University of Massachusetts Amherst

Engin112 – Lecture 15

Binary Adders

Maciej Ciesielski
Department of Electrical and Computer Engineering
10/09/2012

Recap from Last Lectures

- Design and analysis of Boolean circuits
  - Two-level logic (sum of products, product of sums)
- Truth table for multiple-output functions:
  - One output column per function (output)
  - Some minterms can be shared among several outputs
  - One K-map per function
- This lecture:
  - Binary adders (arithmetic circuit)
  - Overflow condition

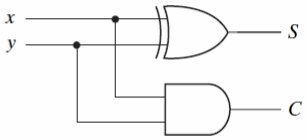
Binary Adders

- Addition is important function in computer system
- What does an adder have to do?
  - Add binary digits
  - Generate carry if necessary
  - Consider carry from previous digit
- Binary adders operate bit-wise
  - A 16-bit adder uses 16 one-bit adders
- Binary adders come in two flavors
  - Half adder : adds two bits and generate result and carry
  - Full adder : also considers carry input
  - Two half adders make one full adder

Binary Half Adder

- Specification:
  - Design a circuit that adds two bits and generates the sum and a carry
- Outputs:
  - Two inputs: *x, y*
  - Two output: *S* (sum), *C* (carry)
- Truth table:

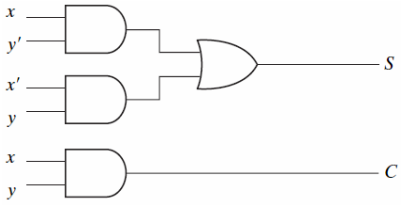
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



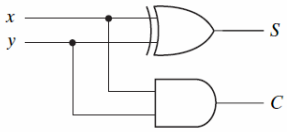
(b) $S = x \oplus y$
$C = xy$

Binary Half Adder - Circuit

- Circuit implementation:



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

- Works for single bit, but not for multi-bit numbers
  - Need to consider input carry from prior stage

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

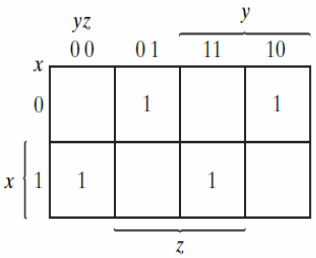
Full Adder

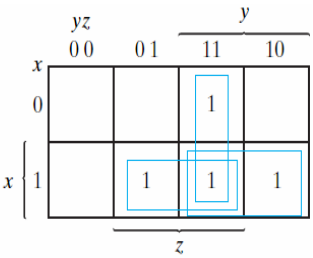
- Specification:
  - A circuit that adds three bits and generates the sum and a carry
- Inputs:
  - Three inputs: *x,y,z*
  - Two outputs: *S, C*
- Truth table:

x	y	z	C	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

Full Adder - Minimization

- Minimization of Boolean functions for *S* and *C*:



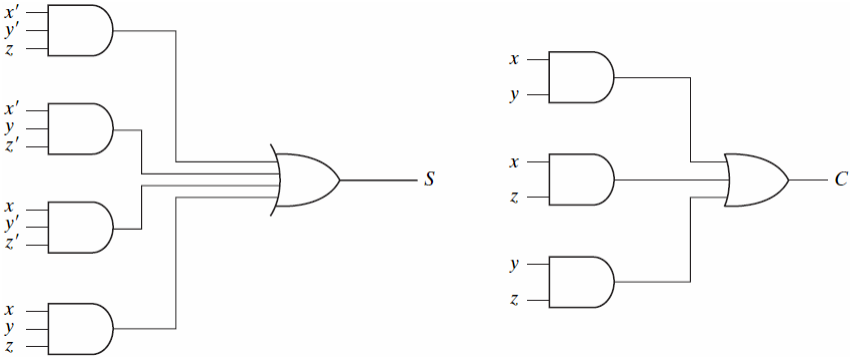


$S = x \oplus y \oplus z$

$C = x\,y + (x \oplus y)\,z$

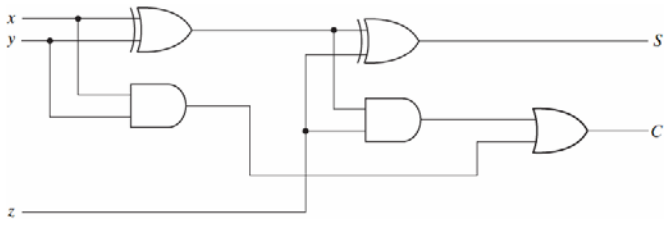
Full Adder - Circuit

- Circuit:



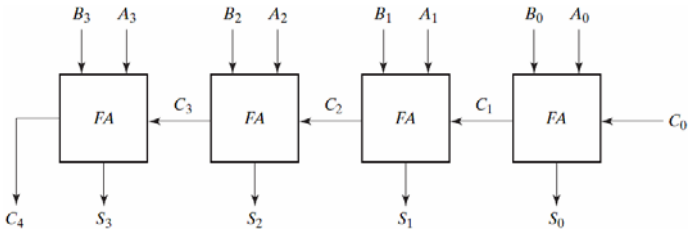
Full Adder from Half Adders

- How can two half adders make a full adder?
- Observations:
  - Three inputs x, y, z can be added in two steps
    - » $x+y+z = (x+y) + z$
  - What about the carry?
    - » Carry can occur when adding $x+y$ and when adding z
- Full adder: $S = x \oplus y \oplus z$, $C = x\,y + (x \oplus y)\,z$



Binary n-bit Adder

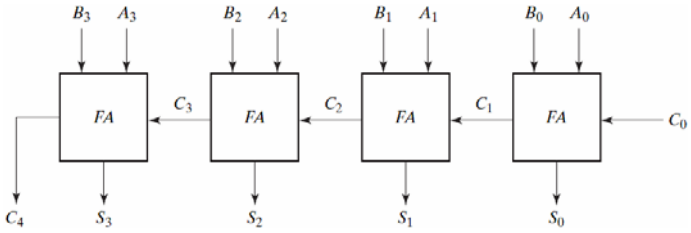
- How can we build an *n*-bit adder from full adders?
  - One adder for each bit (*n* total)
  - Connect carry to next adder's input
  - Output: sequence of sums and a final carry
- 4-bit adder circuit (Ripple Carry Adder):



- Classical example of standard components
  - Would require truth table with 2^9 entries!

Ripple-Carry Adder

- How long does it take to complete an addition?
  - Carry needs to propagate through circuit – a problem



- Speed of addition is critical
  - Fundamental arithmetic operation
- How can we speed up addition?
  - Determine carries ahead of time (carry look-ahead)

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
  - Can be detected by XOR gate
  - Can be used as input carry for next adder circuit

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 wih 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

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
  - $S_i = P_i \oplus C_i$
  - $C_{i+1} = G_i + P_i C_i$

x	y	z	C	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

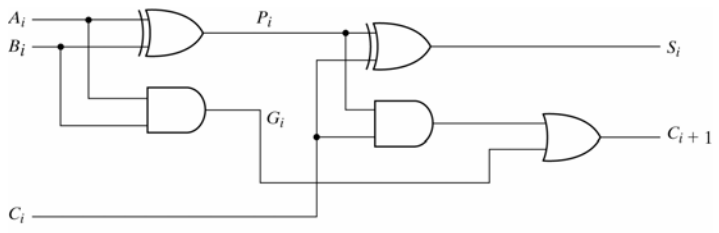


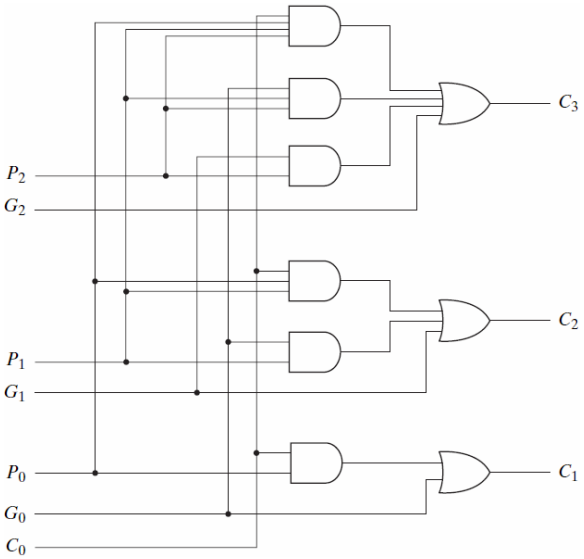
Fig. 4-10 Full Adder with P and G Shown

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 = *input carry*
  - $C_1 = G_0 + P_0 C_0$
  - $C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$
  - $C_3 = G_2 + P_2 C_2 = \ldots = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
  - $C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

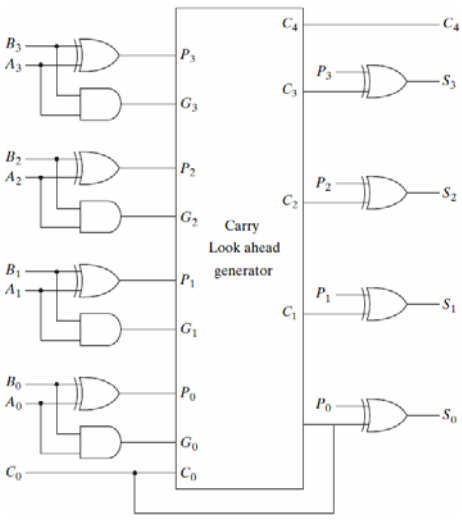
Carry Lookahead - Circuit

- 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



EXAM 1 Coming Up !

- Midterm 1 exam
  - Tuesday, Oct. 16 at 7:00 pm
  - Location: Hasbrouck Lab Addition, room 20.
  - Closed books, closed notes
    - » you will be given Table 2.1 with basic axioms and theorems
- Material:
  - Chapter 1
    - » Basics of Boolean algebra
    - » Number systems (signed, 2's complement)
  - Chapter 2
    - » Boolean theorems, basic proofs
    - » Logic simplification using Boolean theorems
    - » Duality, De Morgan's law
  - Chapter 3 (Sections 3.1 – 3.8)
    - » K-maps, incompletely specified functions
  - Chapter 4 (Sections 4.1 – 4.4)
    - » Analysis and design of logic circuits