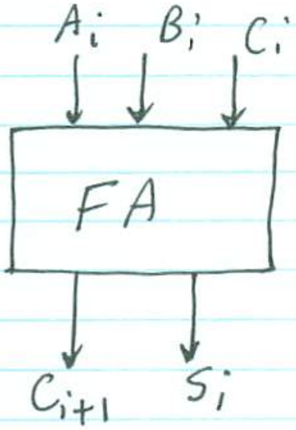
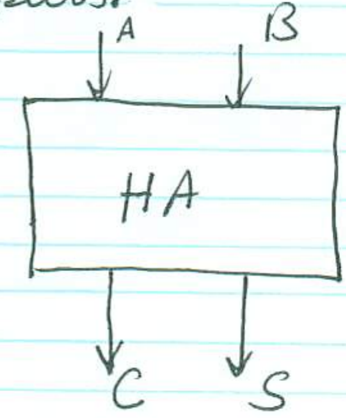
Lecture 9, Feb. 5, 2007

Binary adders

In order two binary strings, we need circuitry to add them bit-by-bit. Adding the least significant bits, we get 1 if one is one and the other one is zero. We get a zero if both are zero or both are one. In the latter case, we get a carry to be added togetter with the next two bits. So, we need a building block that takes in three bits and outputs a sum bit and a carry bit. This is called a full adder.



A full adder can be implemented directly as we Saw in the previous lecture or implemented using two instances of a simpler circuit called a half adder. A half adder has two inputs and two outputs.



Truth table for a half adder is:

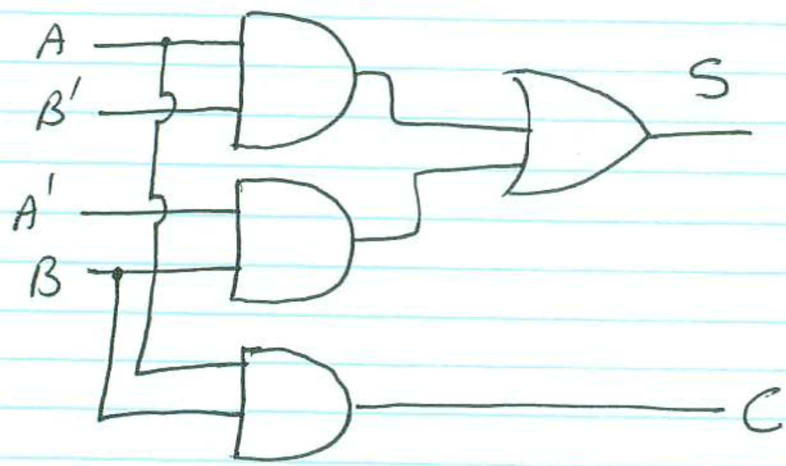
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

So,

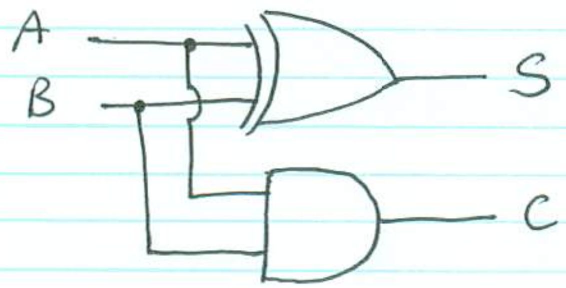
$$S = A'B + AB' = A \oplus B$$

$$C = AB$$

The implementation of a half adder using Sum of products is:



An implementation using XOR is:



Full adder implementation

Let the inputs of the full adder be A_i, B_i and C_i and the outputs be labeled S_i and C_{i+1}. Here C_i is the carry from previous stage and C_{i+1} is the carry to be propagated to the next stage. The truth table is:

A_i	B_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

The K-map for S_i is:

A_i \ B_iC_i	00	01	11	10
0		1		1
1	1		1	

So,

$$S_i = A_i' B_i' C_i + A_i' B_i C_i' + A_i B_i' C_i' + A_i B_i C_i$$

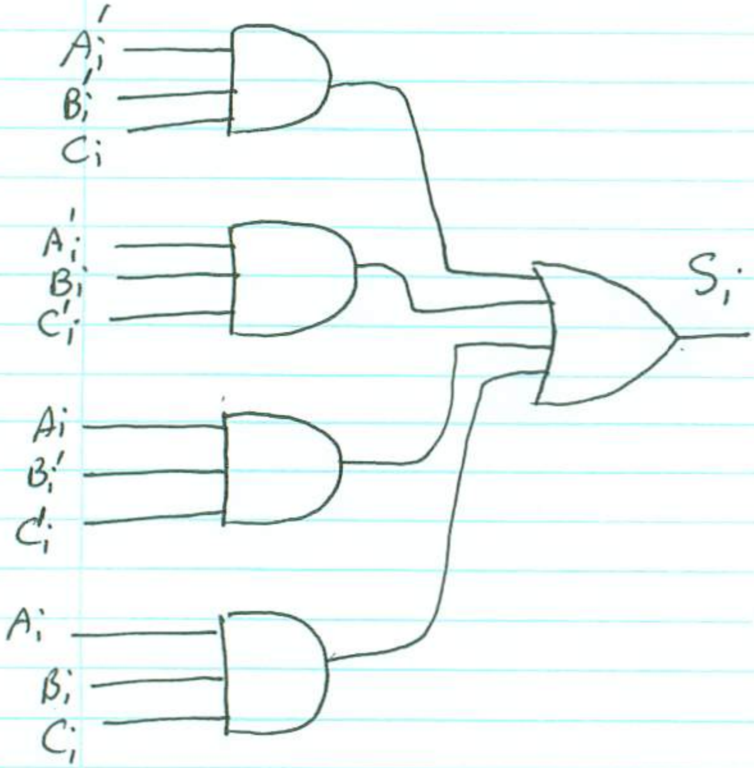
The truth table for the output carry, C_{i+1}, is

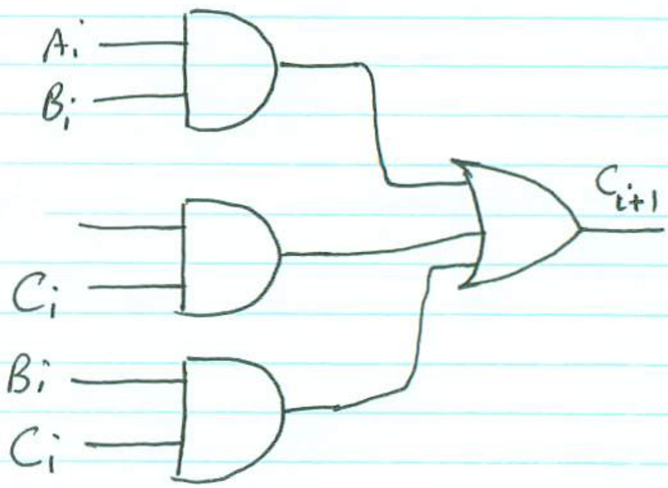
A_i \ B_iC_i	00	01	11	10
0			1	
1		1	1	1

So,

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

The direct implementation will be:





Implementation of full adder using two half adders:

Note that S_i is true when an odd number of its inputs are 1. So:

$$S_i = A_i \oplus B_i \oplus C_i$$

also, we have:

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

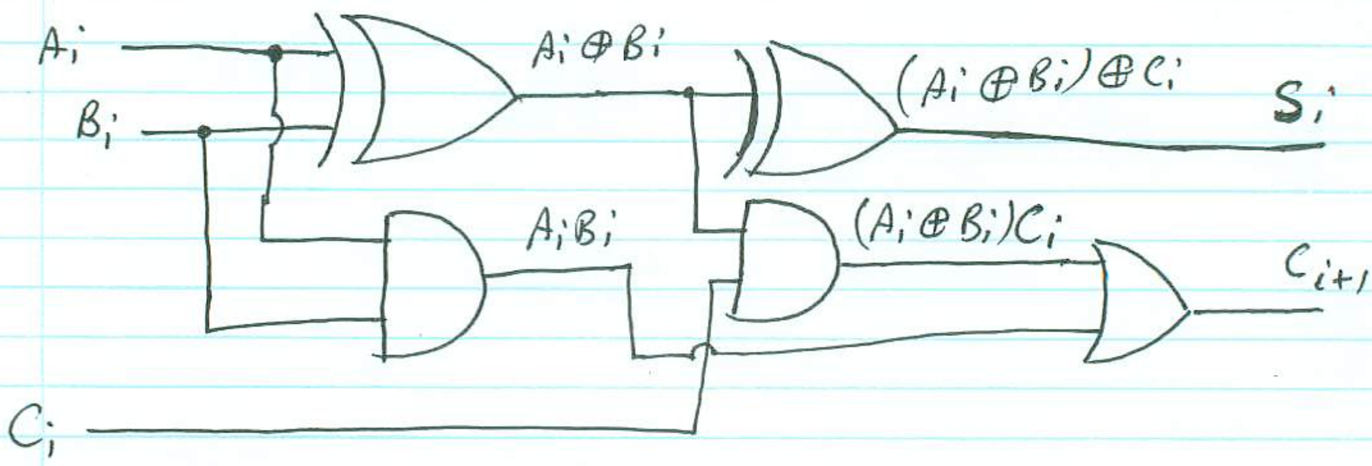
$$= A_i (B_i + C_i) + B_i (A_i + C_i)$$

$$= A_i (B_i + B_i' C_i) + B_i (A_i + A_i' C_i)$$

$$= C_i (A_i B_i' + A_i' B_i) + A_i B_i$$

$$= C_i (A_i \oplus B_i) + A_i B_i$$

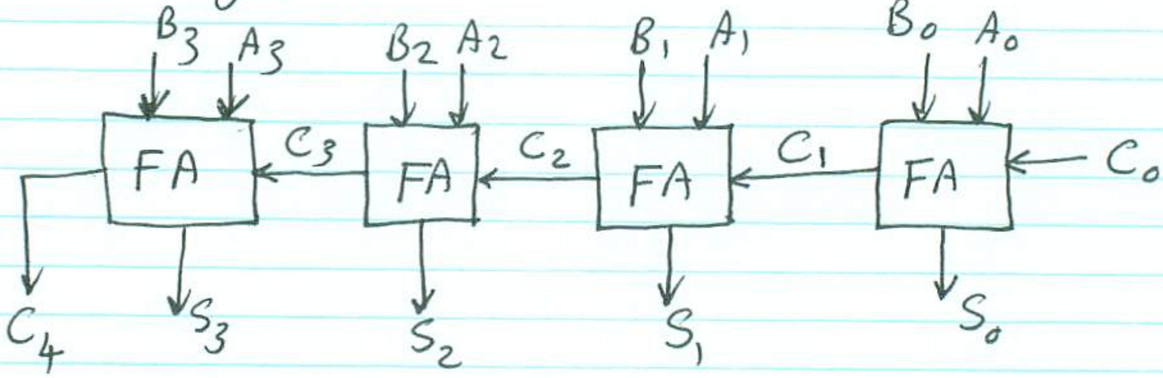
So, we can implement full adder using two



Half adders and one OR gate.

Binary adders / Carry propagation

Assume that we want to implement an adder to add two 4-bit binary numbers. We can do this using 4 full adders (in fact 3 full adders and one half adder if there is no carry in the first stage)

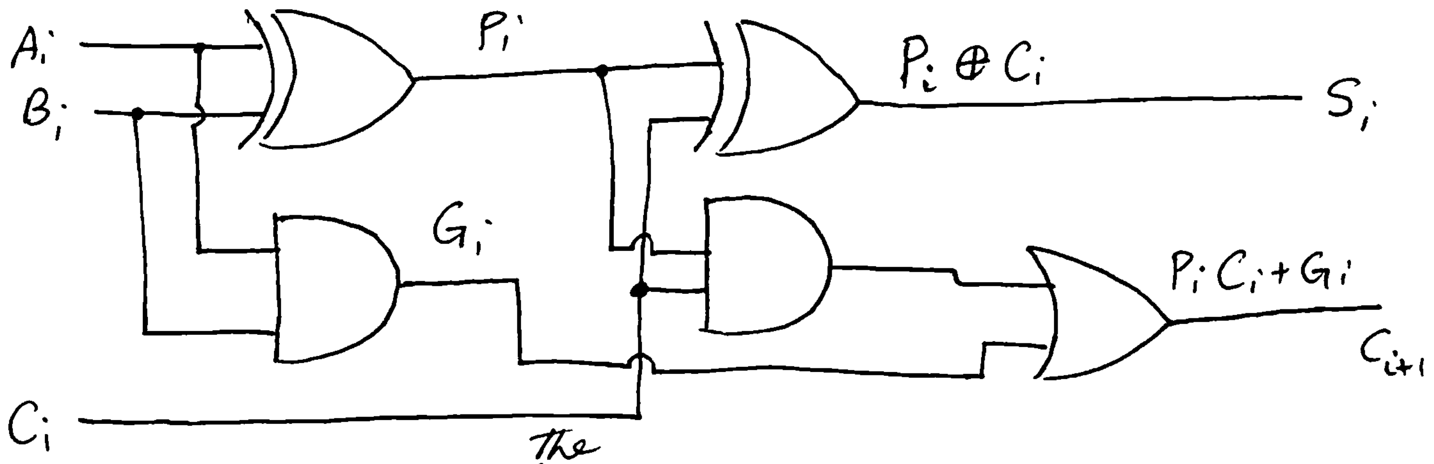


Note that the second Full adder cannot start finding the carry output before having C_1. It takes two gate delays. Similarly to find C_3, the 3rd. full adder needs C_2. In general the delay for a full adder is $2m$ where m is the number of bits to be added. In the above example, the delay is $4 \times 2 = 8$ gate delays.

The above problem can be avoided using Carry lookahead generator.

Carry Lookahead Generator

Let's look at a full adder circuit more carefully:



We have labeled the outputs of the first half adder as P_i and G_i where,

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

Note that,

$$S_i = P_i \oplus C_i$$

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

Note that G_i generates a carry when both the inputs A_i and B_i are 1 regardless of the value of C_i. So, it is called a carry generator. P_i is called a carry propagate since it decides whether a carry will propagate from stage i to stage $i+1$.

Note that P_i and G_i are generated from A_i and B_i in one gate delay.

Now, let's consider the example of 4-bit adder.

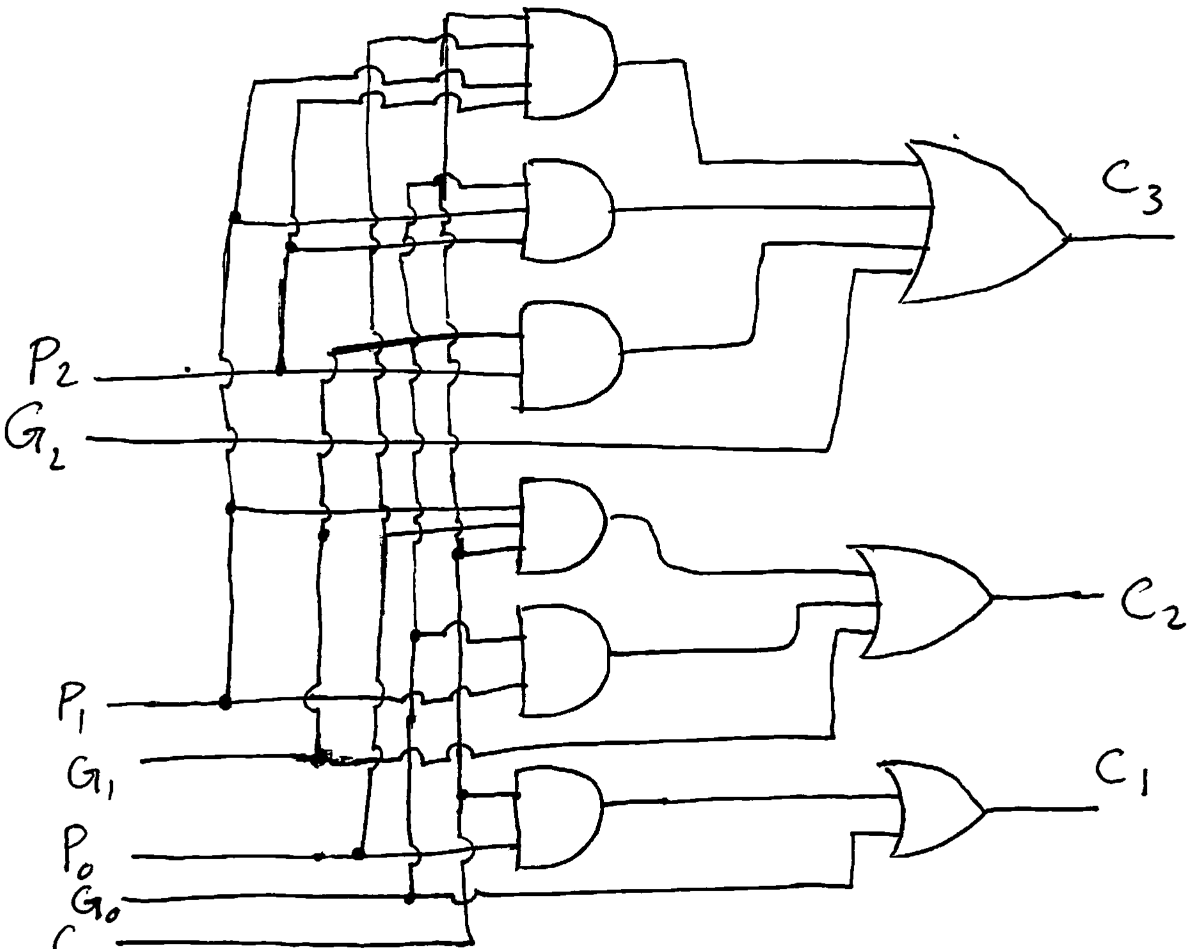
C_0 = the input carry

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

Following is the circuit for the carry lookahead generator.



Note that all carry bits are generated with only two gate delays. We have not shown logic for C_4 to save space.

A four-bit adder with carry lookahead will be: