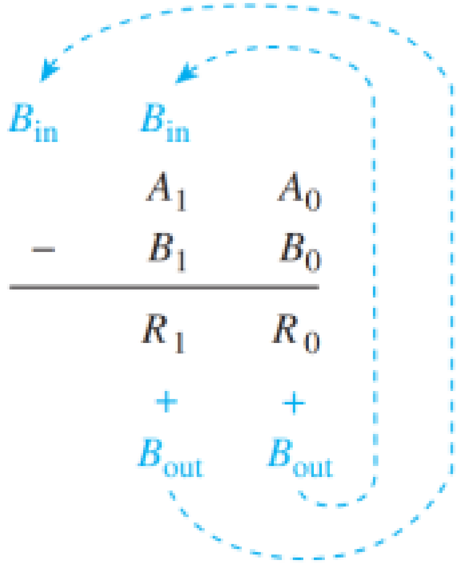
A_0	B_0	R_0	B_{out}	
0	0	0	0	
0	1	1	1	← Borrow required because $A_0 < B_0$
1	0	1	0	
1	1	0	0	

حساب باینری

- در تفریق گر باینری **چند بیتی** رقم قرضی ایجادشده در تفریق بیت های قبلی را باید درنظر گرفت .

$$\begin{array}{r} 4_{10} \\ -\ 1_{10} \\ \hline 3_{10} \end{array} \qquad \begin{array}{c} A_3A_2A_1A_0 \\ A_3A_2A_1A_0 \\ \hline R_3R_2R_1R_0 \end{array} \qquad \begin{array}{rcccc} & 0 & 1 & 0 & 0 \\ - & 0 & 0 & 0 & 1 \\ \hline & 0 & 0 & 1 & 1 \end{array} = 3_{10}$$

(Borrow marks: 1 → 0, 0 → 2 → 1, 0 → 2)



A_1	B_1	B_{in}	R_1	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Borrow (B_{out}) required because B_{in} needs to borrow from A_1, which is zero.

حساب باینری

- چند مثال

(a)

$$\begin{array}{r} 27 \\ -\ 10 \\ \hline 17 \end{array} \qquad \begin{array}{r} 0001\ \ 1011 \\ -\ 0000\ \ 1010 \\ \hline 0001\ \ 0001 = 17_{10} \checkmark \end{array}$$

(b)

$$\begin{array}{r} 9 \\ -\ 4 \\ \hline 5 \end{array} \qquad \begin{array}{r} 0000\ \ 1001 \\ -\ 0000\ \ 0100 \\ \hline 0000\ \ 0101 = 5_{10} \checkmark \end{array}$$

(c)

$$\begin{array}{r} 172 \\ -\ 42 \\ \hline 130 \end{array} \qquad \begin{array}{r} 1010\ \ 1100 \\ -\ 0010\ \ 1010 \\ \hline 1000\ \ 0010 = 130_{10} \checkmark \end{array}$$

حساب باینری

- چند مثال

(d)

$$\begin{array}{r} 154 \\ -\ 54 \\ \hline 100 \end{array} \qquad \begin{array}{r} 1001\ \ 1010 \\ -\ 0011\ \ 0110 \\ \hline 0110\ \ 0100 \end{array} = 100_{10} \checkmark$$

(e)

$$\begin{array}{r} 192 \\ -\ 3 \\ \hline 189 \end{array} \qquad \begin{array}{r} 1100\ \ 0000 \\ -\ 0000\ \ 0011 \\ \hline 1011\ \ 1101 \end{array} = 189_{10} \checkmark$$

حساب باینری

- ضرب باینری هم مانند ضرب دسیمال است فقط با اعداد ۰ و ۱ سر و کار داریم.

Decimal	*Binary*
13	0000 1101 (multiplicand)
× 11	× 0000 1011 (multiplier)
13	0000 1101
13	00001 101
143	000000 00
	0000110 1
	0001000 1111 (product)

حساب باینری

- تقسیم باینری

۳۰	۷
۲۸	۴
۲۰	
۱۴	
۶	

خارج قسمت

- $30 = 16 + 8 + 4 + 2 = (11110)_b$

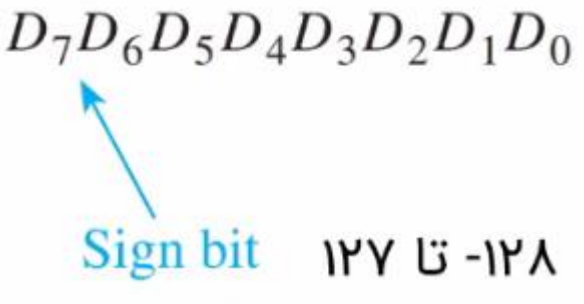
11110	111
111	100
0001	
0000	
00010	

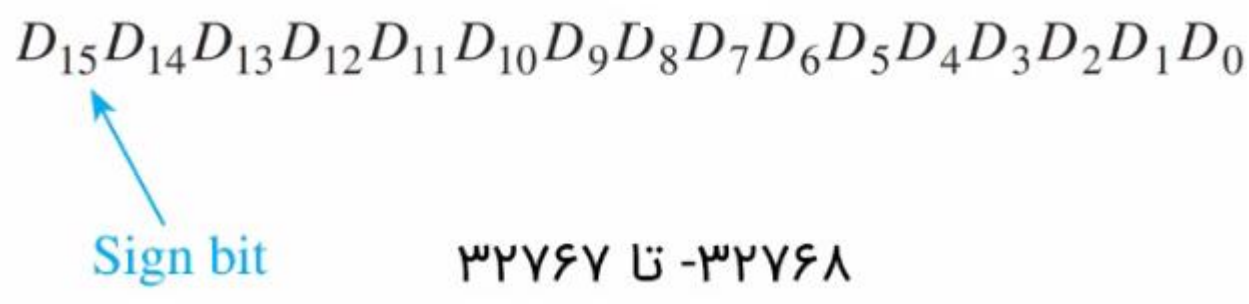
خارج قسمت

باقیمانده

نمایش مکمل دو

- برای انجام محاسبات حسابی در سیستم های دیجیتال اغلب از نمایش مکمل دو است.
- در این نمایش می توان اعداد مثبت و منفی را نمایش داد .
- پرارزش ترین بیت(MSB) به عنوان بیت علامت استفاده می شود





نمایش مکمل دو

- اگر عدد مثبت باشد، نمایش مکمل دو همانند نمایش باینری معمولی است .
- اگر عدد منفی باشد، ابتدا نمایش باینری اندازه آن تعیین می شود. سپس تمام بیت ها قرینه و در نهایت یک واحد به آن اضافه می کنیم .

$+35_{10}$ True binary = 0010 0011 Two's complement = 0010 0011

-35_{10} True binary = 0010 0011

One's complement = 1101 1100

Add 1 = +1

Two's complement = 1101 1101

از طرف LSB تمام بیت‌ها تا رسیدن به اولین ۱ نگه داشته شود، بقیه بیت‌ها قرینه شود.

نمایش مکمل دو

- اگر عدد مثبت باشد، نمایش مکمل دو همانند نمایش باینری معمولی است .
- اگر عدد منفی باشد، ابتدا نمایش باینری اندازه آن تعیین می شود. سپس تمام بیت ها قرینه و در نهایت یک واحد به آن اضافه می کنیم .

-98_{10}

True binary	=	0110 0010
One's complement	=	1001 1101
Add 1	=	+1
Two's complement	=	1001 1110

از طرف LSB تمام بیت‌ها تا رسیدن به اولین ۱ نگه داشته شود، بقیه بیت‌ها قرینه شود.

نمایش مکمل دو

- برای تبدیل اعداد مکمل دو به دسیمال دو روش وجود دارد :
- روش اول :دوباره مکمل دو گرفته شود تا اندازه عدد به دست بیاید (علامت=بیت MSB).
- روش دوم :مشابه با تبدیل باینری به دسیمال معمولی ولی وزن MSB برابر با -2^{N-1} فرض شود .

Two's complement	1011 0010
Complement	0100 1101
Add 1	+1
True binary	0100 1110
Decimal complement	−78

Two's complement	1101 1101
Complement	0010 0010
Add 1	+1
True binary	0010 0011
Decimal complement	−35

حساب مکمل دو

- حسابی برای اعداد علامت دار به راحتی توسط مکمل دو انجام می شود.
- برای تفریق کافی است عدد اول را با مکمل دو عدد دوم جمع کنیم .
- نتیجه :از یک جمع کننده یکسان می توان برای جمع و تفریق اعداد باینری استفاده کرد .
- توجه :اگر جمع و تفریق دو عدد در خارج از بازه قابل تعریف برای سیستم دیجیتال باشد، نتیجه نهایی اشتباه خواهد بود .

حساب مکمل دو

- چند مثال

$$
\begin{array}{rl}
+18 = & 0001\ \ 0010 \\
-7 = & \underline{1111\ \ 1001} \\
\text{Sum} = & 0000\ \ 1011 = 11_{10}
\end{array}
$$

$$
\begin{array}{rl}
+21 = & 0001\ \ 0101 \\
-13 = & \underline{1111\ \ 0011} \\
\text{Sum} = & 0000\ \ 1000 = 8_{10}
\end{array}
$$

$$
\begin{array}{rl}
+118 = & 0111\ \ 0110 \\
-54 = & \underline{1100\ \ 1010} \\
\text{Sum} = & 0100\ \ 0000 = 64_{10}
\end{array}
$$

$$
\begin{array}{rl}
+59 = & 0011\ \ 1011 \\
-96 = & \underline{1010\ \ 0000} \\
\text{Sum} = & 1101\ \ 1011 = -37_{10}
\end{array}
$$

حساب مکمل دو

- تبدیل ضرب به جمع ۳×۳=۳+۳+۳=۹
- تبدیل تقسیم به تفریق ۳۰ تقسیم بر ۷ – با استفاده از شمارنده و تفریق‌های پیاپی

جمع هگزادسیمال

•سازی گسترده در نمایش اعداد توسط بسیاری از سازندگان کامپیوترهای دیجیتال به کار می رود .

•نمونه کاربردها :آدرس دهی حافظه، آدرس دهی وسایل جانبی و کدگذاری دستورات برنامه نویسی

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

- جمع هگزادسیمال همانند جمع دسیمال انجام می شود فقط رقم
- نقلی زمانی ایجاد می شود که حاصل جمع بزرگتر از ۱۵ شود .
- اگر از بیت های بالاتر یک بیت قرض گرفته شد، 16واحد به رقم مذکور اضافه می شود .

جمع هگزادسیمال

- برای محاسبات سریع تر بهتر است به صورت ذهنی جمع هگزادسیمال به دسیمال تبدیل شود و سپس نتیجه به هگزادسیمال برگردانده شود .

$$9 + C = 15_{16}$$

$$\begin{array}{r} A\,7\,C\,5 \\ +\ 2\,D\,A\,8 \\ \hline D\,5\,6\,D \end{array}$$

$$\begin{array}{r} 4\ F \\ +\ 2\,D \\ \hline 7\,C \end{array}$$

جمع هگزادسیمال

- برای محاسبات سریع تر بهتر است به صورت ذهنی جمع هگزادسیمال به دسیمال تبدیل شود و سپس نتیجه به هگزادسیمال برگردانده شود .

$$\begin{array}{r} D7 \\ -\ A8 \\ \hline 2F \end{array} \qquad \begin{array}{r} A05C \\ -\ 24CA \\ \hline 7B92 \end{array}$$

جمع BCD

- در نمایشگرها اغلب برای محاسبات سریع تر از سیستم اعداد BCD استفاده می کنیم.
- هر رقم دسیمال به صورت یک کد باینری ۴–بیتی نمایش داده می شود.
- الگوریتم جمع BCD :

ابتدا همانند جمع باینری عادی، رقم ها جمع شوند .

اگر حاصل جمع بین ۰ تا ۹ باشد، معتبر است.

در غیر این صورت، ۶ واحد به آن اضافه و حامل ایجاد شده به رقم بعد منتقل شود.

گام های ۱ تا ۳ به رقم های LSB تا MSB اعمال شود.

جمع BCD

- چند مثال

$$
\begin{aligned}
8 &= 1000 \\
+\ 7 &= \underline{0111} \\
\text{Sum} &= 1111 \text{ (invalid BCD, so add six)} \\
\text{Add } 6 &= \underline{0110} \\
1\quad 0101 &= 0001\ \ 0101_{\text{BCD}} = 15_{10} \checkmark
\end{aligned}
$$

$$
\begin{aligned}
9 &= \quad 1001 \\
+\ 9 &= \quad \underline{1001} \\
\text{Sum} &= 1\ \ 0010 \text{ (invalid because of carry)} \\
&\quad\ \nwarrow \text{cy} \\
\text{Add } 6 &= \quad \underline{0110} \\
1\ \ 1000 &= 0001\ \ 1000_{\text{BCD}} = 18_{10} \checkmark
\end{aligned}
$$

جمع BCD

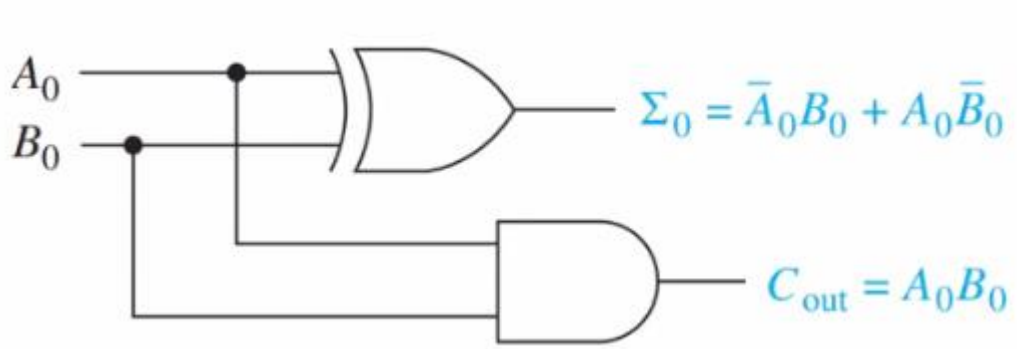
- چند مثال

78 =	0111 1000	
+ 69 =	0110 1001	
Sum =	1110 0001	(both groups of 4
	cy	BCD bits are invalid)
Add 6 =	0110	
	1110 0111	
Add 6 =	0110	
	1 0100 0111 = 0001 0100 0111 = 147_{10} ✓	

مدارات حسابی

- تمام عملیات های حسابی را می توان توسط مدارات جمع کننده پیاده سازی کرد.
- جمع کننده Half-Adder: فقط دو بیت را بدون درنظر گرفتن حامل ورودی جمع می کند.

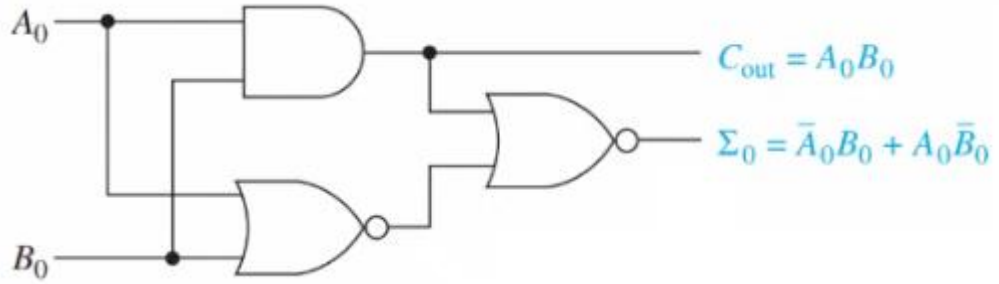
2 inputs		2 outputs	
A_0	B_0	Σ_0	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



مدارات حسابی

- تمام عملیات های حسابی را می توان توسط مدارات جمع کننده پیاده سازی کرد.
- جمع کننده Half-Adder: فقط دو بیت را بدون درنظر گرفتن حامل ورودی جمع می کند.

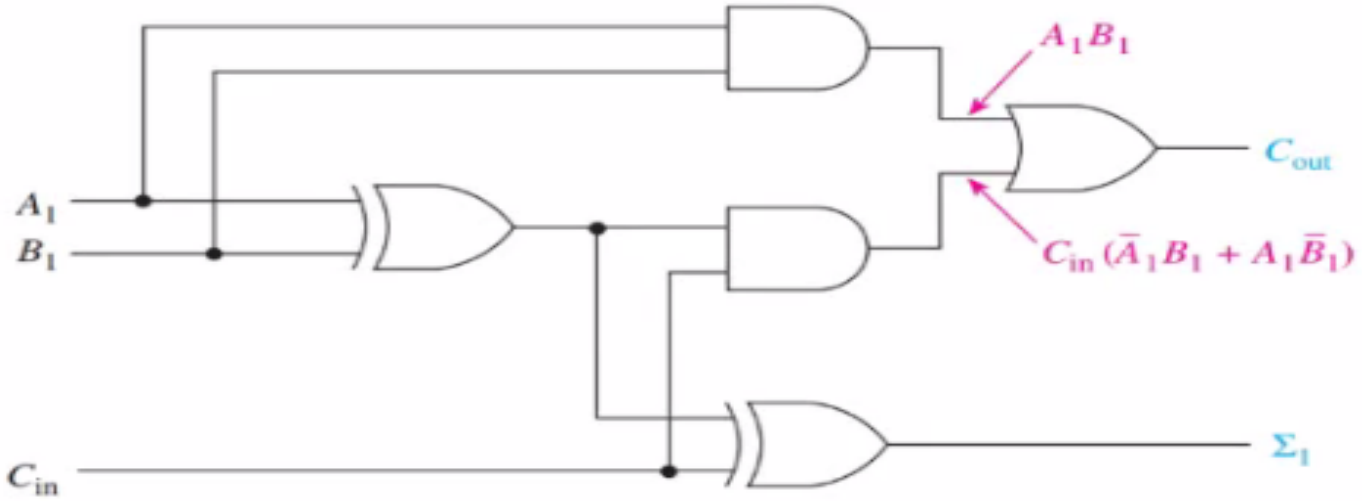
2 inputs		2 outputs	
A_0	B_0	Σ_0	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



مدارات حسابی

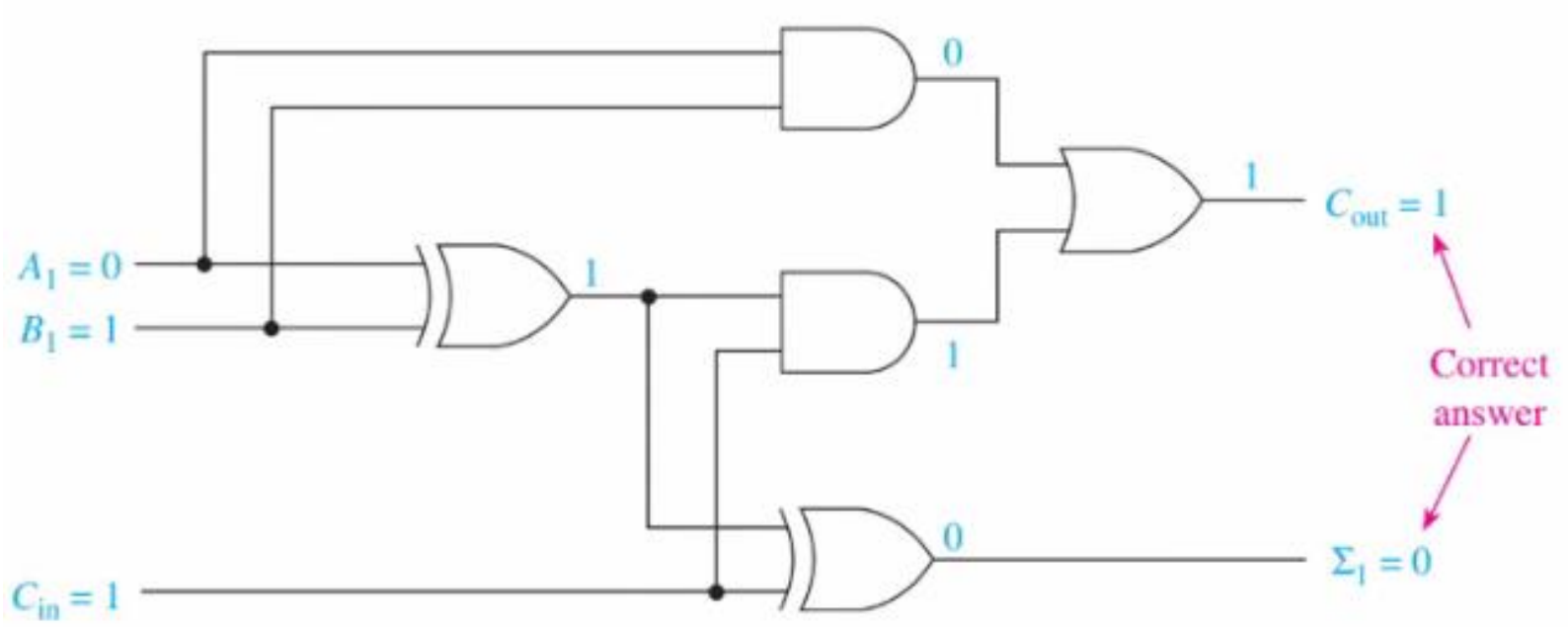
- جمع کننده Full-Adder: دو بیت را با درنظر گرفتن حامل ورودی جمع می کند.

3 inputs			2 outputs	
A_1	B_1	C_{in}	Σ_1	C_{out}
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



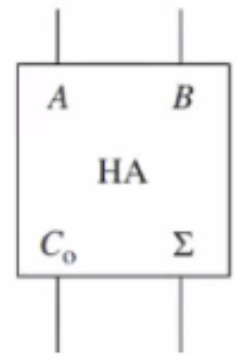
مدارات حسابی

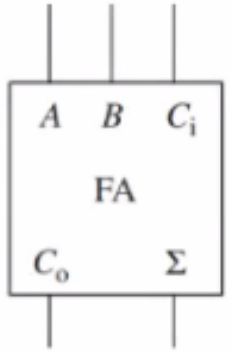
- جمع کنندهFull-Adder: دو بیت را با درنظر گرفتن حامل ورودی جمع می کند.
- تست مدار

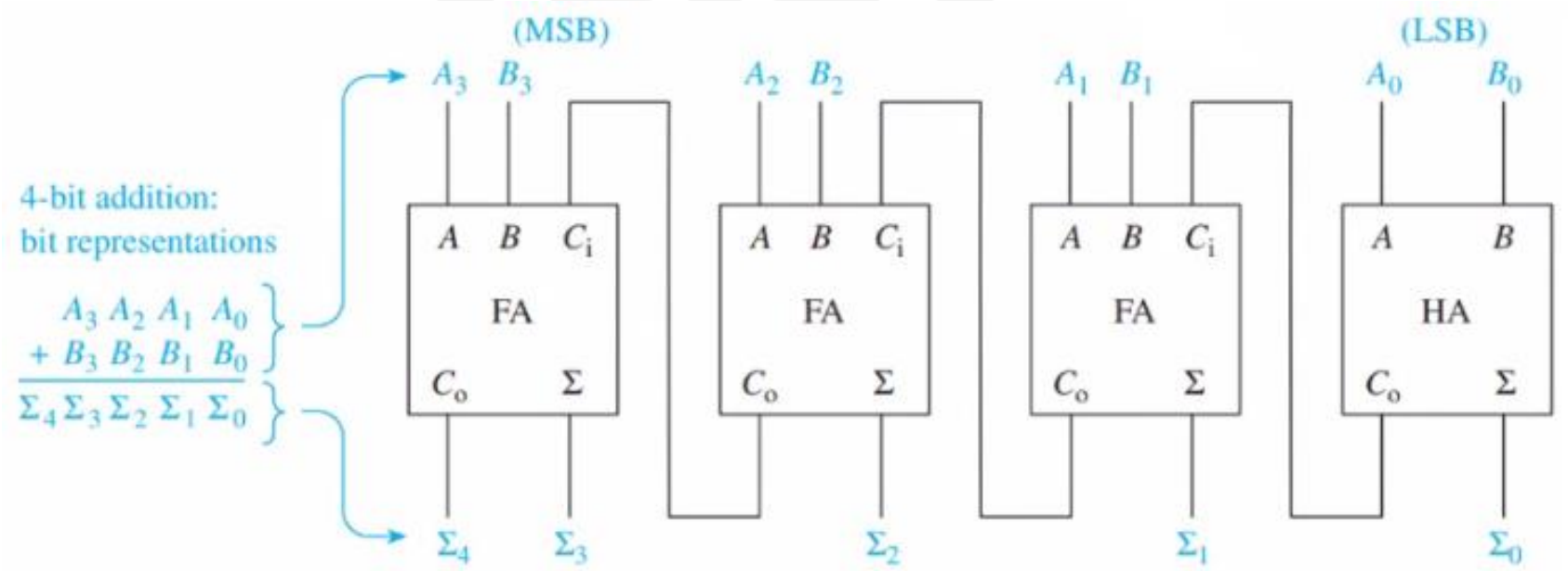


مدارات حسابی

- با ترکیب چند بلوک جمع کننده تک بیتی می توان جمع کننده های چندبیتی را هم ساخت .



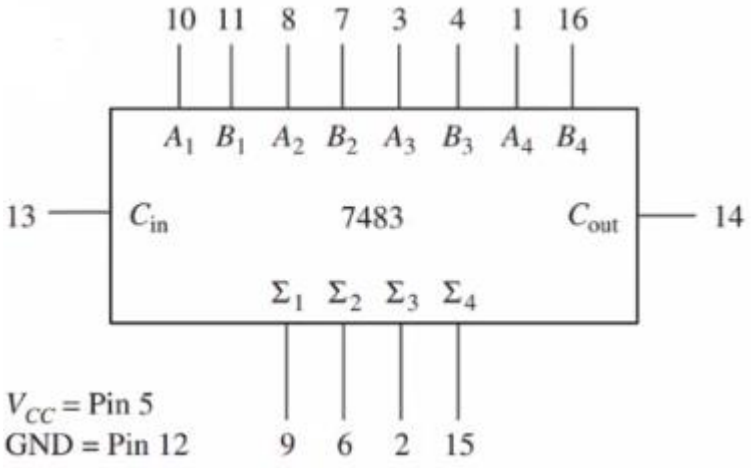




مدارات حسابی

- آی سی های جمع کننده

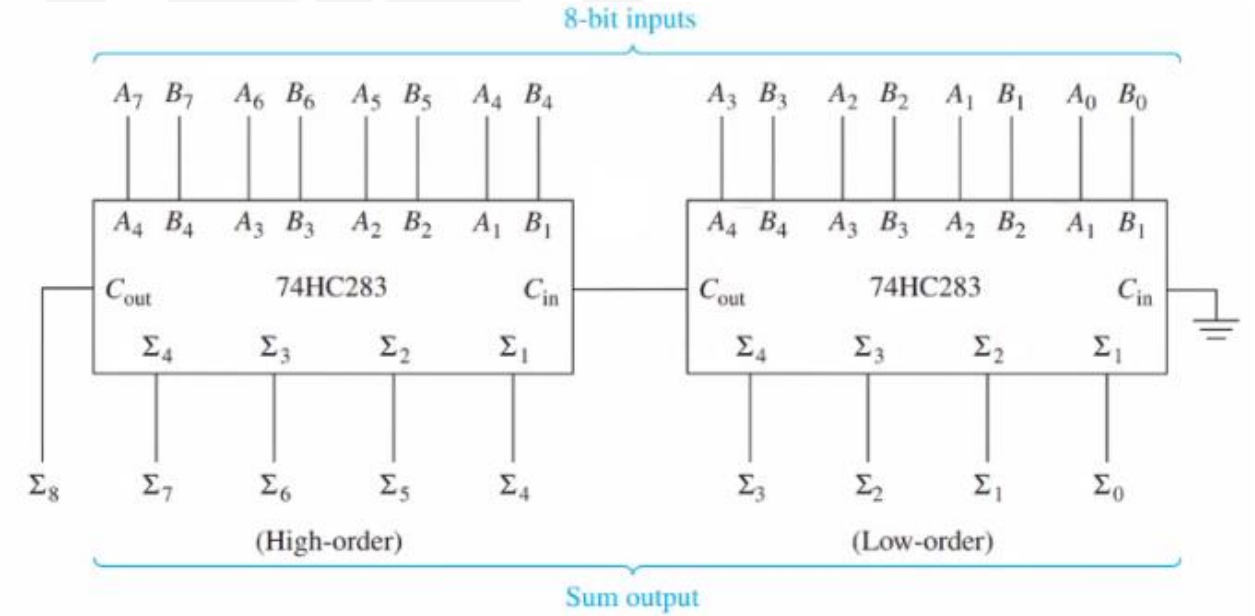
Device	Family	Description
7483	TTL	4-bit binary full-adder, fast carry
74HC283	CMOS	4-bit binary full-adder, fast carry
4008	CMOS	4-bit binary full-adder, fast carry



مدارات حسابی

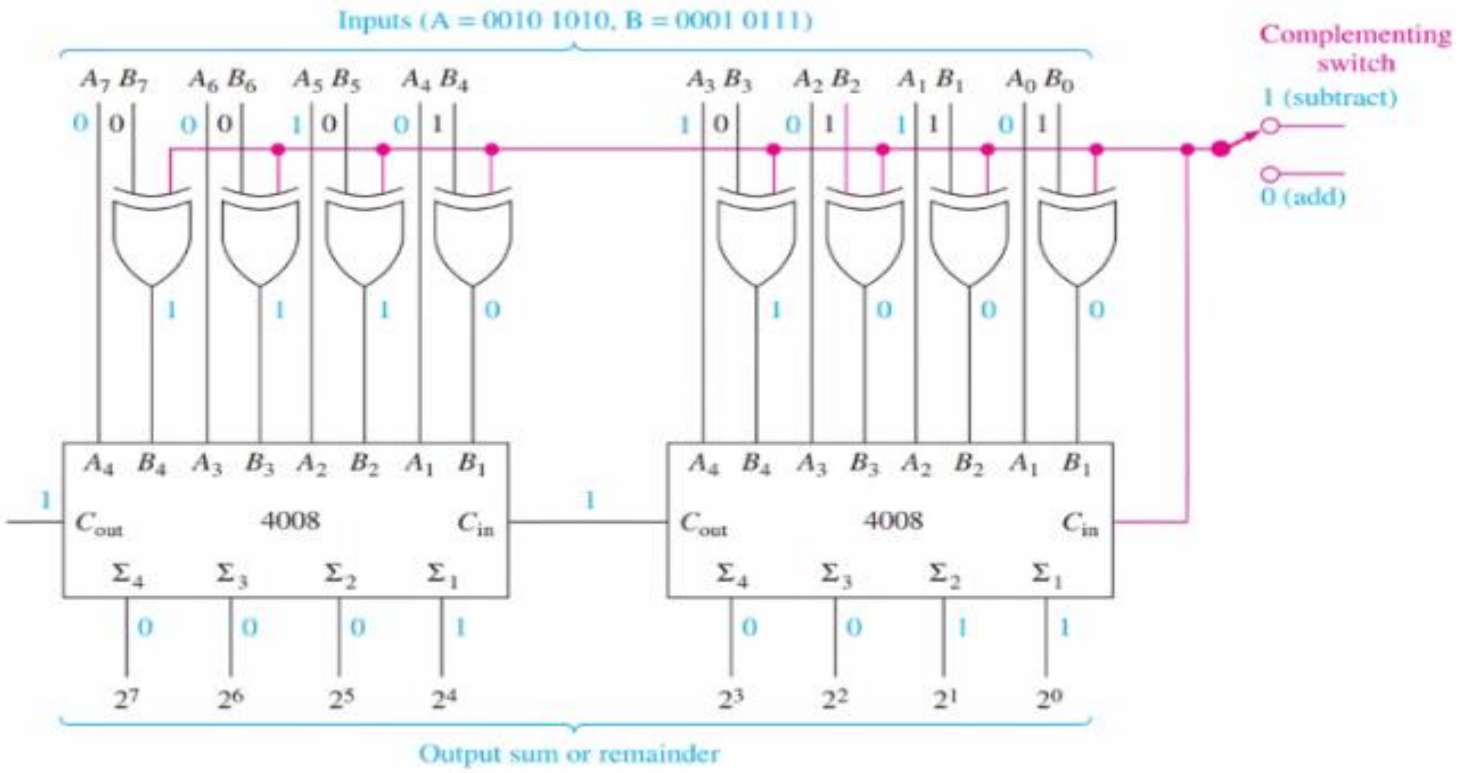
- حامل خروجی آی سی ها برای ساخت جمع کننده های مرتبه بالاتر استفاده می شود .
- جمع کننده ۸–بیتی

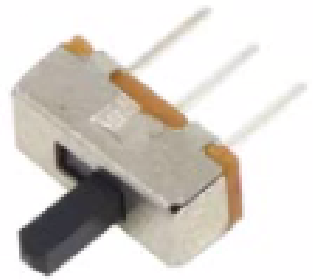
$$\begin{array}{r} A_7A_6A_5A_4A_3A_2A_1A_0 \\ +\ B_7B_6B_5B_4B_3B_2B_1B_0 \\ \hline \Sigma_8\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1\Sigma_0 \end{array}$$



مدارات حسابی

- جمع کننده و تفریق کننده ۸-بیتی: استفاده از گیت XOR برای انتخاب مد عملیاتی





مدارات حسابی

- آی سی منطق و حساب (ALU): SN74181N و 74HC181
- ۱۶ عملیات حسابی + ۱۶ عملیات منطقی

Mode select				Logic functions	Arithmetic operations
S_3	S_2	S_1	S_0	$(M = H)$	$(M = L)(\overline{C}_n = H)$
L	L	L	L	$F = \overline{A}$	$F = A$
L	L	L	H	$F = \overline{A + B}$	$F = A + B$
L	L	H	L	$F = \overline{A}B$	$F = A + \overline{B}$
L	L	H	H	$F = 0$	$F =$ minus 1 (2's comp.)
L	H	L	L	$F = \overline{AB}$	$F = A$ plus $A\overline{B}$
L	H	L	H	$F = \overline{B}$	$F = (A + B)$ plus $A\overline{B}$
L	H	H	L	$F = A \oplus B$	$F = A$ minus B minus 1
L	H	H	H	$F = A\overline{B}$	$F = A\overline{B}$ minus 1
H	L	L	L	$F = \overline{A} + B$	$F = A$ plus AB
H	L	L	H	$F = \overline{A \oplus B}$	$F = A$ plus B
H	L	H	L	$F = B$	$F = (A + \overline{B})$ plus AB
H	L	H	H	$F = AB$	$F = AB$ minus 1
H	H	L	L	$F = 1$	$F = A$ plus A^*
H	H	L	H	$F = A + \overline{B}$	$F = (A + B)$ plus A
H	H	H	L	$F = A + B$	$F = (A + \overline{B})$ plus A
H	H	H	H	$F = A$	$F = A$ minus 1

$F = A$ means: $F_0=A_0, F_1=A_1, F_2=A_2, F_3=A_3$