Department of Computer Science & Engineering (CSE)

Course Title: Digital Logic Design

Shahriar Rahman Khan
Lecturer
Dept. of CSE, MIST

Course Code: CSE 103
Credit Hr: 3.00
Contact Hr: 3.00

Overview

- What is MSI and PLD?
- Binary parallel Adder
- Binary Adder-Subtractors
- Carry Propagation
- Decimal Adder
  - BCD Adder
- Magnitude Comparator
- Decoders and Encoders
- Priority Encoders
- De-multiplexers and Multiplexers

MSI Components

- 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 has millions of gates on it.

- A combinational circuit designed with individual gates can be implemented with SSI circuits that contain several independent gates.
- The number of gates in an SSI circuit is limited by the number of pins in it, (generally 14 – 16 pins).
- Medium Scale Integration components perform specific digital functions commonly needed in the design of digital systems.
- MSI components: adder, subtracter, comparator, decoder, encoder, multiplexer.

PLD Components

- 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)

Full-Adder

You remember this combinational circuit named Full-Adder from chap 4. As this is a combinational circuit that adds three 1 bit binary digits, so there will be 3 input variables and 2 output variables.

C 10
01
+ 01
10

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

Full-Adder

0	1	0	1
1	0	1	0

0	0	1	0
0	1	1	1

Corresponding minimal sums:

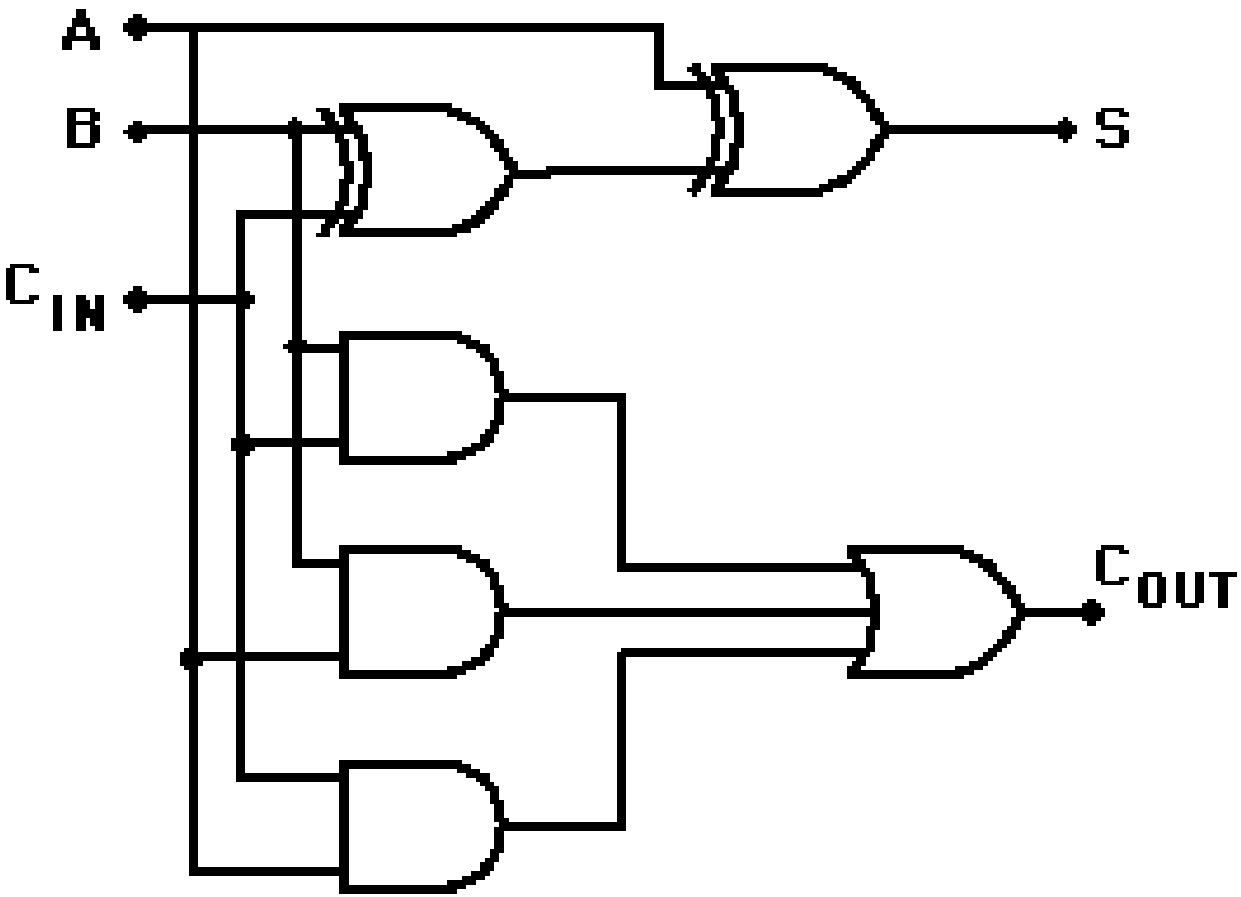
$$s_i = \overline{x}_i\overline{y}_i c_i + \overline{x}_i y_i \overline{c}_i + x_i \overline{y}_i \overline{c}_i + x_i y_i c_i$$

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

We can simplify the sum for s_i by using xor:

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

Full-Adder



Full-Adder

- So far, we have seen that, we can add 2 binary numbers, each consisting of 1 bit. For example,

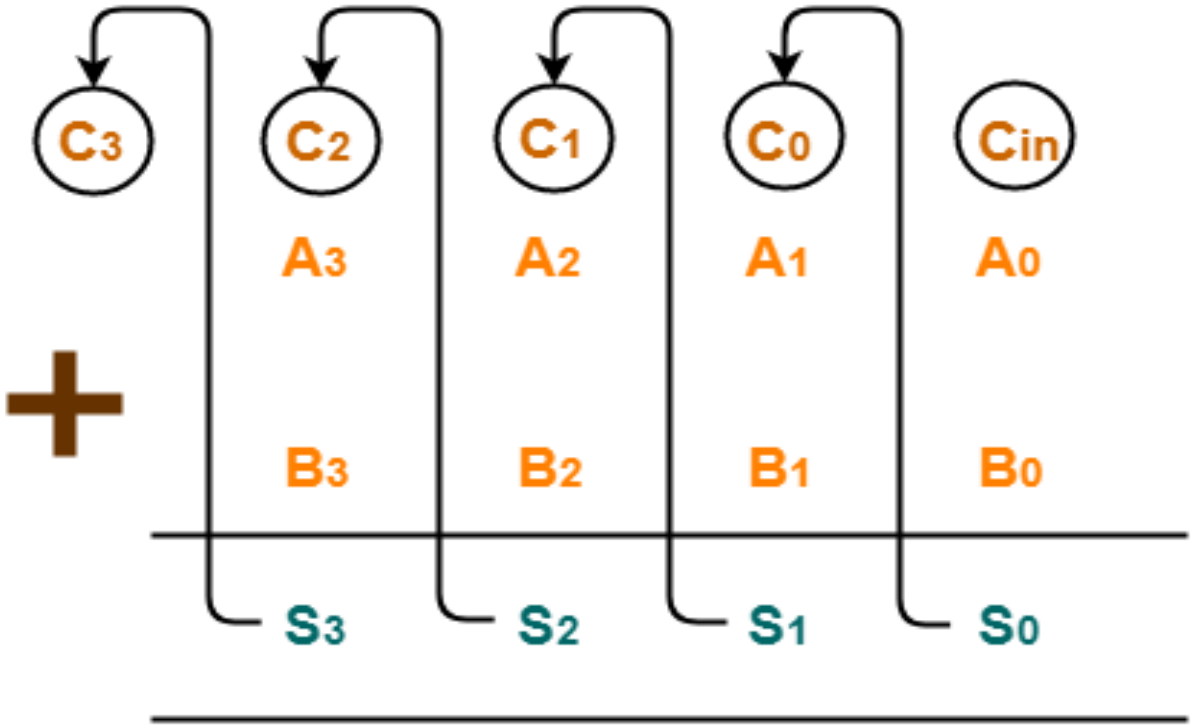
```
  0
 +1
 ---
  1
```

  - But consider this scenario, we want to add two binary numbers, each consisting of n bits.

```
  0110101
 +1011001
 --------
```

- One direct approach, write a truth table with 2^2n rows corresponding all the combinations of values and also specifying the values of the sum bits. But, that's really time consuming.

What About Many Bits?



Adding two 4-bit Numbers

What About Many Bits?

Subscript i	4	3	2	1		Full-adder of Fig. 4-5
Input carry	0	1	1	0	C_i	z
Augend	1	0	1	1	A_i	x
Addend	0	0	1	1	B_i	y
Sum	1	1	1	0	S_i	S
Output carry	0	0	1	1	C_{i+1}	C

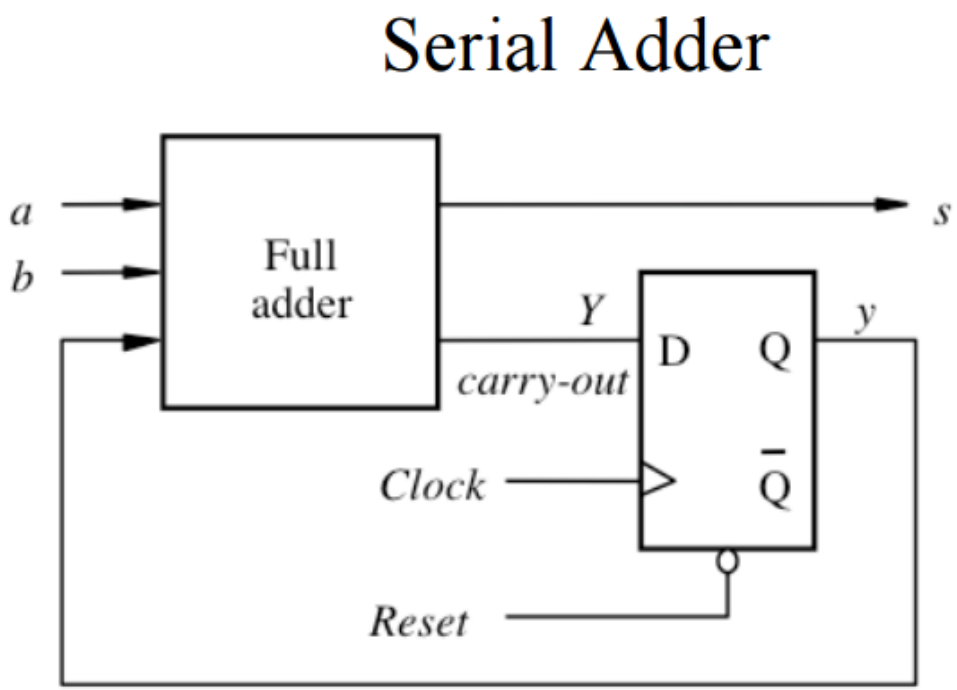
- Here, Let A=1011 and B=0011. The bits are added with full adders, starting from the least significant position. This will form a sum bit and a carry bit.
- The input carry C1 must be 0. The value of C_{i+1} in a given significant position is the output carry of the full adder.

What About Many Bits?

Subscript i	4	3	2	1		Full-adder of Fig. 4-5
Input carry	0	1	1	0	C_i	z
Augend	1	0	1	1	A_i	x
Addend	0	0	1	1	B_i	y
Sum	1	1	1	0	S_i	S
Output carry	0	0	1	1	C_{i+1}	C

- There are two ways to implement this: 1. Serial addition 2. Parallel addition.
- Serial addition: It uses only one full adder and a storage device to hold the generated output carry.
- Parallel addition: It uses n full adder and all bits of A and B are applied simultaneously. The output carry of one full adder is connected to the input carry of next full adder.

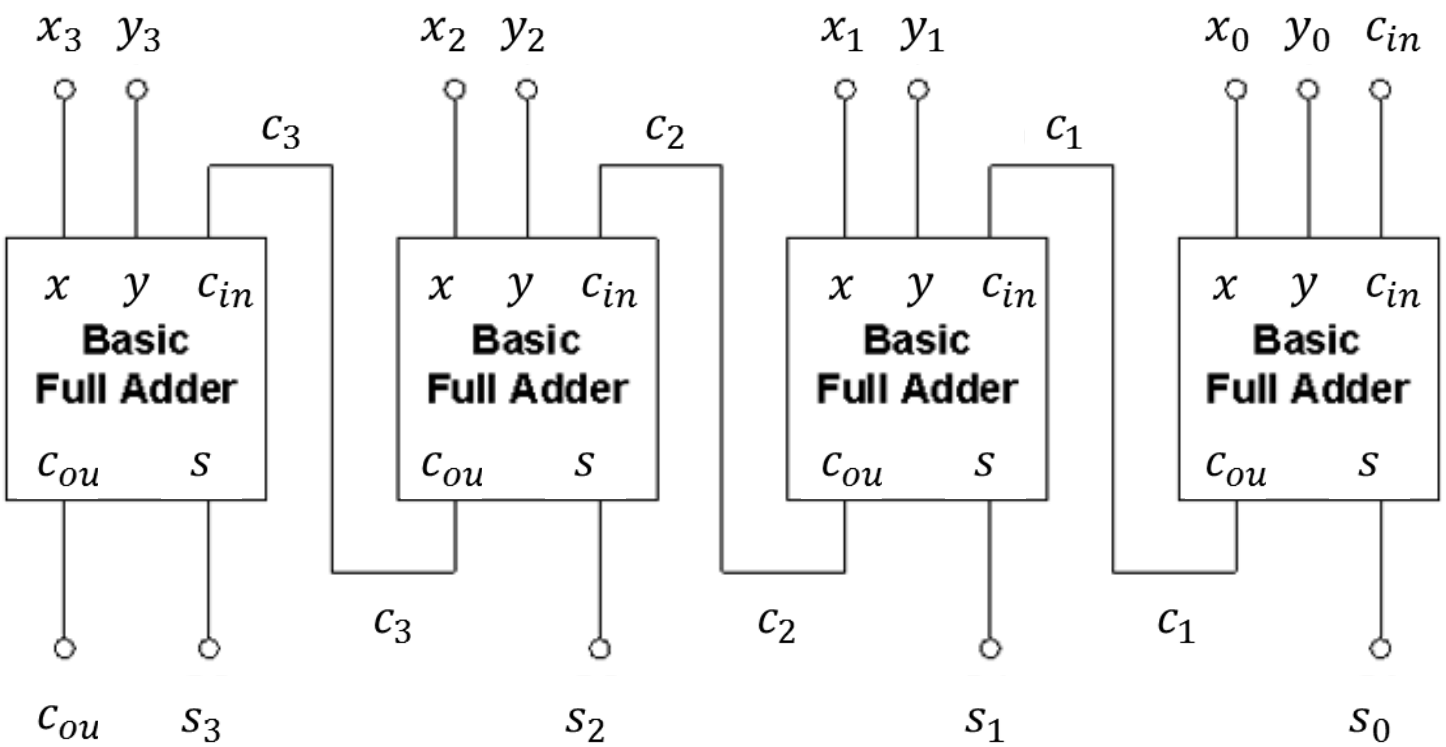
Serial Binary Adder



Q. Why don't we connect Carry out directly with Carry in?

=>Because I have to make Cin=0 while adding the least significant bit, and to do that, I have to make a system that will initially provide that 0 to Cin.

Parallel (ripple) Binary Adder



Why is it called “ripple” adder?

Full-Subtractor

A Full-Subtractor is a combinational circuit that performs a subtraction between two bits, assuming that 1 may have been borrowed by a lower significant stage.

Compute: $x_i - y_i$.

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

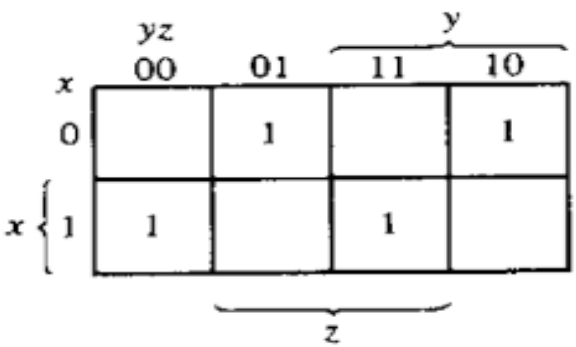
b_{i+1} is a borrow-out bit.

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	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

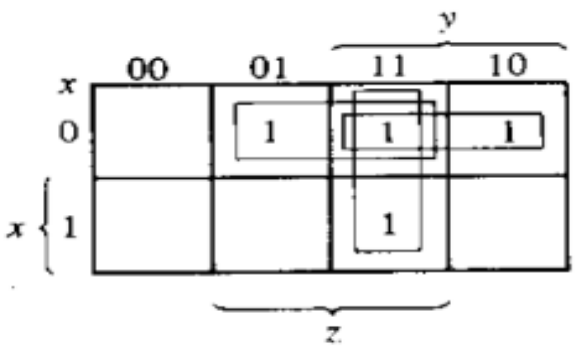
Full-Subtractor

$$D = x'y'z + x'yz' + xy'z' + xyz$$

$$B = x'y + x'z + yz$$



$D = x'y'z + x'yz + xy'z' + xyz$



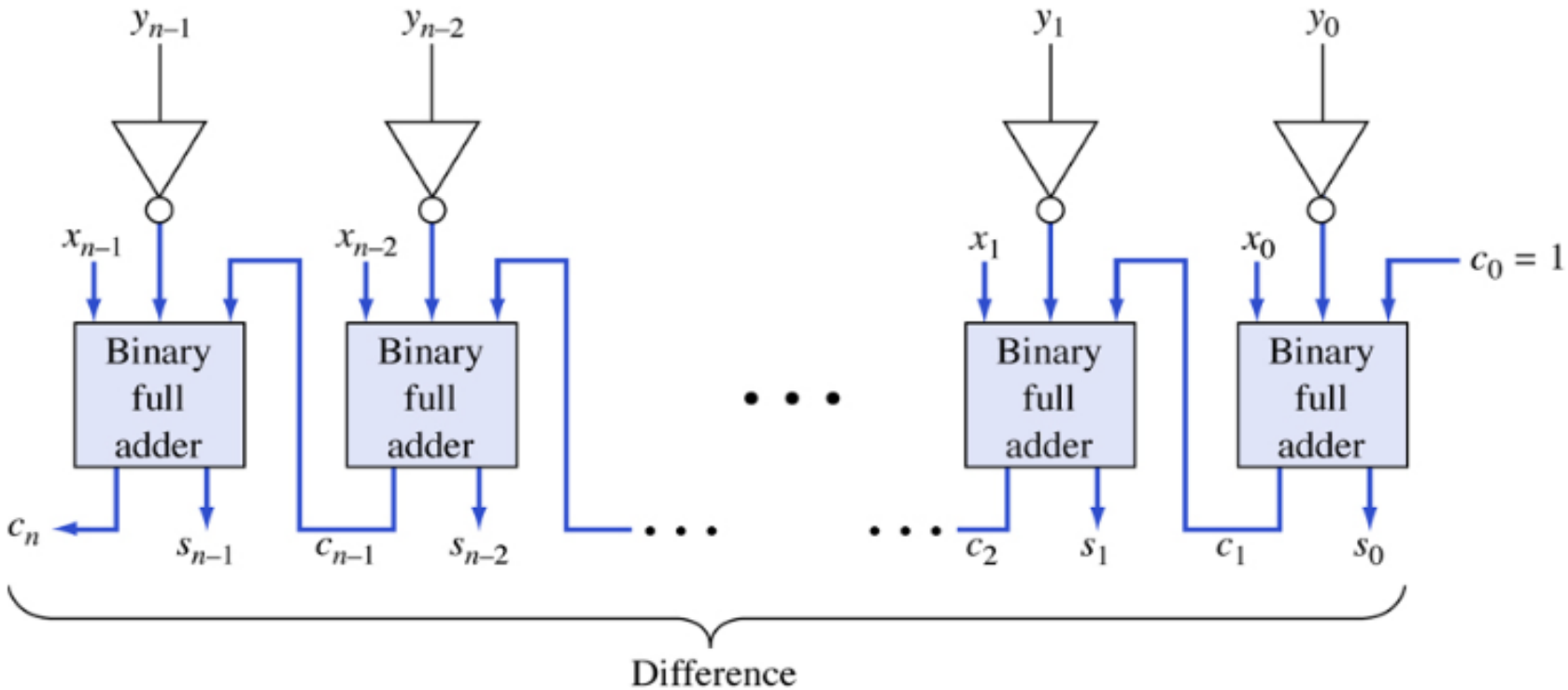
$B = x'y + x'z + yz$

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

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

Binary Subtractor using 2's complement

- Here, we are doing x-y. We know from 2's complement, (-y) can be achieved by $\bar{y}$+1. So, x-y = x+$\bar{y}$+1.



Binary Subtractor using 2's complement

- Here, we are doing x-y. We know from 2's complement, (-y) can be achieved by $\bar{y}$+1. So, x-y = x+$\bar{y}$+1. Which means the 2's complement of a number can be achieved by taking 1's complement and adding 1 to the least significant bit.

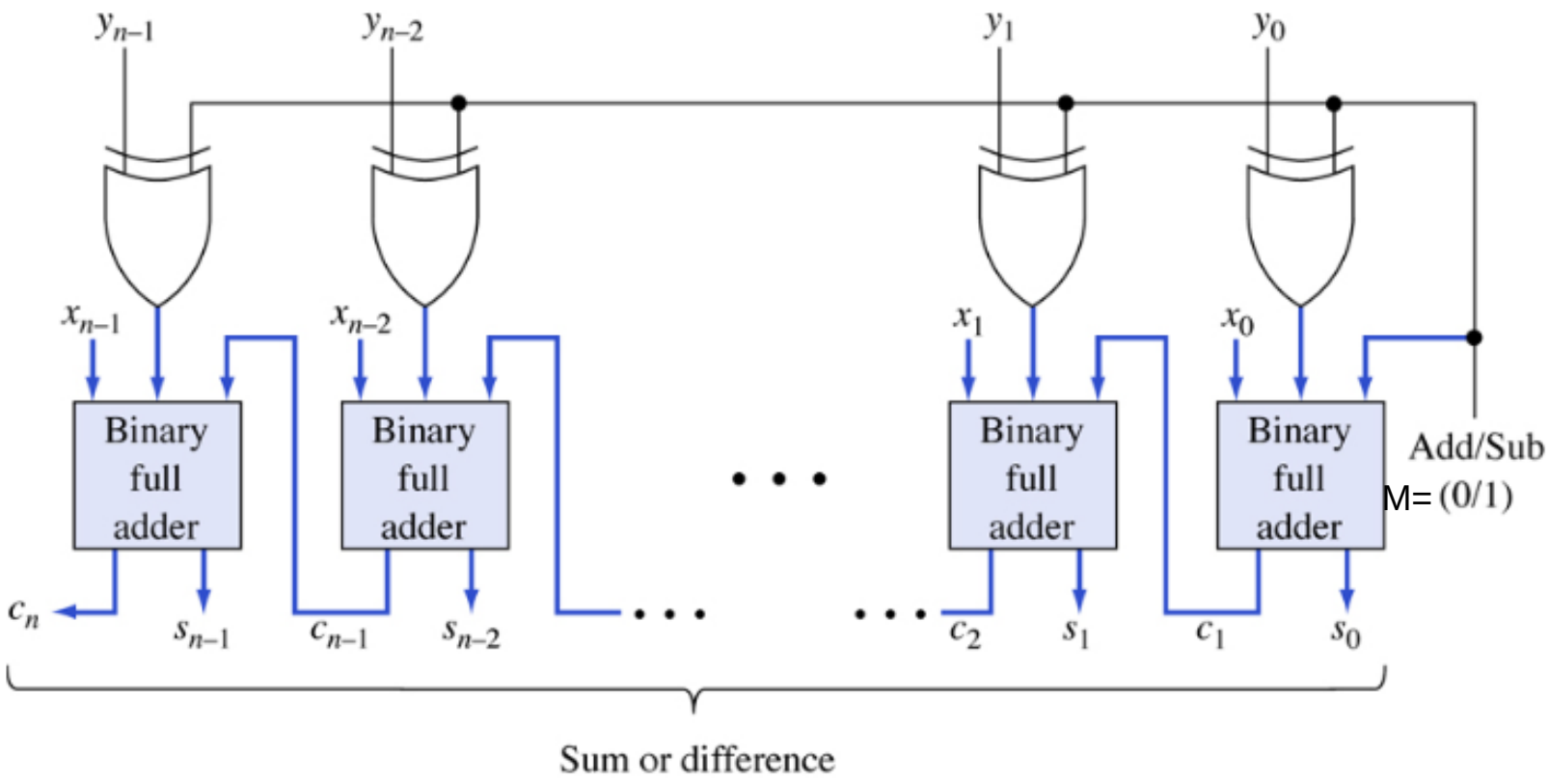
- The 1's complement can be implemented with inverters and a one can added to the sum through the input carry. Thus, the input carry C1 must be equal to 1 to perform subtraction.

Parallel Binary Adder/Subtractor

- This is an XOR gate. As you can see, when y = 0, then output=x
- When y = 1, then output= x'. We will use this concept to create a subtractor which will be called parallel binary adder/subtractor.

x	y	Output
0	0	0
1	0	1
0	1	1
1	1	0

Parallel Binary Adder/Subtractor



Parallel Binary Adder/Subtractor

- Here, mode input M controls the operation.
- When M=0, the circuit acts as an adder. Because that time, B xor 0 = B. As we saw previously. Also, C_1=0.The full adders receive the value of B and the input carry is 0. Thus, this circuit performs as A+B, which is addition.

- When M=1, the circuit acts as a subtractor. Because that time, B xor 1 = B'. Also, C_1=1. That means, the B inputs are all complemented and a 1 is added through the input carry. Thus, this circuit performs as A+B'+1 = A – B, which is subtraction.

Parallel (ripple) Binary Adder

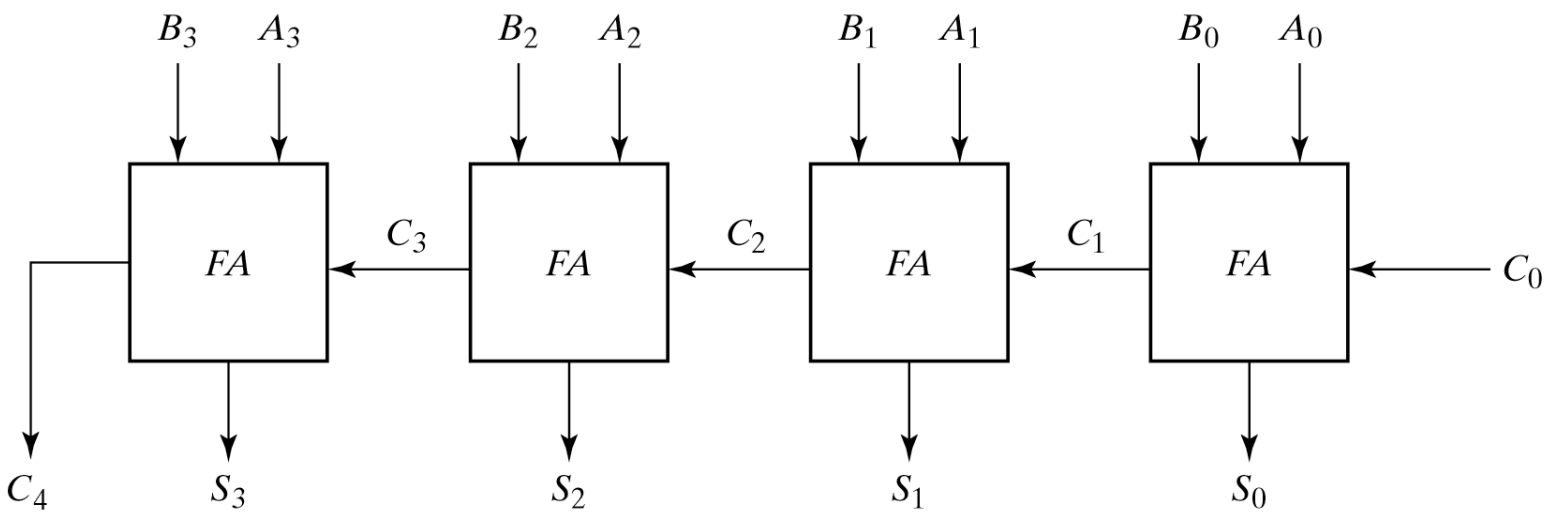


Fig. 4-9 4-Bit Adder

Carry Propagation Delay

- Ripple effect: If a carry is generated in the least-significant-bit the carry must propagate through all the remaining stages.
- Since each bit of the sum output S_i depends on the value of the input carry, the value of Si in any given stage in the adder will be in it's steady state (give a final value) only after the input carry to that stage has been propagated.
- Consider output S_3 in previous slide. Inputs A_3 and B_3 reach a steady state value as soon as input signals are applied to the adder. But input carry C_3 doesn't settle to it's final steady state, until C_2 is available in it's steady state value.
- Similarly, C_2 has to wait for C_1, C_1 has to wait for C_0 and so on.
- Must speed up propagation of the carries. Adders designed with this consideration in mind are called high-speed adders.

Carry Propagation Delay

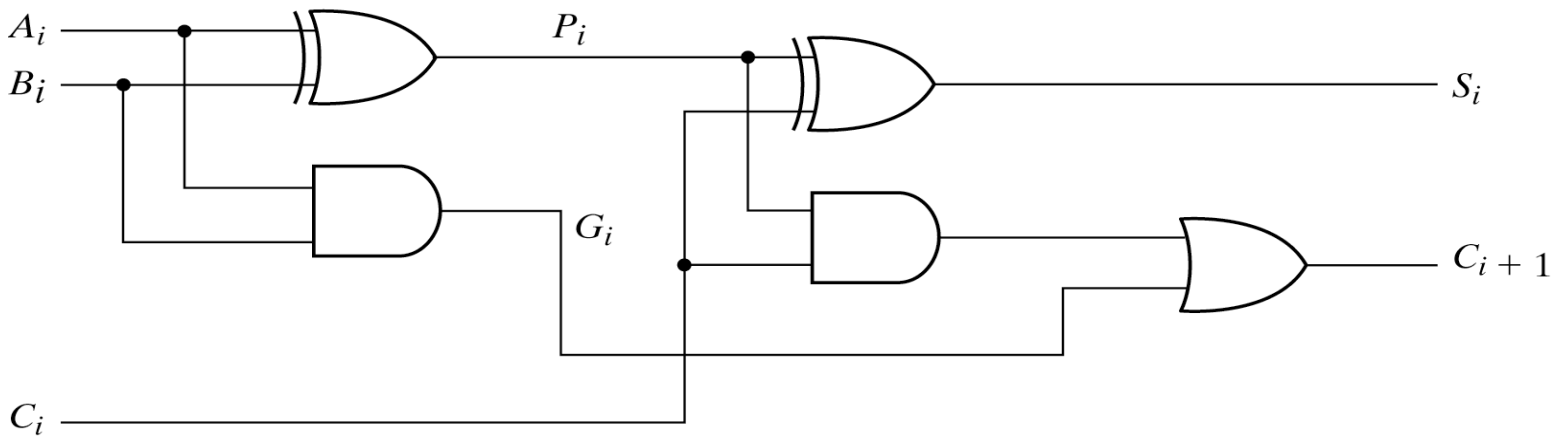


Fig. 4-10 Full Adder with P and G Shown

- P_i and G_i depends only on the input augend(A_i) and addend(B_i) bits.
- But the signal from the input carry Ci to the output carry Ci+1 propagates through an AND gate and an OR gate (Two level gates).

Carry Propagation Delay

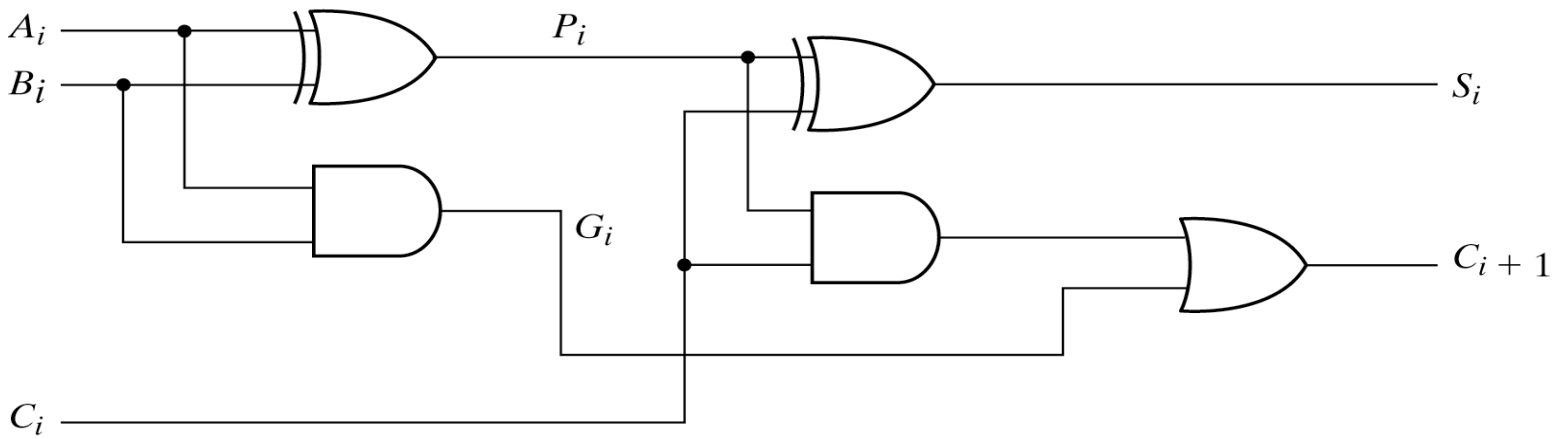


Fig. 4-10 Full Adder with P and G Shown

- If there are 4 full adders in a parallel adder, the output carry C4 will have 2x4=8 gate levels from Cin to C4.
- Total propagation time = one half adder + 8 gate levels
- For n bit parallel adder, there are 2n gate levels for the carry to propagate through.
- Carry propagation time is a limiting factor on the speed.

Carry Look-ahead Adder

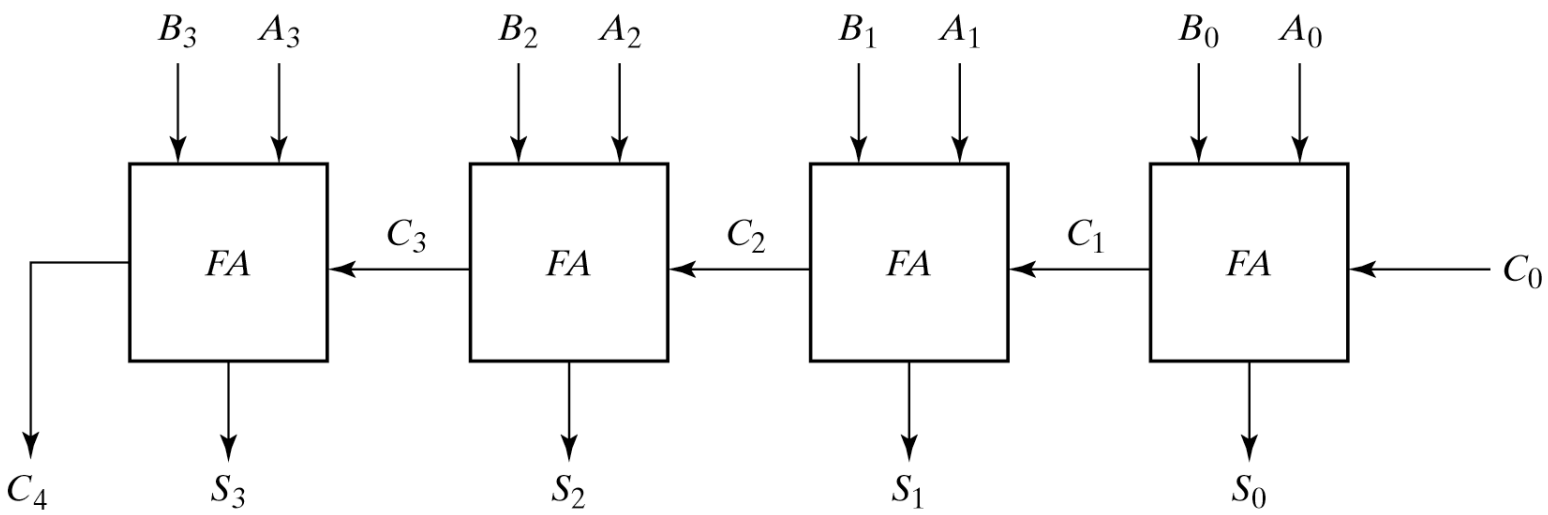


Fig. 4-9 4-Bit Adder

Carry Look-ahead Adder

- There are several techniques for reducing the carry propagation time in a parallel adder. Most widely used technique: Look-ahead carry

- The principle of carry look-ahead solves this problem by calculating the carry in advance, based on the input.

Carry Look-ahead Adder

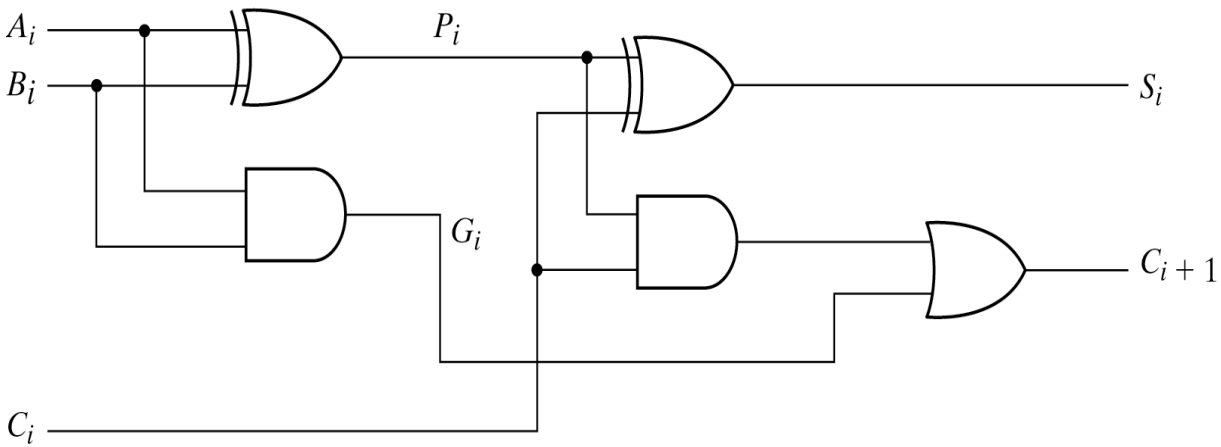


Fig. 4-10 Full Adder with P and G Shown

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

- Here in C_{i+1}, first term G_i is called a carry generate function, because it produces an output carry when $A_i = B_i = 1$.
- The second term $P_i C_i$ corresponds to a previously generated Carry C_i, that must propagate past the i-th stage to the next stage.

Carry Look-ahead Adder

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

- Since for each output carry is expressed in SOP, each function can be implemented with one level AND gates followed by an OR gate (or by a two-level NAND)
- In the next slide, you will see that, C4 doesn't have to wait for C3 and C2 to propagate. It's propagated at the time as C3 and C2.

Carry Look-ahead Adder

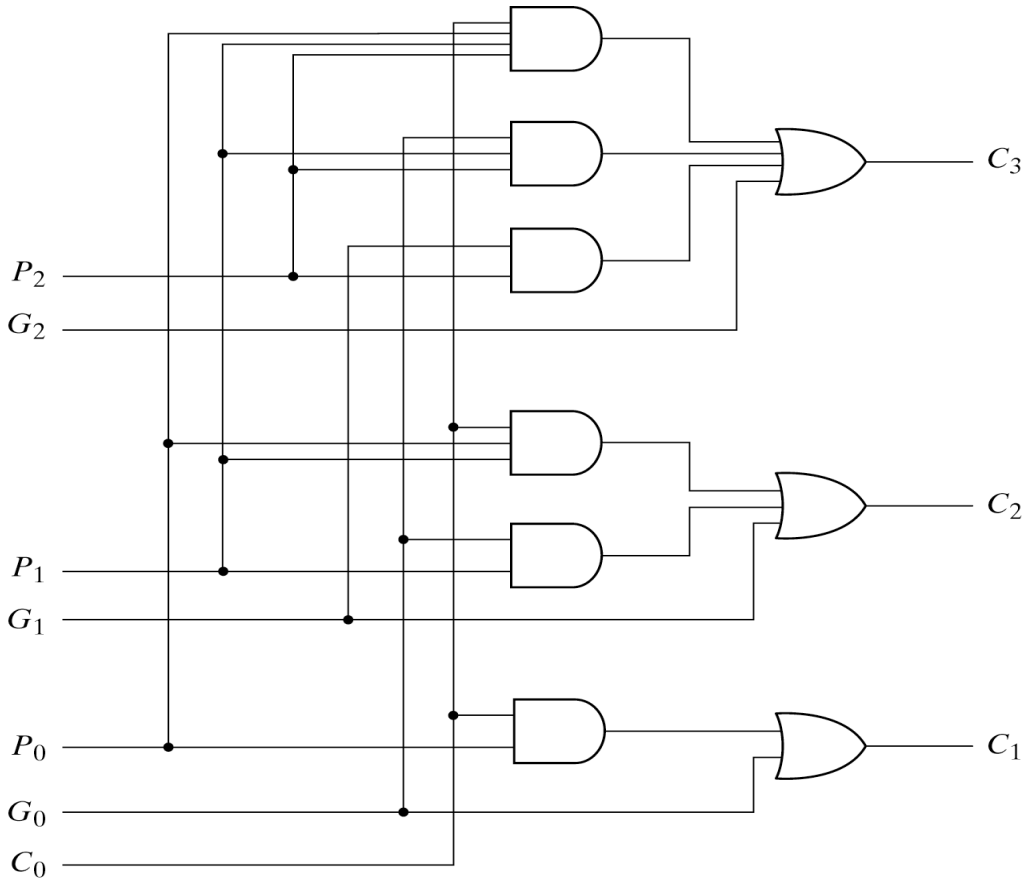
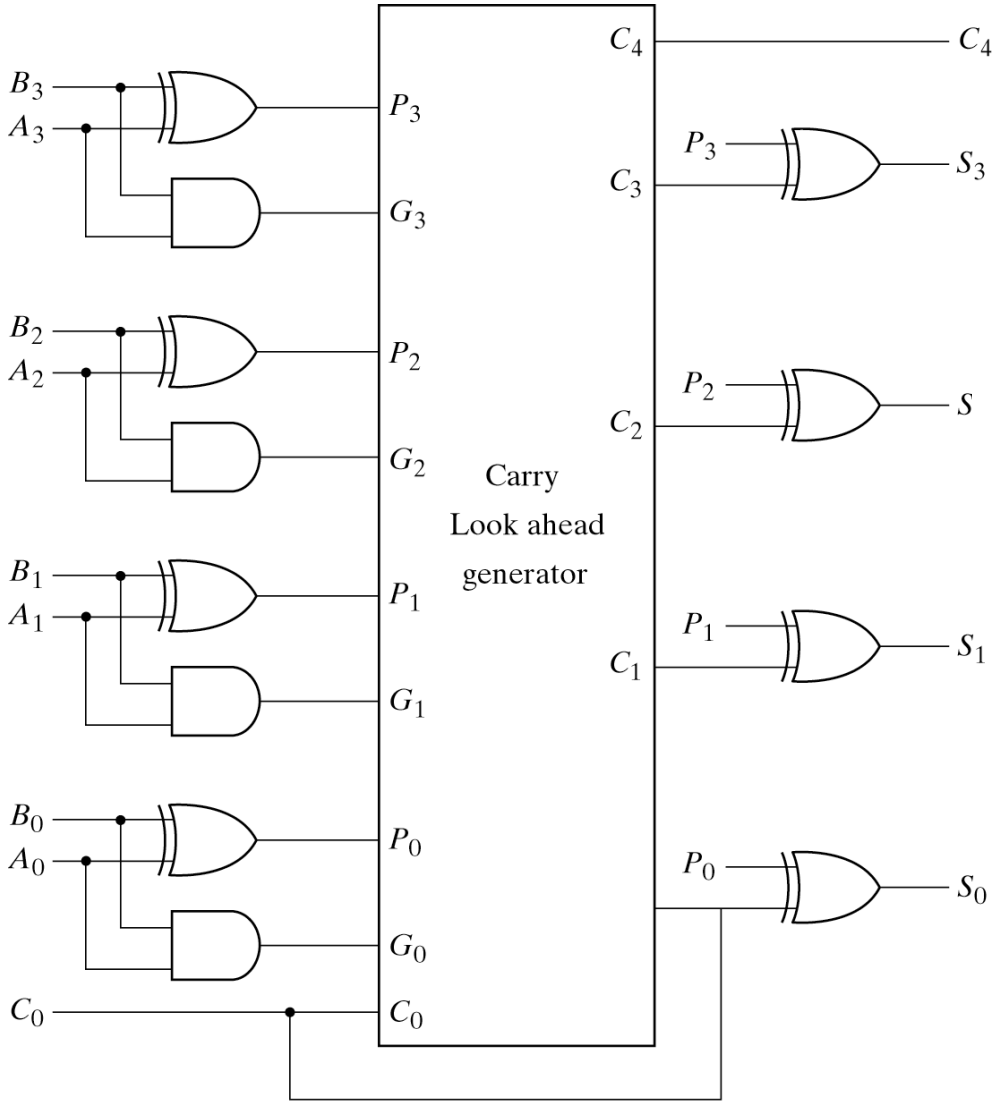


Fig. 4-11 Logic Diagram of Carry Lookahead Generator



$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i$$

Fig. 4-12 4-Bit Adder with Carry Lookahead

BCD Adder

- In a 4 bit parallel adder, we can add two 4 bit binary numbers and the output is in Binary sum.
- But, when we are talking about BCD, the inputs ranges from (0-9), which means (0000 to 1001). 10 to 15 are invalid in BCD.
- If we want to add two numbers in parallel binary adder,
  - 6+7 = 13 = 1101 (Binary Sum)
  - Now, 13 has 2 digits, 1 and 3. BCD of 1 = 0001 and BCD of 3 = 0011
  - So, 6+7 = 13 = 00010011 (BCD Sum) = 10011 (ignoring 0 bits in MSB)
- So, need to modify the Parallel adder to convert the Binary sum to BCD.

BCD Adder

Another important thing to mention:

- In a 4 bit parallel adder, it has 5 outputs= C_{out}, S_3, S_2, S_1, S_0
- When Considering BCD, the input A and B must be range from 0 to 9 as each input digit doesn't exceed 9.
- The output will vary from (0 to 19)
- The output Binary sum can't be greater than:
  - 1001 + 1001 = 10011
  - 9 + 9 + 1= 19. Here, 1 is the input carry Cin
- When the binary sum exceeds 1001, we obtain a non-valid BCD representation. If we add 6 = 0110 with Binary Sum, it becomes the correct equivalent to BCD sum.

BCD Adder

Derivation of a BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

$C=K+Z8.Z4+Z8.Z2$

BCD Adder

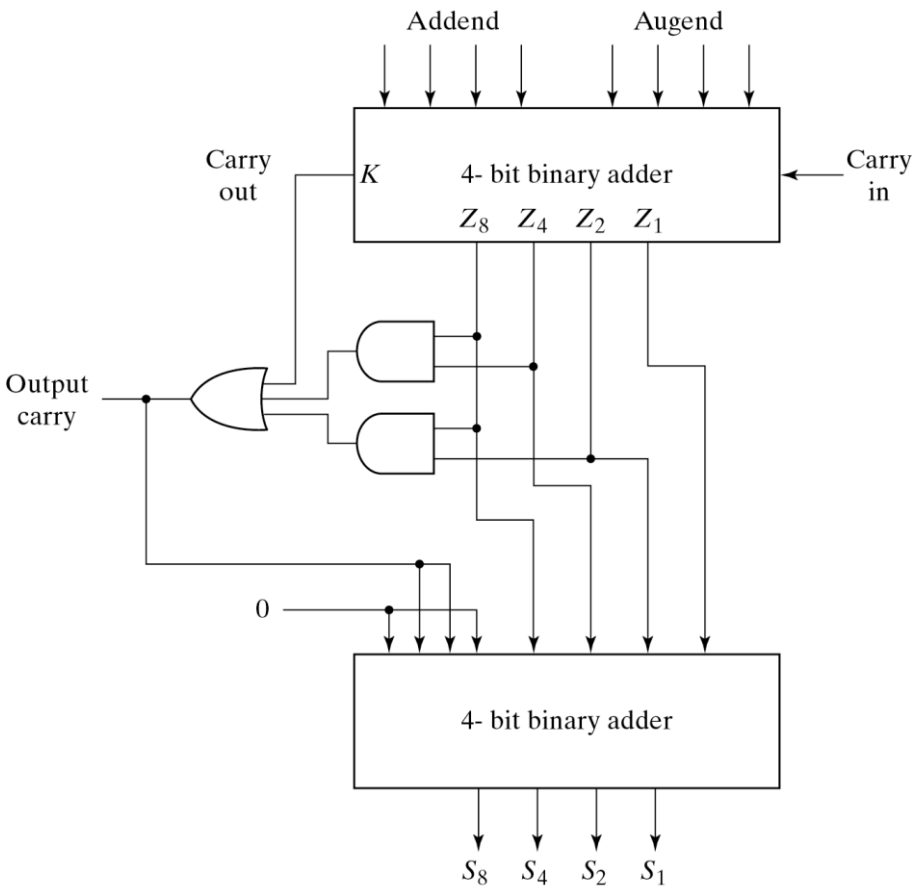


Fig. 4-14 Block Diagram of a BCD Adder


GOOD NEWS!
THE CLASS IS OVER…
THANK YOU!