# LC Se9 Dr. Shaabaany

سؤال مهم: حاصل FA را با استفاده از HA بسازید.

جمع اعداد در 2's complement

← در نمایش 2's compl. اعداد را حتی قابل جمع کردن هستند در صورتیکه carry-out ایجاد شود از آن صرف نظر می‌کنیم و حاصل بدون آن carry کاملاً صحیح است.

$$\begin{array}{r} 0101 \\ +0010 \\ \hline 0111 \end{array} \qquad \begin{array}{rl} 1011 & (-5) \\ +0010 & (+2) \\ \hline 1101 & \end{array} \qquad \begin{array}{rl} 0101 & (5) \\ +1110 & (-2) \\ \hline 0011 & (+3) \end{array} \qquad \begin{array}{r} 1011 \\ +1110 \\ \hline \cancel{1}\,1001 \end{array}$$

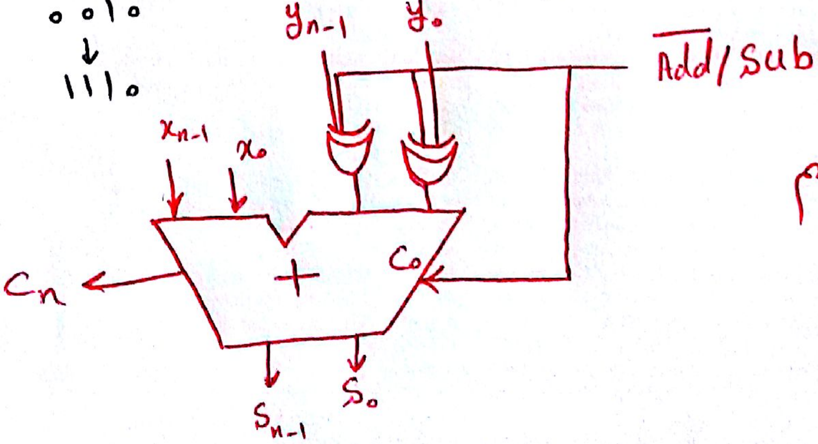
حذف

← بنابراین جمع در این نمایش صرف نظر از علامت اعداد صحیح است.

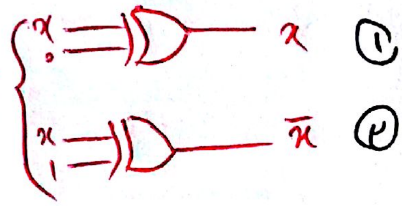
تفریق:

$$\begin{array}{r} 5 \\ -2 \\ \hline \end{array} \qquad \begin{array}{l} +5 \\ +2 \rightarrow -2 \rightarrow \end{array} \qquad \begin{array}{r} 0101 \\ +1110 \\ \hline * \; 0011 \end{array}$$

$$0010 \downarrow 1110$$

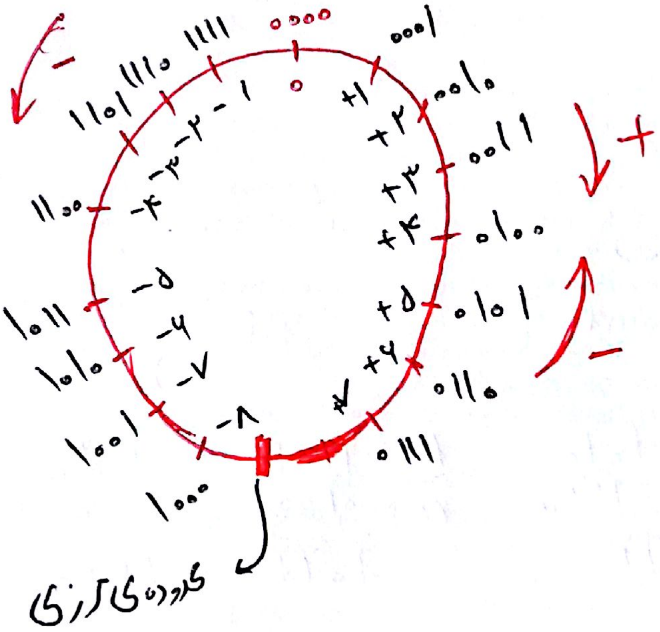



مهم *




* در اینجا به FA خود می‌گوییم اگر تفریق است از XOR شماره ۲ استفاده کند و $y_i$ را با ۱ XOR کند ولی در جمع $y_i$ با ۰ باید XOR شود تا خودش استفاده شود.

# نمایش گرافیکی جمع و تفریق 2's complement




* مهم: در جمع و تفریق اجازه نداریم از محدوده‌ی مرزی عبور کنیم.

## overflow :

← با یک عدد n بیتی در نمایش 2's compl ← $[-2^{n-1}, 2^{n-1}-1]$ تنها این محدوده را می‌توان نشان داد.

← اگر جمع دو عدد n بیتی از این محدوده بزرگتر شود نمی‌توان آن را با n بیت نمایش داد.

مثال:

$$\begin{array}{rcr} +6 & \rightarrow & 0110 \\ +3 & & 0011 \\ \hline & & 1001 \rightarrow \text{غلط} \\ & & (-7) \end{array}$$

→ در این صورت overflow (سرریز) اتفاق افتاده است

## معیار تشخیص overflow :

$$\begin{array}{ccccc} c_4 & c_3 & c_2 & c_1 & c_0 \\ & x_3 & x_2 & x_1 & x_0 \\ + & y_3 & y_2 & y_1 & y_0 \\ \hline c_4 & s_3 & s_3 & s_1 & s_0 \end{array}$$

* در واقع overflow زمانی رخ می‌دهد که

$x_3 = 0$ و $y_3 = 0$ و $s_3 = 1$

یا

$x_3 = 1$ و $y_3 = 1$ و $s_3 = 0$

← یا به تعبیر دیگر زمانی رخ می‌دهد که $c_3$ و $c_4$ عکس هم باشند.

Scanned by CamScanner

$$\begin{array}{r} -۷ \\ +۲ \\ \hline -۵ \end{array} \qquad \begin{array}{r} 00000 \\ 1001 \\ 0010 \\ \hline 1011 \end{array}$$

$$OV = C_{n-1} \oplus C_n$$

$$\begin{array}{r} +۷ \\ +۲ \\ \hline +۹ \end{array} \qquad \begin{array}{r} 01100 \\ 0111 \\ 0010 \\ \hline 1001 \end{array} \rightarrow -۷$$

overflow $c_{n-1}$, $c_n$

---

راه عبور از overflow ← sign extension

$$\begin{array}{r} +۷ \\ +۲ \\ \hline +۹ \end{array} \qquad \begin{array}{r} 0111 \\ 0010 \\ \hline 1001 \end{array} \longrightarrow \begin{array}{r} 00111 \\ 00010 \\ \hline 01001 \end{array}$$

$-۷$ غلط  (+۹) صحیح

$$\begin{array}{r} -۷ \\ -۲ \\ \hline -۹ \end{array} \qquad \begin{array}{r} 1001 \\ 1110 \\ \hline 10111 \end{array} \xrightarrow{SE} \begin{array}{r} 11001 \\ +\,11110 \\ \hline \cancel{1}\,10111 \end{array}$$

حذف  (-۹) صحیح

در verilog → assign C = {x[n-1], x} + {y[n-1], y};

که رقم آخر x را در سمت چپش تکراری کند.

---

تمرین : طراحی مقایسه کننده comparator :

← مداری طراحی کنید که دو عدد چهار بیتی x و y را مقایسه کرده و هر کدام از حالت‌های $x>y$ ، $x=y$ و $x<y$ را گزارش کنید. (امتحانی)

Scanned by CamScanner

# نمایش BCD (binary-coded decimal)

← نمایشی که در آن به ازای هر رقم در یک عدد decimal معادل باینری آن را قرار می‌دهیم

← چون ۱۰ رقم داریم نیاز به ۴ بیت داریم.

| decimal digit | BCD code |
|---|---|
| ۰ | 0000 |
| ۱ | 0001 |
| ۲ | 0010 |
| ۳ | 0011 |
| ۴ | 0100 |
| ۵ | 0101 |
| . | . |
| . | . |
| ۸ | 1000 |
| ۹ | 1001 |

تنها این دو رقم آخرشان ۱ است.

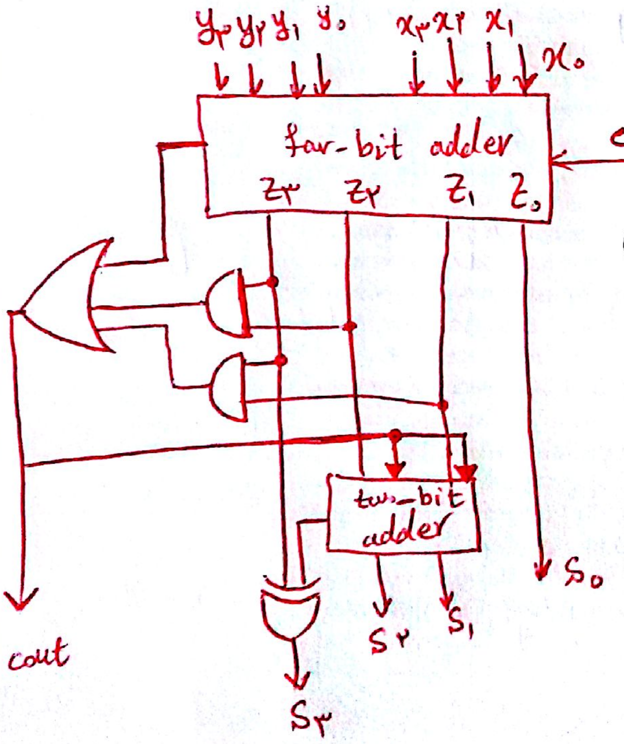
$۳۷ \rightarrow (0011\ 0111)_{BCD}$

← برای جمع در BCD اگر جمع دو ۴ بیتی از مقدار ۹ بیشتر شود باید با ۶ جمع شود (تا از مقدار ۱۵ عبور کند.)

$$
\begin{array}{r}
6 \\
+\,5
\end{array}
\rightarrow
\begin{array}{r}
0110 \\
0101 \\
\hline
1011 \rightarrow >9 \\
0110 \\
\hline
0001\,0001 \rightarrow
\end{array}
\quad \ldots
$$

---

: BCD Adder




۱) زمانیکه جمع $x+y$ ایجاد carry-out کند $c_4=1$

۲) زمانیکه $z_3=1$ , $(z_1=1$ یا $z_2=1)$

$x+y>9$

## نمایش fixed-point , floating-point :

### نمایش fixed-point :

← یک عدد B را در این نمایش بصورت زیر نشان می‌دهیم.

$$\underbrace{b_{n-1} \cdots b_1 b_0}_{\text{قسمت صحیح}} \Big| \underbrace{b_{-1} b_{-2} \cdots b_{-k}}_{\text{قسمت اعشار}}$$

$$v(B) = \sum_{i=-k}^{n-1} b_i \times 2^i$$

B = ۱۰۱۰.۰۱۰۱ ← v(B) = ۱۰,۳۱۲۵

hardware
این چیزا را نمی‌بیند

B = ۰۰.۰۰۰۱۱ ← v(B) = ۰.۰۹۳۷۵

A = ۱۱۰۱.۰۱۱۰۱

B = ۱۱.۰۱

C = A+B; ← در verilog معمولاً غلط است

= {A[۸], A} + {{۳{B[۳]}}, B, ۳'b۰}