THE UNIVERSITY OF TEXAS AT DALLAS

CS4341 Digital Logic & Computer Design

Lecture Notes 22

Omar Hamdy
Assistant Professor
Department of Computer Science

Digital Building Blocks

- Now it is time to go one-level up in our abstraction hierarchy to introduce more complex digital components, how they can be used and interconnected, rather than the internal build of each.
- These building blocks are the basis for the microprocessor design
- The scope of our study includes:
  - Arithmetic circuits: addition, subtraction, comparators, ALUs, shifters/rotators, multiplication and division
  - Sequential building blocks: counters and shift registers
  - Memory arrays: memory cells, DRAM. SRAM and ROM
  - Logic arrays: programmable logic arrays (PLAs) and Field Programmable Gate Array (FPGA)

Binary Addition

Start with the least significant bit (rightmost bit):

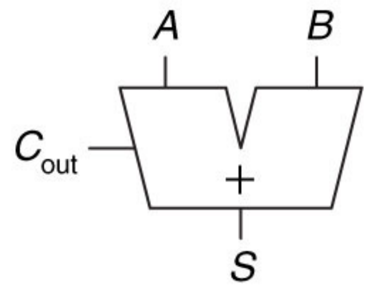
- Add each pair of bits
- Include the carry in the addition, if present
- Discard the carry if it exceeds the available maximum number size (overflow)

carry		1	1	1	1				
	0	0	1	1	0	1	1	0	(54)
+	0	0	0	1	1	1	0	1	(29)
	0	1	0	1	0	0	1	1	(83)
bit position:	7	6	5	4	3	2	1	0	

Binary Addition Circuits: Half Adder

- 1-bit half adder is the simplest addition building block. It has two 1-bit inputs A & B and two 1-bit outputs S and C_{out}.
- S is the sum of A & B, and if the addition is > 1, then C_{out} and S together represent the result of the addition (2).
- Careful study of the truth table shows that S is the A XOR B, and C_{out} is AB
- The half adder cannot be used for more than 1-bit numbers addition (why?)

Half Adder



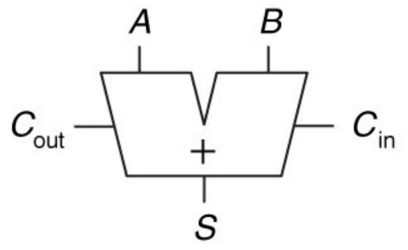
A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$
$$C_{out} = AB$$

Binary Addition Circuits: Full Adder

- A full adder introduces a 3rd input C_{in} to the half adder circuit
- This allows the circuit to accept carry from a previous bit-addition.
- Careful study of the truth table can verify the functional logic of S and C_{out}
- Recall XOR of N inputs produces a 1 if the input has odd number of 1, and 0 otherwise.
- $S = A \oplus B \oplus C$
- $C_{out} = AB + (A + B)\, C_{in}$

Full Adder

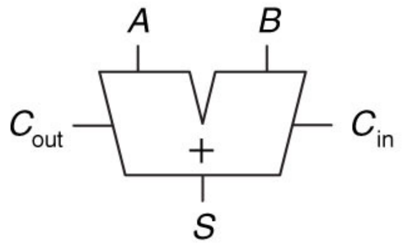


C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

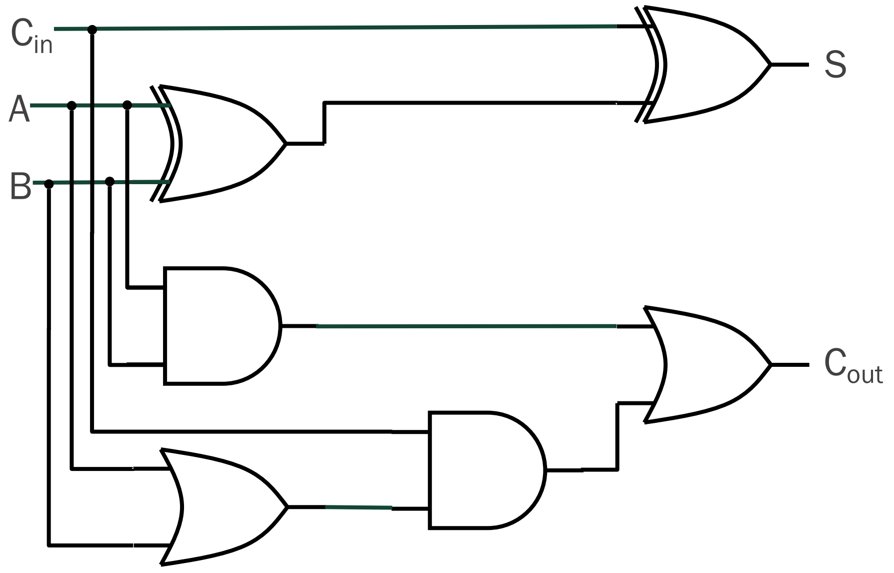
Full Adders Circuit Implementation

Full Adder



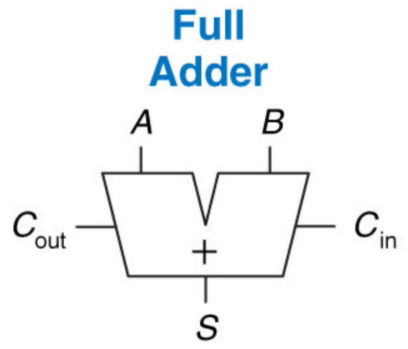
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$



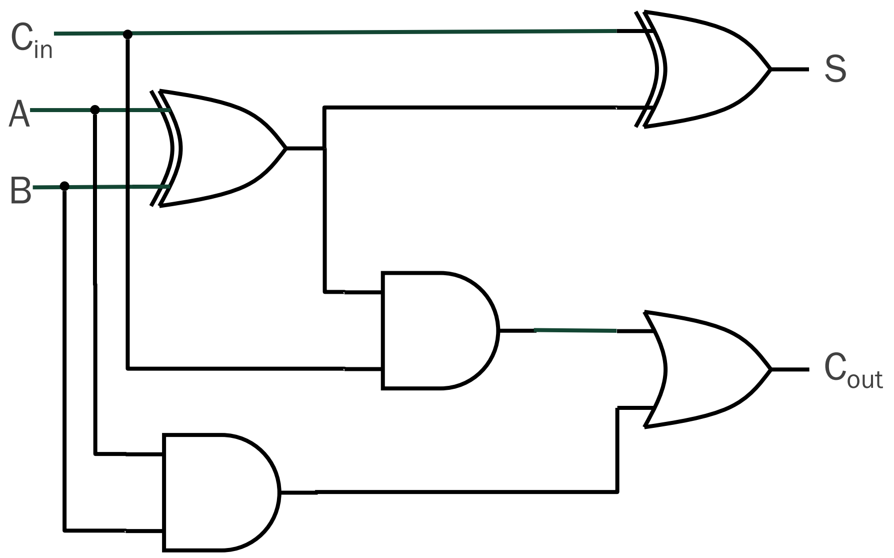
The full adder can be simplified further by noticing that C_{out} can be expressed as $AB + C_{in}(A \oplus B)$ (why?)

Full Adders - Simplified



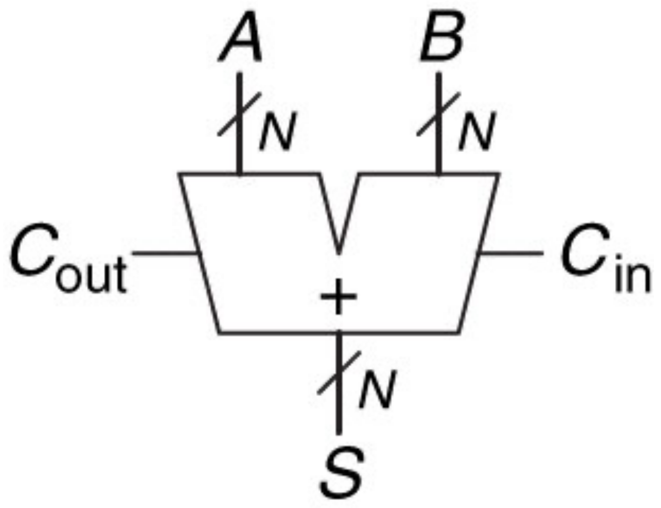
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + C_{in}(A \oplus B)$$



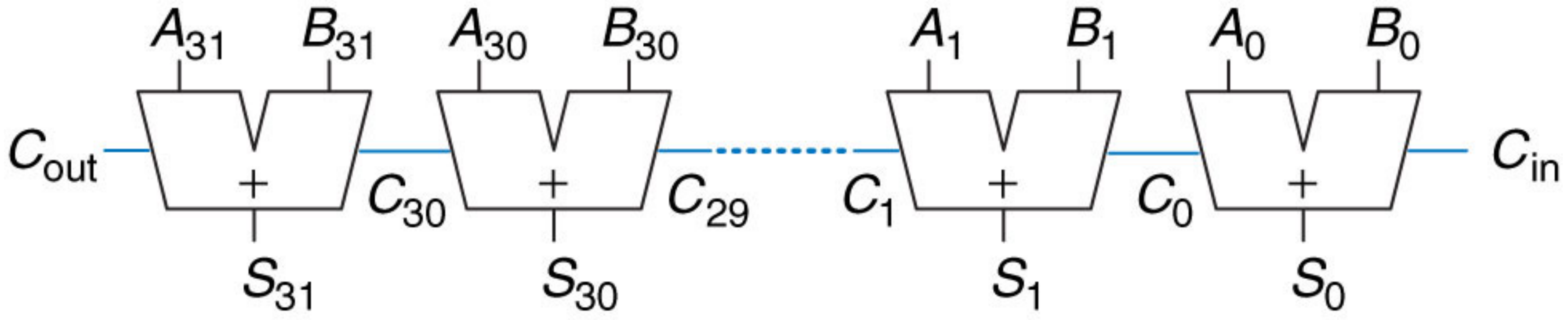
Carry Propagate Adder (CPA)

- Now we introduce an adder capable of adding more than one bit.
- The circuit is similar to the full adder, except that the inputs are two busses instead of single-bit each.
- The shown circuit is an abstraction, which requires implementation strategy
- Three common implementations are ripple-carry adder, carry-lookahead adder and prefix adder.



CPA Implementation: Ripple-Carry

- The ripple carry is the simplest CPA implementation.
- To add two N-bit numbers, N full-address are chained together: the C_{out} of one adder becomes the C_{in} of the next adder.
- That model is slow O(N), and for large N, CPA can be very slow.
- $t_{ripple} = Nt_{FA}$



CPA Implementation: Carry-Lookahead

- The ripple carry delay is caused by the fact that each full adders needs to wait for the carry from the previous full adders to be generated.
- To overcome this bad delay, we need to find a way to produce (compute) all the carries simultaneously and not in sequence.
- This will require added logic, and hence there needs to be a balance between the delay and the added circuit complexity.
- The basic idea is to design each adder circuit in a way that it only considers the inputs and ignores any interim outputs (intermediate carries).
- That means only consider A&B inputs and the initial C_{in} to the circuit.

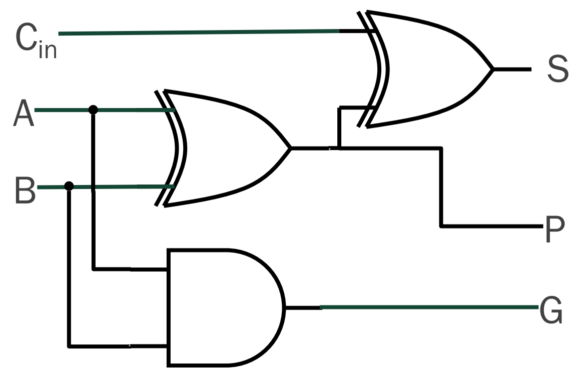
Carry-Lookahead Model

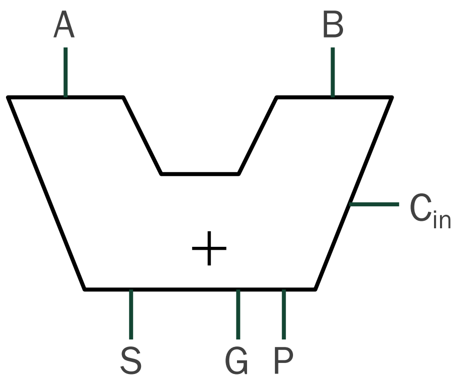
- Carry-lookahead uses a mathematical model to produce C_{in} simultaneously
- The model defines two terms Generate (G) and Propagate (P) as follows:
  - Generate is when C_{in} is 0, and the adder is generating a C_{out} of 1
    - From the truth table, G = AB
  - Propagate is when C_{in} is 1, and the adder is propagating it to C_{out}
    - From the truth table, P = A + B, and following same circuit simplification, we can say $P = A \oplus B$
- Therefore, $C_{out} = AB + (A \oplus B)C_{in} = G + PC_{in}$

C_{in}	A	B	C_{out}	S
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

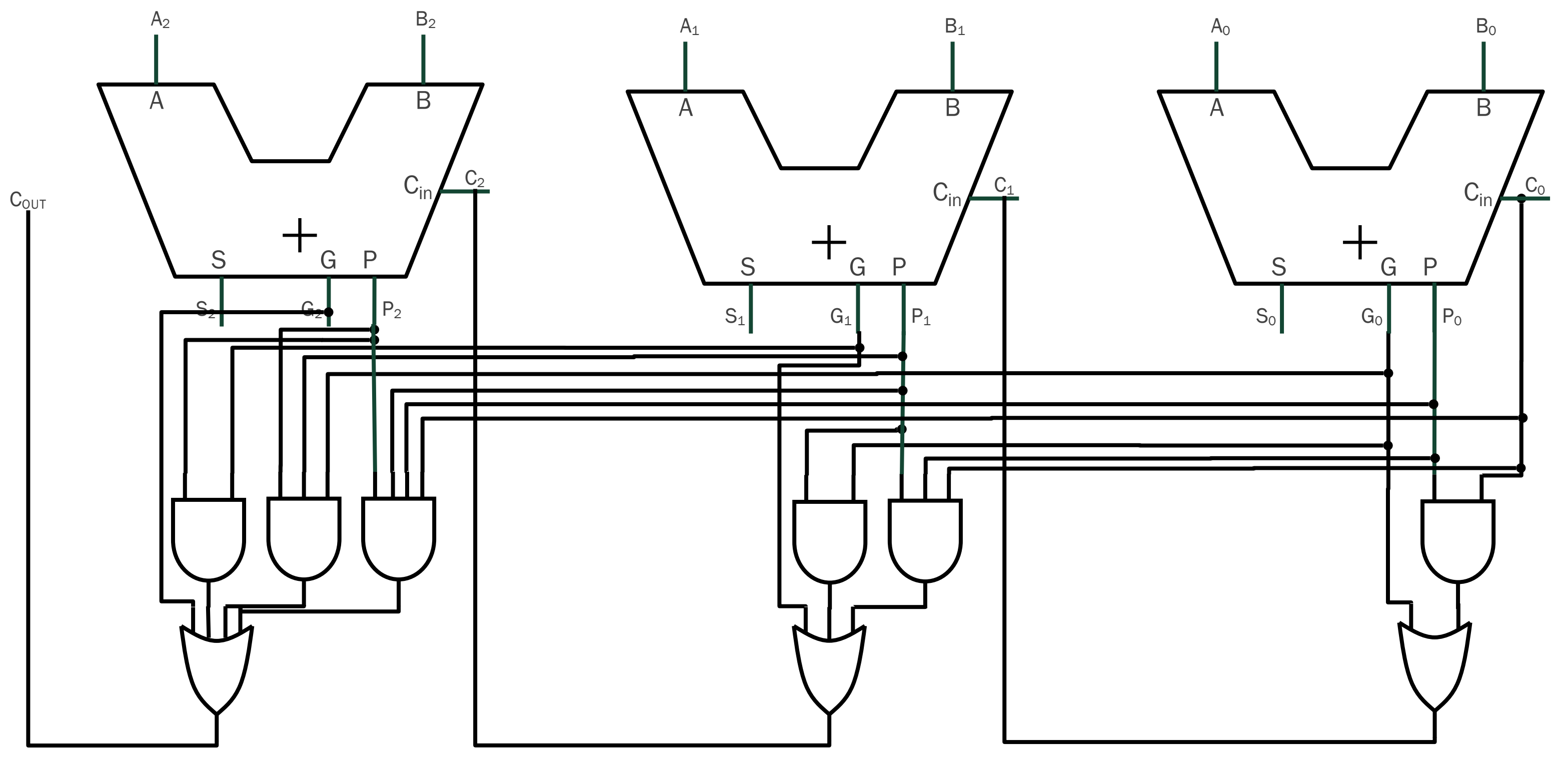
Calculating Carry-Lookahead

- First, we modify the full-adder circuit to produce G and P instead of C_{out}.
- For any full adder i: $G_i = A_iB_i$ and $P_i = A_i \oplus B_i$
- Therefore, $C_{out(i)} = G_i + P_iC_{in(i)}$
- It then follows that $C_{in(i+1)} = C_{out(i)}$
- For convenience, we will refer to $C_{in(i+1)}$ as C_{i+1}
- Therefore, $C_{i+1} = g_i + p_iC_i$
  - $C_1 = g_0 + p_0C_0$, and $C_2 = g_1 + p_1C_1$
  - Then, $C_2 = g_1 + p_1(g_0 + p_0C_0) = g_1 + p_1g_0 + p_1p_0C_0$
  - Then, $C_3 = g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0C_0$
  - Also, $C_4 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0C_0$





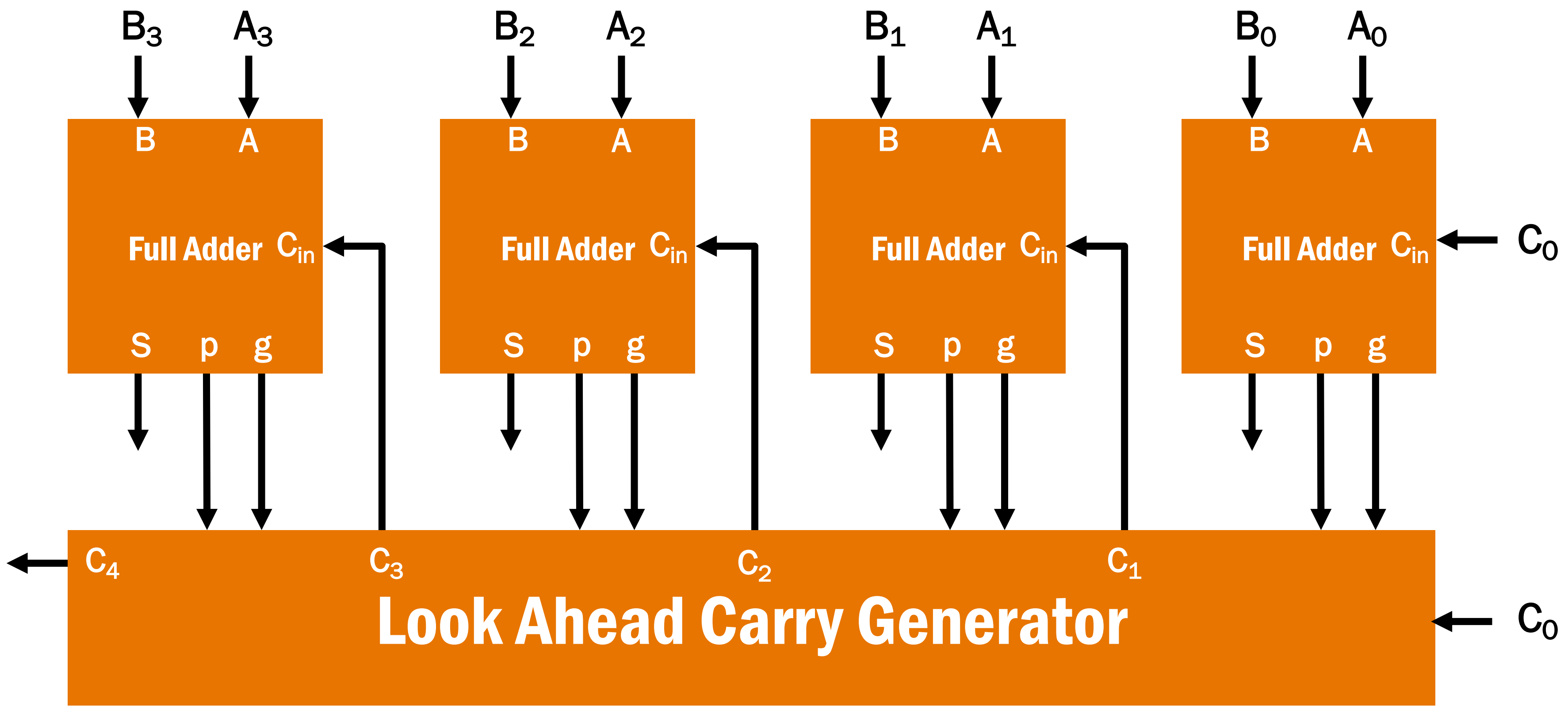
3-Bit Carry Lookahead Circuit



$C_3 = g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0C_0$ $C_2 = g_1 + p_1g_0 + p_1p_0C_0$ $C_1 = g_0 + p_0C_0$

CPA Implementation: Carry-Lookahead

- Putting the circuit all together, we get the following abstracted view



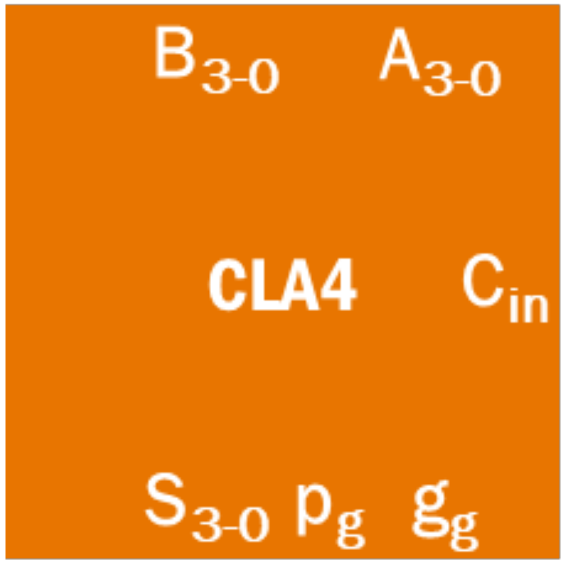
Carry-Lookahead Limitations

- Recall $C_4 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0C_0$
- Therefore, for the case of 4-bit carry lookahead adder, the final carry out C_4 requires a 5-input OR gate, and 5, 4, 3 and 2-input AND gates.
- Therefore a 32-bit CLA will require 33-input gates, which is not practically feasible.
- Alternatively, 4-bit CLA's can be combined in a hierarchal manner.
- For example a 16-bit CLA can be implemented using four 4-bit CLAs and an additional level of carry lookahead logic.
- This will add delay to the overall circuit.

Group Carry and Propagate

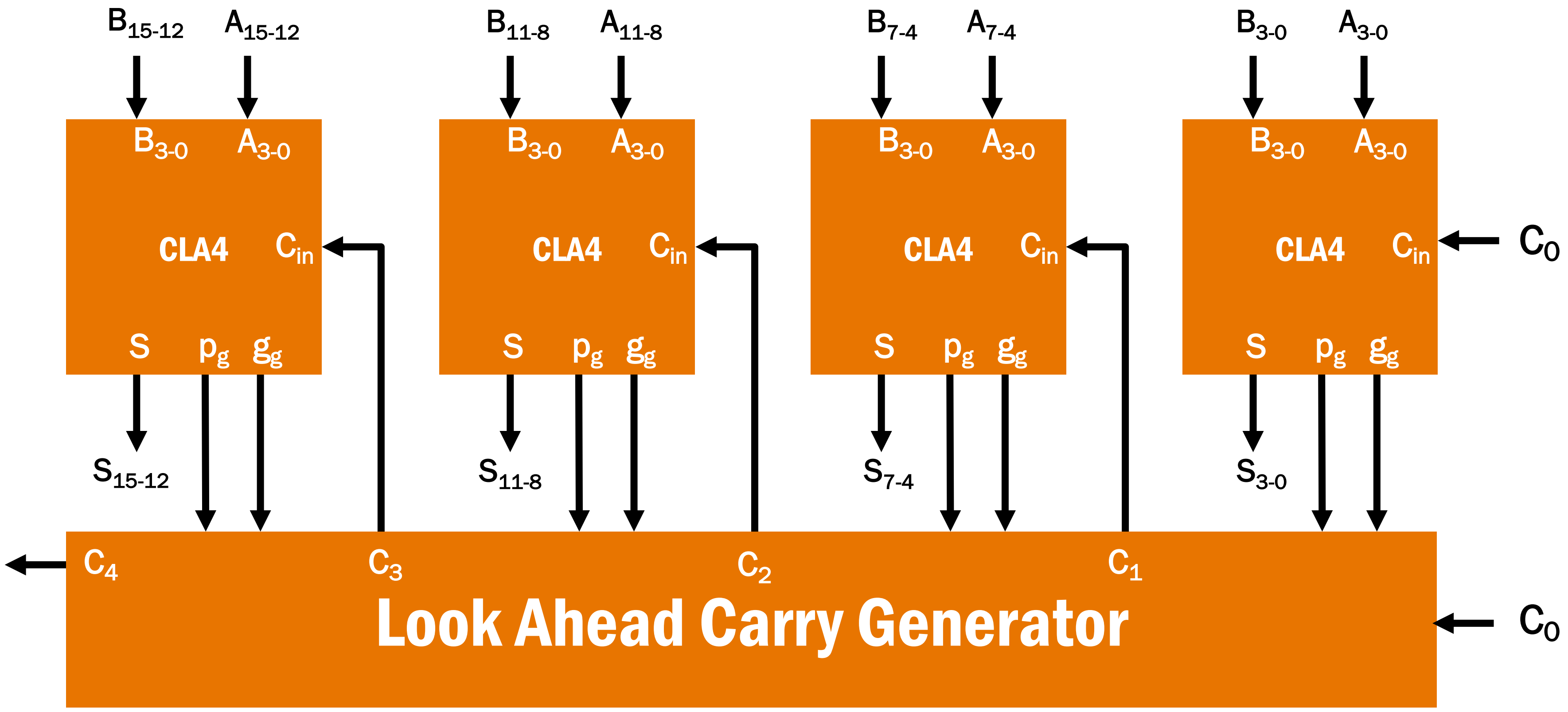
- The hierarchical model will require 4-bit CLA building block as follows

- $p_g = p_3p_2p_1p_0$ follows
- $g_g = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1p_0g_0$



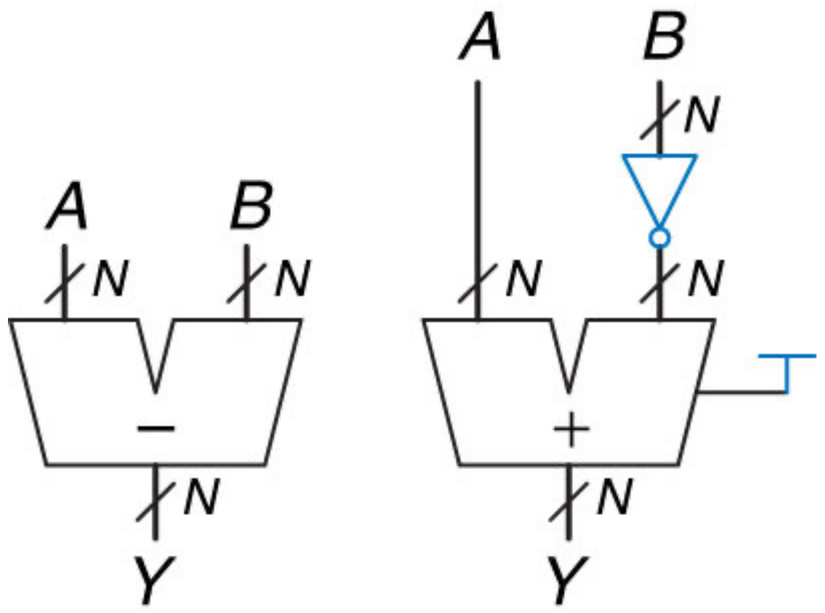
CPA Implementation: Carry-Lookahead

- Putting the circuit all together, we get the following abstracted view



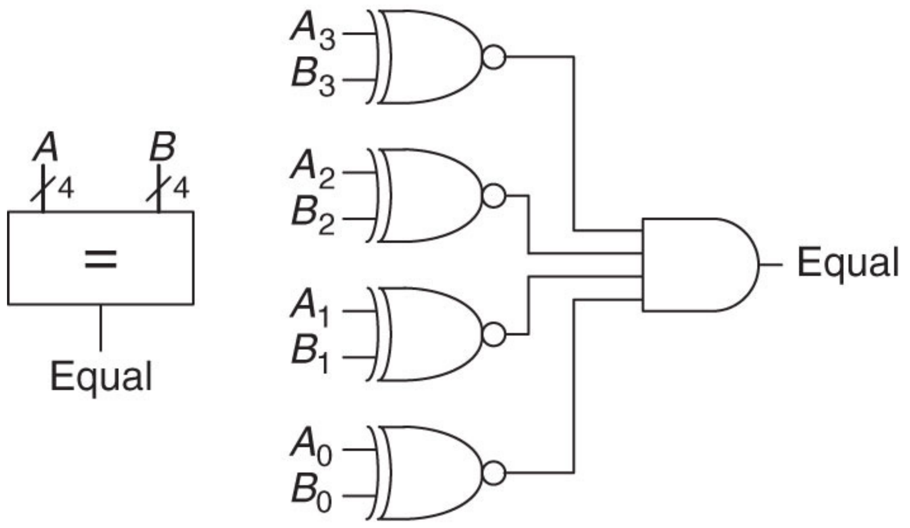
Subtraction

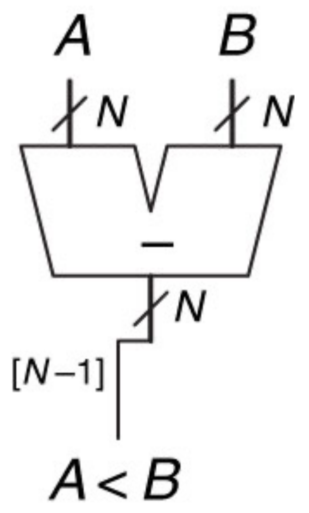
- Subtraction can be done by adding A to the 2's complement of B.



Comparators

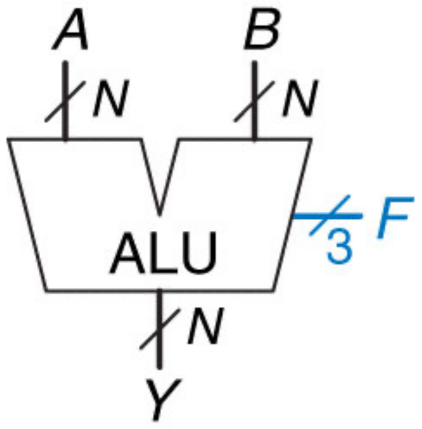
- Comparator determines if two binary numbers are equal, or one is greater (smaller) than the other.
- Two common types:
  - Equality comparator: produces a 1 if A is equal to B (How?)
  - Magnitude comparator: produces a 1 if A < B, and 0 if A ≥ B (How?)





ALU

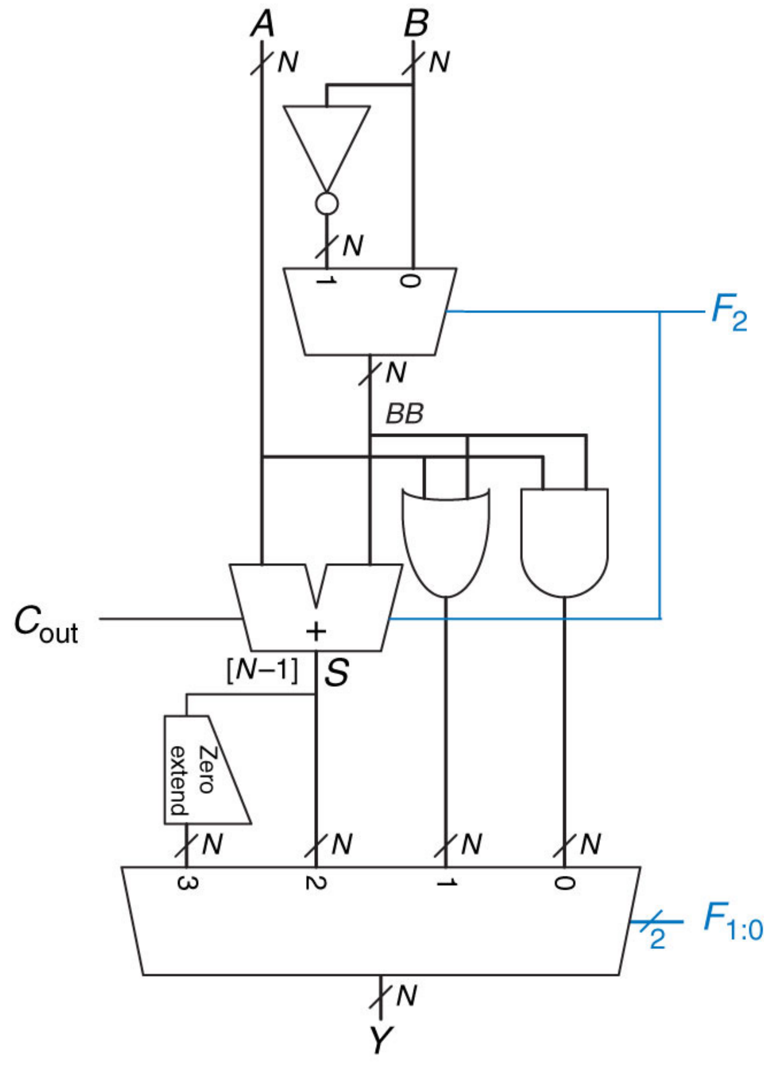
- Arithmetic/Logical Unit (ALU) combines a variety of mathematical and logical operations into a single unit.
- Therefore, the ALU needs an additional input F which tells the ALU which function to perform.
- A good ALU design efficiently uses the digital building blocks with minimum redundancy.
- Can you think of a design?

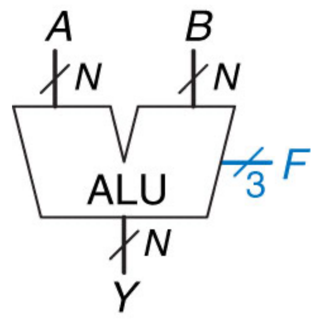


$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A − B
111	SLT

ALU Internal Design

$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A − B
111	SLT





To Do List

- Review lecture notes
- Study Chapter 5, sections 1 through 3