Carry Lookahead Generator, Adder/Subtractor, BCD Adder, Multiplier

Instructor Name: Engr. Rimsha

Binary Adder

- A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.
- A binary adder can be implemented using multiple full adders (FA) connected in cascade with the output carry from each full adder to the input carry of the next full adder in the chain

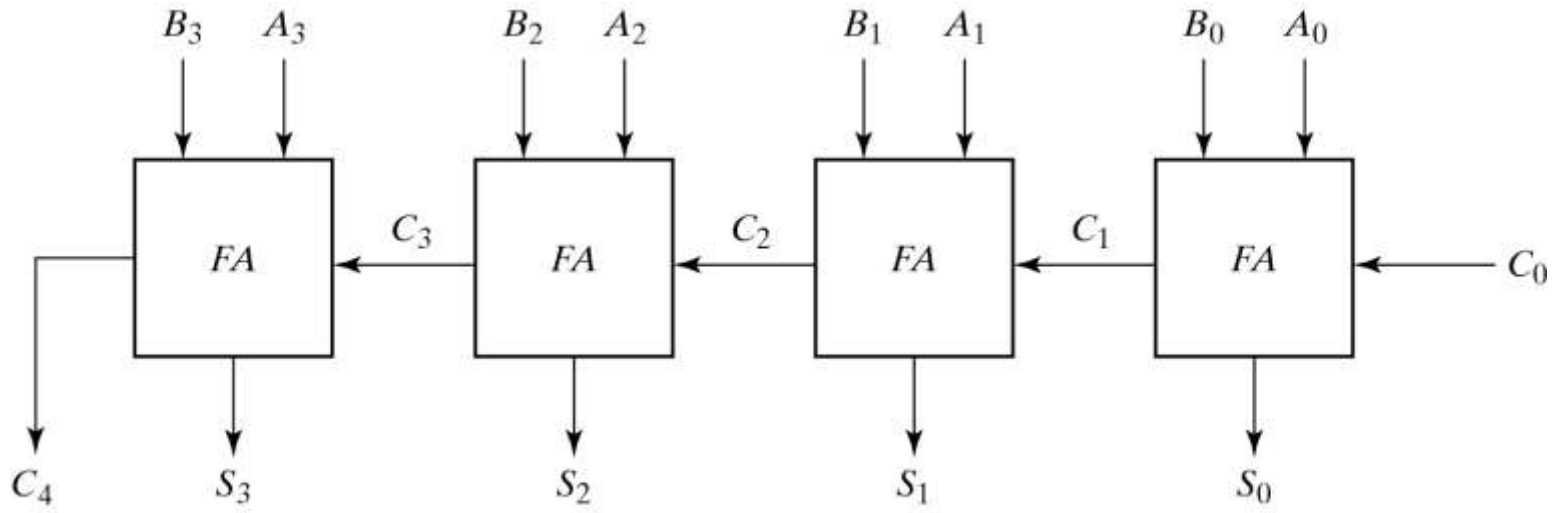


Fig. 4-9 4-Bit Adder

Carry Propagation

- The number of gate levels for the carry propagation can be found from the circuit of the full adder
- The input and output variables use the subscript *i* to denote a typical stage in the adder
- The signals at P_i and G_i settle to their steady state values after they propagate through their respective gates

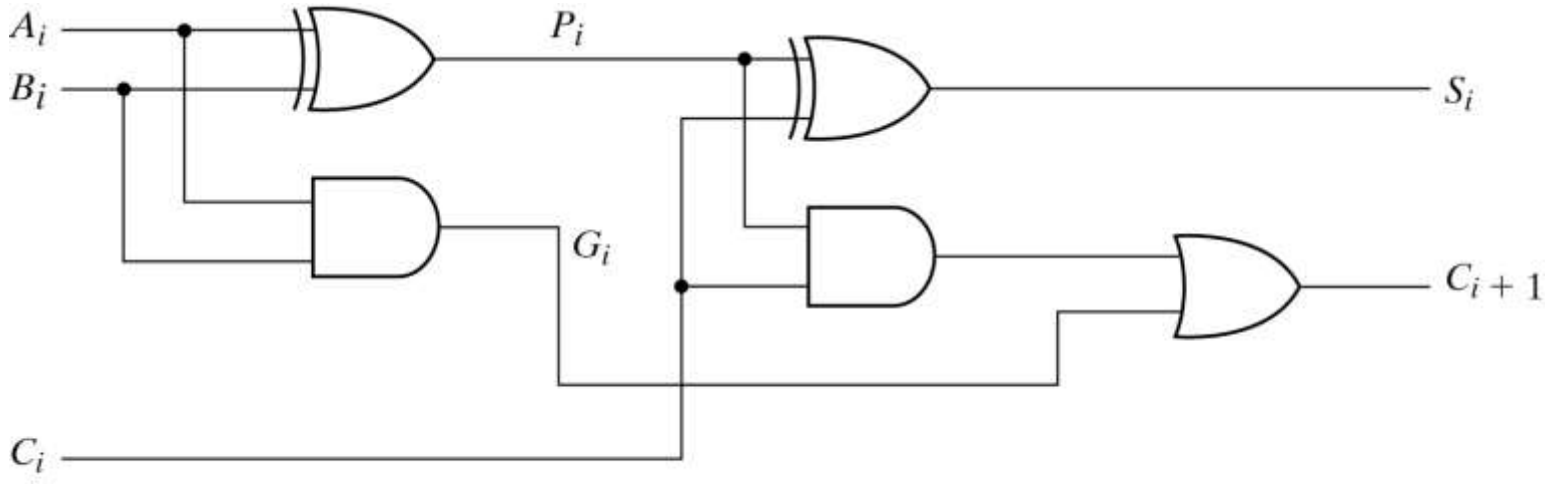


Fig. 4-10 Full Adder with P and G Shown

Carry Propagation

- These two signals are common to all full adders and depend only on the augend and addend bits
- The signal from the input carry C_i to the output carry C_{i+1} propagates through an AND gate and an OR gate, which constitutes two gate levels
- If there are four full adders in the adder, the output carry C_4 would have 2 x 4 = 8 gate levels from C_0 to C_4.
- For an n-bit adder, there are 2n gate levels for the carry to propagate from input to output
- The outputs of a combinational circuit will not be correct unless the signals are given enough time to propagate through the gates connected from inputs to outputs
- All other arithematic operations are implemented by successive additions, the time consumed during the addition process is very critical.

Carry Propagation

- One choice to reduce the carry propagation delay is to employ faster gates but the most widely used technique for reducing the carry propagation time in parallel adder is principle of carry lookahead
- Consider the circuit of full adder shown in fig 4-10
- If we define two binary variables
  - carry propagate: $P_i = A_i \oplus B_i$ (Term associated with the propagation of carry from C_i to C_{i+1})
  - carry generate: $G_i = A_i B_i$ (Produces 1 when both Ai and Bi are 1)
- Output sum and carry can be expressed as
  - sum: $S_i = P_i \oplus C_i$
  - carry: $C_{i+1} = G_i + P_i C_i$

Carry Propagation

- We now write the Boolean functions for the carry outputs of each stage and substitute for each C_i, its value from the previous equation
  - C_0= input carry
  - $C_1 = G_0+P_0C_0$
  - $C_2 = G_1+P_1C_1$
  - $= G_1+P_1(G_0+P_0C_0)$
  - $= G_1+P_1G_0+P_1P_0C_0$
  - $C_3 = G_2+P_2C_2$
  - $= G_2+P_2G_1+P_2P_1G_0+ P_2P_1P_0C_0$
  - $C_4 = G_3+P_3C_3$
  - $=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0+P_3P_2P_1P_0C_0$
- Since the Boolean function for each output carry is expressed in sum of products, each function can be implemented with one level of AND gates followed by an OR gate (or by two-level NAND)

Carry Lookahead Generator

- The three Boolean functions for C_1, C_2 and C_3 are implemented in the carry lookahead generator shown in figure 4-11
- Here C_3 doesn't have to wait for C_2 and C_1 to propagate and C_3 is propagated at the same time as C_1 and C_2

Carry Lookahead Generator

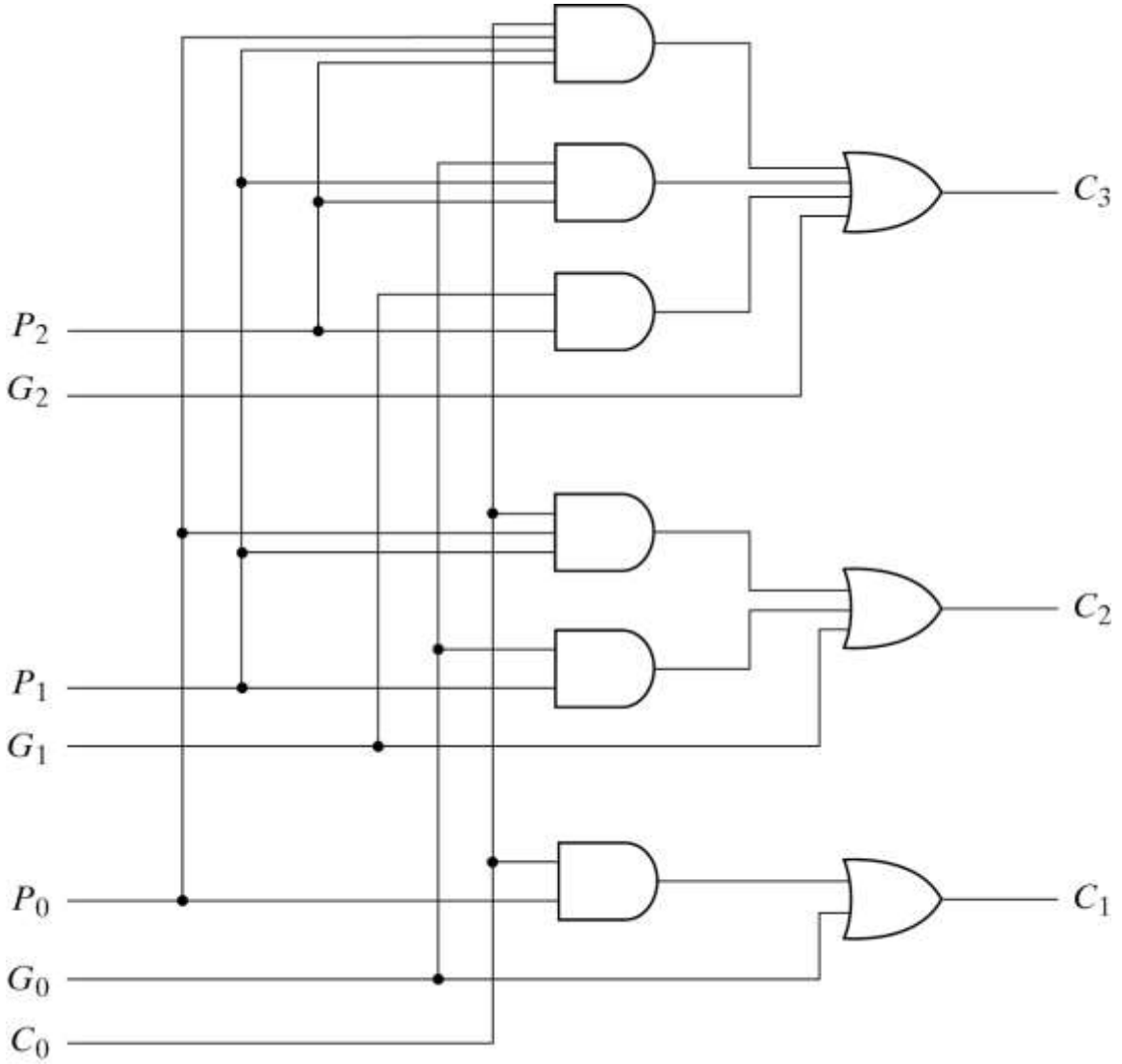
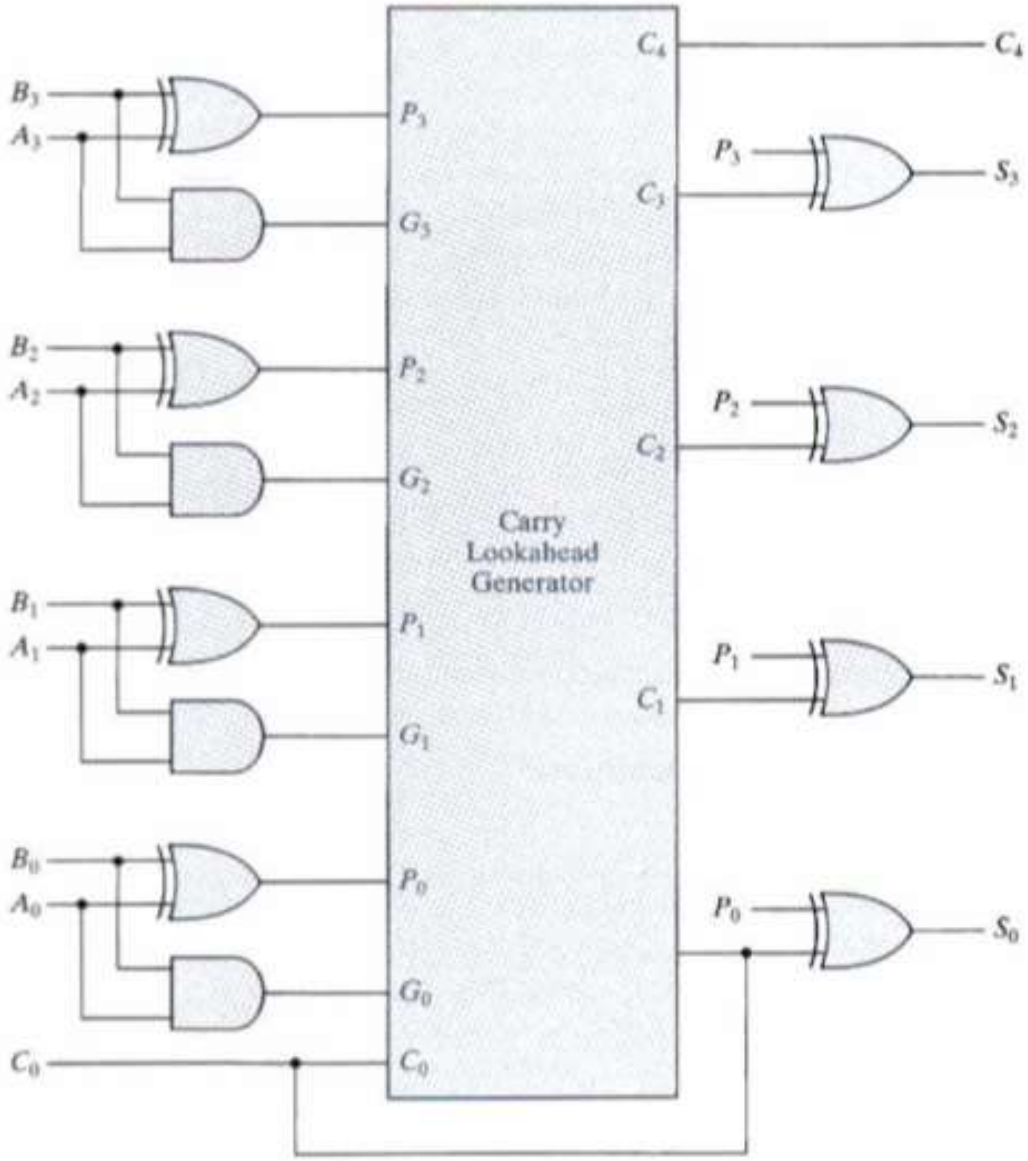


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

Four Bit Adder with Look Ahead Carry Generator



4-Bit Adder with Carry Lookahead

- The construction of 4-bit adder with a carry lookahead generator is shown in fig 4-12
- Each sum output requires two exclusive-OR gates.
- The output of first exclusive-OR gate generates the P_i variable and the AND gate generates the G_i variable
- The carries are propagated through the carry lookahead generator and applied as inputs to the second exclusive-OR gate
- All outputs carries are generated after a delay through two levels of gates
- Outputs S_1 through S_3 have equal propagation delay times

- **Binary Subtractor**
- **BCD Adder**

Binary subtractor

- The subtraction of unsigned binary numbers can be easily done by means of complements.(section 1-5)
- A-B = A+(2's complement of B)
- The 2's complement can be obtained by taking the 1's complement and adding 1 to least significant pair of bits
- To obtain 2's complement, the 1's complement can be implemented with inverters and one can be added to the sum through the input carry

Binary Subtractor

- The circuit for subtracting A – B consists of an adder with inverters placed between each data input B and the corresponding input of full adder
- The input carry C_0 must be equal to 1 when performing subtraction
- The operation thus performed becomes A, plus the 1's complement of B, plus 1 (A+1's complement of B+1). This is equal to A plus 2's complement of B (A+2's complement of B)
- For unsigned numbers this gives A – B if A $\geq$ B or the 2's complement of (B – A) if A<B

Binary subtractor

- The addition and subtraction operation can be combined into one circuit with one common binary adder
- This is done by including an exclusive-OR gate with each full adder
- The mode input M controls the operation. When M=0, we have B⊕0= B. The full adder receives the value of B, the input carry is 0 and the circuit performs A+B
- When M=1, we have B⊕1= B'. The full adder receives the value of B', the input carry C_0 is 1 and the circuit performs A plus the 2's complement of B. (A+1's complement of B+1).
- The exclusive-OR with output V is for detecting an overflow (detail later, in overflow)

Binary subtractor

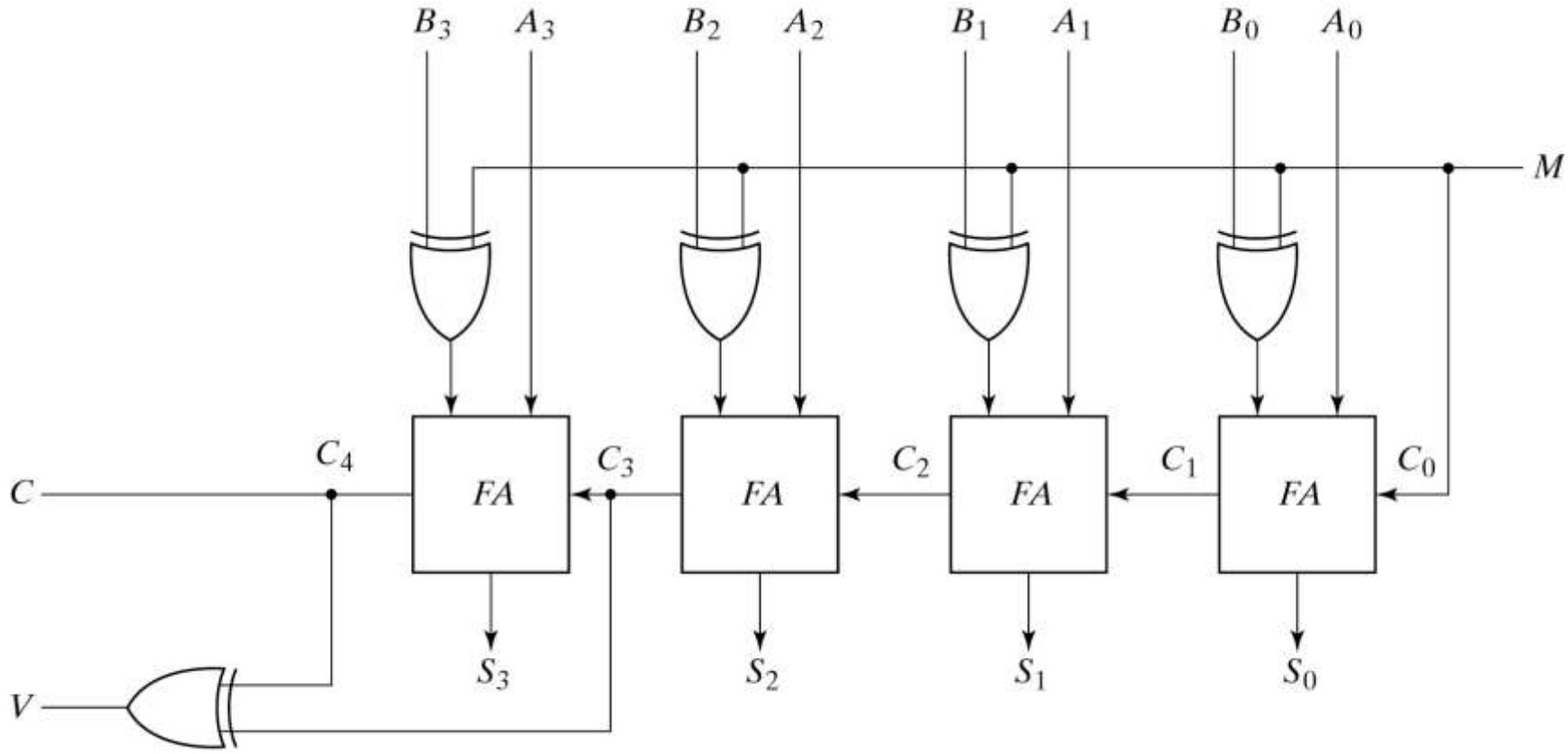


Fig. 4-13 4-Bit Adder Subtractor

Overflow

- When two numbers of n digits are added and sum occupies n+1 digits we say that an overflow occurred
- Overflow is problem in digital computers because the number of bits that hold the number (limited storage) is finite and a result that contains n+1 can't be accommodated.
- The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be signed or unsigned
- When two unsigned numbers are added an overflow is detected from the end carry out of the most significant position
- In the case of signed numbers, the leftmost bit always represent the sign and negative numbers are in 2's complement form

Overflow

- **When two signed numbers are added, the sign bit is treated as part of the number and end carry doesn't indicate an overflow**
- **An overflow can't occur after an addition if one number is positive and the other is negative, since adding a positive number to a negative number produces a result which is smaller than the larger of the two original numbers.**
- **An overflow may occur if the two numbers added are both positive or both negative**
- **For n-bit registers, the range of numbers (signed numbers) that each register can accommodate is from binary 2^{n-1}-1 to binary -2^{n-1}.**

Overflow

- **For 8-bit registers, the range of numbers that each register can accommodate is from binary +127 to binary –128**
- **Consider two signed numbers +70 and +80 stored in two 8-bit registers. The sum of two numbers is +150 which exceeds the capacity of eight bit register**

Carries:	0 1	Carries:	1 0
+70	0 1000110	-70	1 0111010
+80	0 1010000	-80	1 0110000
+150	1 0010110	-150	0 1101010

Note: Negative numbers are in 2's complement form

Overflow

- If we just consider the 8-bit result from the last example, we go wrong as we
  - Add two positive numbers and obtain a negative number
  - Add two negative numbers and obtain a positive number
- The carry out of sign bit position is taken as the sign bit of the result, then the 9-bit answer so obtained will be correct. This answer can't be accommodated within 8-bits we say that an overflow has occurred
- An overflow has occurred if carry into the sign bit position and carry out of the sign bit position are not equal. This can be found by applying the two carries to an exclusive-OR gate. 1 at the output of the gate indicates overflow
  - V=0, no overflow; V=1, overflow

The End