Digital Logic Design

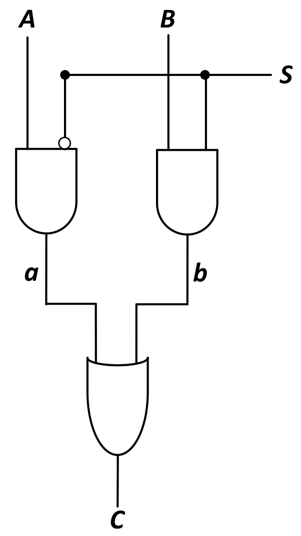
Hajar Falahati

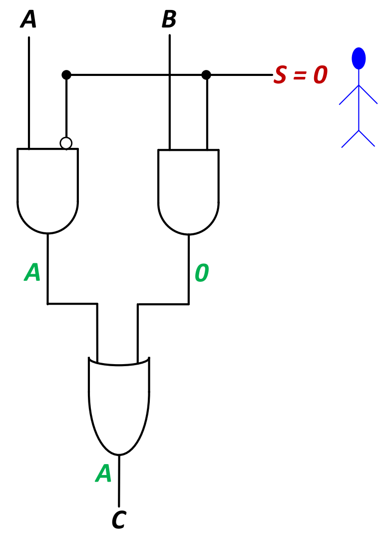
Department of Computer Engineering
IRAN University of Science and Technology

hfalahati@iust.ac.ir

Multiplexer (MUX)

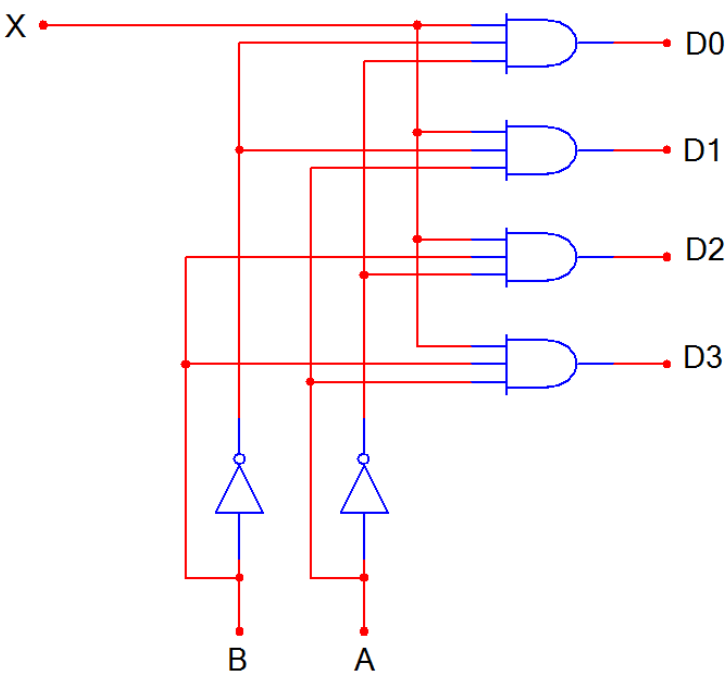
- Selects one of the *N* inputs to connect it to the output
- Needs $\log_2 N$-bit control input
- 2:1 MUX
- How is it useful?

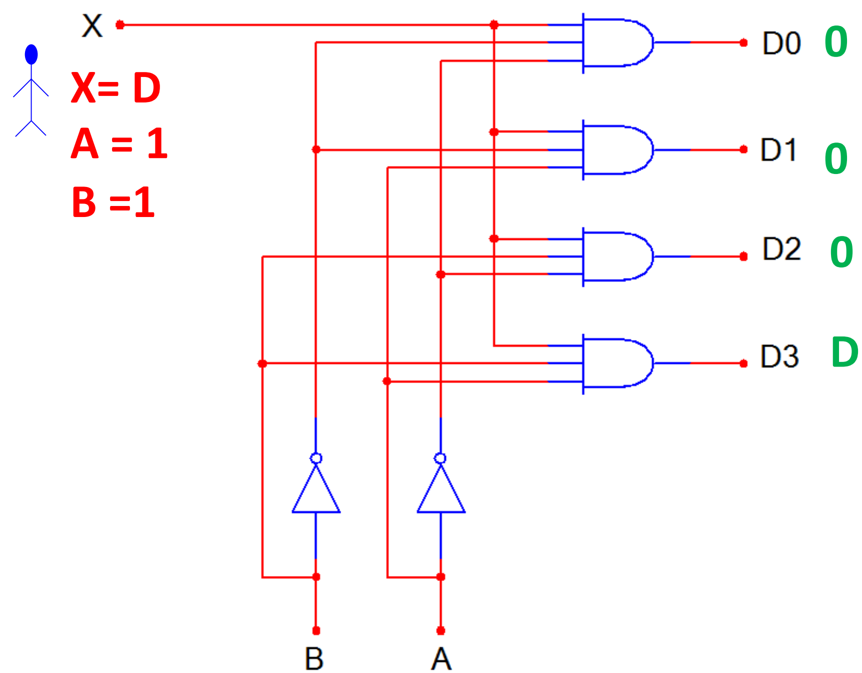




DeMultiplexer (DeMUX)

- Selects one of the N outputs and send the data
- Needs $\log_2 N$-bit control input
- 1 input and 2^n output lines





Outline

- Binary Arithmetic Units
  - Adder
  - Subtractor
  - Comparator

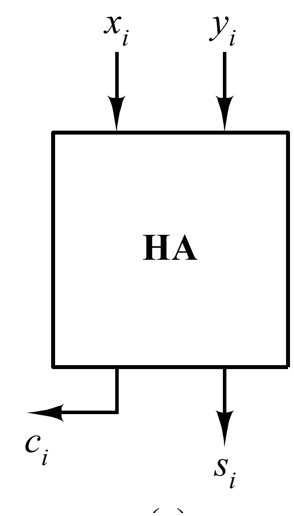
Adder

Half Adder (HA)

- Inputs
  - 2 1-bit numbers
  - x_i, y_i

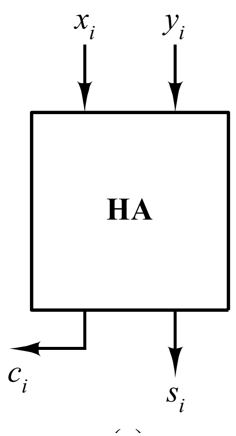
- Output
  - 1 1-bit number
    - A.k.a., sum (s_i)
  - 1 1-bit number
    - A.k.a., carry (c_i)

- Functionality
  - Add two numbers



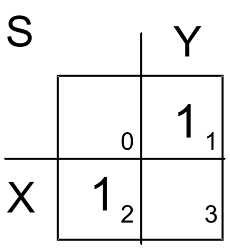
$$\begin{array}{r} x_i \\ +\ \ y_i \\ \hline c_i\ s_i \end{array}$$

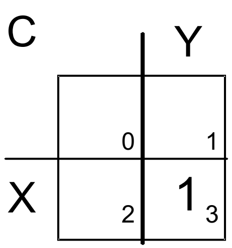
Half Adder (cont'd)



$$\begin{array}{r} x_i \\ +\ \ y_i \\ \hline c_i\ s_i \end{array} \qquad \begin{array}{r} 0 \\ +\ \ 0 \\ \hline 0\ 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ \ 1 \\ \hline 0\ 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ \ 0 \\ \hline 0\ 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ \ 1 \\ \hline 1\ 0 \end{array}$$

x_i	y_i	c_i	s_i
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

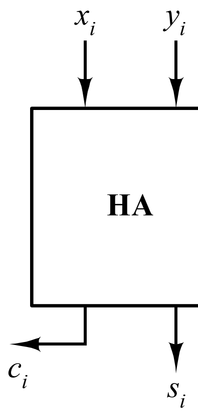




$$s_i = \bar{x}_i . y_i + x_i . \bar{y}_i$$

$$c_i = x_i . y_i$$

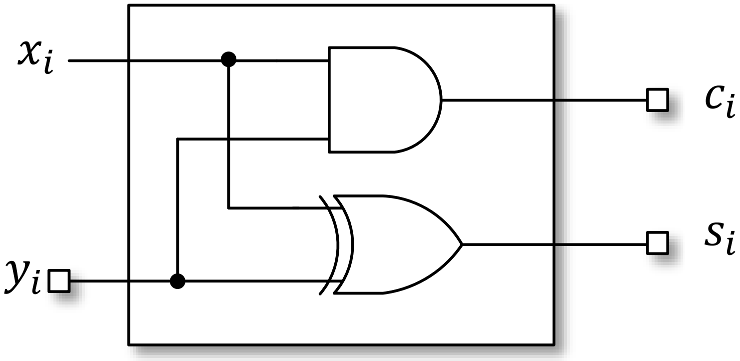
Half Adder (cont’d)

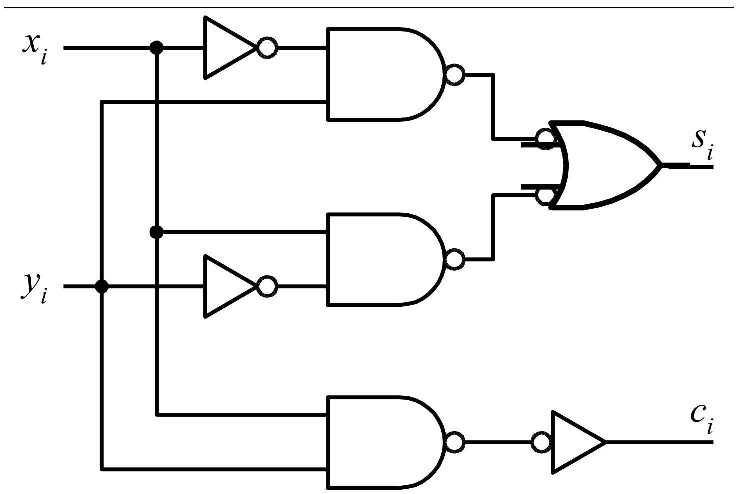


$$
\begin{array}{r}
x_i \\
+\quad y_i \\
\hline
c_i\ s_i
\end{array}
$$

$$s_i = \bar{x}_i . y_i + x_i . \bar{y}_i$$

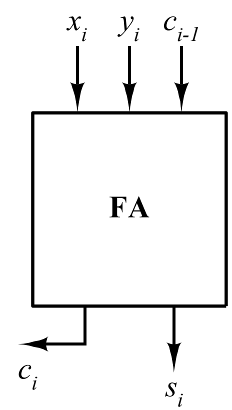
$$c_i = x_i . y_i$$





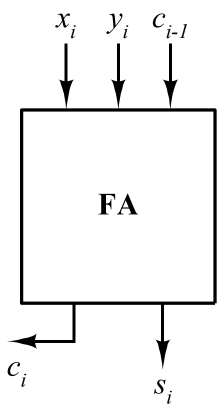
Full Adder (FA)

- Inputs
  - 2 1-bit numbers
    - x_i, y_i
  - 1 1-bit number
    - c_{in} or c_{i-1}
- Output
  - 1 1-bit number
    - A.k.a., sum (s_i)
  - 1 1-bit number
    - A.k.a., carry (c_i)
- Functionality
  - Add two numbers



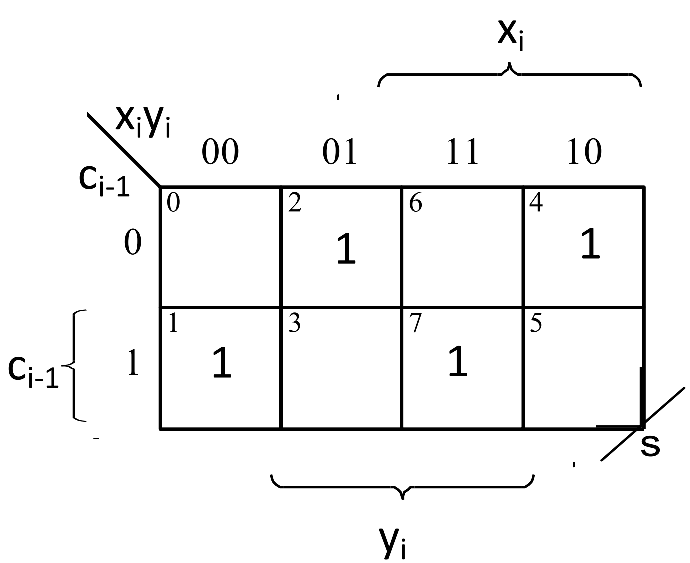
$$\begin{array}{cr} & c_{i-1} \\ & x_i \\ + & y_i \\ \hline & c_i\ s_i \end{array}$$

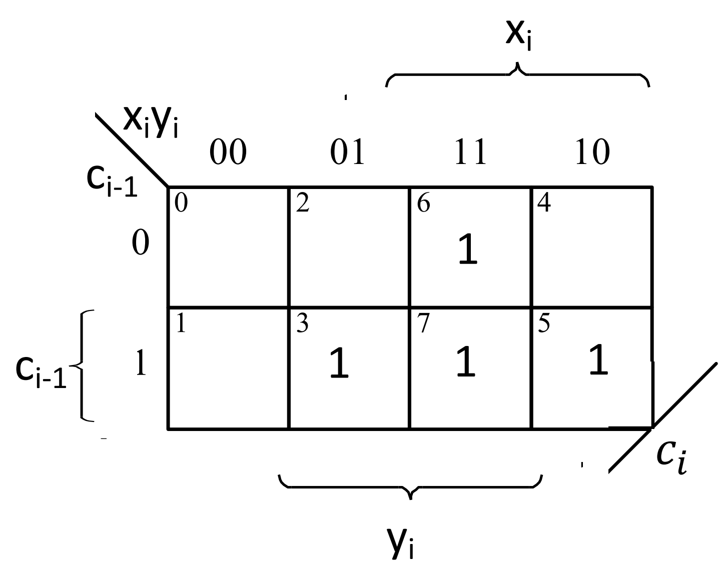
Full Adder (cont'd)



$$\begin{array}{r} x_i \\ + \quad y_i \\ \hline c_i \; s_i \end{array} \qquad \begin{array}{r} 1 \\ 0 \\ + \quad 0 \\ \hline 0 \; 1 \end{array} \qquad \begin{array}{r} 1 \\ 0 \\ + \quad 1 \\ \hline 1 \; 0 \end{array} \qquad \begin{array}{r} 1 \\ 1 \\ + \quad 0 \\ \hline 1 \; 0 \end{array} \qquad \begin{array}{r} 1 \\ 1 \\ + \quad 1 \\ \hline 1 \; 1 \end{array}$$

x_i	y_i	c_{i-1}	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





Full Adder (cont'd)

$$s_i = \bar{x}_i . y_i . \bar{c}_{i-1} + x_i . \bar{y}_i . \bar{c}_{i-1} + x_i . y_i . c_{i-1} + \bar{x}_i . \bar{y}_i . c_{i-1}$$

$$s_i = x_i \oplus y_i \oplus c_{i-1}$$

s (K-map; x_i = 1 in columns 11, 10; y_i = 1 in columns 01, 11; c_{i-1} = 1 in row 1)

c_{i-1} \ $x_i y_i$	00	01	11	10
0	0:	2: 1	6:	4: 1
1	1: 1	3:	7: 1	5:

c_i (K-map; x_i = 1 in columns 11, 10; y_i = 1 in columns 01, 11; c_{i-1} = 1 in row 1)

c_{i-1} \ $x_i y_i$	00	01	11	10
0	0:	2:	6: 1	4:
1	1:	3: 1	7: 1	5: 1

$$c_o = x_i . y_i + y_i . c_{i-1} + x_i . c_{i-1}$$

$$c_o = x_i . y_i + (x_i + y_i) . c_{i-1}$$

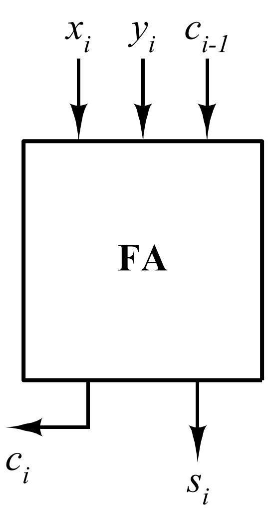
$$c_i = x_i . y_i + (x_i \oplus y_i) . c_{i-1}$$

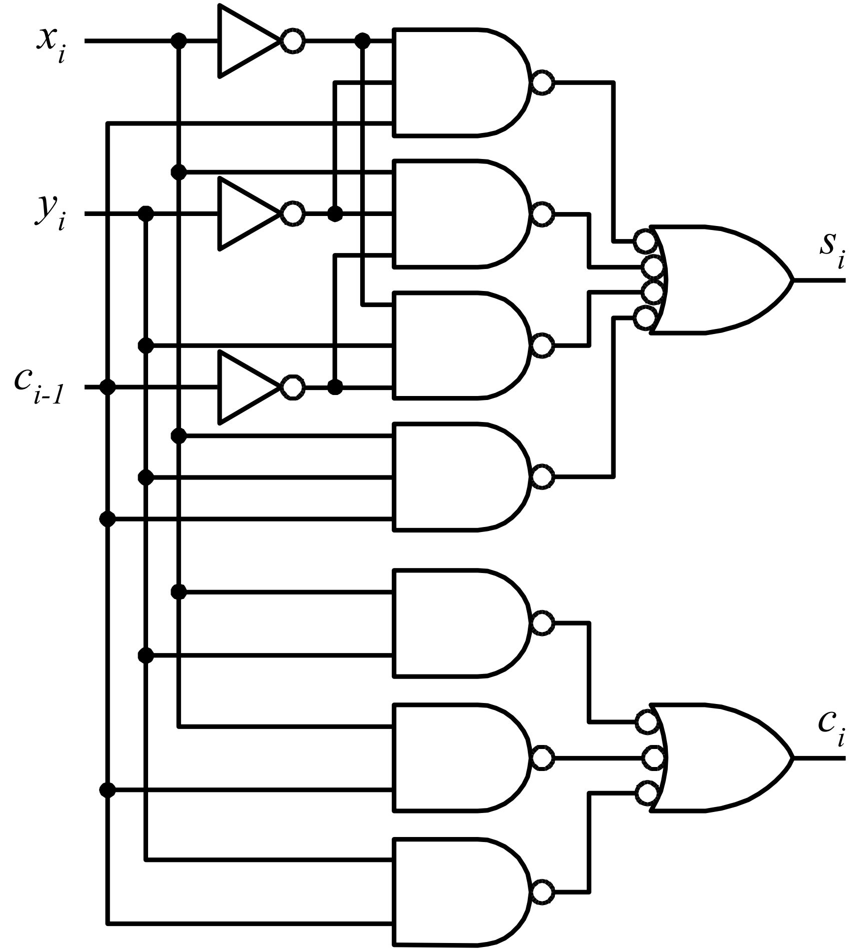
Full Adder (cont'd)

$$s_i = \bar{x}_i . y_i . \bar{c}_{i-1} + x_i . \bar{y}_i . \bar{c}_{i-1} + x_i . y_i . c_{i-1} + \bar{x}_i . \bar{y}_i . c_{i-1}$$

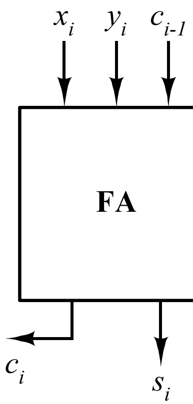
$$s_i = x_i \oplus y_i \oplus c_{i-1}$$

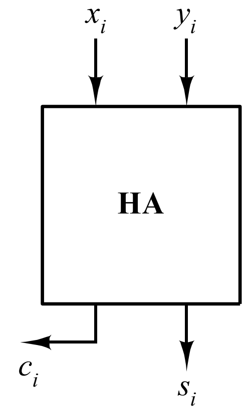
$$c_i = x_i . y_i + y_i . c_{i-1} + x_i . c_{i-1}$$

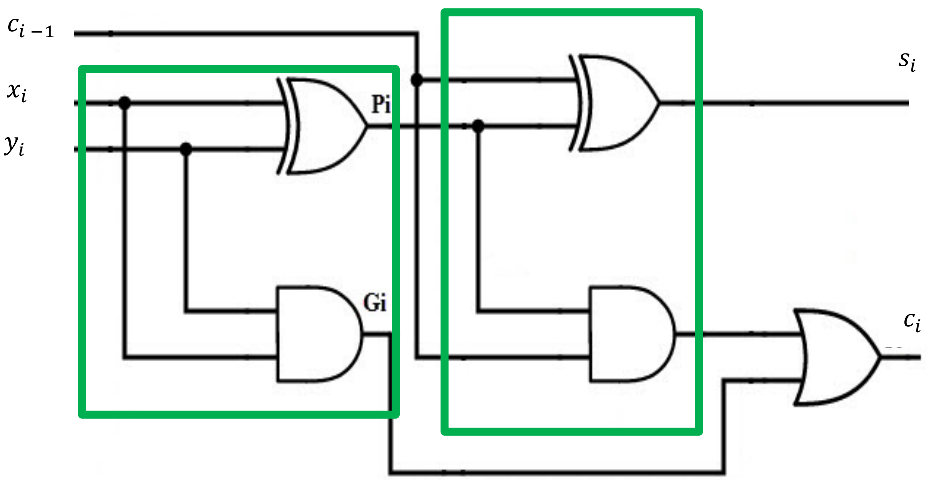


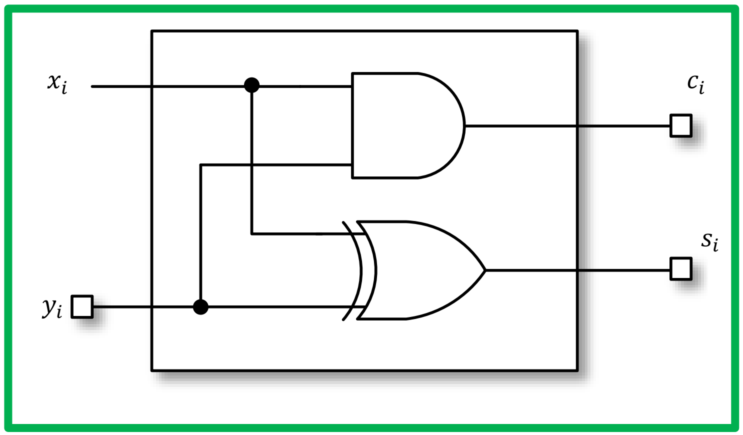


Full Adder Vs. Half Adder

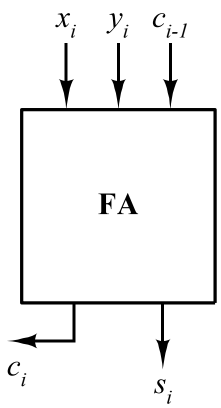


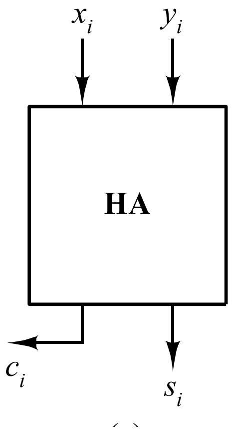


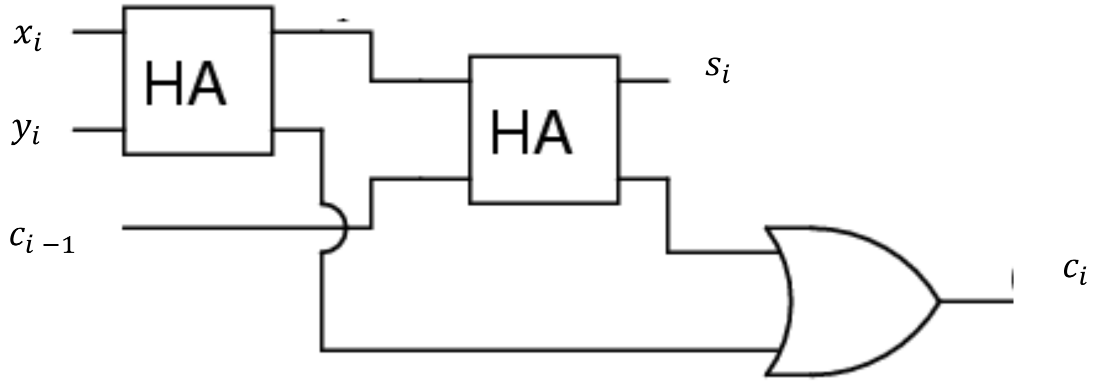




Full Adder Vs. Half Adder(cont'd)







Binary Adder

- Add N-bit Numbers
- Add Two vectors

```
        0
  1 1 1 0 1
+ 1 0 0 0 1
-----------
```

```
1  0 0 0 1 0
   1 1 1 0 1
+  1 0 0 0 1
------------
1  0 1 1 1 0
```

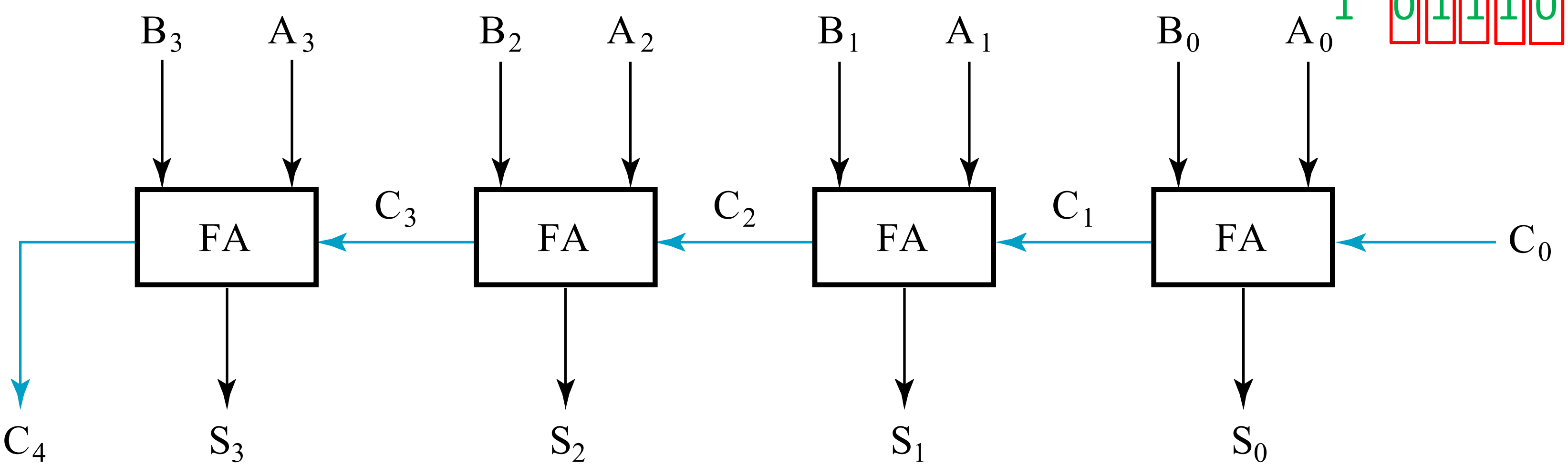
```
            FA
1  0 0 0 1 0
   1 1 1 0 1
+  1 0 0 0 1
------------
1  0 1 1 1 0
```

Binary Adder

- Add N-bit Numbers
- Add Two vectors

FA

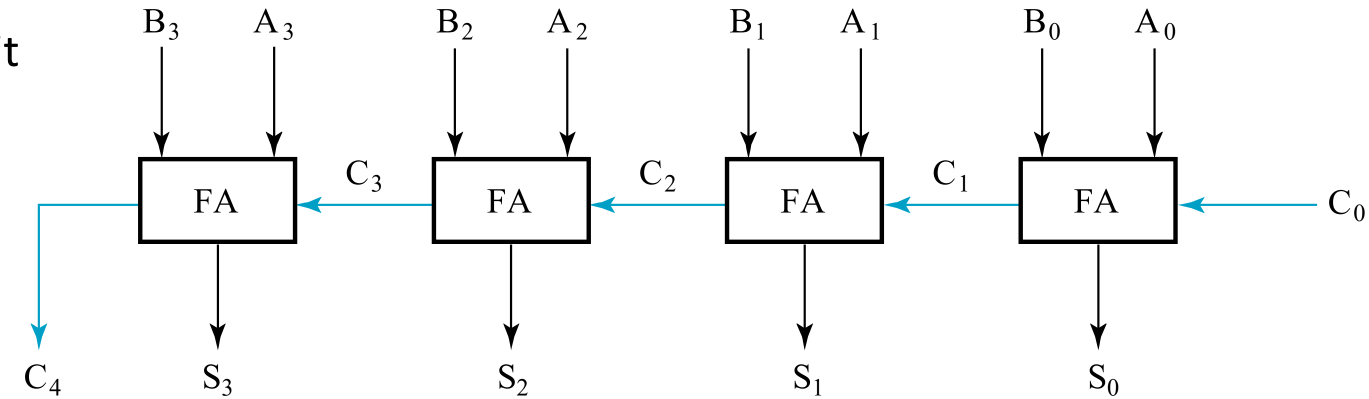
```
 1 0 0 0 1 0
   1 1 1 0 1
+  1 0 0 0 1
------------
 1 0 1 1 1 0
```



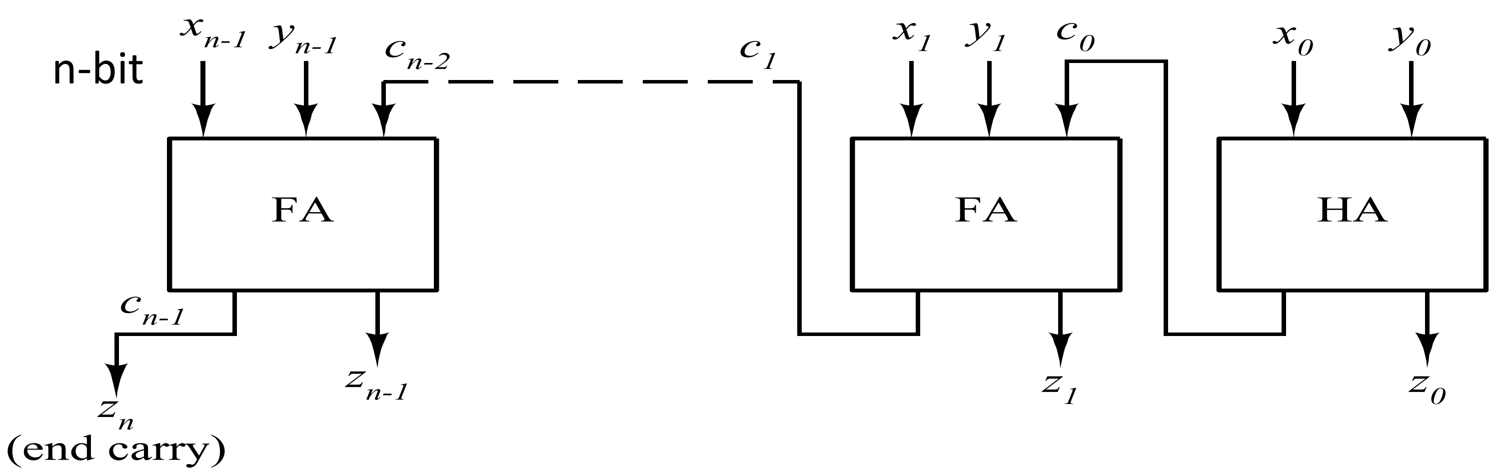
Ripple Carry Adder

Ripple-Carry-Adder (RCA)

4-bit



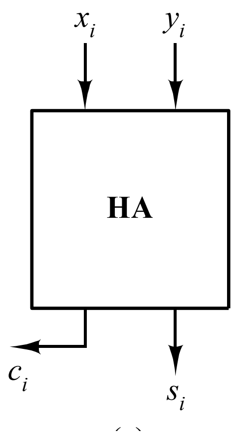
n-bit

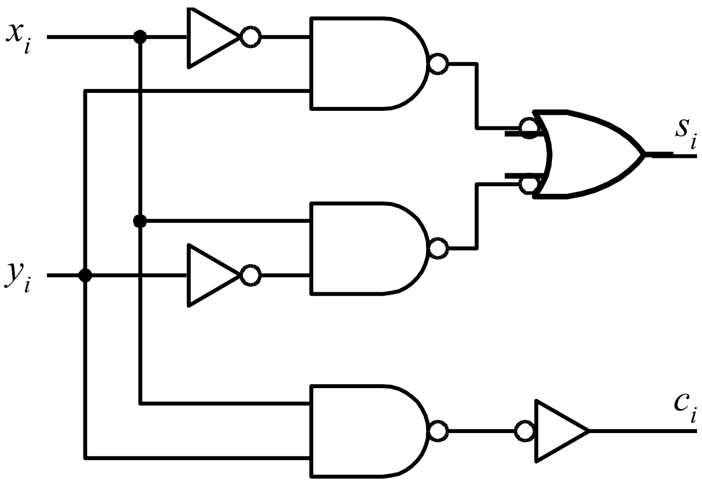


Delay Analysis

- Propagation delay through a typical logic gate
  - t_{gate}
- Calculate the delay of n-bit Ripple carry Adder
  - *?*

Delay Analysis: HA

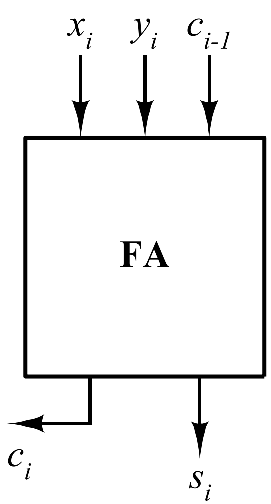




$t_{\text{add}} = 3\ t_{\text{gate}}$

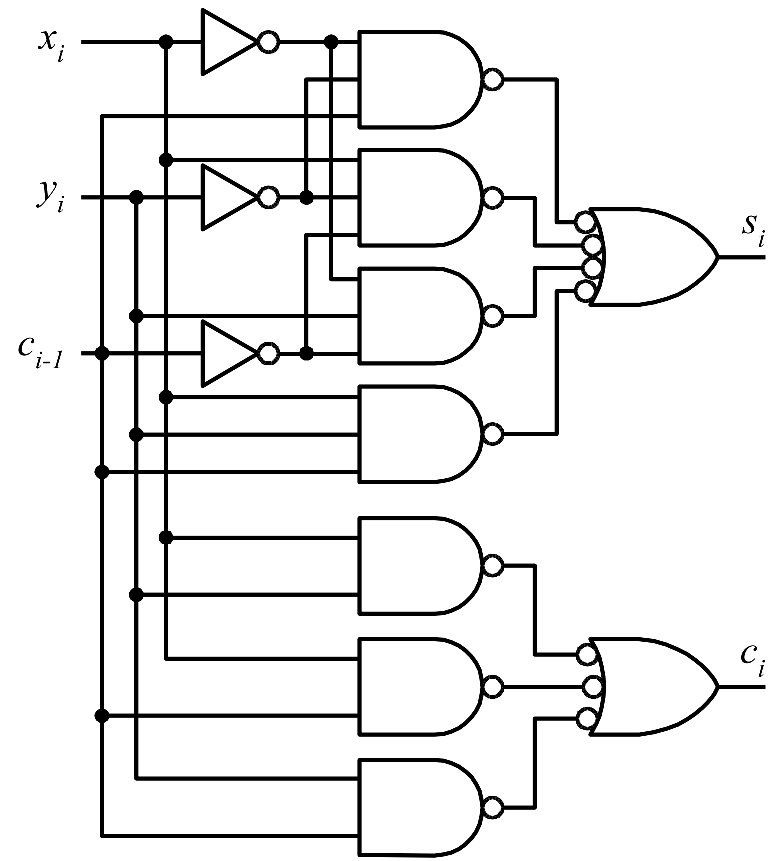
$t_{\text{carry}} = 2\ t_{\text{gate}}$

Delay Analysis: FA

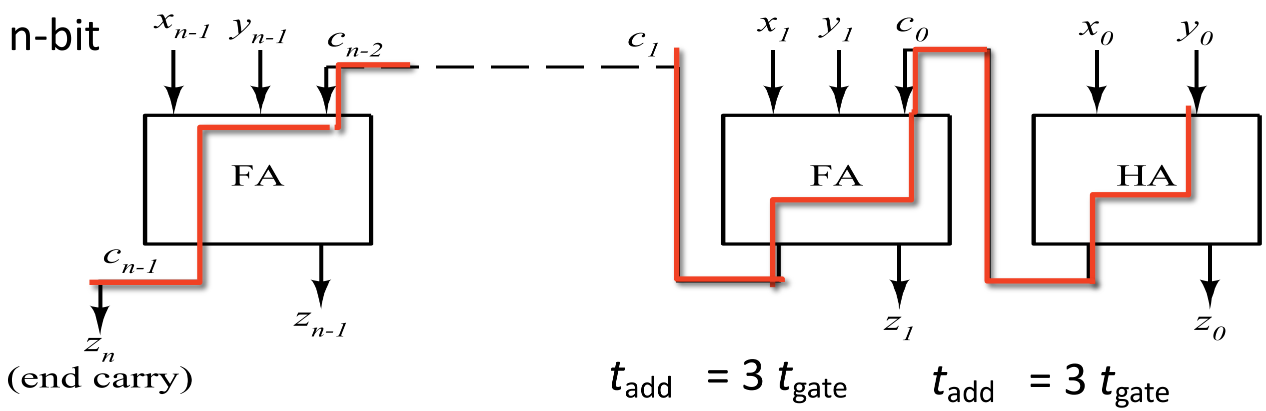


$t_{\text{add}} = 3\ t_{\text{gate}}$

$t_{\text{carry}} = 2\ t_{\text{gate}}$



Delay Analysis: Ripple-Carry-Adder



$t_{add} = 3\ t_{gate}$ $t_{add} = 3\ t_{gate}$

$t_{carry} = 2\ t_{gate}$ $t_{carry} = 2\ t_{gate}$

	delay
S_0	*3*
C_0	*2*
S_1	*5 = 3+2*
C_1	*4 = 2+2*
S_2	*7 = 3+ 4*
C_2	*6 = 2+4*
S_3	*9 = 3+ 6*
C_3	*8 = 2+6*

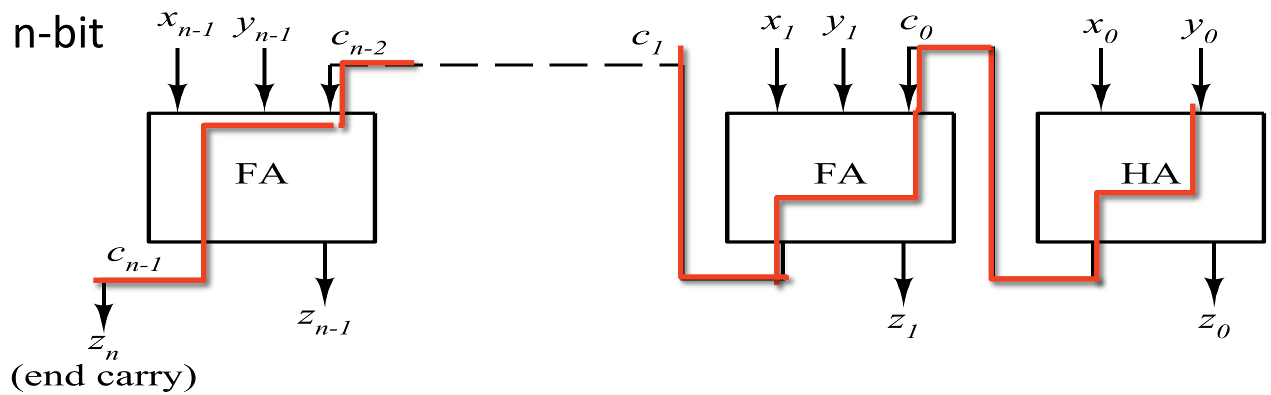
$$t_{add} = 3\ t_{gate} + (n-1)\ 2\ t_{gate} = (2n + 1)\ t_{gate}$$

$$t_{carry} = 2\ t_{gate} + (n-1)\ 2\ t_{gate} = 2n\ \ t_{gate}$$

Delay Analysis: Ripple-Carry-Adder (cont'd)

- Each bit adder waits for its carry input
  - Generate and propagate carry

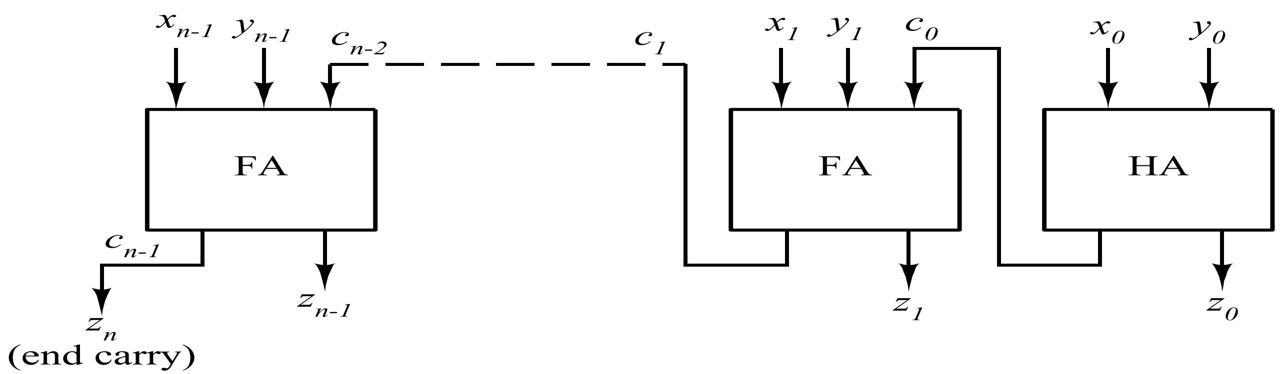
- Carry propagation
- => long sequential wait chain
- => Ripple-carry-adder is slow



$t_{\text{add}} = 3\, t_{\text{gate}} + (n-1)\, 2\, t_{\text{gate}} = (2n + 1)\, t_{\text{gate}}$ $\quad$ $t_{carry} = 2\, t_{\text{gate}} + (n-1)\, 2\, t_{\text{gate}} = 2n\ t_{\text{gate}}$

Faster Binary Adder

- Can we make ripple-carry-adder faster?
- Can we generate all terms in parallel?
- Can we say anything about Cout without having Cin?



$t_{add} = 3\ t_{gate} + (n-1)\ 2\ t_{gate} = (2n + 1)\ t_{gate}$ $t_{carry} = 2\ t_{gate} + (n-1)\ 2\ t_{gate} = 2n\ t_{gate}$

Faster Binary Adder (cont'd)

$s_i = x_i \oplus y_i \oplus c_{i-1}$

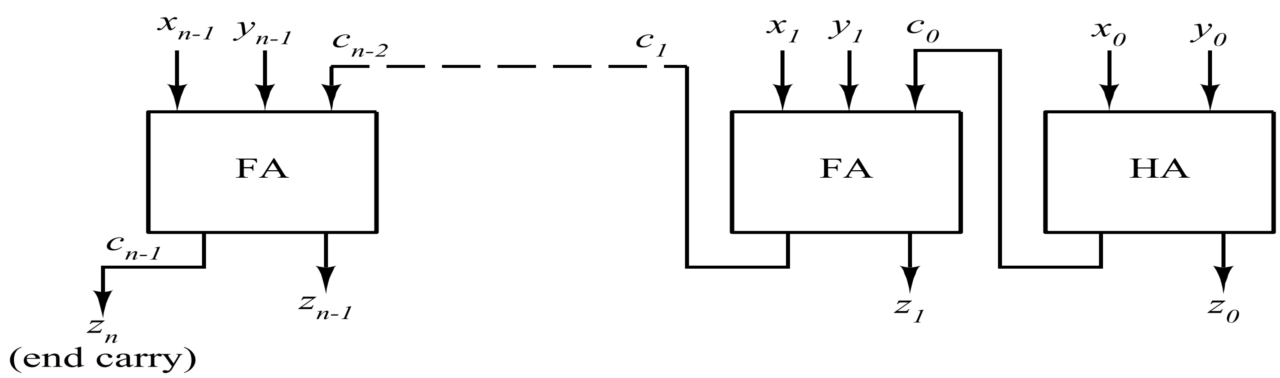
$c_i = x_i . y_i + (x_i \oplus y_i) . c_{i-1}$

$g_i = x_i . y_i$: Carry generate (G)

$p_i = (x_i \oplus y_i)$: Carry propagate (P)

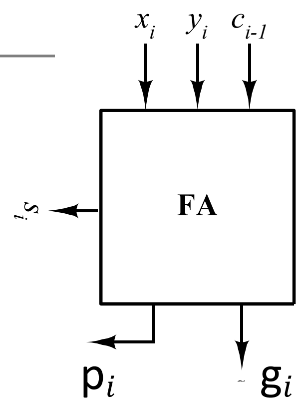
$$s_i = pi \oplus c_{i-1}$$

$$c_i = g_i + p_i . c_{i-1}$$

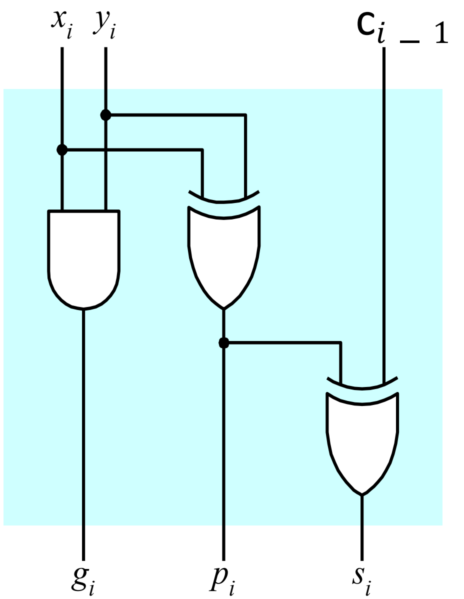


Carry Look Ahead

- Inputs
  - 2 1-bit numbers
    - x_i, y_i
  - 1 1-bit number
    - c_{in} or c_{i-1}
- Output
  - 1 1-bit number
    - A.k.a., sum (s_i)
  - 1 1-bit number
    - A.k.a., carry propagate (p_i)
  - 1 1-bit number
    - A.k.a., carry generate (g_i)



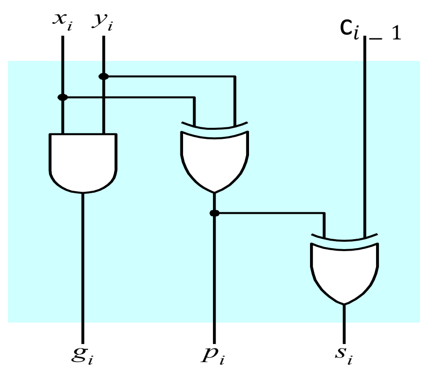
$$g_i = x_i . y_i$$
$$p_i = (x_i \oplus y_i)$$



Carry Look Ahead (cont'd)

$s_i = pi \oplus c_{i-1}$

$c_i = g_i + p_i \cdot c_{i-1}$



$c_0 = g_0 + p_0 c_{in}$

$s_0 = p_0 \oplus c_{in}$

$c_1 = g_1 + p_1 c_0$

$= g_1 + p_1 g_0$

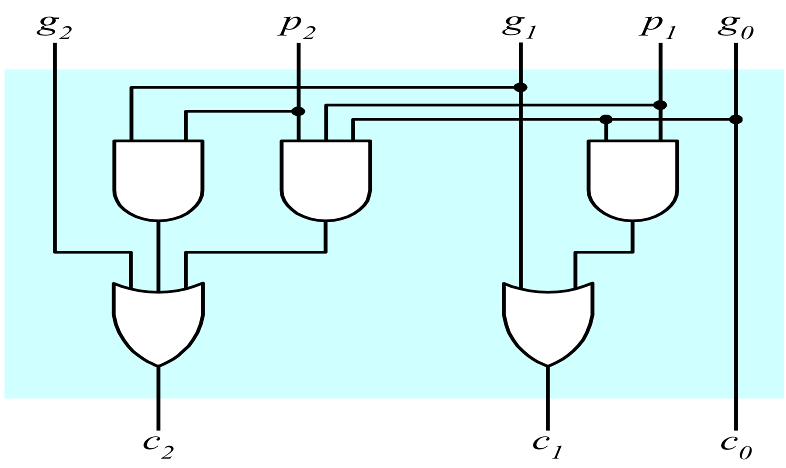
$s_1 = p_1 \oplus c_0$

$c_2 = g_2 + p_2 c_1$

$= g_2 + p_2(g_1 + p_1 g_0)$

$= g_2 + p_2 g_1 + p_2 p_1 g_0$

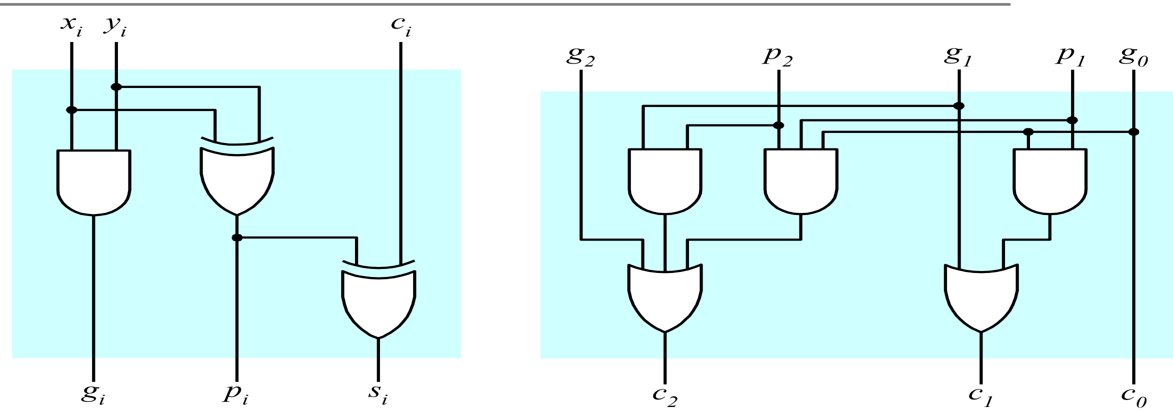
$s_2 = p_2 \oplus c_1$



Carry Look Ahead: Delay

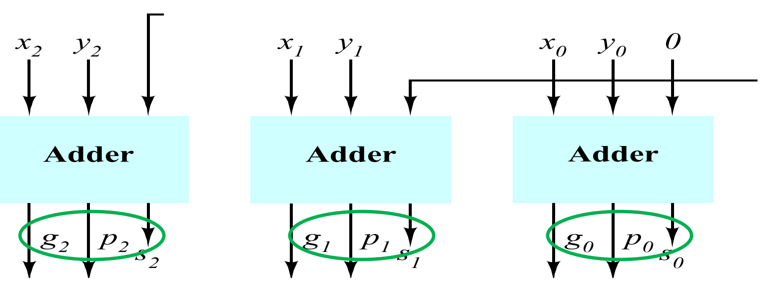
$t_g = t_p = t_{gate}$

$t_c = 2\ t_{gate}$



(a) (b)

	Delay	Total Delay
p_i	1	1
g_i	1	1
S_0	1	
C_0	2	
S_1	1	
C_1	2	
S_2	1	
C_2	2	



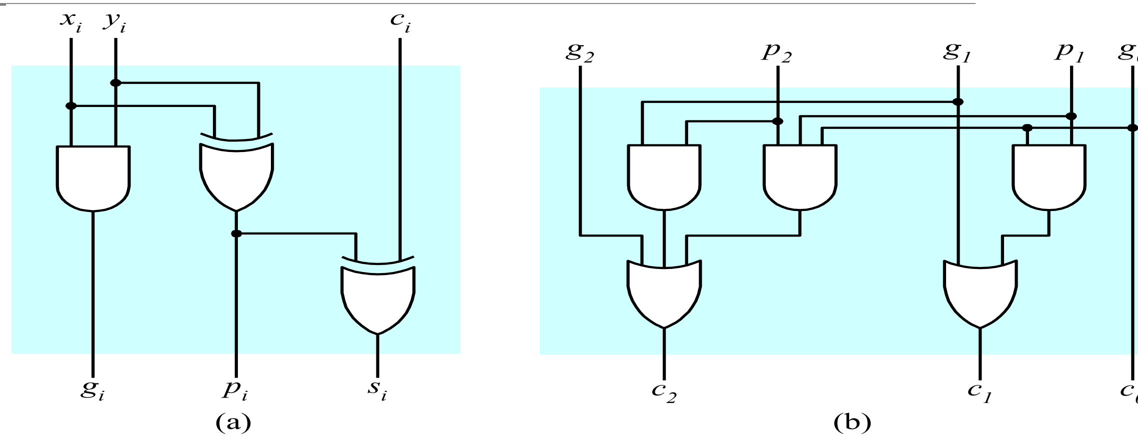
$s_i = pi \oplus c_{i-1} \quad c_i = g_i . + p_i . c_{i-1}$

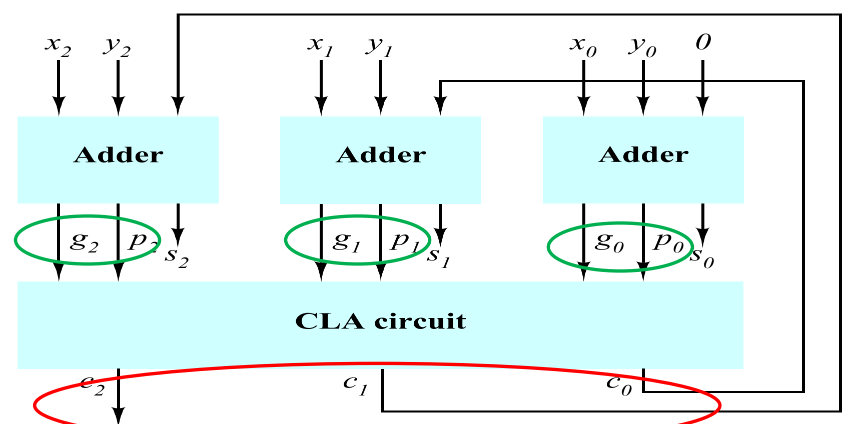
Carry Look Ahead: Delay (cont'd)

$t_g = t_p = t_{gate}$

$t_c = 2\ t_{gate}$

	Delay	Total Delay
p_i	1	1
g_i	1	1
S_0	1	
C_0	2	3
S_1	1	
C_1	2	3
S_2	1	
C_2	2	3





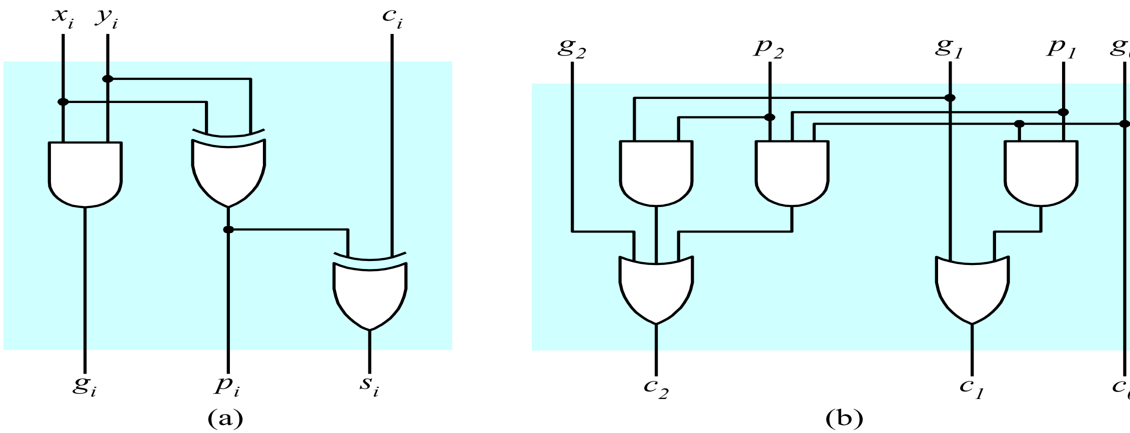
$s_i = p_i \oplus c_{i-1}$ $\quad c_i = g_i + p_i \cdot c_{i-1}$

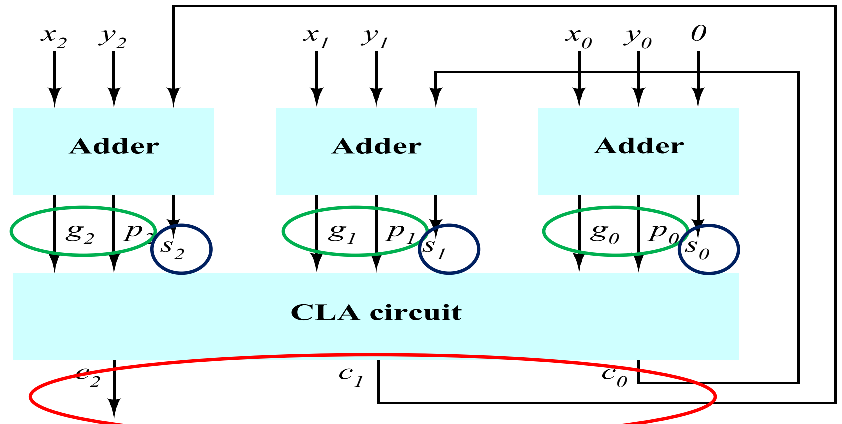
Carry Look Ahead: Delay (cont’d)

$t_g = t_p = t_{gate}$

$t_c = 2\ t_{gate}$

	Delay	Total Delay
p_i	1	1
g_i	1	1
S_0	1	4
C_0	2	3
S_1	1	4
C_1	2	3
S_2	1	4
C_2	2	3

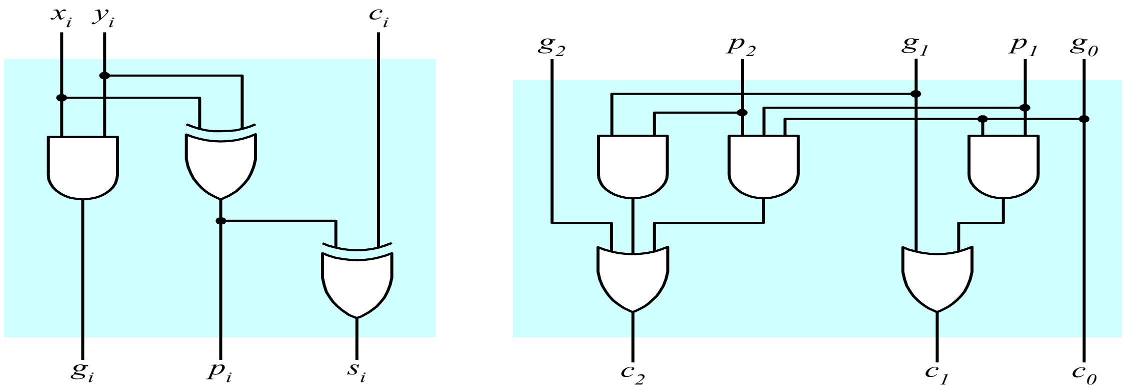




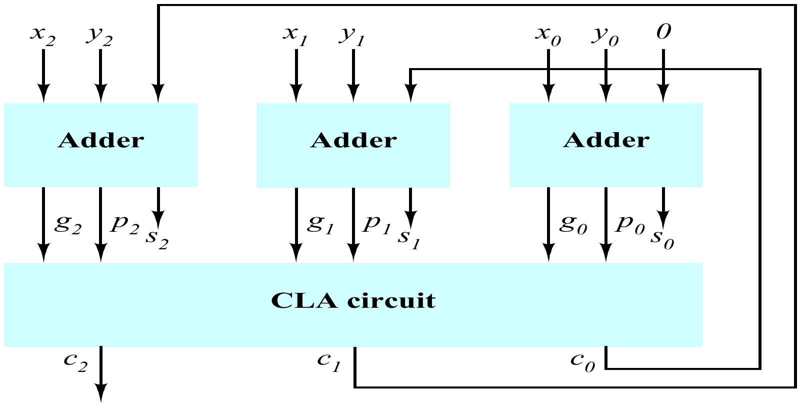
$s_i = pi \oplus c_{i-1} \quad c_i = g_i . + p_i . c_{i-1}$

Carry Look Ahead: Delay (cont'd)

- Adder module
  - $t_g = t_p = t_{gate}$
- CLA module
  - $t_c = 2\ t_{gate}$
- Total
  - $t_{add} = t_{gate} + 2\ t_{gate} + t_{gate} = 4\ t_{gate}$
  - $t_{carry} = t_{gate} + 2\ t_{gate} = 3\ t_{gate}$



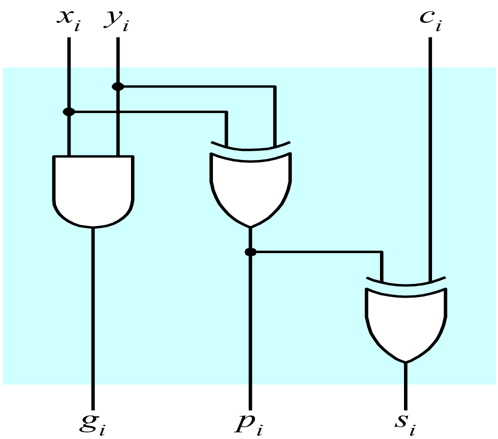
(a) (b)



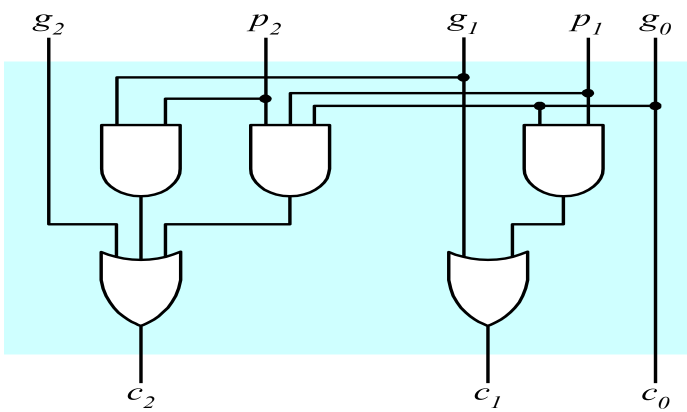
$s_i = p_i \oplus c_{i-1}$ $\quad c_i = g_i + p_i \cdot c_{i-1}$

Carry Look Ahead: Delay (cont'd)

- Adder module
  - $t_g = t_p = t_{gate}$
- CLA module
  - $t_c = 0 - 2\ t_{gate}$



(a)



(b)

$c_0 = g_0 + p_0 c_{in}$

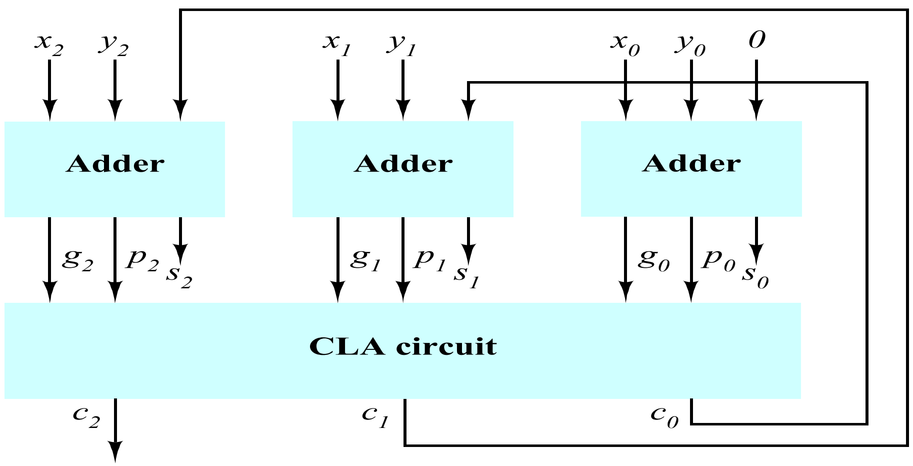
$s_0 = p_0 \oplus c_{in}$

$c_1 = g_1 + p_1 c_0$

$s_1 = p_1 \oplus c_0$

$c_2 = g_2 + p_2 c_1$

$s_2 = p_2 \oplus c_1$

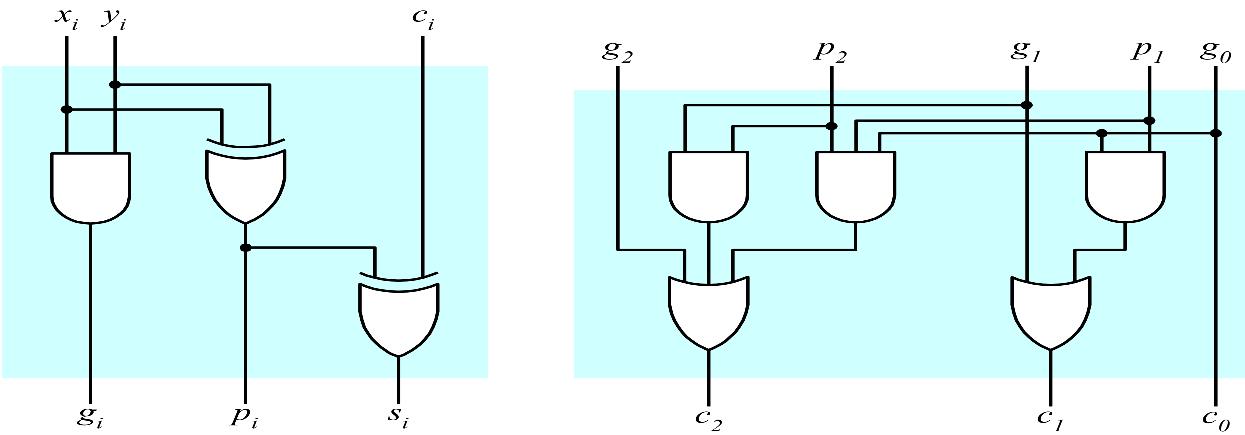


Carry Look Ahead: Delay (cont’d)

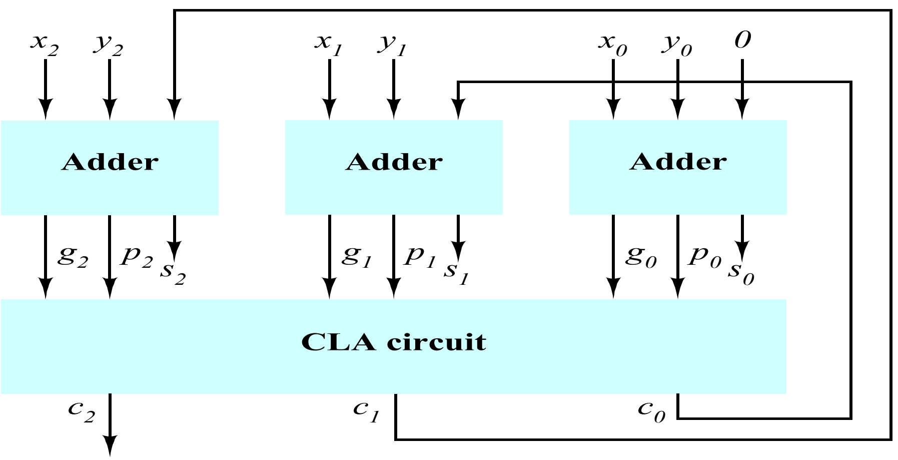
$t_g = t_p = t_{gate}$

$t_c = 0 - 2\ t_{gate}$

	Delay	Total Delay
p_i	1	1
g_i	1	1
S_0	1	2
C_0	0	1
S_1	1	2
C_1	2	3
S_2	1	4
C_2	2	3



(a) (b)



Carry Look Ahead: Summary

- CLA circuit can not be so big
  - Calculating C becomes too long
  - Cannot be evaluated in 2 gate delay
  - => At most 4 bit

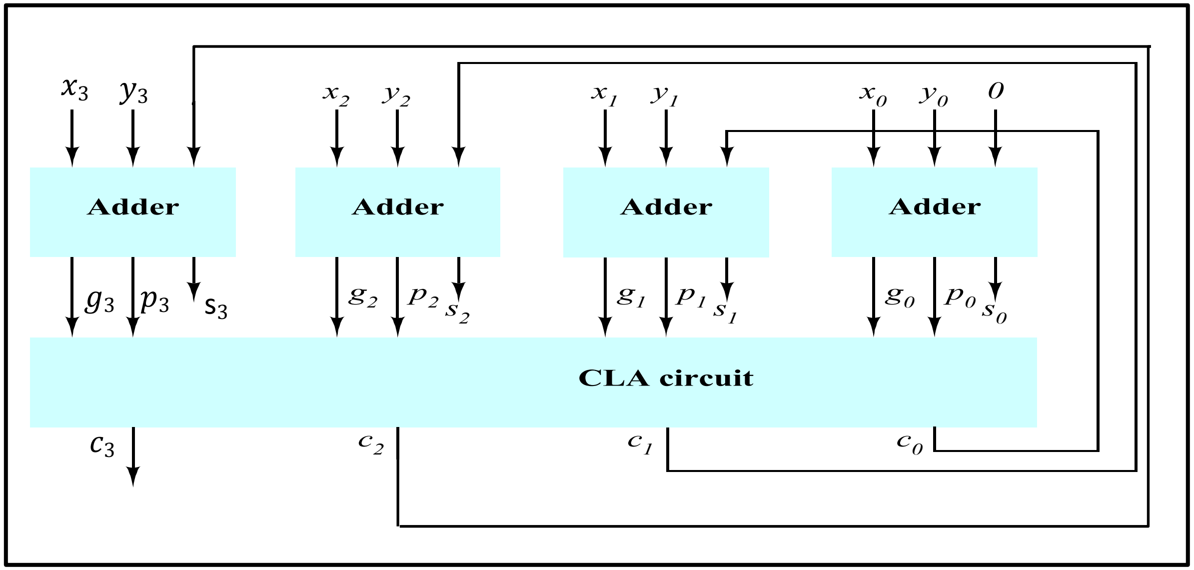
$c_0 = g_0 + p_0 c_{in}$
$s_0 = p_0 \oplus c_{in}$

$c_1 = g_1 + p_1 c_0$
$s_1 = p_1 \oplus c_0$

$c_2 = g_2 + p_2 c_1$
$s_2 = p_2 \oplus c_1$

$c_3 = g_3 + p_3 c_2$
$s_3 = p_3 \oplus c_2$

$CLA - 4$



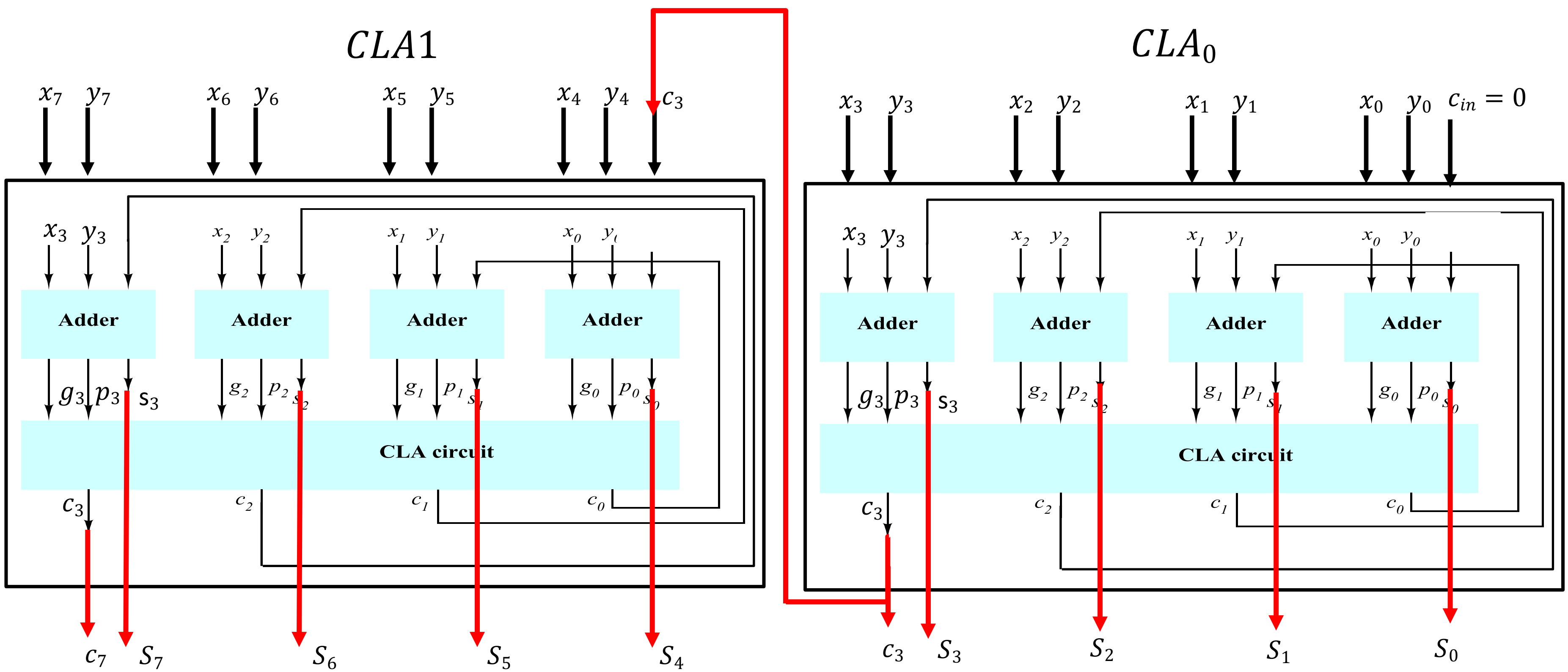
$s_i = p_i \oplus c_{i-1}$ $\quad c_i = g_i + p_i \cdot c_{i-1}$

Carry Look Ahead: Long operands

- How about long operands
- Example: add two 8-bit numbers

8-bit Carry Look Ahead: Cascade

- First design
  - Ripple carry among CLAs



8-bit Carry Look Ahead: Multi-level CLA

- Add two 8-bit numbers

$CLA-4$	$CLA-4$
$c_0 = g_0 + p_0c_{in}$ $s_0 = p_0 \oplus c_{in}$ $c_1 = g_1 + p_1c_0$ $s_1 = p_1 \oplus c_0$ $c_2 = g_2 + p_2c_1$ $s_2 = p_2 \oplus c_1$ $c_3 = g_3 + p_3c_2$ $s_3 = p_3 \oplus c_2$	$c_4 = g_4 + p_4c_3$ $s_4 = p_1 \oplus c_{in}$ $c_5 = g_5 + p_5c_4$ $s_5 = p_1 \oplus c_0$ $c_6 = g_6 + p_6c_5$ $s_6 = p_6 \oplus c_1$ $c_7 = g_7 + p_7c_6$ $s_7 = p_7 \oplus c_2$

$$c_3 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0c_{in}$$

$$c_4 = g_4 + p_4g_3 + p_4p_3g_2 + p_4p_3p_2g_1 + p_4p_3p_2p_1g_0 + p_4p_3p_2p_1p_0c_{in}$$

$$P = p_4p_3p_2p_1p_0$$

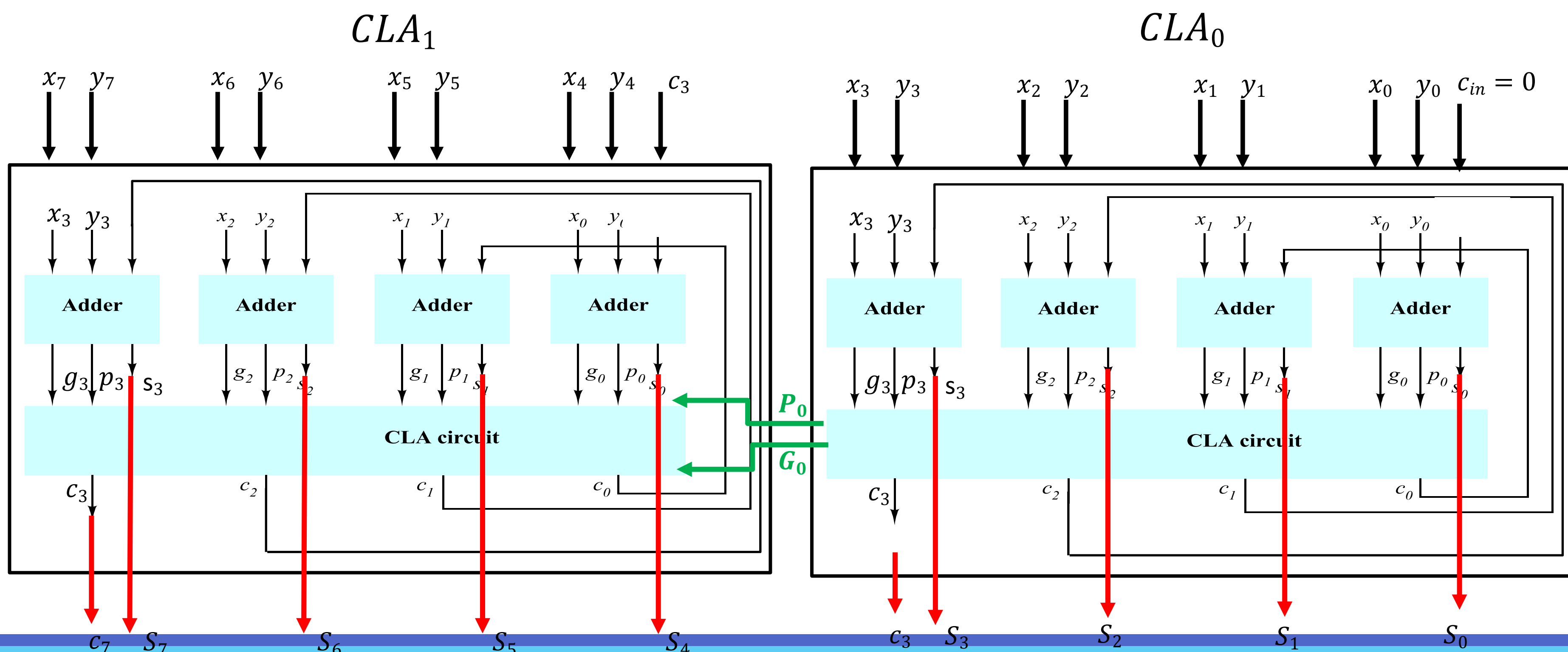
$$G = g_4 + p_4g_3 + p_4p_3g_2 + p_4p_3p_2g_1 + p_4p_3p_2p_1g_0$$

$$c_4 = G + P\,c_{in}$$

$$c_4 = G_0 + P_0\,c_{in}$$

Multi-level CLA

- CLA_1 takes the P,G's from CLA_0 and C_{in} to generate C_4 ("seed C's")
  - 2 gate delay
  - $c_4 = G + P\ c_{in}$



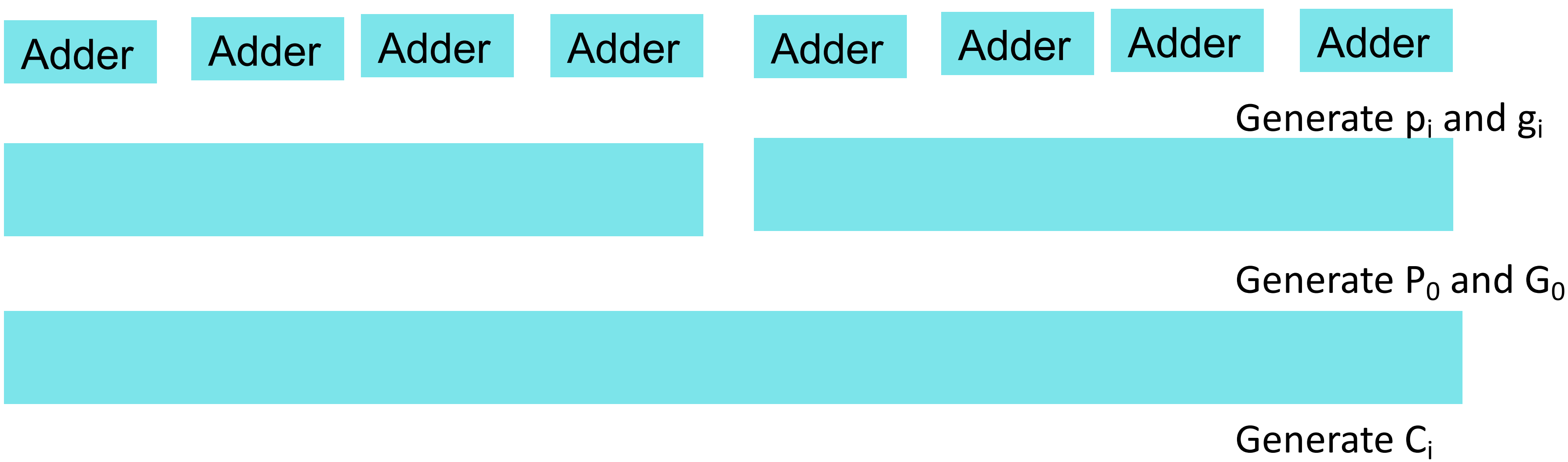
Multi-level CLA: More Detail

- Generate p_i and g_i
- Generate P_0 and G_0
- Generate C_i
- Generate S_i : t_{gate}
  - $c_4 = G + P\ c_{in}$

$P = p_4 p_3 p_2 p_1 p_0$

$G = g_4 + p_4 g_3 + p_4 p_3 g_2 + p_4 p_3 p_2 g_1 + p_4 p_3 p_2 p_1 g_0$

$c_4 = G_0 + P_0\ c_{in}$



Thank You