



L11: Arithmetic Structures



Acknowledgements:

R. Katz, “*Contemporary Logic Design*”, Addison Wesley Publishing Company, Reading, MA, 1993.

J. Rabaey, A. Chandrakasan, B. Nikolic, “*Digital Integrated Circuits: A Design Perspective*” Prentice Hall, 2003.

Kevin Atkinson, Alice Wang



Number Systems Basics



How to represent negative numbers?

- Three common schemes: sign-magnitude, ones complement, twos complement
- Sign-magnitude: MSB = 0 for positive, 1 for negative
 - Range: $-(2^{N-1} - 1)$ to $+(2^{N-1} - 1)$
 - Two representations for zero: 0000... & 1000...
 - Leads to complicated addition/subtraction, but simple multiplication
- Ones complement: if N is positive then its negative is \bar{N}
 - Ex.: 0111 = 7, 1000 = -7
 - Range: $-(2^{N-1} - 1)$ to $+(2^{N-1} - 1)$
 - Still two representations for zero: 0000... & 1111...
 - Subtraction implemented as addition followed by ones complement



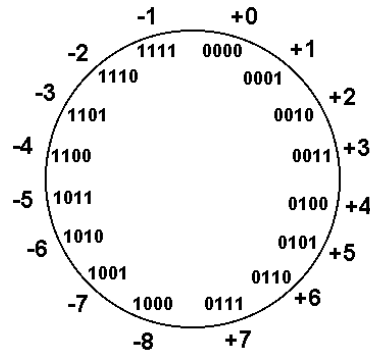
Twos Complement Representation



Twos complement = bitwise complement + 1

0111 -> 1000 + 1 = 1001 = -7
1001 -> 0110 + 1 = 0111 = 7

- Asymmetric range: -2^{N-1} to $+2^{N-1}-1$
- Only one representation for zero
- Simple addition and subtraction
- Most common representation



4	0100	-4	1100	4	0100	-4	1100
+3	0011	+(-3)	1101	-3	1101	+3	0011
7	0111	-7	11001	1	10001	-1	1111

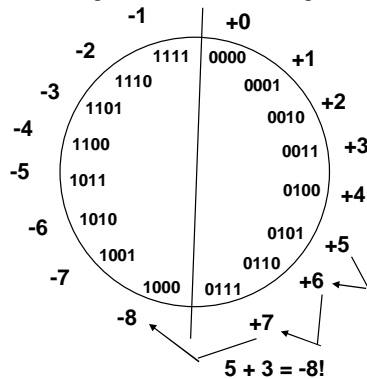
If carry in to sign equals carry out then can ignore carry out, otherwise have overflow



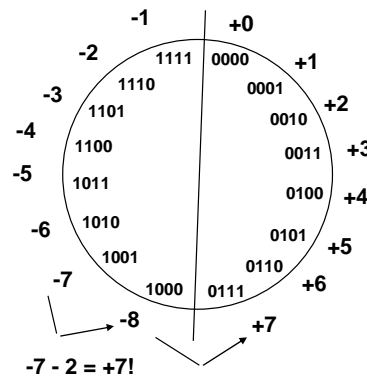
Overflow Conditions



Add two positive numbers to get a negative number
or two negative numbers to get a positive number



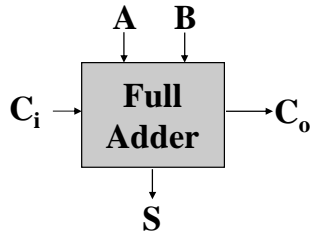
5	0111
	0101
+3	0011
-8	01000



-7	1000
	1001
-2	1100
7	0111



Binary Full Adder



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

$$= \overline{A} \overline{B} C_i + \overline{A} B \overline{C}_i + A \overline{B} \overline{C}_i + A B C_i$$

$$C = AB + BC_i + AC_i$$

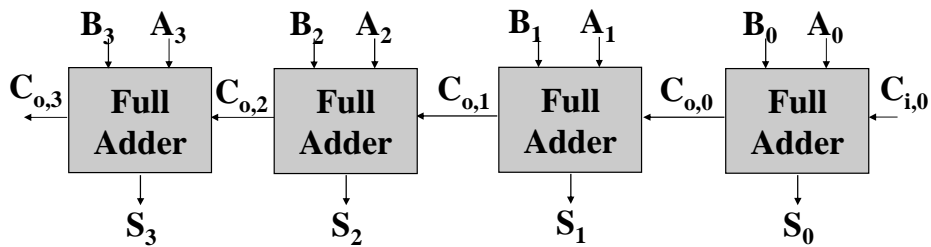
A	B	C _i	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

C _i	A B			
	00	01	11	10
0	0	1	0	1
1	1	0	1	0

C _o	A B			
	00	01	11	10
0	0	0	1	0
1	0	1	1	1



Ripple Carry Adder Structure



Worst case propagation delay linear with the number of bits

$$t_{\text{adder}} = (N-1)t_{\text{carry}} + t_{\text{sum}}$$

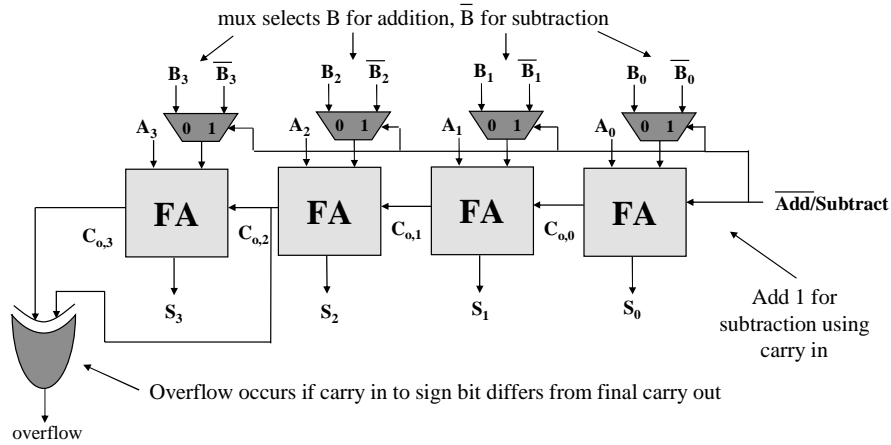


Extension to Subtraction

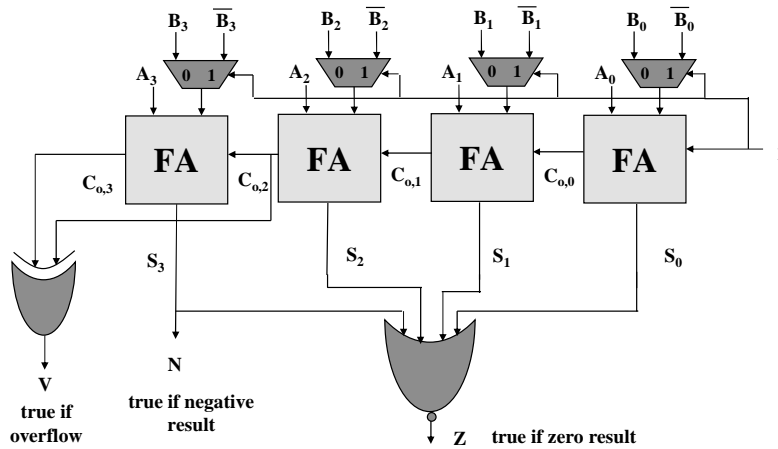


- Under two's complement, subtracting B is the same as adding the bitwise complement of B then adding 1

Combination addition/subtraction system:



Comparator (one approach)



$$\begin{array}{ll}
 A < B = N \oplus V & A > B = \overline{Z + (N \oplus V)} \\
 A \leq B = Z + (N \oplus V) & A \geq B = \overline{N \oplus V} \\
 A = B = Z & A \neq B = \overline{Z}
 \end{array}$$

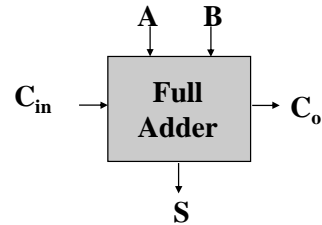


Alternate Adder Logic Formulation



How to Speed up the Critical (Carry) Path? (How to Build a Fast Adder?)

A	B	C_i	S	C_o	Carry status
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate



$$\text{Generate } (G) = AB$$

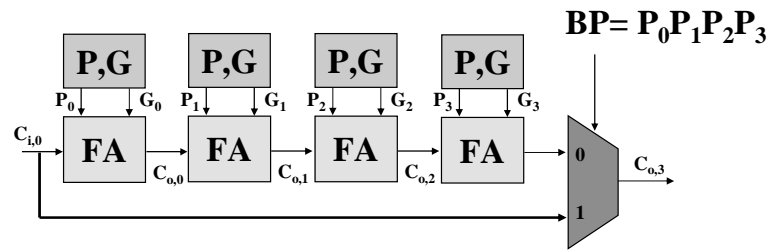
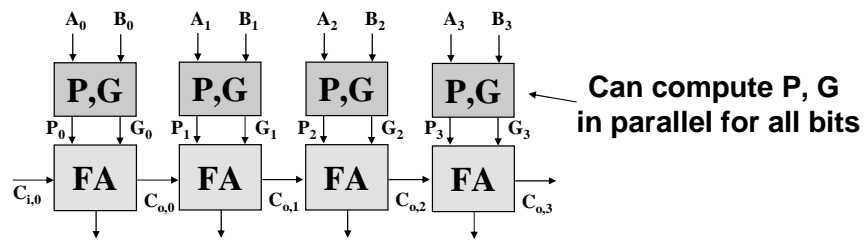
$$\text{Propagate } (P) = A \oplus B$$

$$C_o(G, P) = G + PC_i$$

$$S(G, P) = P \oplus C_i$$



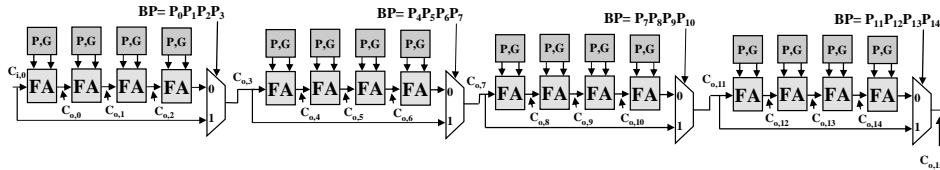
Carry Bypass Adder



Key Idea: if $(P_0 P_1 P_2 P_3)$ then $C_{o,3} = C_{i,0}$



16-bit Carry Bypass Adder



Assume the following for delay each gate:

P, G from A, B: 1 delay unit

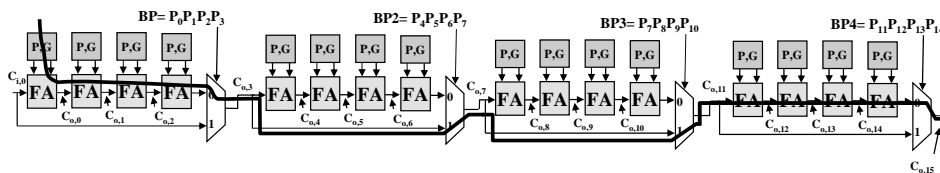
P, G, C_i to C_o or Sum for a FA: 1 delay unit

2:1 mux delay: 1 delay unit

What is the worst case propagation delay for the 16-bit adder?



Critical Path Analysis



For the second stage, is the critical path:

BP2 = 0 or BP2 = 1?

**Message: Timing Analysis is Very Tricky –
Must Carefully Consider Data Dependencies For
False Paths**



Carry Lookahead Adder



Re-express the carry logic as follows:

$$C1 = G0 + P0 C0$$

$$C2 = G1 + P1 C1 = G1 + P1 G0 + P1 P0 C0$$

$$C3 = G2 + P2 C2 = G2 + P2 G1 + P2 P1 G0 + P2 P1 P0 C0$$

$$C4 = G3 + P3 C3 = G3 + P3 G2 + P3 P2 G1 + P3 P2 P1 G0 + P3 P2 P1 P0 C0$$

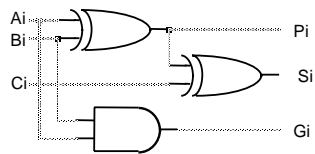
...

- Each of the carry equations can be implemented in a two-level logic network
- Variables are the adder inputs and carry in to stage 0

Ripple effect has been eliminated!

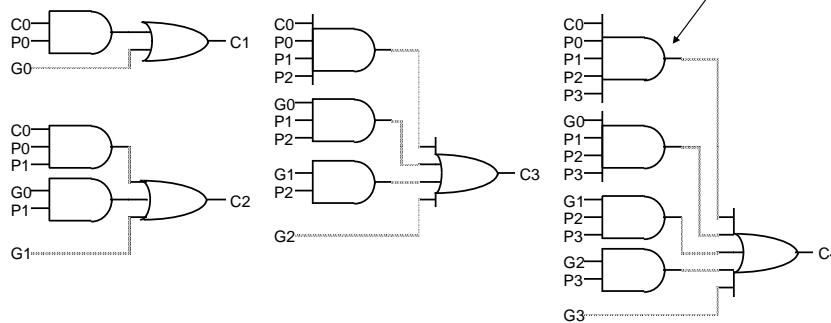


Carry Lookahead Logic



Adder with propagate and generate outputs

Later stages have increasingly complex logic





Block Generate and Propagate



G_{ij} and P_{ij} denote the Generate and Propagate functions, respectively, for a group of bits from positions i to j . We call them Block Generate and Block Propagate. G_{ij} equals 1 if the group generates a carry independent of the incoming carry. P_{ij} equals 1 if an incoming carry propagates through the entire group. For example, $G_{3,2}$ is equal to 1 if a carry is generated at bit position 3, or if a carry is generated at bit position 2 and propagates through position 3. $G_{3,2} = G_3 + P_3 G_2$. $P_{3,2}$ is true if an incoming carry propagates through both bit positions 2 and 3. $P_{3,2} = P_3 P_2$

$$C_{0,1} = (G_1 + P_1 G_0) + (P_1 P_0) C_{i,0} = G_{1:0} + P_{1:0} C_{i,0}$$

$$C_{0,3} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{i,0}$$

$$= (G_3 + P_3 G_2) + (P_3 P_2) C_{0,1} = G_{3:2} + P_{3:2} C_{0,1}$$

$$= G_{3:2} + P_{3:2} (G_{1:0} + P_{1:0} C_{i,0})$$

The carry out of a 4-bit block can thus be computed using only the block generate and propagate signals for each 2-bit section, plus the carry in to bit 0. The same formulation will be used to generate the carry out signals for a 16-bit adder using the block generate and propagate from 4-bit sections.



74181 TTL 4-bit ALU (TI)



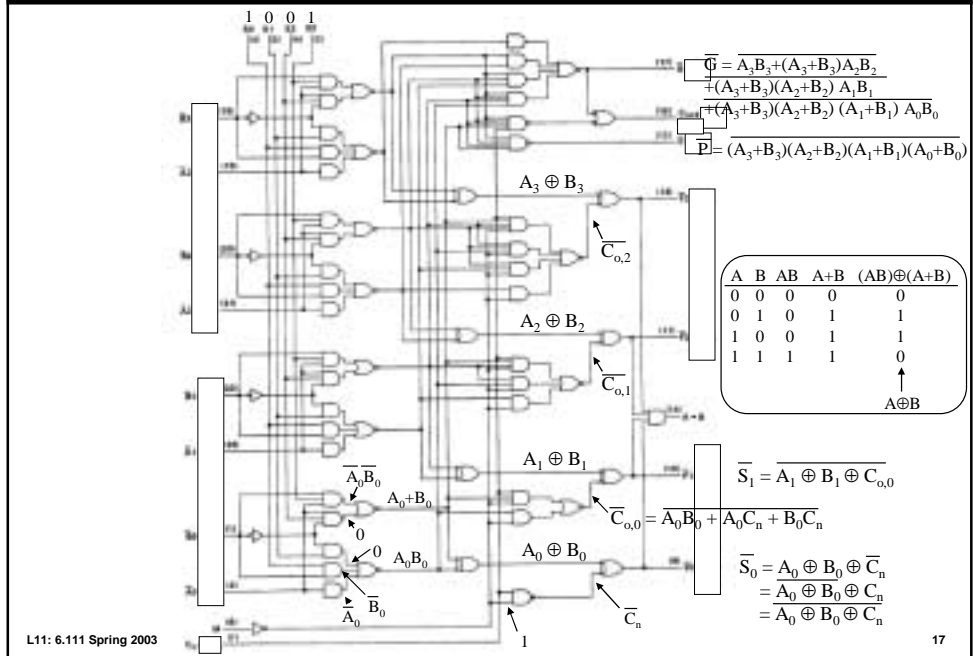
SELECTION	M = H		M = L: ARITHMETIC OPERATIONS	
	LOGIC FUNCTIONS		C _{in} = L (no carry)	C _{in} = H (with carry)
	S ₃	S ₂	S ₁	S ₀
L L L L	$F = A$	$F = A \text{ MINUS } 1$	$F = A$	$F = A$
L L L H	$F = \overline{AB}$	$F = AB \text{ MINUS } 1$	$F = AB$	$F = AB$
L L H L	$F = \overline{A} + B$	$F = \overline{A} \overline{B} \text{ MINUS } 1$	$F = \overline{A} \overline{B}$	$F = \overline{A} \overline{B}$
L L H H	$F = 1$	$F = \text{MINUS } 1$ (2's COMP)	$F = \text{ZERO}$	$F = \text{ZERO}$
L H L L	$F = \overline{A} + \overline{B}$	$F = A \text{ PLUS } (A + \overline{B})$	$F = A \text{ PLUS } (A + \overline{B}) \text{ PLUS } 1$	$F = A \text{ PLUS } (A + \overline{B}) \text{ PLUS } 1$
L H L H	$F = \overline{B}$	$F = \overline{AB} \text{ PLUS } (A + \overline{B})$	$F = \overline{AB} \text{ PLUS } (A + \overline{B}) \text{ PLUS } 1$	$F = \overline{AB} \text{ PLUS } (A + \overline{B}) \text{ PLUS } 1$
L H H L	$F = A \oplus B$	$F = A \text{ MINUS } B \text{ MINUS } 1$	$F = A \text{ MINUS } B$	$F = A \text{ MINUS } B$
L H H H	$F = A + \overline{B}$	$F = A + \overline{B}$	$F = (A + \overline{B}) \text{ PLUS } 1$	$F = (A + \overline{B}) \text{ PLUS } 1$
H L L L	$F = \overline{AB}$	$F = A \text{ PLUS } (A + B)$	$F = A \text{ PLUS } (A + B) \text{ PLUS } 1$	$F = A \text{ PLUS } (A + B) \text{ PLUS } 1$
H L L H	$F = A \oplus B$	$F = A \text{ PLUS } B$	$F = A \text{ PLUS } B$	$F = A \text{ PLUS } B \text{ PLUS } 1$
H L H L	$F = B$	$F = \overline{AB} \text{ PLUS } (A + B)$	$F = \overline{AB} \text{ PLUS } (A + B) \text{ PLUS } 1$	$F = \overline{AB} \text{ PLUS } (A + B) \text{ PLUS } 1$
H L H H	$F = A + B$	$F = (A + B)$	$F = (A + B) \text{ PLUS } 1$	$F = (A + B) \text{ PLUS } 1$
H H L L	$F = 0$	$F = A \text{ PLUS } A^2$	$F = A \text{ PLUS } A \text{ PLUS } 1$	$F = A \text{ PLUS } A \text{ PLUS } 1$
H H L H	$F = \overline{AB}$	$F = \overline{AB} \text{ PLUS } A$	$F = \overline{AB} \text{ PLUS } A \text{ PLUS } 1$	$F = \overline{AB} \text{ PLUS } A \text{ PLUS } 1$
H H H L	$F = \overline{AB}$	$F = \overline{AB} \text{ PLUS } A$	$F = \overline{AB} \text{ PLUS } A \text{ PLUS } 1$	$F = \overline{AB} \text{ PLUS } A \text{ PLUS } 1$
H H H H	$F = A$	$F = A$	$F = A \text{ PLUS } 1$	$F = A \text{ PLUS } 1$

¹Each bit is shifted to the next more significant position.

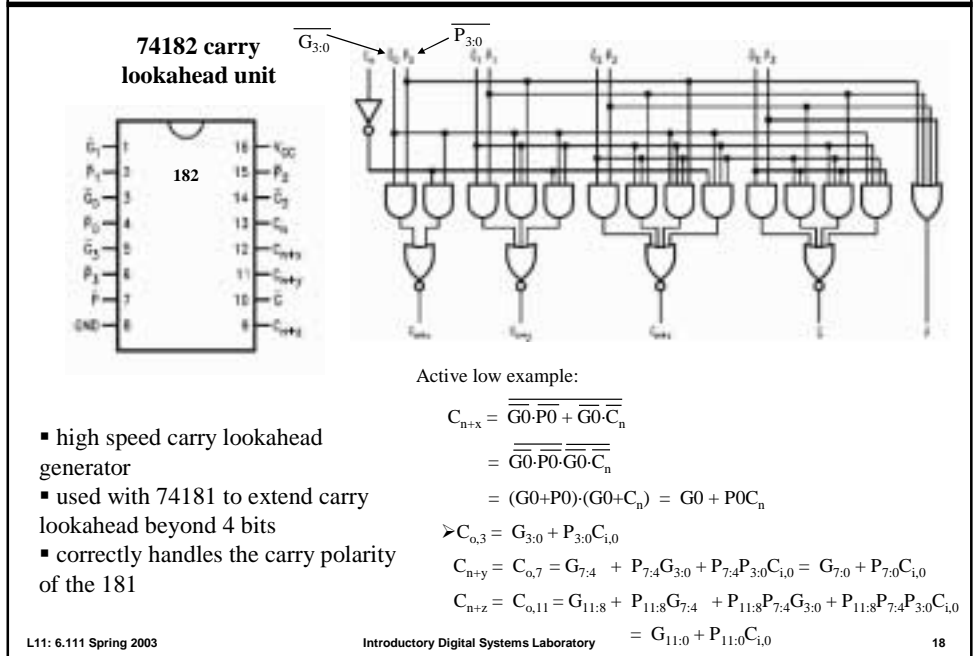
- 16 logic functions and 16 arithmetic operations
- Internal 4-bit carry lookahead adder
- Inputs can be active high or active low (active low is shown here)
- Carry in and out are opposite polarity from other inputs/outputs



74181 Addition (Active Low)

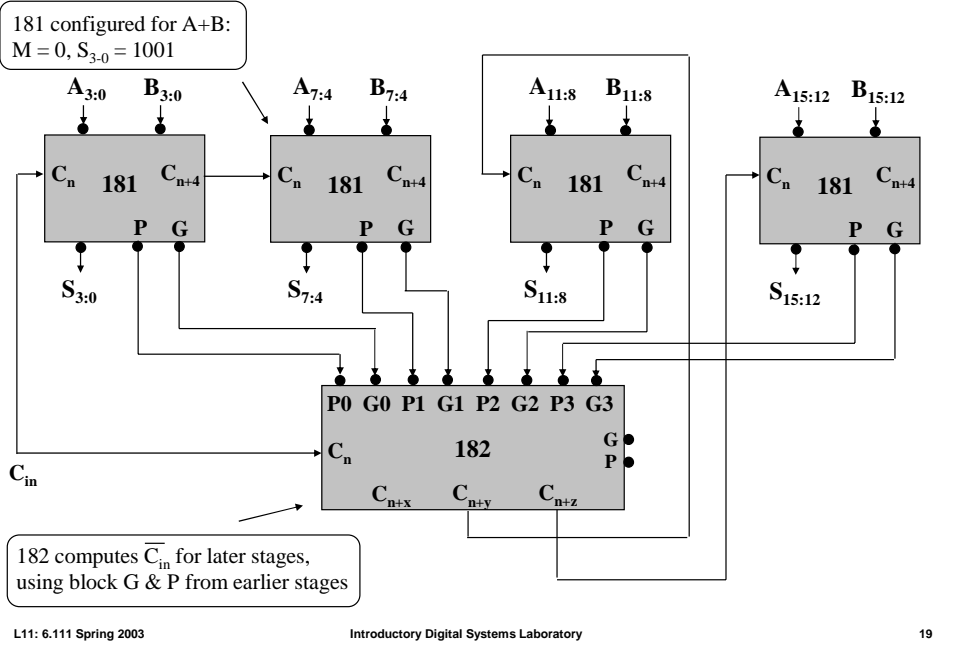


74182

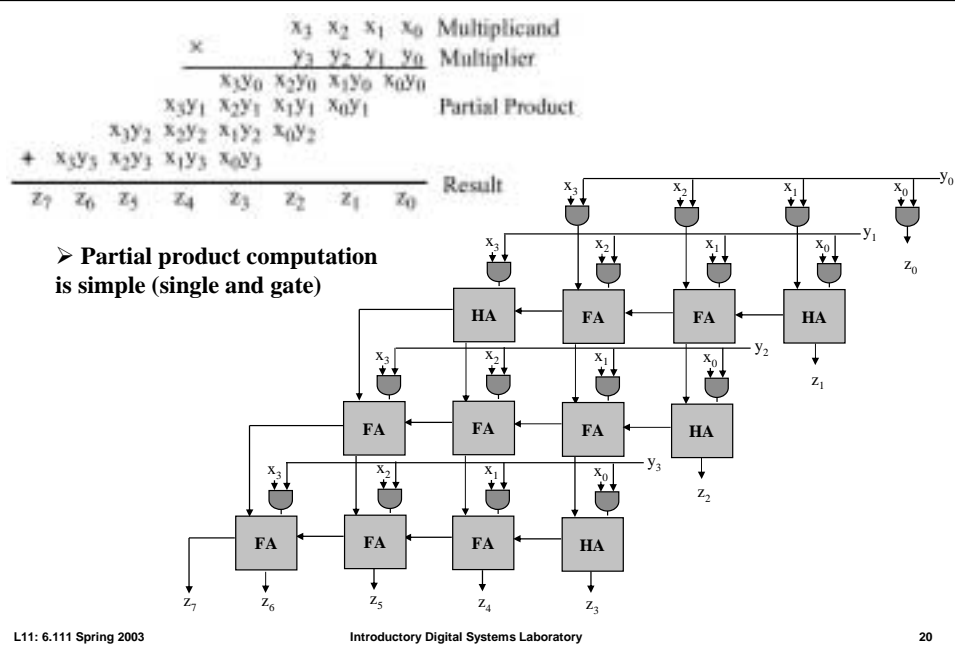


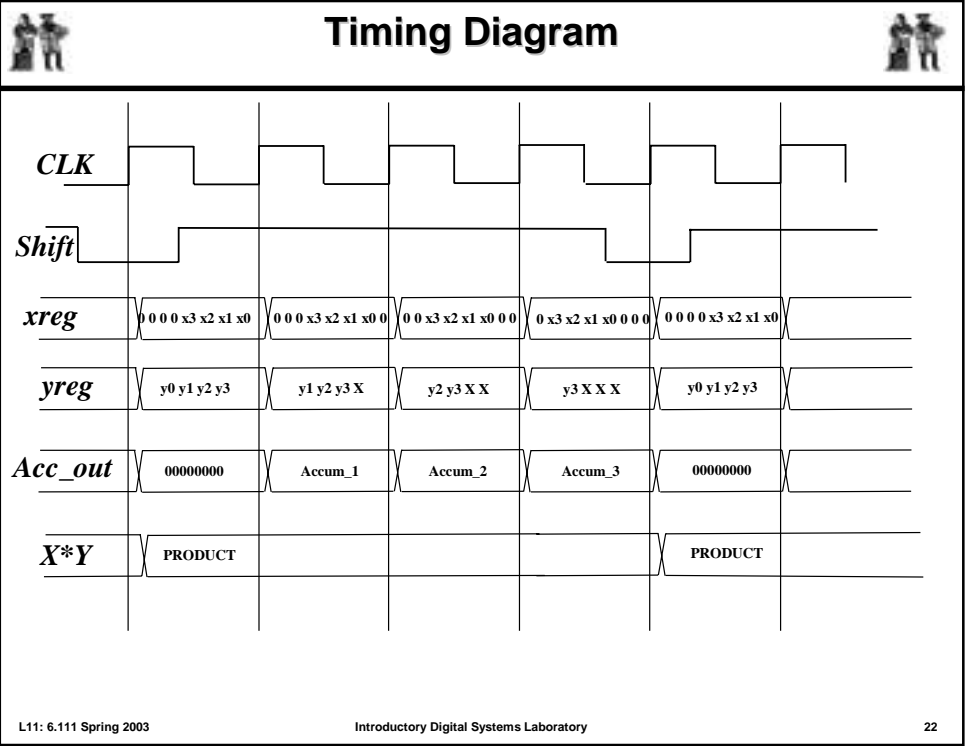
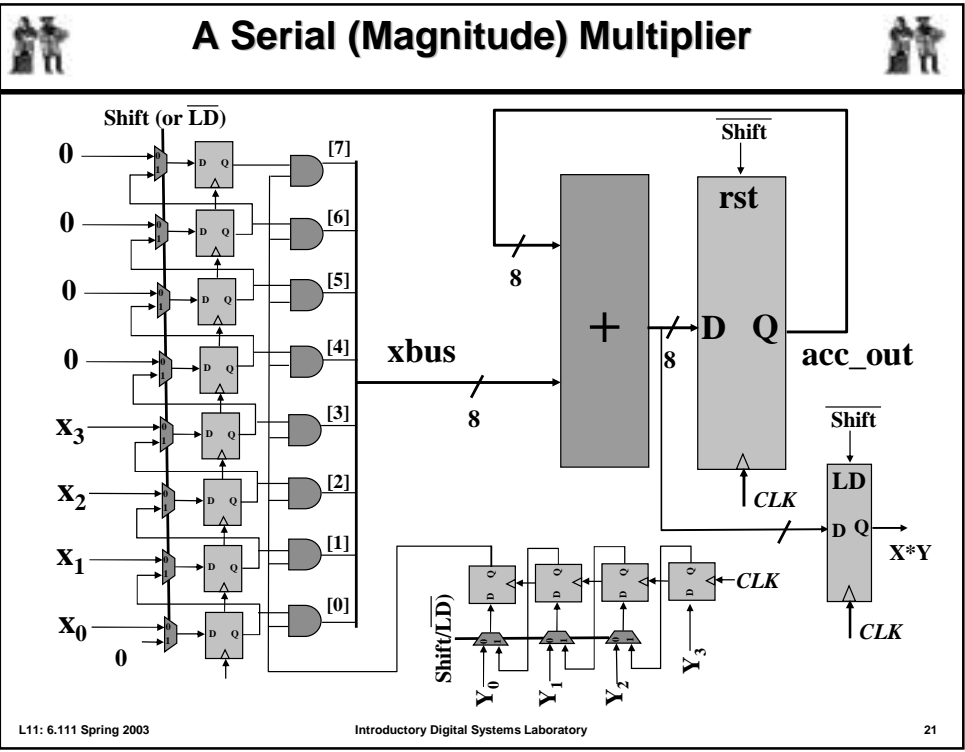


16-bit Carry Lookahead Schematic



Binary Multiplication







VHDL of Serial Multiplier



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity serialmult is
  port (SHIFT, CLK: in std_logic;
        X, Y : in std_logic_vector (3 downto 0);
        XY: out std_logic_vector (7 downto 0));
end serialmult ;

architecture behavior of serialmult is

  signal XREG, XBUS, acc_out, XY_int, add_out: std_logic_vector (7 downto 0);
  signal YREG: std_logic_vector (3 downto 0);
begin
  add_out <= XBUS + acc_out;
  XY <= XY_int;
  process (YREG(0), XREG)
  begin
    if (YREG(0) = '0') then
      XBUS <= "00000000";
    else
      XBUS <= XREG;
    end if;
  end process;
```



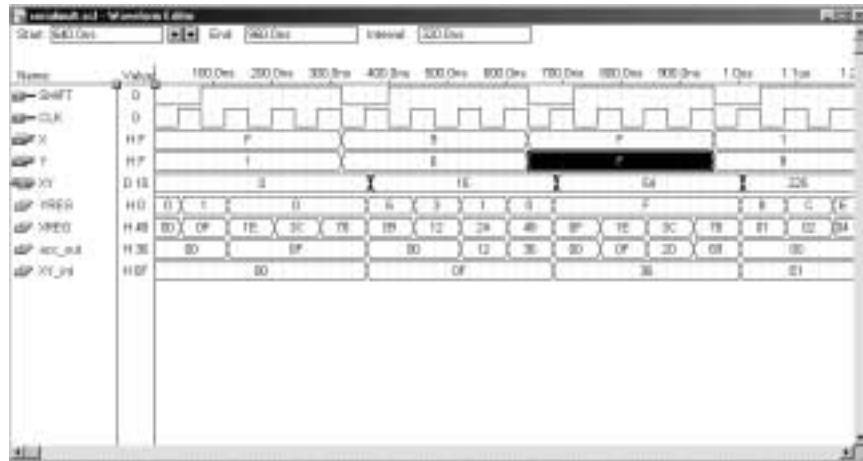
VHDL of Serial Multiplier (cont.)



```
process (CLK)
begin
  if (CLK'event and CLK = '1') then
    if (SHIFT = '0') then
      XREG (7 downto 4) <= "0000";
      XREG (3 downto 0) <= X;
      YREG <= Y;
      acc_out <= "00000000";
      XY_int <= add_out;
    else
      XREG (7 downto 1) <= XREG (6 downto 0);
      XREG (0) <= '0';
      YREG (2 downto 0) <= YREG (3 downto 1);
      YREG (3) <= Y(3);
      acc_out <= add_out;
      XY_int <= XY_int;
    end if;
  end if;
end process;
end architecture behavior;
```



Simulation



Baugh Wooley Formulation



Assuming X and Y are 4-bit two's complement numbers:

$$X = -2^3x_3 + \sum_{i=0}^2 x_i 2^i \quad Y = -2^3y_3 + \sum_{i=0}^2 y_i 2^i$$

The product of X and Y is:

$$XY = x_3y_32^6 - \sum_{i=0}^2 x_iy_32^{i+3} - \sum_{j=0}^2 x_3y_j2^{j+3} + \sum_{i=0}^2 \sum_{j=0}^2 x_iy_j2^{i+j}$$

Recall for two's complement:

$$-\sum_{i=0}^3 x_i 2^i = -2^4 + \sum_{i=0}^3 \bar{x}_i 2^i + 1$$

The product then becomes:

$$\begin{aligned} XY &= x_3y_32^6 + \sum_{i=0}^2 \bar{x}_i y_3 2^{i+3} + 2^3 - 2^6 + \sum_{j=0}^2 \bar{x}_3 y_j 2^{j+3} + 2^3 - 2^6 + \sum_{i=0}^2 \sum_{j=0}^2 x_i y_j 2^{i+j} \\ &= x_3y_32^6 + \sum_{i=0}^2 \bar{x}_i y_3 2^{i+3} + \sum_{j=0}^2 \bar{x}_3 y_j 2^{j+3} + \sum_{i=0}^2 \sum_{j=0}^2 x_i y_j 2^{i+j} + 2^4 - 2^7 \\ &= -2^7 + x_3y_32^6 + (\bar{x}_2 y_3 + \bar{x}_3 y_2)2^5 + (\bar{x}_1 y_3 + \bar{x}_3 y_1 + x_2 y_2 + 1)2^4 \\ &\quad + (\bar{x}_0 y_3 + \bar{x}_3 y_0 + x_1 y_2 + x_2 y_1)2^3 + (x_0 y_2 + x_1 y_1 + x_2 y_0)2^2 + (x_0 y_1 + x_1 y_0)2^1 \\ &\quad + (x_0 y_0)2^0 \end{aligned}$$

