



University of
Massachusetts
Amherst

Engin112 – Lectures 18-19

Arithmetic Circuits

Maciej Ciesielski
Department of Electrical and Computer Engineering
10/19-21/2011

Recap from Last Lectures

- Binary adders – single bit
 - Half adders
 - Full adders
- Binary adders – n bits
 - Ripple carry adder
 - Overflows during arithmetic operations

- This lecture
 - Carry lookahead adder
 - Subtractor
 - » built from adder
 - Multipliers

Carry-Lookahead Adder

- Full adder: $S_i = A_i \oplus B_i \oplus C_i$, $C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i$
- Create new signals:
 - $G_i = A_i B_i$ “carry generate” for stage i
 - $P_i = A_i \oplus B_i$ “carry propagate” for stage i
- Full adder equations expressed in terms of G_i and P_i
 - $S_i = P_i \oplus C_i$
 - $C_{i+1} = G_i + P_i C_i$

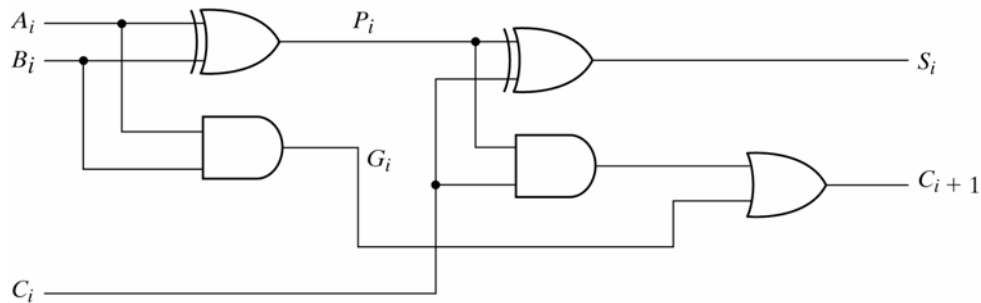


Fig. 4-10 Full Adder with P and G Shown

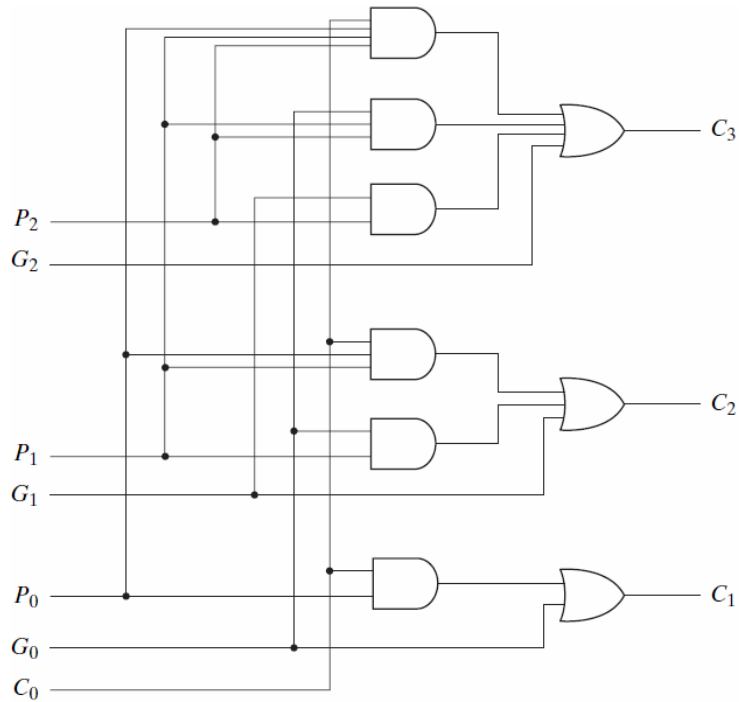
Carry Lookahead - Equations

- Full adder functionality can be expressed recursively
 - $S_i = P_i \oplus C_i$
 - $C_{i+1} = G_i + P_i C_i$
- Carry of each stage
 - $C_0 = \text{input carry}$
 - $C_1 = G_0 + P_0 C_0$
 - $C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$
 - $C_3 = G_2 + P_2 C_2 = \dots = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
 - $C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

Carry Lookahead - Circuit

▪ **Circuit:**

- Sum of products
- Only two stages



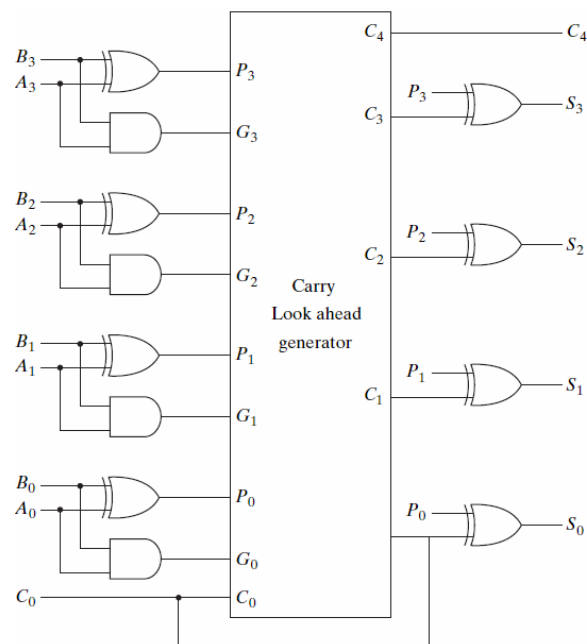
4-bit Adder with Carry Lookahead

▪ **Complete adder:**

- Same number of stages for each bit

▪ **Drawback?**

- Increasing complexity of lookahead logic for more bits



Subtraction

- How can we implement subtraction?

- Subtraction is addition of complement
 - $N - M = N + (\text{two's complement of } M) = N + (2^n - M)$
- How do we determine 2's complement?
 - 1's complement (flip bits) and add 1

- How can we flip bits?

- NOT gate (subtraction only)
- XOR gate (to provide control: Add/Sub):
 - $x \oplus 0 = x$ (use for Add)
 - $x \oplus 1 = x'$ (use for Sub)

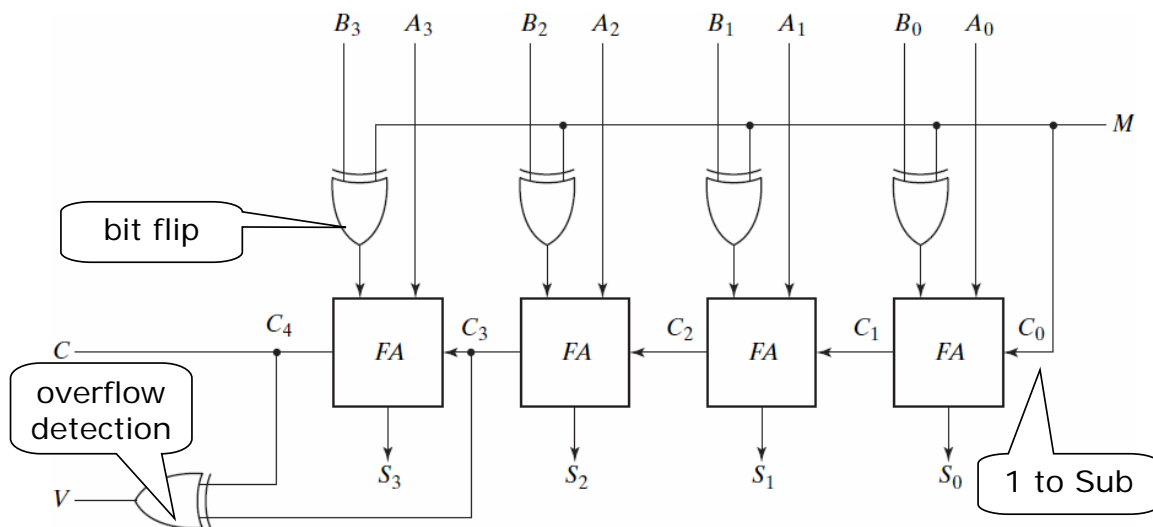
x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

- How can we add 1?

- Input carry

Binary Subtractor

- Adder/Subtractor circuit:



- M sets mode: $M=0$ addition and $M=1$ subtraction

- M is a "control signal" (not "data") switching between Add and Sub

Overflow

- n -bit addition can generate $(n+1)$ -bit number
 - Called “overflow”
 - Needs to be detected by computer system
- How can we detect overflow in addition?
 - End carry
- Also necessary for signed numbers or subtraction
 - Most significant bit indicates sign
- If carry into sign position and out of sign position differ, then overflow
 - Result would be correct with extra position
 - Detected by XOR gate
 - Can be used as input carry for next adder circuit

10/19-21/2011

Engin 112 - Intro to ECE

9

Overflow Conditions

- Overflow conditions
 - There is no overflow if signs are different ($pos + neg$, or $neg + pos$)
 - Overflow can happen only when both numbers have same sign, and
 - If carry *into* sign position and *out* of sign position differ
 - Example: 2's complement signed numbers with $n = 4$ bits

+6	0 110	-6	1 010
+7	0 111	-7	1 001

+13	0 1 101	-13	1 0 011
	↖ 1		↖ 0
	↖ 0		↖ 1

- Result would be correct with extra position
- Detected by XOR gate (output =1 when inputs differ)
- Can be used as input carry for next adder circuit

10/19-21/2011

Engin 112 - Intro to ECE

10

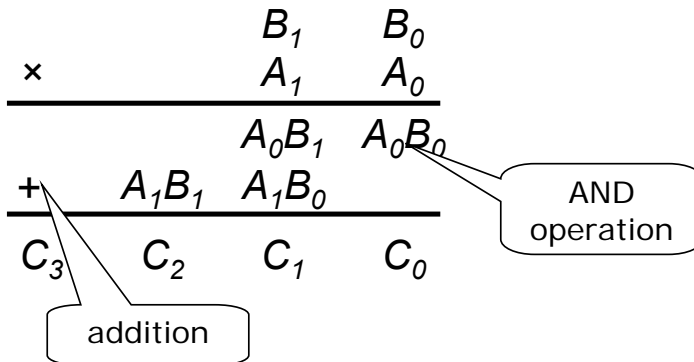
Multiplier

- Multiplication in binary

- Product is 1 if both inputs are 1
- Can be implemented with AND

$$\begin{array}{r}
 1101 \\
 \times 1010 \\
 \hline
 0000 \\
 1101 \\
 0000 \\
 1101 \\
 \hline
 10000010
 \end{array}$$

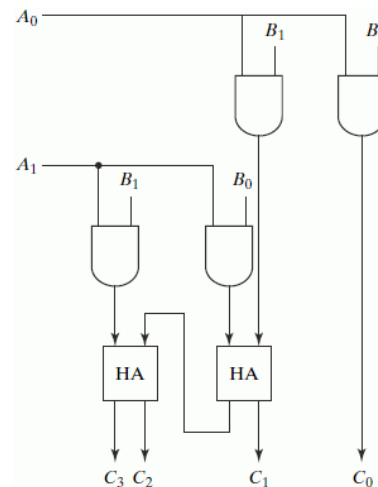
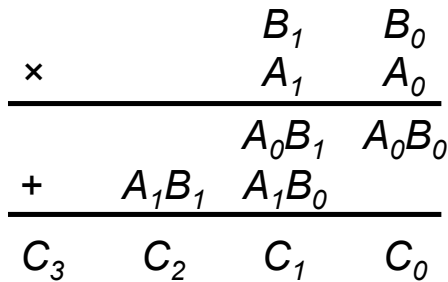
- 2-bit multiplier:



Multiplier

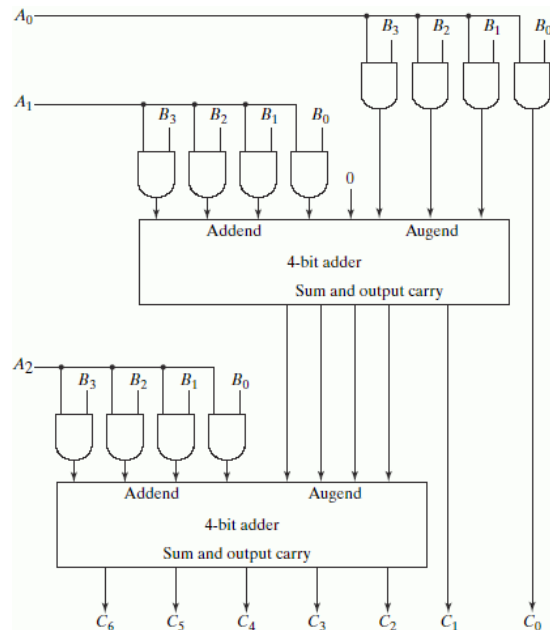
- Circuit diagram:

- Direct translation of computation



Multipliers

- 4-bit by 3-bit multiplier:
- Larger multipliers:
 - J bit multiplier
 - K bit multiplicand
- How many AND gates?
 - $J \times K$ AND gates
- How many adders?
 - $(J-1)$ K -bit adders
- How many output bits?
 - $J + K$ output bits



Magnitude Comparator

- Need to compare two numbers: A and B
 - $A > B ?$, $A = B ?$, $A < B ?$
- How many truth table entries for n -bit numbers?
 - 2^{2n} entries
 - Impractical for design
- How can we determine that two numbers are equal?
 - Equal if every digit is equal
 - $A_3A_2A_1A_0 = B_3B_2B_1B_0$ iff $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 = B_0$
- New function: x_i indicates if $A_i = B_i$
 - $x_i = A_iB_i + A_i'B_i'$ (XNOR)
 - Thus, $(A = B) = x_3x_2x_1x_0$
- What about $A < B$ and $A > B$?

Magnitude Comparator

- **Case 1: $A > B$**
 - How can we tell that $A > B$?
 - Look at most significant bit where A and B differ
 - » If $A = 1$ and $B = 0$, then $A > B$
 - » If not, then $A \leq B$
- **Function ($n = 4$) :**
 - If difference in first digit: A_3B_3'
 - If difference in second digit: $x_3A_2B_2'$
 - » Conditional that $A_3 = B_3$ ($x_3 = 1$ if $A_3 = B_3$)
 - Similar for all other digits
- **Comparison function $A > B$:**
 - $(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$
- **Case 2: $A < B$**
 - swap A and B for $A < B$

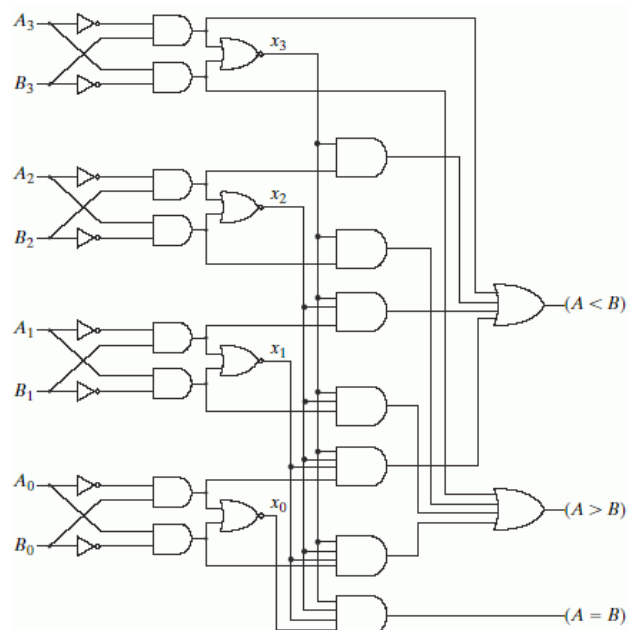
10/19-21/2011

Engin 112 - Intro to ECE

15

Magnitude Comparator Circuit

- **Functions:**
 - $(A = B) = x_3x_2x_1x_0$
 - $(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$
 - $(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$
- **Can be extended to arbitrary number of bits**
 - Size grows with n^2
($n =$ number of bits)



10/19-21/2011

Engin 112 - Intro to ECE

16

Summary

- Standard arithmetic components (combinational)
 - Adders, subtractors
 - Multipliers
 - Comparators

- Next time: Other arithmetic blocks
 - Decoders
 - Encoders
 - Multiplexers (MUX)
 - Demultiplexers