

Digital Logic Design

ENEE 244-020x

Lecture 11

Announcements

- Exams available at the end of lecture if you have not picked yours up yet.
- Anything not picked up after today's class will be given to TA's to hand back during discussion session.
- **One Homework will be dropped from your grade**

Agenda

New Topic: MSI Components

- Today:
 - Binary Adders and Subtractors (4.5)
 - Carry Lookahead Adders (4.5)
 - Verilog Tutorial

Scale of Integration

- Scale of Integration = Complexity of the Chip
 - SSI: small-scale integrated circuits, 1-10 gates
 - MSI: medium-scale IC, 10-100 gates
 - LSI: large scale IC, 100-1000 gates
 - VLSI: very large-scale IC, 1000+ gates
 - Today's chip can contain billions of transistors.
- MSI components: adder, subtracter, comparator, decoder, encoder, multiplexer.

Scale of Integration

- LSI technology introduced highly generalized circuit structures known as programmable logic devices (PLDs).
 - Can consist of an array of and-gates and an array of or-gates. Must be modified for a specific application.
 - Modification involves specifying the connections using a hardware procedure. Procedure is known as programming.
- Three types of programmable logic devices:
 - Programmable read-only memory (PROM)
 - Programmable logic array (PLA)
 - Programmable array logic (PAL)

MSI Components

Binary Full Adder

x_i	y_i	c_i	c_{i+1}	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

Finding a Simplified Circuit

0	1	0	1
1	0	1	0

0	0	1	0
0	1	1	1

Corresponding minimal sums:

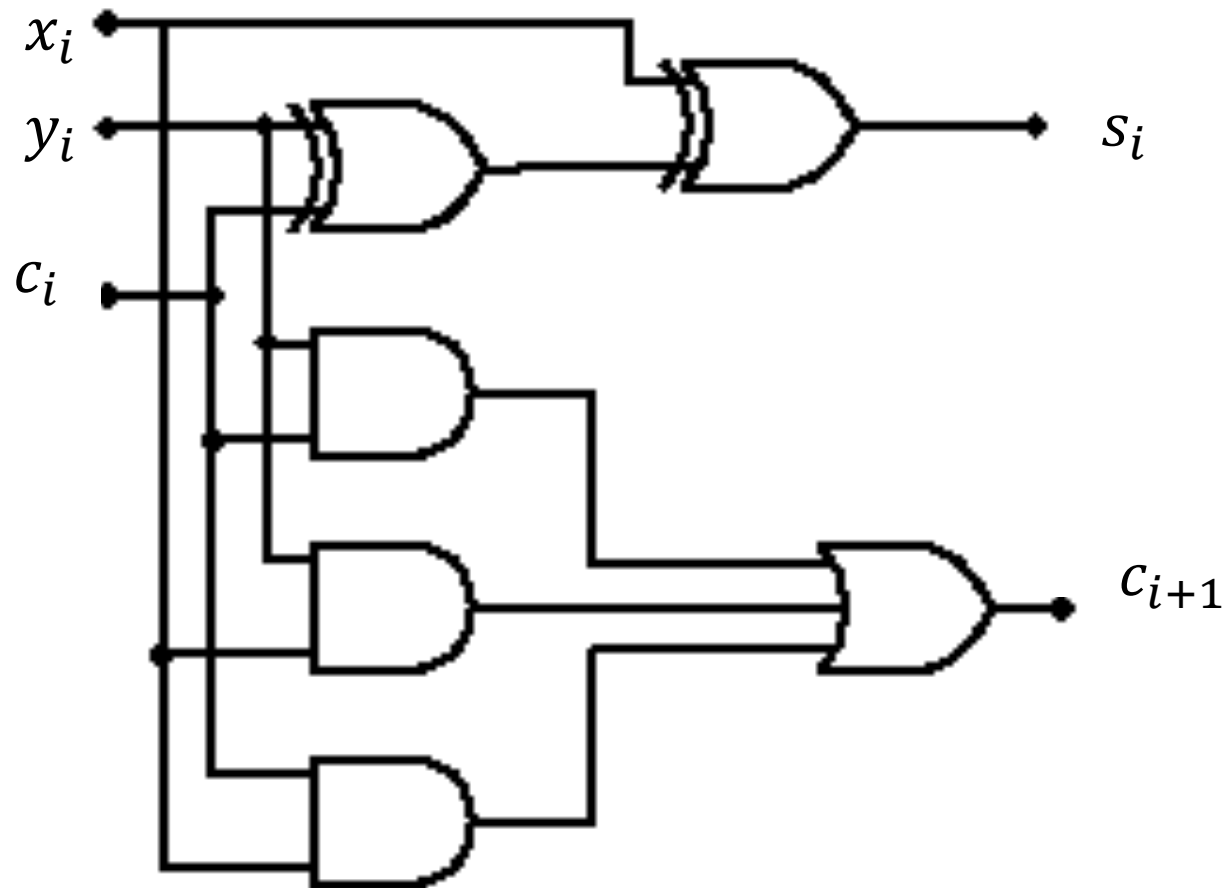
$$s_i = x'_i y'_i c_i + x'_i y_i c'_i + x_i y'_i c'_i + x_i y_i c_i$$

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

We can simplify the sum for s_i by using xor:

$$s_i = c_i \oplus x_i \oplus y_i$$

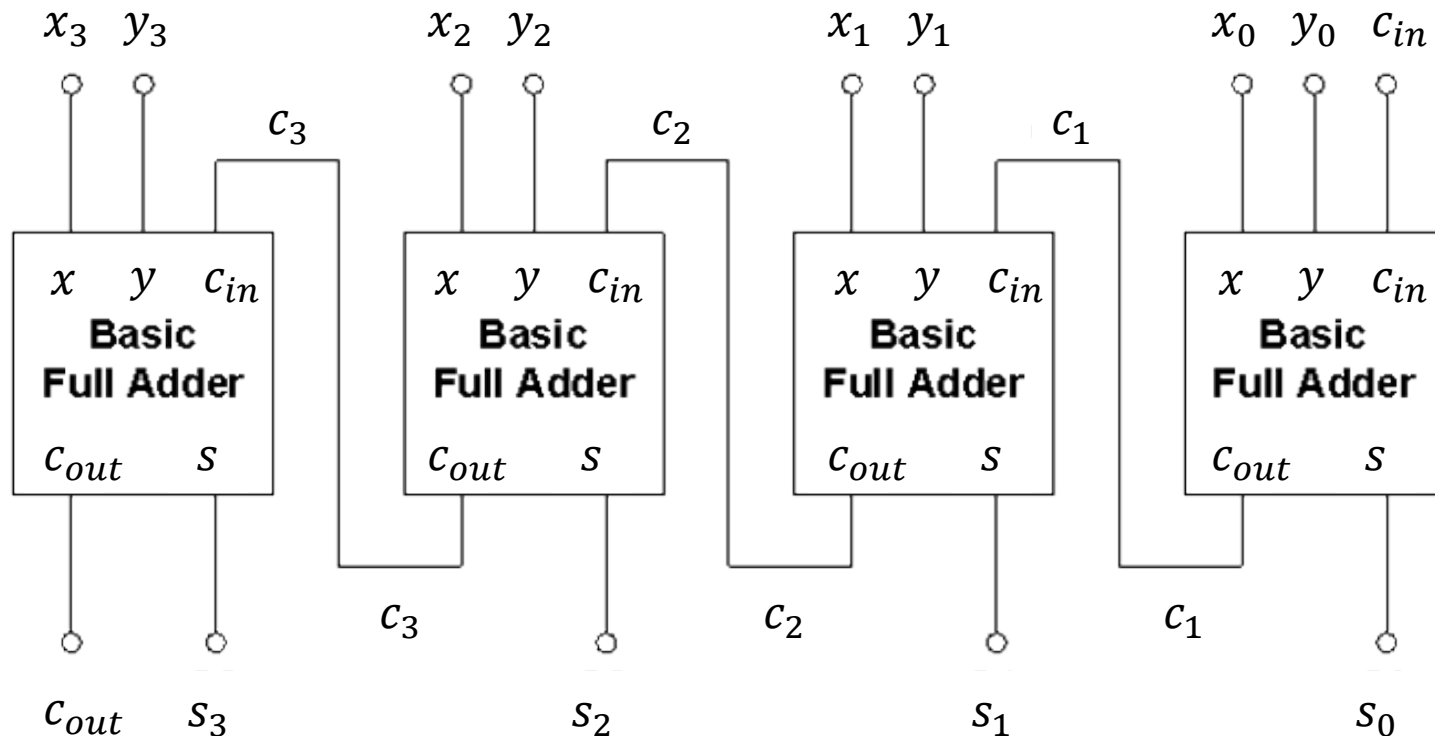
Realization of Full Binary Adder



What about many bits?

- Consider addition of two binary numbers, each consisting of n bits.
- Direct approach: Write a truth table with 2^{2n} rows corresponding to all the combinations of values and specifying the values of the sum bits. Then find a minimal combinational network.
- This will be intractable.

Parallel (ripple) Binary Adder



Why is it called “ripple” adder?
Recall—signed binary numbers,
final carry-out may signal overflow.

Carry Lookahead Adder

- Ripple effect:
 - If a carry is generated in the least-significant-bit the carry must propagate through all the remaining stages.
 - Assuming two-levels of logic are need to propogate the carry through each of the next higher-order stages. Delay is $2n$.
- Must speed up propagation of the carries. Adders designed with this consideration in mind are called **high-speed adders**.

Carry Lookahead Adder

- Consider $c_{i+1} = x_i y_i + x_i c_i + y_i c_i$
$$= x_i y_i + (x_i + y_i) c_i$$
- The first term $x_i y_i$ is called the **carry-generate function** since it corresponds to the formation of a carry at the i-th stage.
- The second term $(x_i + y_i) c_i$ corresponds to a previously generated carry c_i that must propagate past the i-th stage to the next stage.
- The $(x_i + y_i)$ part of this term is called the **carry-propagate function**.
- Carry-generate function will be denoted by g_i , carry-propagate function will be denoted by p_i .

Carry Lookahead Adder

$$g_i = x_i y_i$$

$$p_i = x_i + y_i$$

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

Using this general result, the output carry at each of the stages can be written in terms of the g 's, p 's and initial input carry c_0 .

Carry Lookahead Adder

$$\begin{aligned}
 & \underline{c_1 = g_0 + p_0 c_0} \\
 & \underline{c_2 = g_1 + p_1 c_1} \\
 & = g_1 + p_1(g_0 + p_0 c_0) \\
 & = \underline{g_1 + p_1 g_0 + p_1 p_0 c_0} \\
 & \underline{c_3 = g_2 + p_2 c_2} \\
 & = g_2 + p_2(g_1 + p_1 g_0 + p_1 p_0 c_0) \\
 & = \underline{g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0}
 \end{aligned}$$

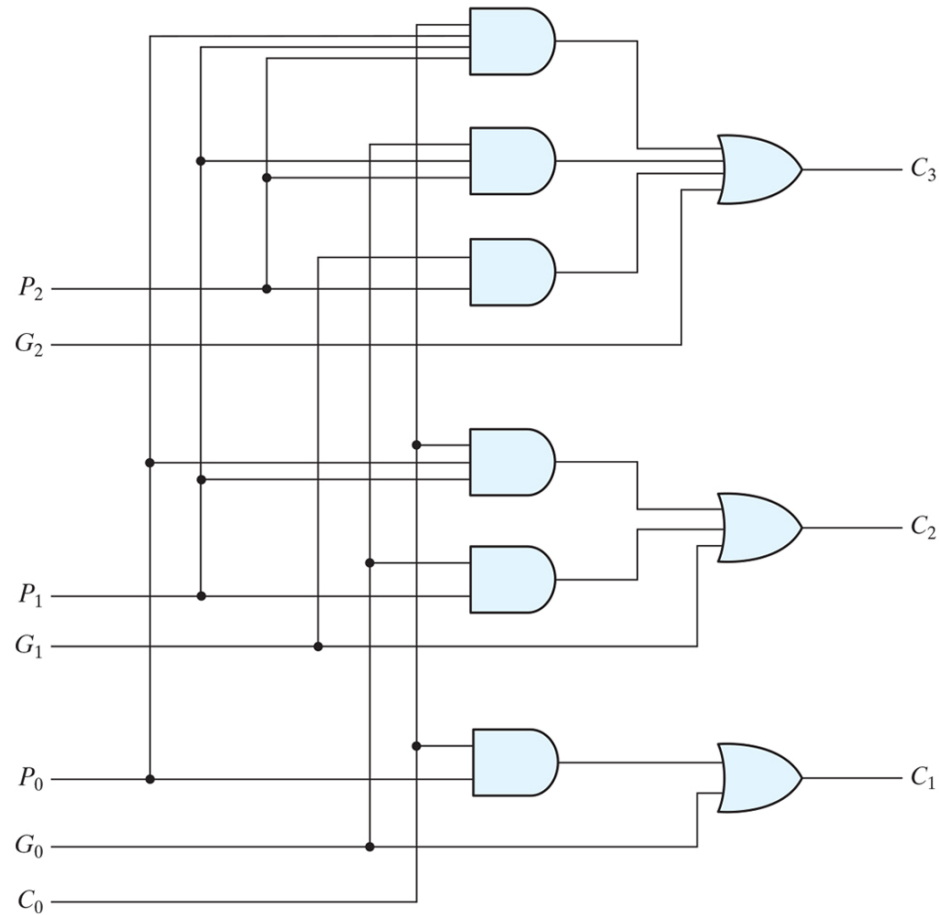
C_{i+1} can be
computed in
2 layers of logic
↓ give n all
the p_i 's, g_i 's
and C_0

C_{i+1}

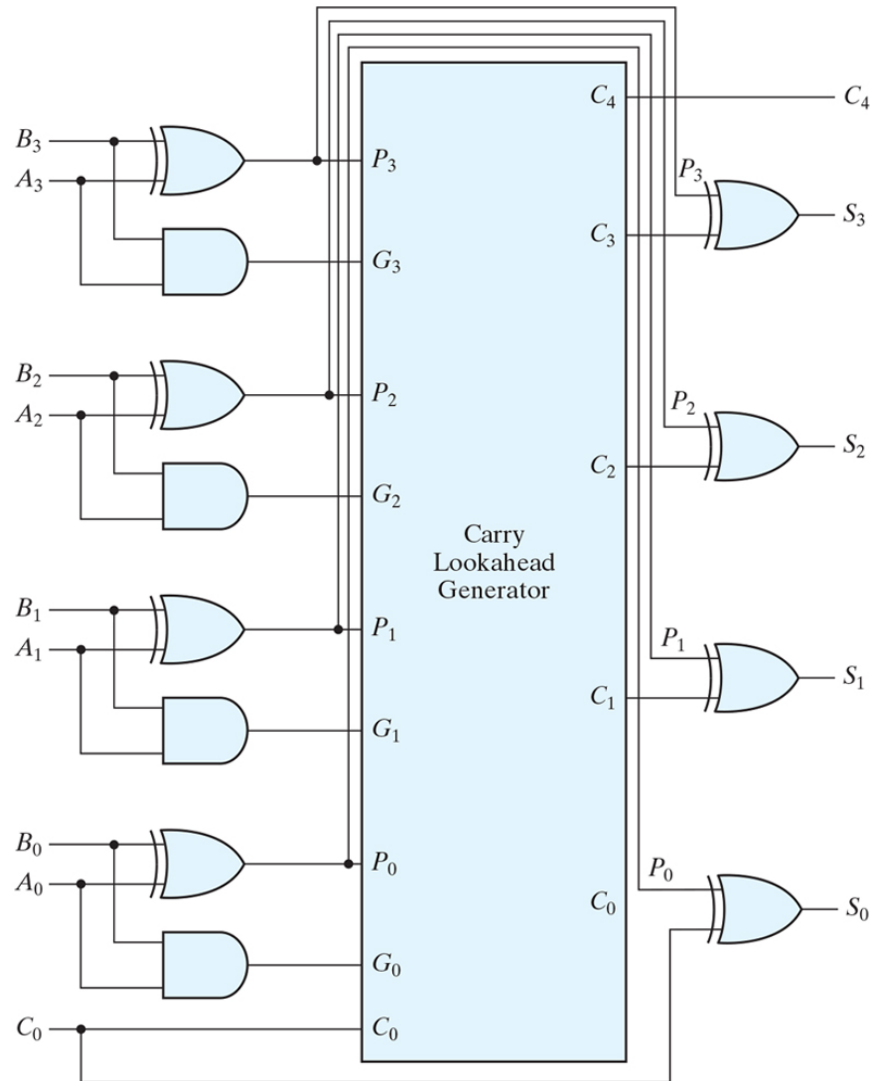
$$= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \dots + p_i p_{i-1} \dots p_1 g_0 + p_i p_{i-1} \dots p_0 c_0$$

Why is this a good idea? Do we save on computation?

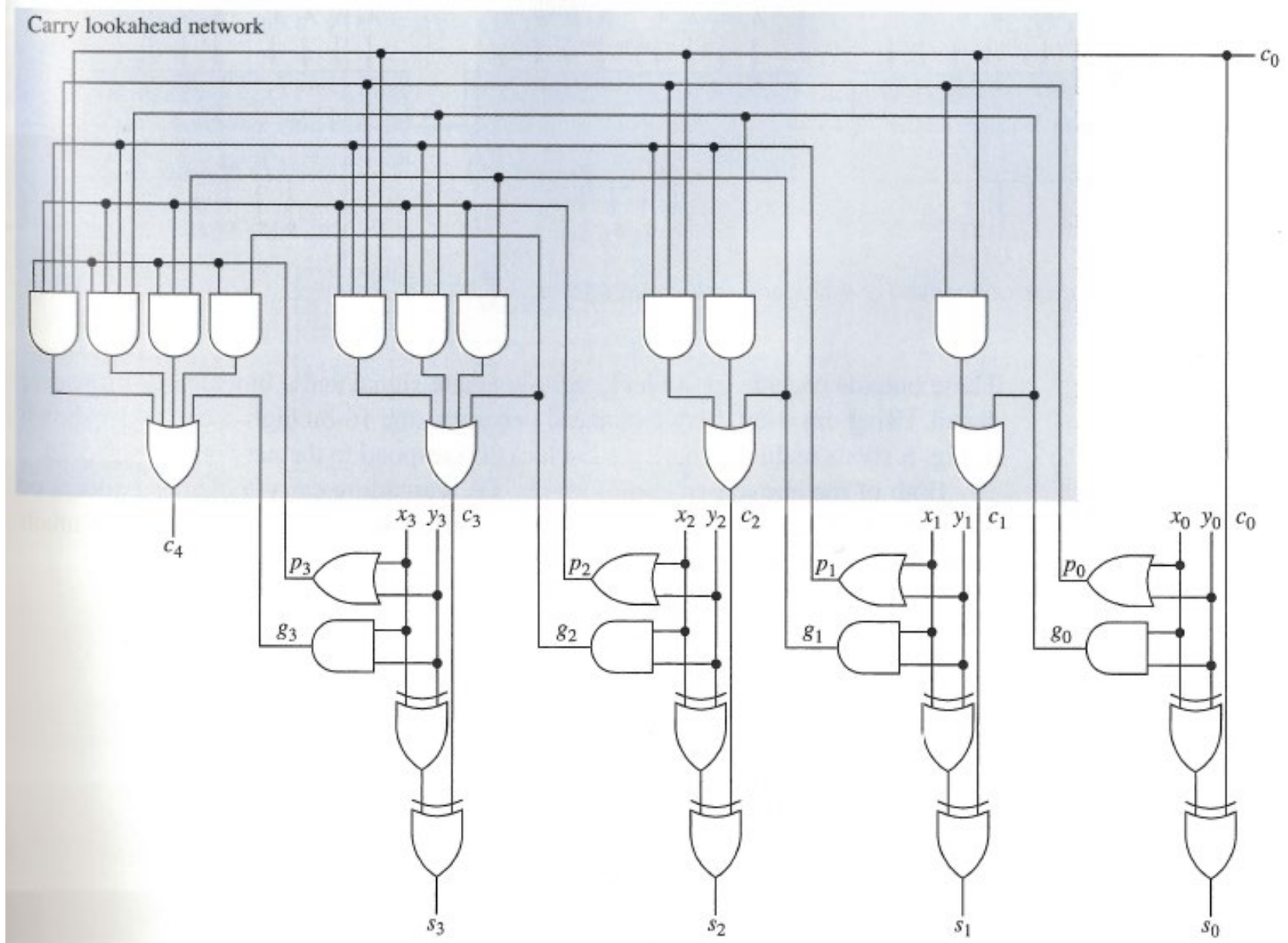
Carry Lookahead Generator



Carry Lookahead Adder



Carry Lookahead Adder



Carry Lookahead Adder

- What is the delay?
 - One level of logic to form g 's, p 's
 - Two levels of logic to propagate through the carry lookahead
 - One level of logic to have the carry effect a sum output.
 - Total: 4 units of time.
- Delay of 4-bit ripple-adder?
 - 2 levels of logic for each c_1, c_2, c_3, c_4
 - 8 levels of logic

Large High-Speed Adders

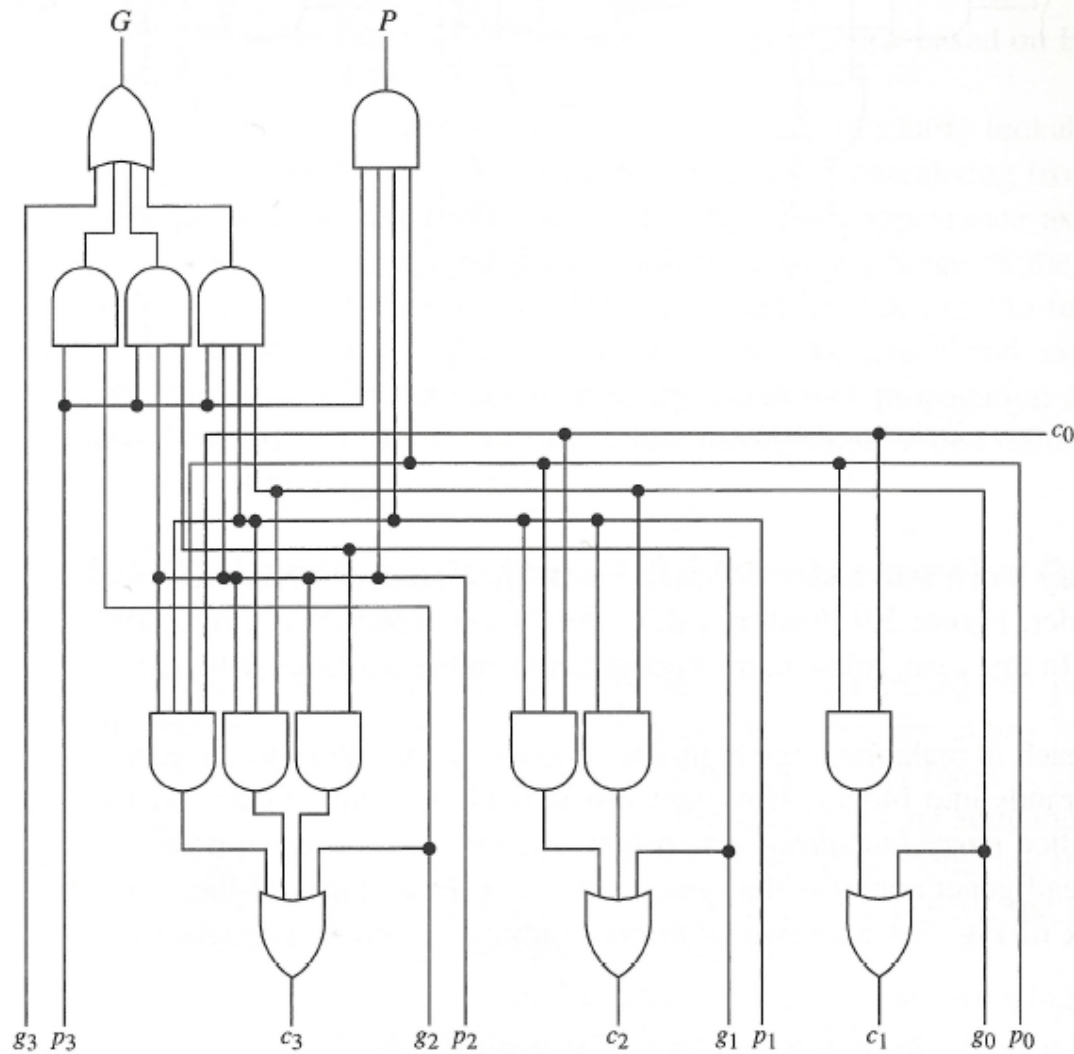
- The carry lookahead network can very large as the number of bits increases.
- Approach: Divide bits of the operands into blocks, use carry lookahead adders for each block. Cascade the adders for the blocks.
- Ripple carries occur between the cascaded adders.

Another Approach to Large High-Speed Adders

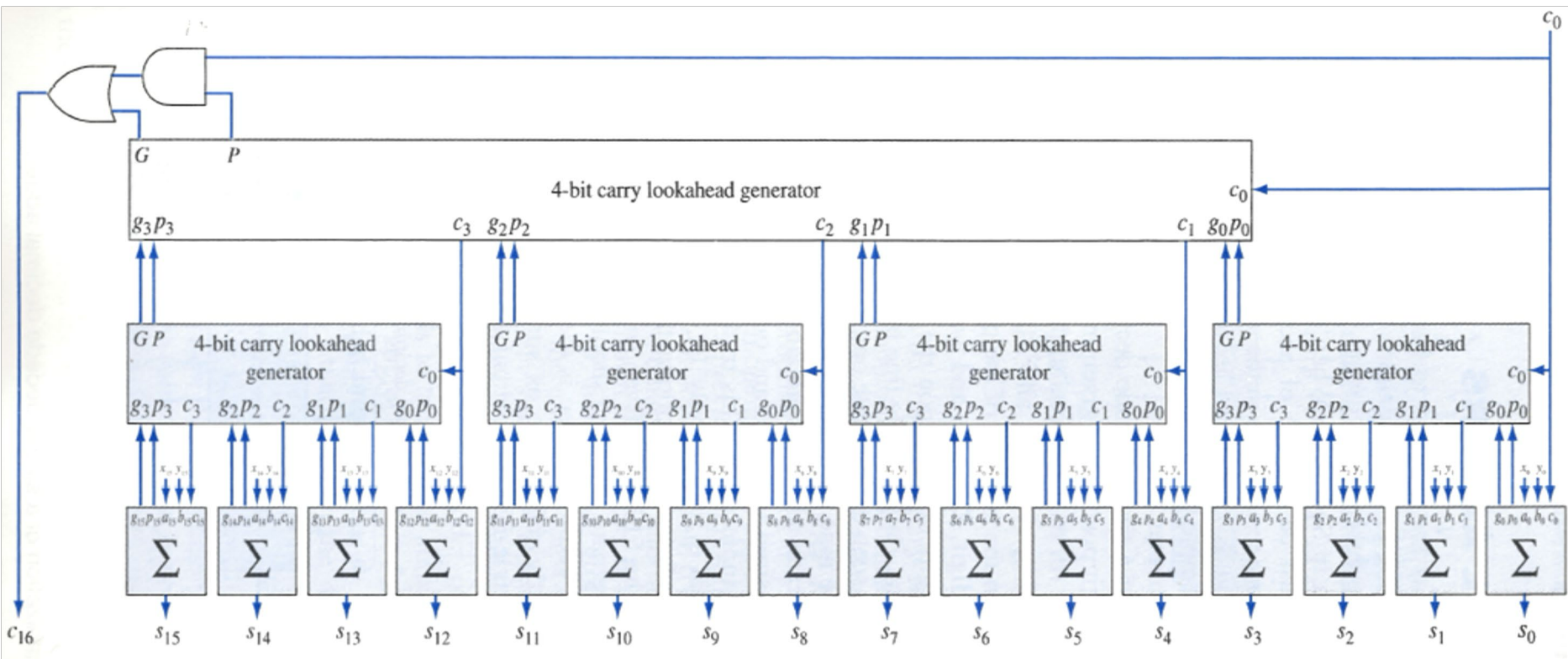
- Carry lookahead generators that generate the carry of an entire block.
- Assume 4-bit blocks.
- For each block, 4-bit carry lookahead generator outputs:

$$G = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0$$
$$P = p_3p_2p_1p_0$$

Carry Lookahead Generator



Large High-Speed Adders



Binary Subtractors

Compute: $x_i - y_i$.

b_i is a borrow-in bit from previous bit-order position.

b_{i+1} is a borrow-out bit.

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

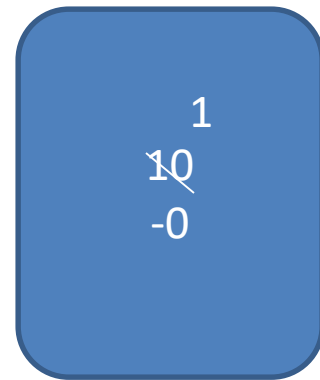
Binary Subtractors

Compute: $x_i - y_i$.

b_i is a borrow-in bit from previous bit-order position.

b_{i+1} is a borrow-out bit.

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



Binary Subtractors

Compute: $x_i - y_i$.

b_i is a borrow-in bit from previous bit-order position.

b_{i+1} is a borrow-out bit.

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

10
-1

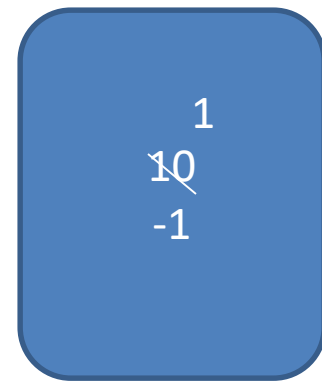
Binary Subtractors

Compute: $x_i - y_i$.

b_i is a borrow-in bit from previous bit-order position.

b_{i+1} is a borrow-out bit.

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0		
1	0	1		
1	1	0		
1	1	1		



Binary Subtractors

Compute: $x_i - y_i$.

b_i is a borrow-in bit from previous bit-order position.

b_{i+1} is a borrow-out bit.

$$\begin{array}{r} 0100 \\ - 010 \\ \hline 10 \end{array}$$

$i=2 \quad i=1 \quad i=0$

$$b_0 = 0$$

$$b_1 = 0$$

$$b_2 = 1$$



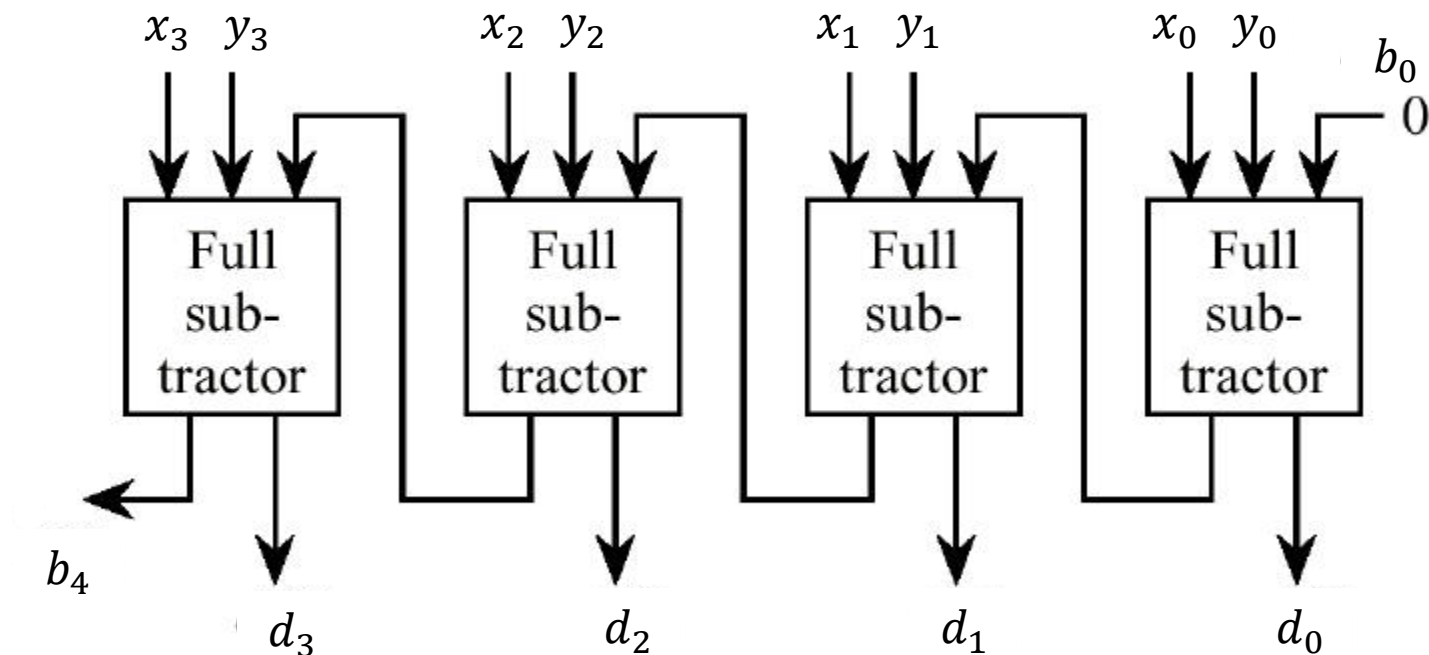
x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



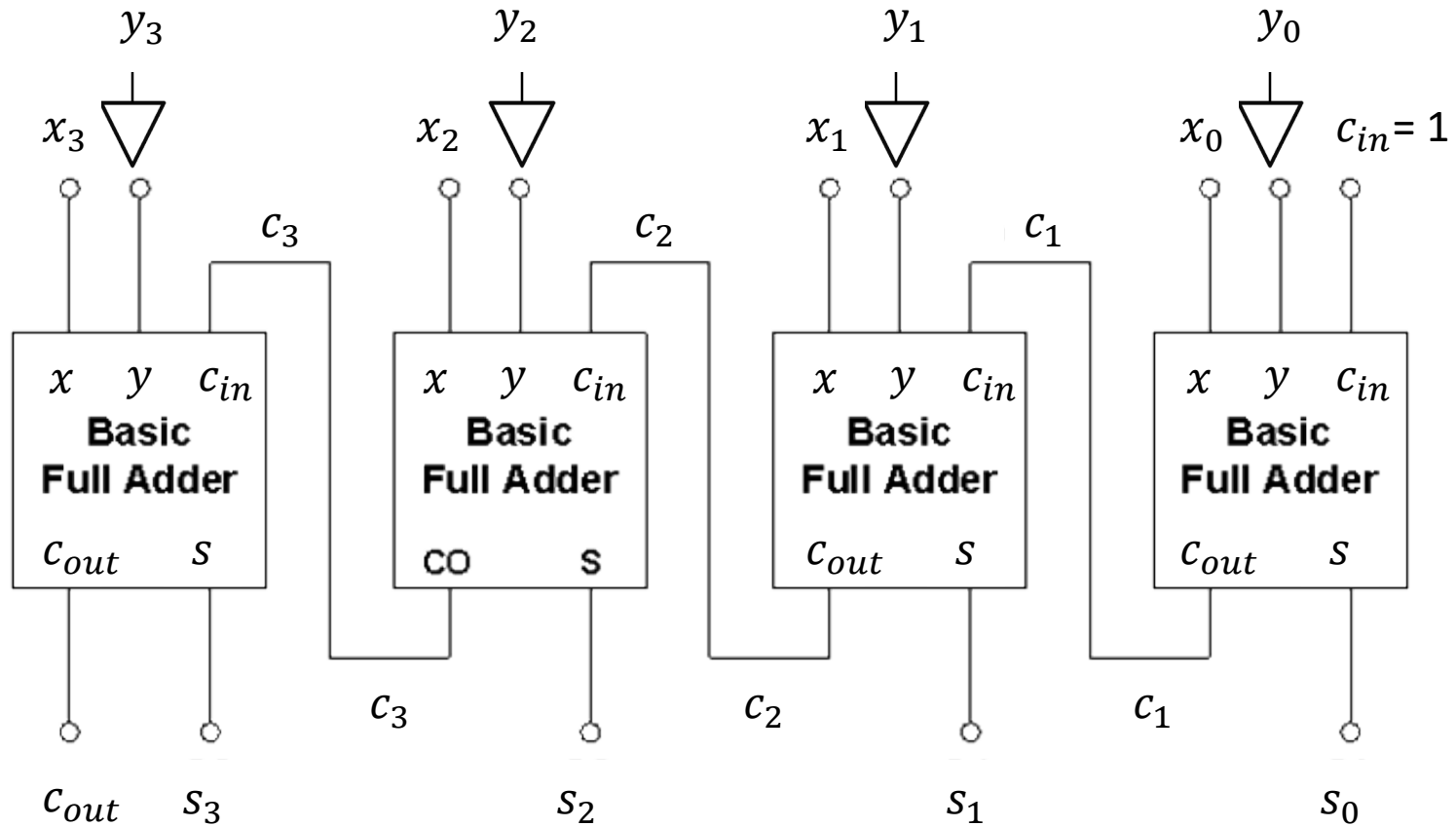
Finding a Simplified Circuit

$$d_i = b_i \oplus x_i \oplus y_i \text{ (Same as sum in adder)}$$

$$b_{i+1} = x'_i y_i + x'_i b_i + y_i b_i$$



A better approach using 2's complement



Parallel Adder/Subtractor

