

Lecture 6

Arithmetic

2's complement

- Recall how to represent negative numbers in binary? (2E6)
- 2's complement makes addition and subtraction seamless for hardware
- To convert to 2's complement (quick way), invert all bits and add 1 to LSB
- Addition
 - add normally, discard any carry out from sign bit position
- Subtraction
 - Convert subtrahend to 2's complement and then add, again discard any carry out from sign bit position

Half Adder & Full Adder

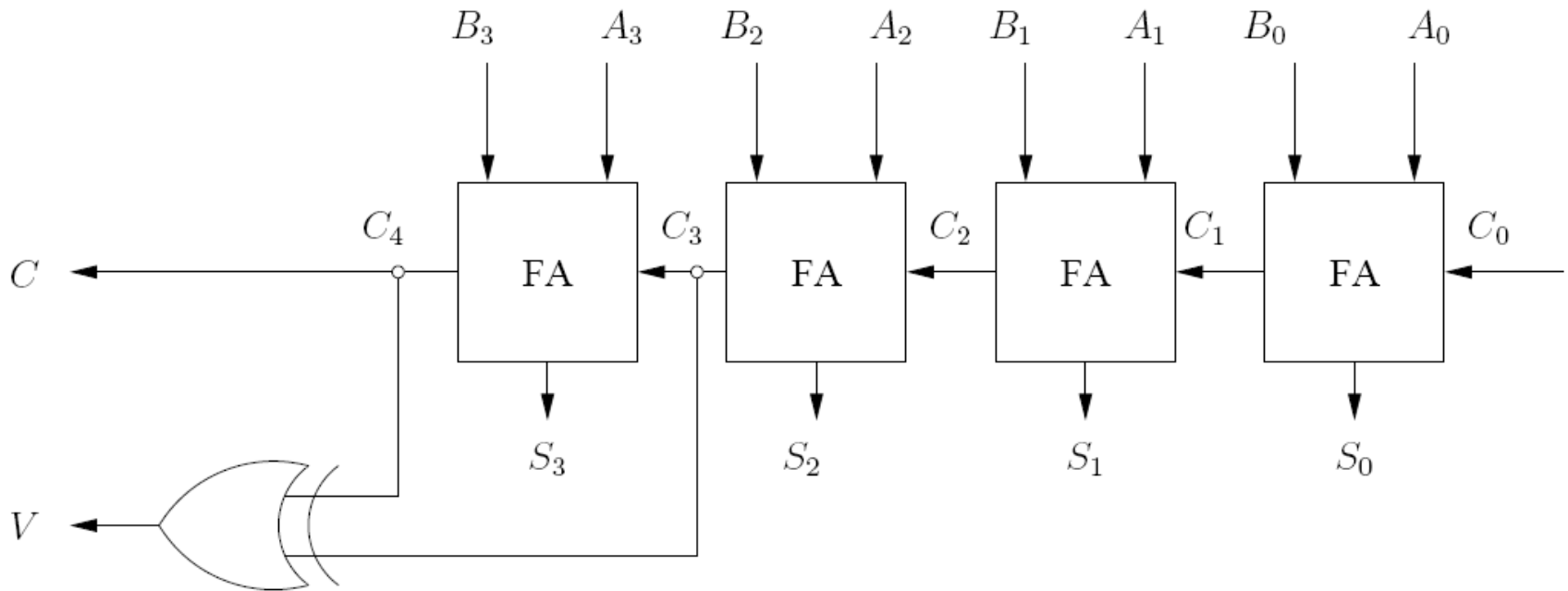
- Half Adder
 - inputs A, B
 - outputs Carry, Sum
- Full Adder
 - inputs A, B, Carry-in
 - outputs Sum, Carry-out

$$S = C_i \oplus A \oplus B$$

$$C_o = BC_i + AC_i + AB$$

- Cascade to make multiple-bit adder

4-Bit Ripple-Carry Adder



Overflow for 2C addition

- Output of arithmetic operation exceeds wordlength
- In unsigned binary addition, just check C_o of MSB full adder
- More complicated in 2C!
- Think:
 - A and B have to have same sign for overflow
 - Sign of sum different to A and B for overflow
 - Check these three

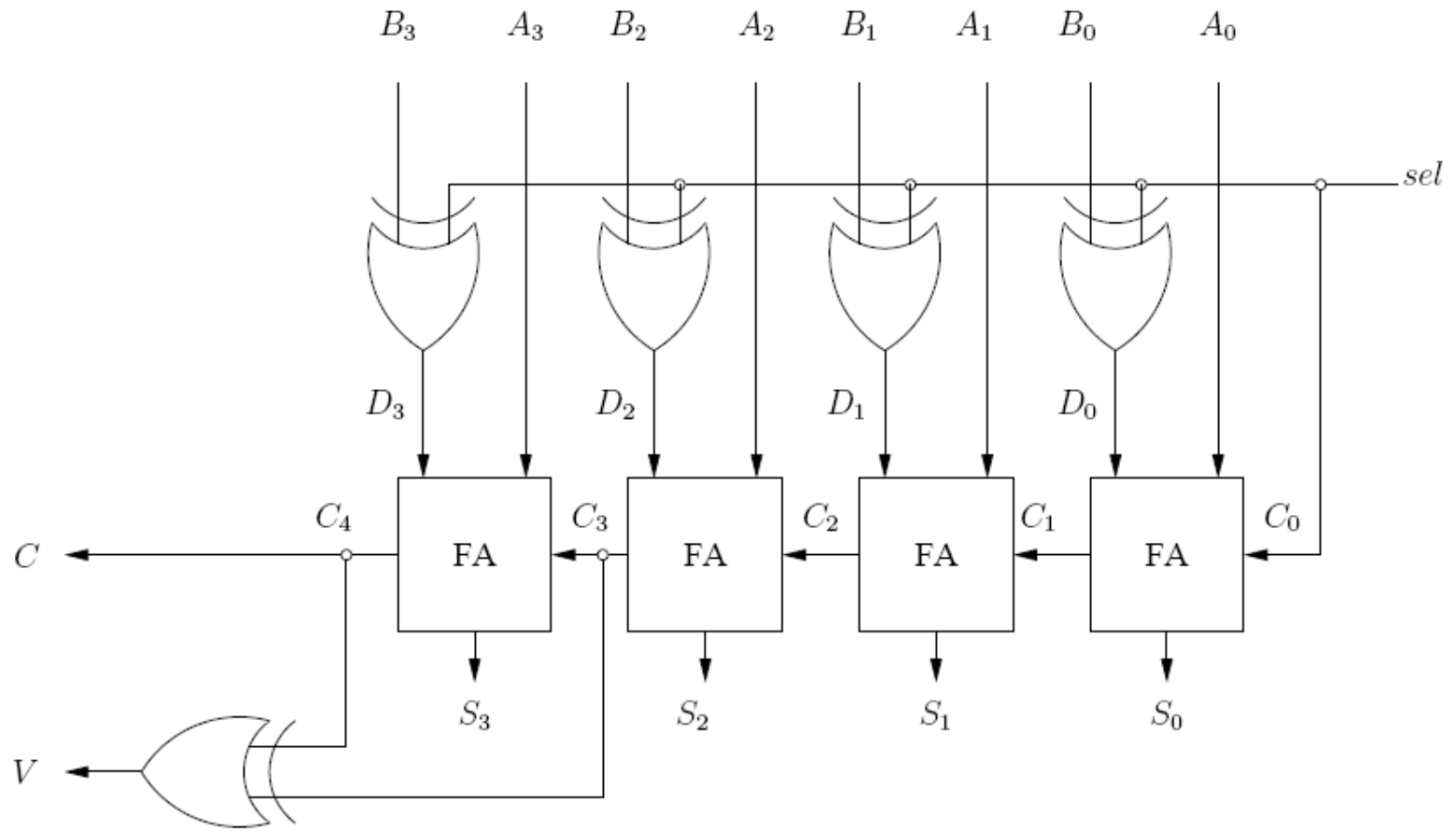
$$\begin{array}{r} A = a_{n-1}a_{n-2}a_{n-3}\cdots a_1a_0 \\ + B = b_{n-1}b_{n-2}b_{n-3}\cdots b_1b_0 \\ \hline \text{Sum} = s_{n-1}s_{n-2}s_{n-3}\cdots s_1s_0 \end{array}$$

$$V = a_{n-1}b_{n-1}\overline{s_{n-1}} + \overline{a_{n-1}}\overline{b_{n-1}}s_{n-1}$$

Overflow ctd

- An alternative: $V = c_{o_{n-1}} \oplus c_{o_{n-2}}$
- XOR the carry-in and carry-out of MSB adder
- Scenario: Carry in 0, carry out 1
 - only way to get carry-out 1 is if a_{n-1} and b_{n-1} are both 1 (negative numbers). Gives sum 0 and carry out 1 i.e. overflow
 - Added two negative numbers, got positive result
- Scenario 2: Carry in 1, carry out 0
 - Since you have carry-in of 1, both a_{n-1} and b_{n-1} must be 0 to give no carry out. Hence sum is 1.
 - Added two positive numbers and got negative result

4-bit Adder/Subtractor



Carry Lookahead Circuit

- Ripple adder requires propagation of carry bits from previous stage
 - Delay, increases with number of bits
- Carry Lookahead logic
 - express each c_i in terms of $a_i a_{i-1} \cdots a_0$ and $b_i b_{i-1} \cdots b_0$ and c_0 directly

Carry Lookahead

- Define functions carry generate and carry propagate

$$\text{Generate : } G_i = a_i b_i$$

$$\text{Propagate : } P_i = a_i \oplus b_i$$

- Recall expression for carry out

$$c_{out_i} = G_i + P_i c_{in_i}$$

$$s_i = P_i \oplus c_{in_i}$$

Carry Lookahead

- Carry out logic can be rewritten
- Refer to Katz
- i-th carry signal is OR of i+1 product terms with up to i+1 literals
- 3 gate delays
 - One for the generate and propagate
 - one to AND these
 - one to OR all product terms

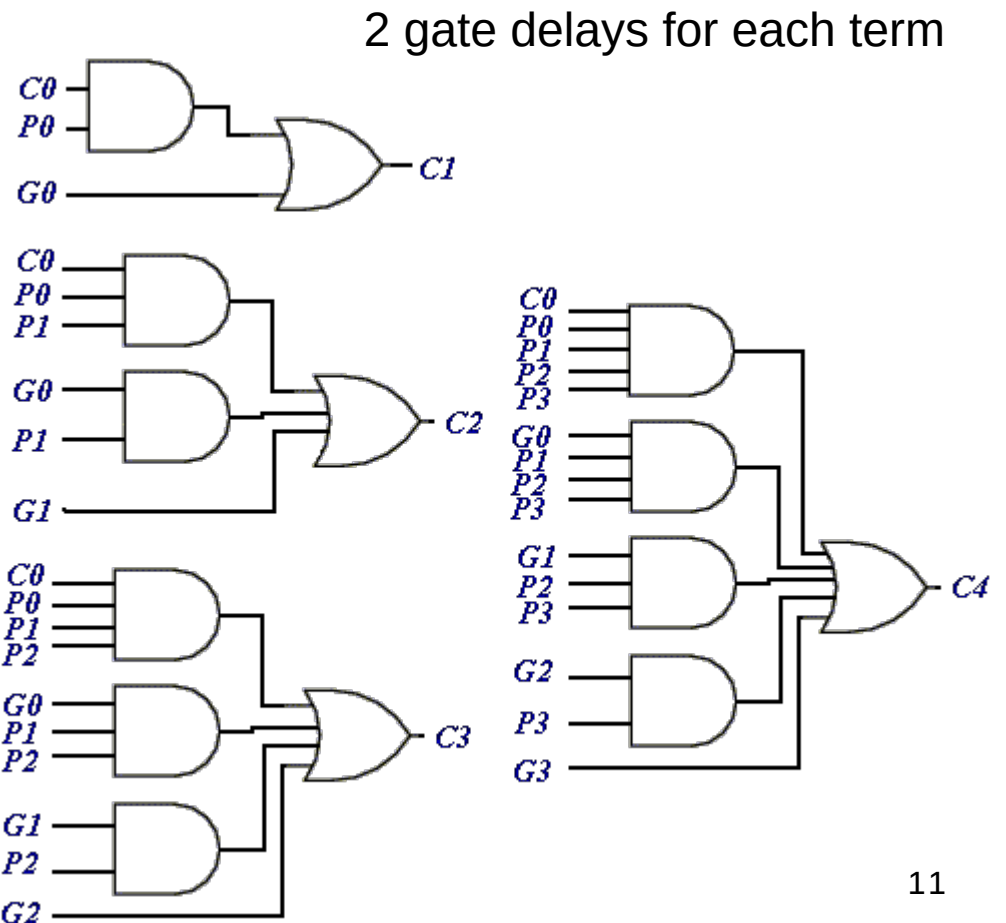
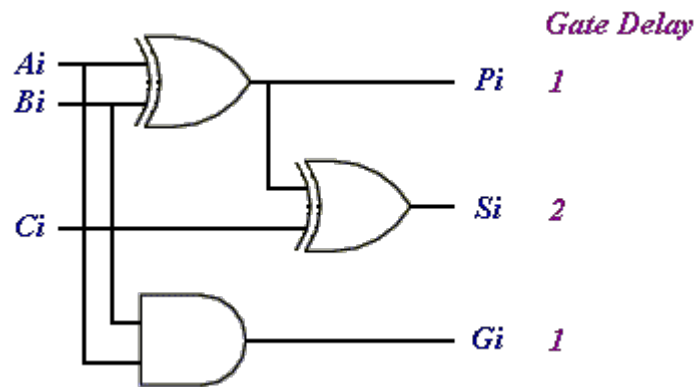
$$C_{out_0} = G_0 + P_0 C_{in_0}$$

$$C_{out_1} = G_1 + P_1 G_0 + P_1 P_0 C_{in_0}$$

$$C_{out_2} = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in_0}$$

$$C_{out_3} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in_0}$$

Logic for 4-bit Carry Lookahead



Overall delay in carry lookahead

