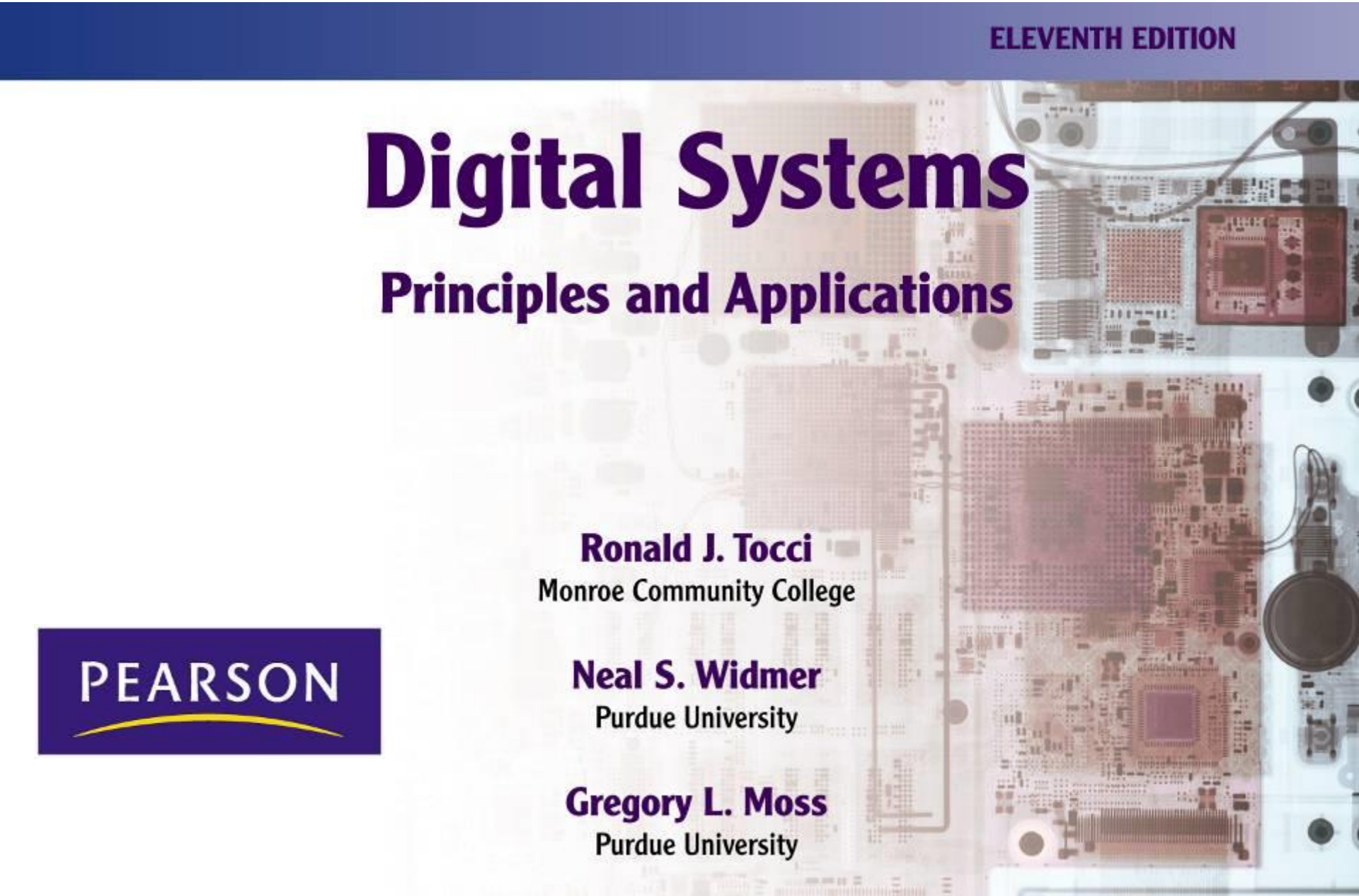

PEARSON

Chapter 6 – Digital Arithmetic: Operations & Circuits



Chapter 6 Objectives

- *Selected areas covered in this chapter*:
  - Binary addition, subtraction, multiplication, division.
  - Differences between binary addition and OR addition.
  - Advantages & disadvantages among three different systems of representing signed binary numbers.
  - Basic operation of an arithmetic/logic unit.
  - Operation of a parallel adder/subtractor circuit.
  - ALU integrated circuits for various logic and arithmetic operations on input data.
  - Digital functions from libraries to implement more complex circuits.
  - Boolean equation description to perform operations on entire sets of bits.

6-1 Binary Addition & Subtraction

- Binary numbers are added like decimal numbers.
  - In decimal, when numbers sum more than 9, a carry results.
  - In binary when numbers sum more than 1, a carry takes place.
- Addition is the basic arithmetic operation used by digital devices for subtraction, multiplication & division.

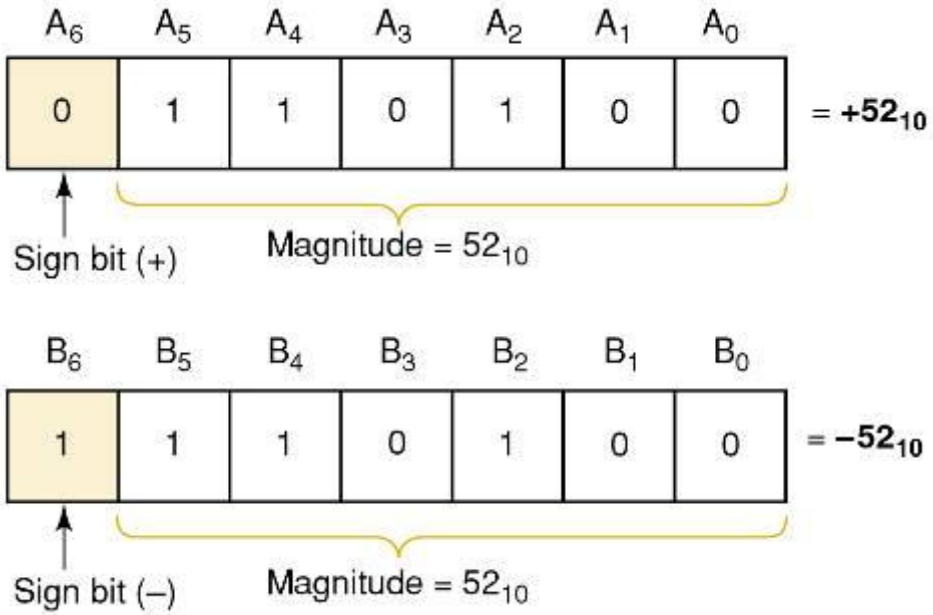
- Binary subtraction is performed just like the subtraction of decimal numbers.

Four possible situations when subtracting one bit from another in any position of a binary number.

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$0 - 1 \Rightarrow \text{needs to borrow} \Rightarrow 10 - 1 = 1$$

6-2 Representing Signed Numbers

- Since it is only possible to show magnitude with a binary number, the sign (+) or (-) is shown by adding an extra “sign” bit.
  - A sign bit of 0 indicates a positive number.
  - A sign bit of 1 indicates a negative number.



6-2 Representing Signed Numbers

- The 2's complement system is the most commonly used way to represent signed numbers.
- To change a binary number to 2's complement it must first be changed to 1's complement.
  - Change each bit to its complement (opposite).
  - To convert 1's complement to 2's complement, add 1 to the 1's complement.
- A number is negated when converted to the opposite sign.
  - A binary number can be negated by taking the 2's complement of it.

6-3 Addition in the 2's Complement System

- Perform normal binary addition of magnitudes.
  - The sign bits are added with the magnitude bits.
- If addition results in a carry of the sign bit, the carry bit is ignored.
  - If the result is *positive*, it is in pure binary form.
  - If the result is *negative*, it is in 2's complement form.

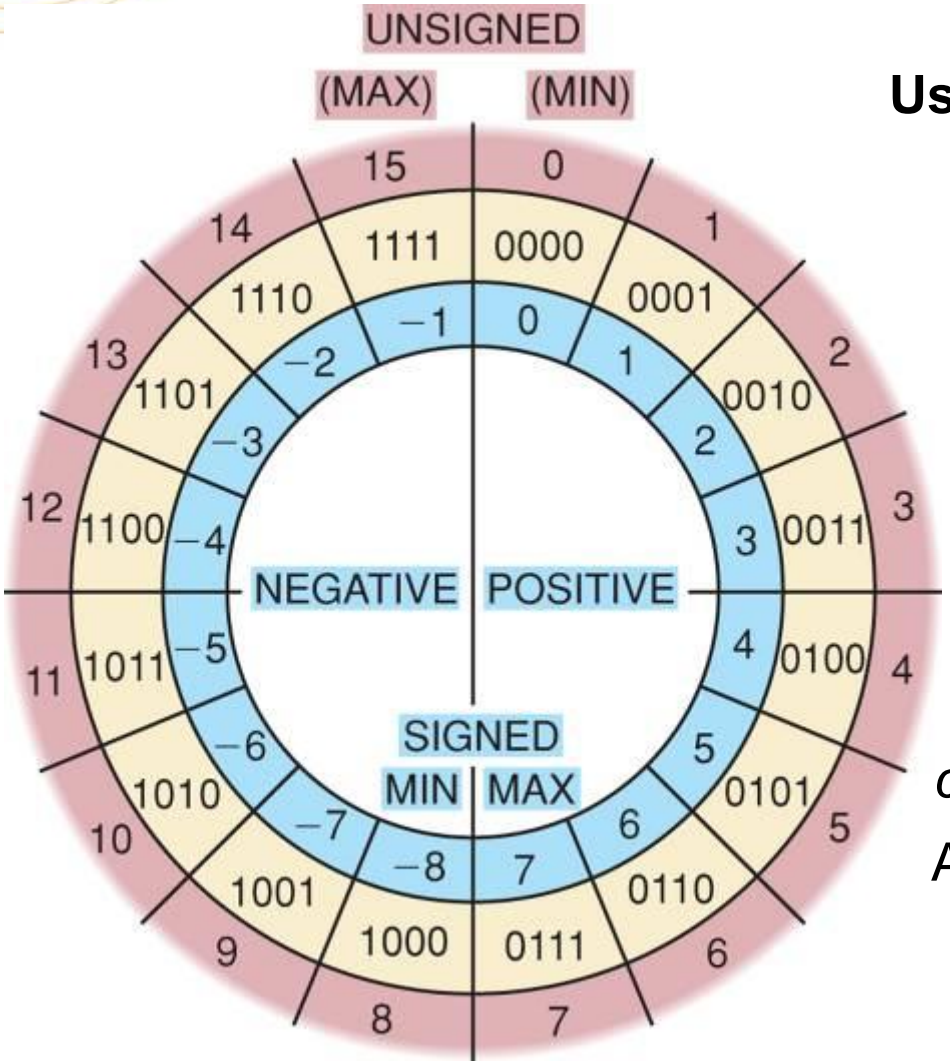
6-4 Subtraction in the 2's Complement System

- Subtraction using the 2's-complement system actually involves the operation of addition.
  - The number subtracted (subtrahend) is negated.
  - The result is added to the minuend.
  - The answer represents the difference.

6-4 Subtraction in the 2's Complement System

- **Overflow** can occur only when two positive or two negative numbers are being added—which always produces an incorrect result.
  - If the answer exceeds the number of magnitude bits, an overflow results.

6-4 Subtraction in the 2's Complement System



Using a number circle to add.

Start at the value of the augend.

Advance around the number circle clockwise by the number of spaces in the addend.

The most apparent way to subtract is to move *counterclockwise* around the circle

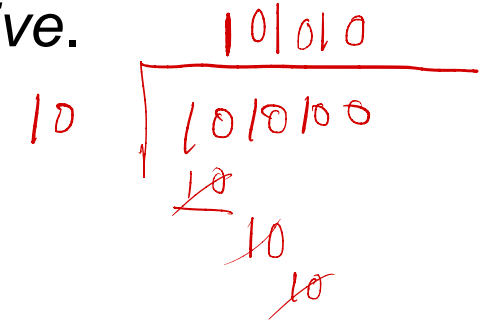
Any subtraction operation between four-bit numbers of opposite sign producing a result greater than 7 or less than -8 is an overflow.

6-5 Multiplication of Binary Numbers

- Similar to multiplication of decimal numbers.
  - Each bit in the multiplier is multiplied by the multiplicand.
- Results are shifted as we move from LSB to MSB in the multiplier.
  - All of the results are added to obtain the final product.

6-6 Binary Division

- Similar to decimal long division—simpler, as only 1 or 0 are possible.
- The subtraction part of the operation is done using 2's complement subtraction.
  - If the signs of the dividend & divisor are the *same*…
    - The answer will be *positive*.
  - If the signs of the dividend & divisor are *different*…
    - The answer will be *negative*.



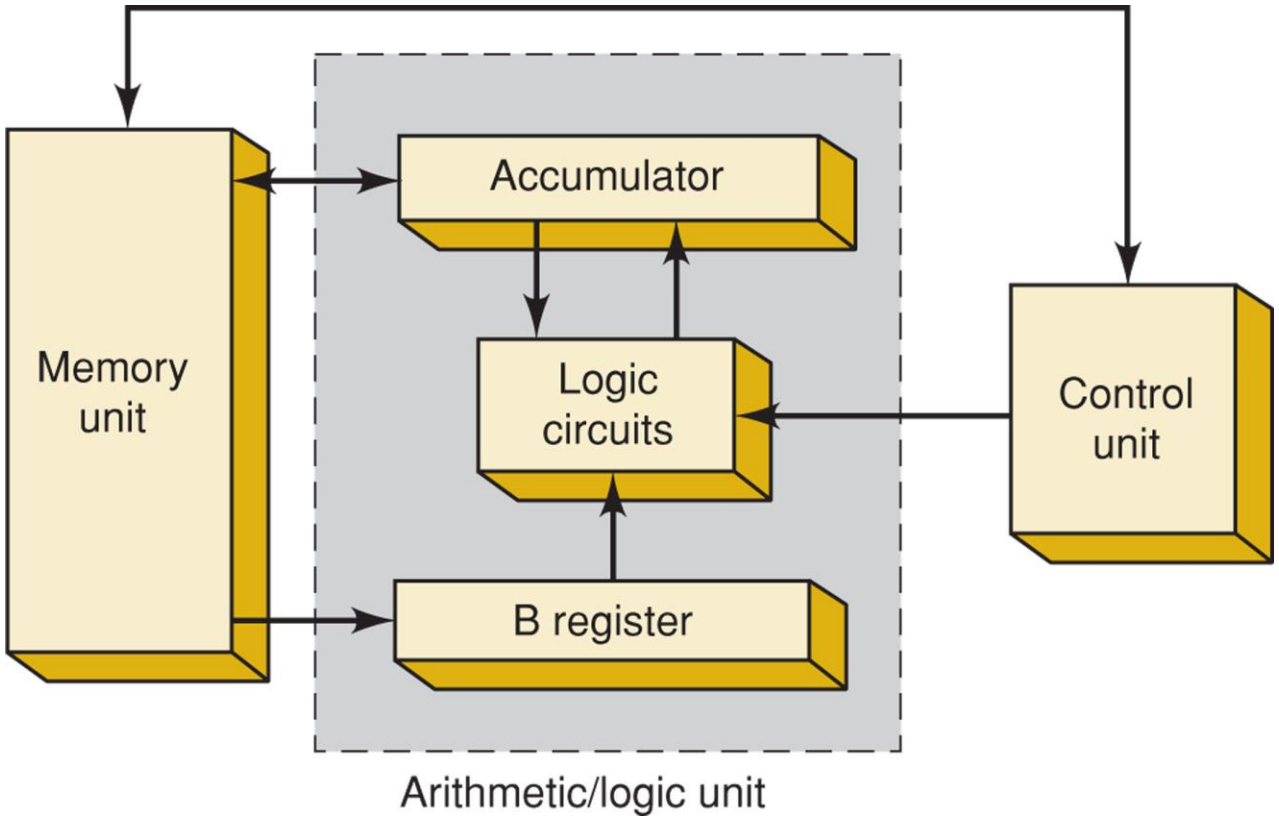
6-7 BCD Addition

- When the sum of each decimal digit is less than 9, the operation is the same as normal binary addition.
- When the sum of each decimal digit is greater than 9, a binary 6 is added.
  - This will always cause a carry.

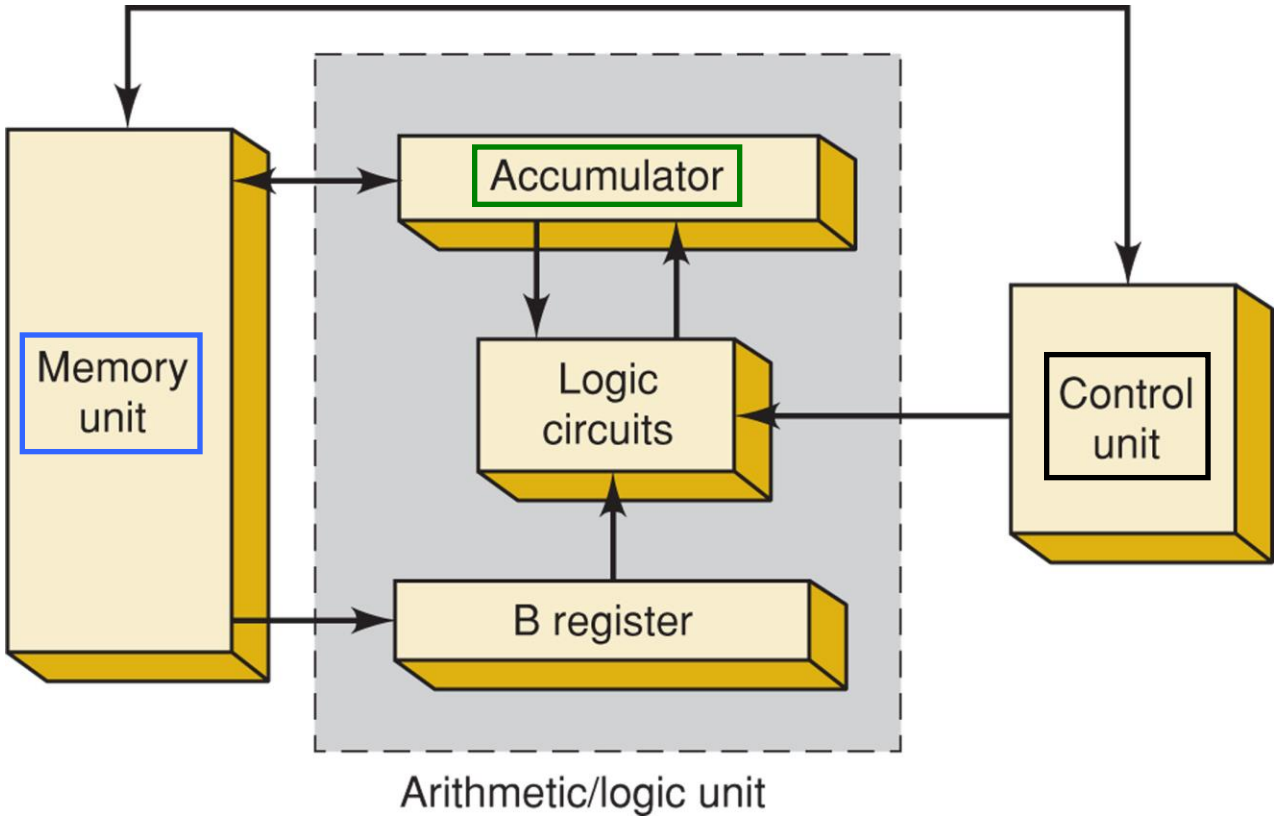
6-8 Hexadecimal Arithmetic

- **Hex addition**:
  - Add the hex digits in decimal.
  - If the sum is 15 or less—express directly in hex digits.
  - If the sum is greater than 15—subtract 16 and carry 1 to the next position.
- **Hex subtraction**—use the same method as for binary numbers.
  - When the MSD in a hex number is 8 or greater, the number is negative.
  - When the MSD is 7 or less, the number is positive.

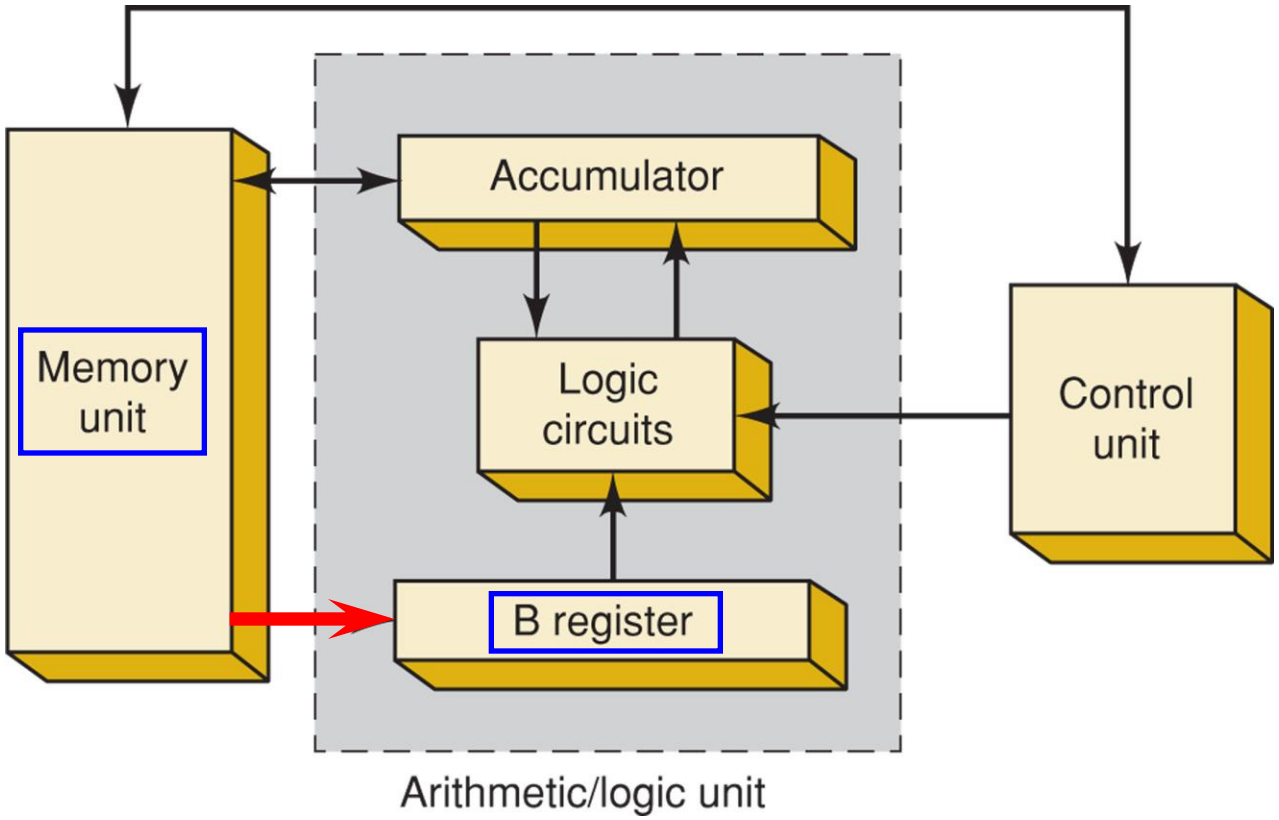
An arithmetic/logic unit (ALU) accepts data stored in memory, and executes arithmetic and logic operations as instructed by the control unit.



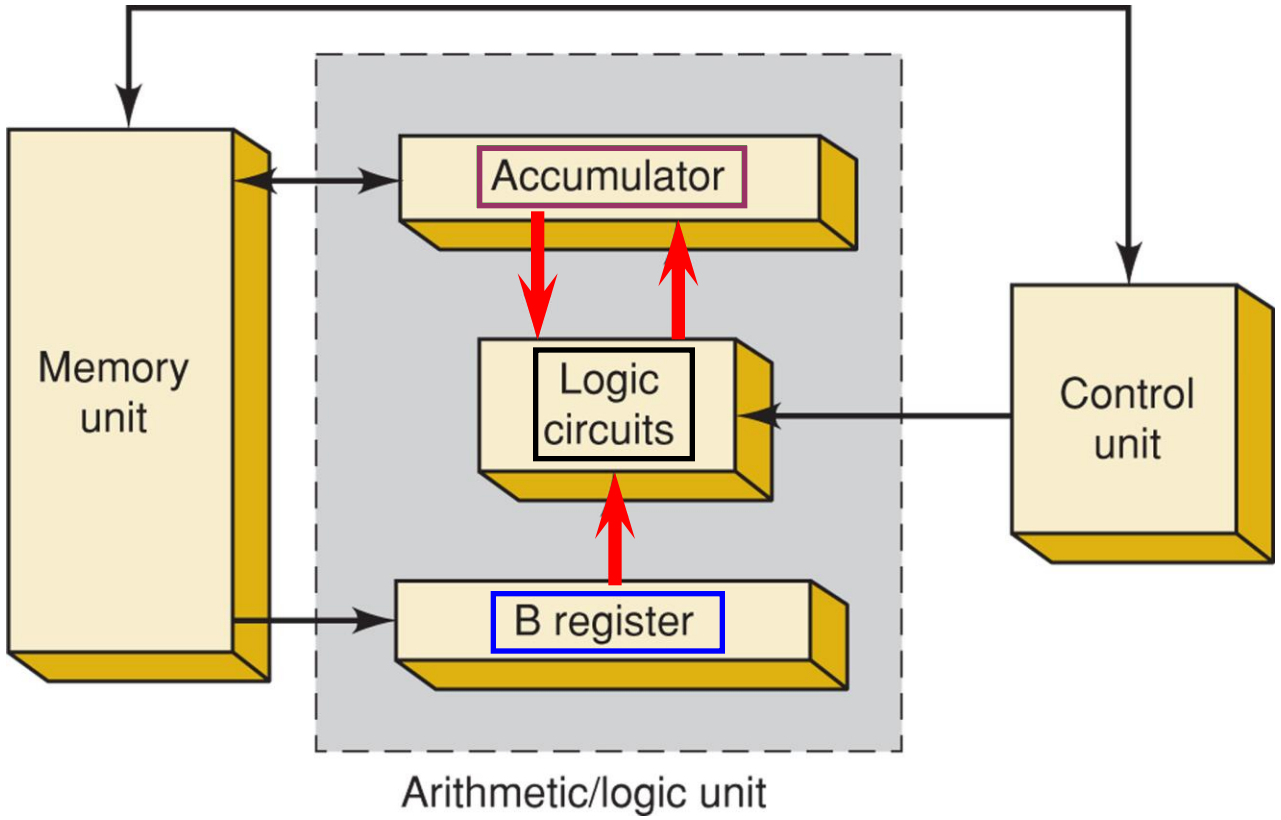
The control unit is instructed to add a specific number from a memory location to a number stored in the accumulator register.



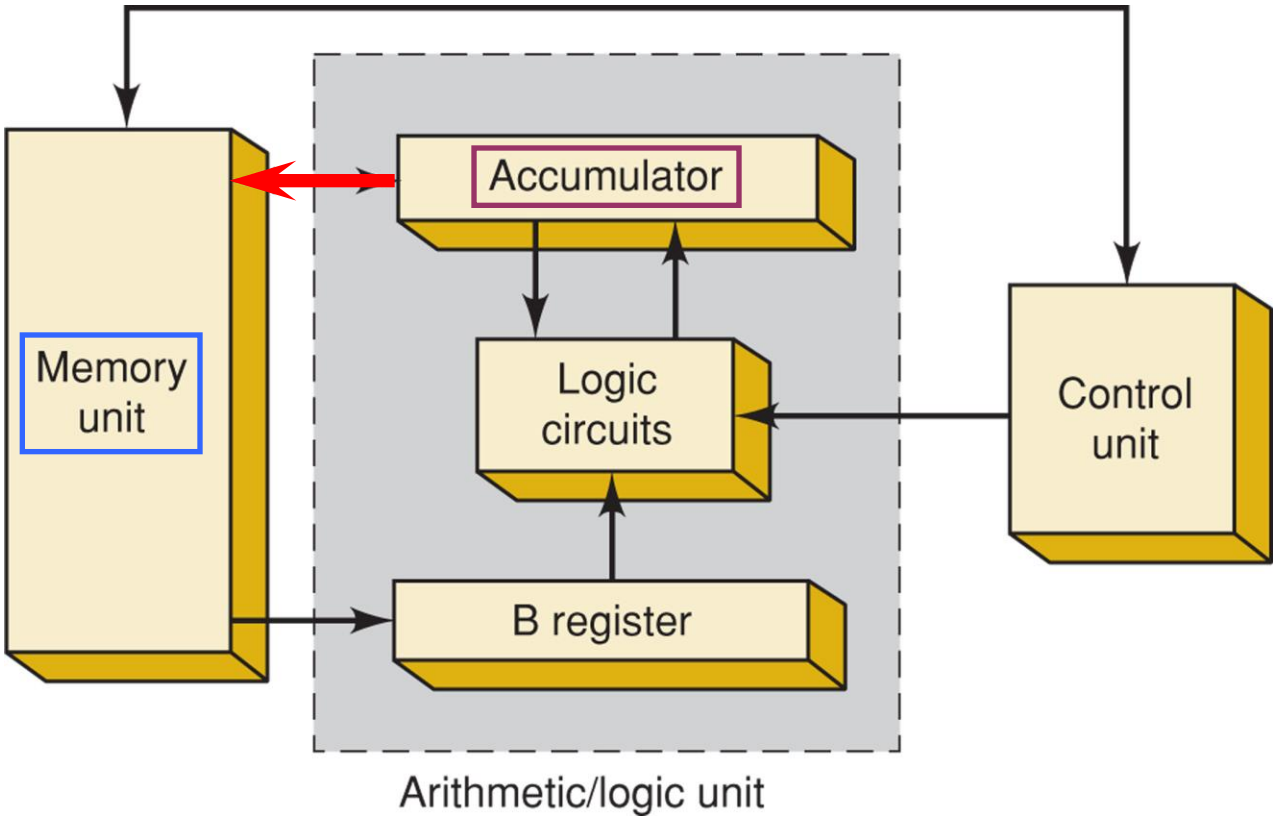
The number is transferred from memory to the B register.



The number in B register and accumulator register are added in the logic circuit, with sum sent to accumulator for storage.

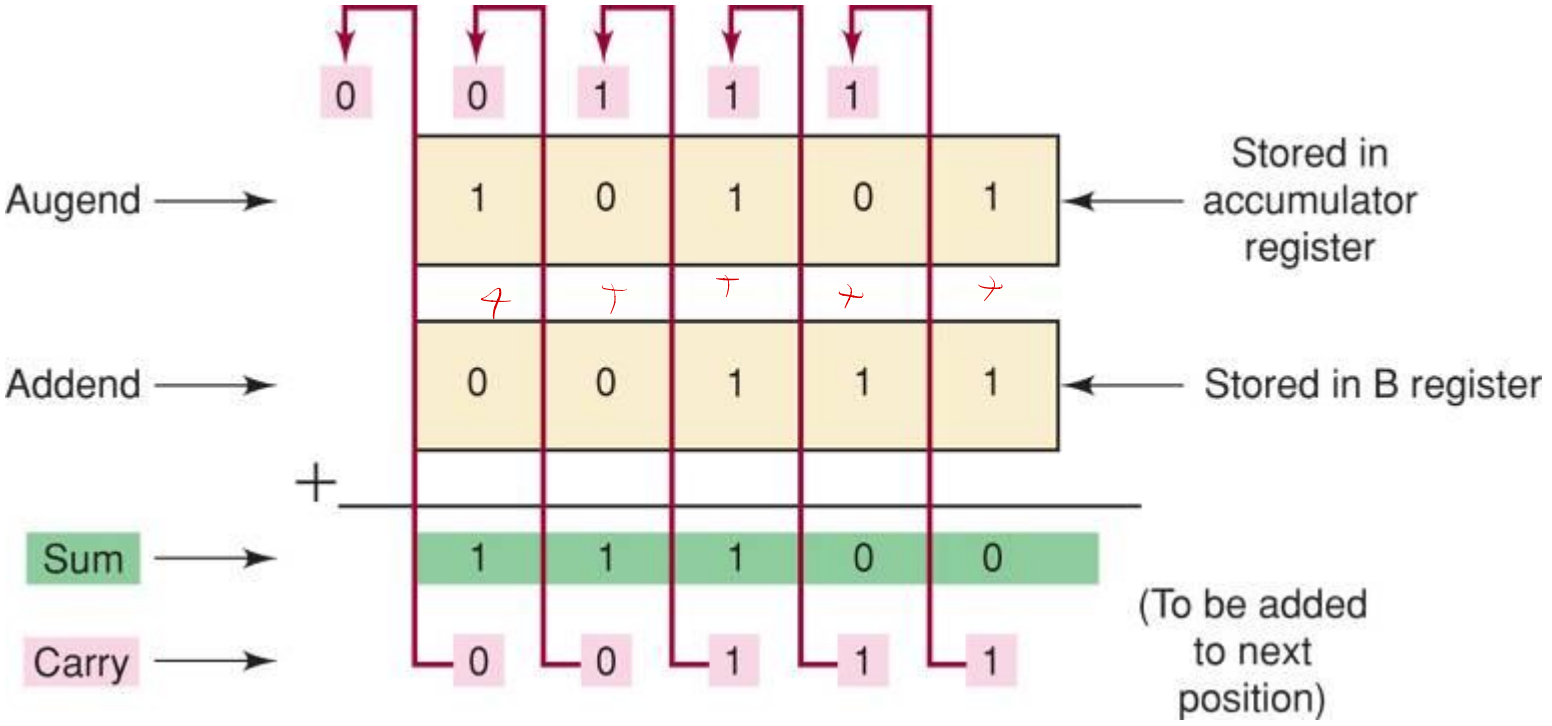


The new number remains in the accumulator for further operations—or can be transferred to memory for storage

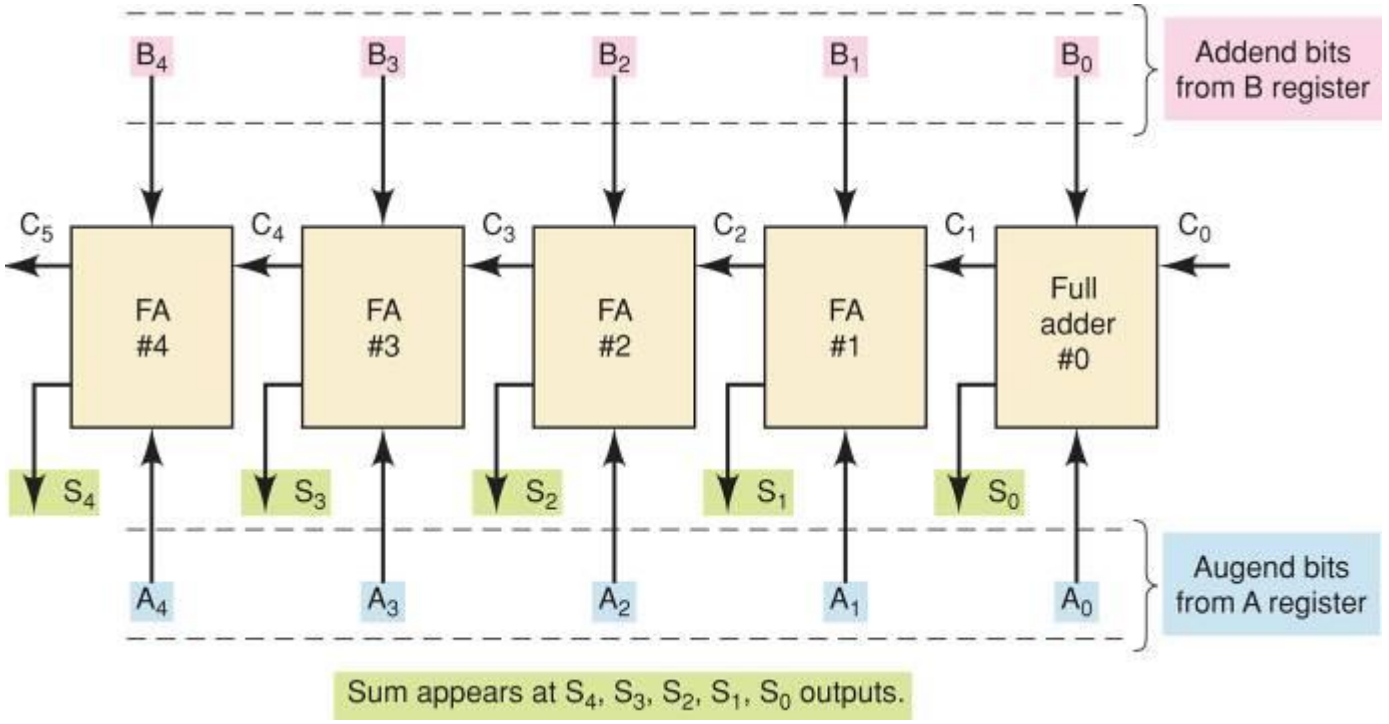


6-10 Parallel Binary Adder

- Computers and calculators perform the addition operation on two binary numbers at a time.
  - Each binary number can have several binary digits.



Block diagram of a five-bit parallel adder circuit using full adders.



6-11 Design of a Full Adder

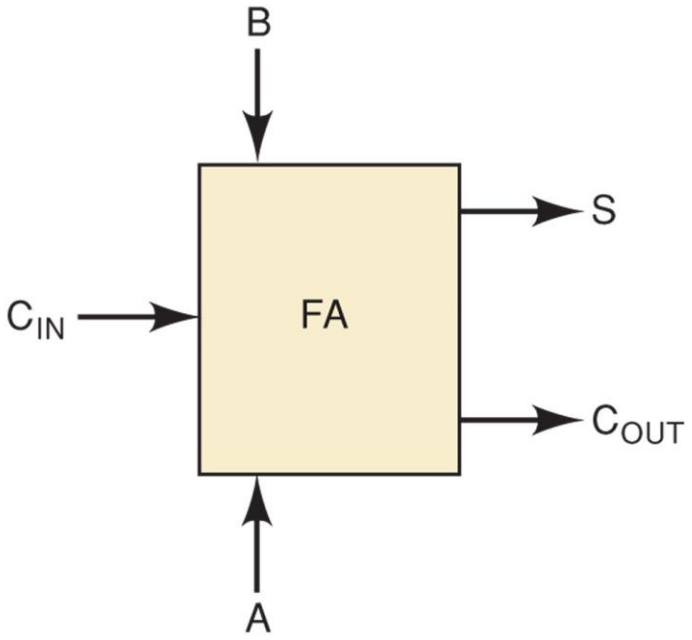
- Construct a truth table…
  - 3 inputs (2 numbers to be added and carry in).
  - 2 outputs (sum and carry out).

Augend bit input	Addend bit input	Carry bit input		Sum bit output	Carry bit output
A	B	C_{IN}		S	C_{OUT}
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

6-11 Design of a Full Adder

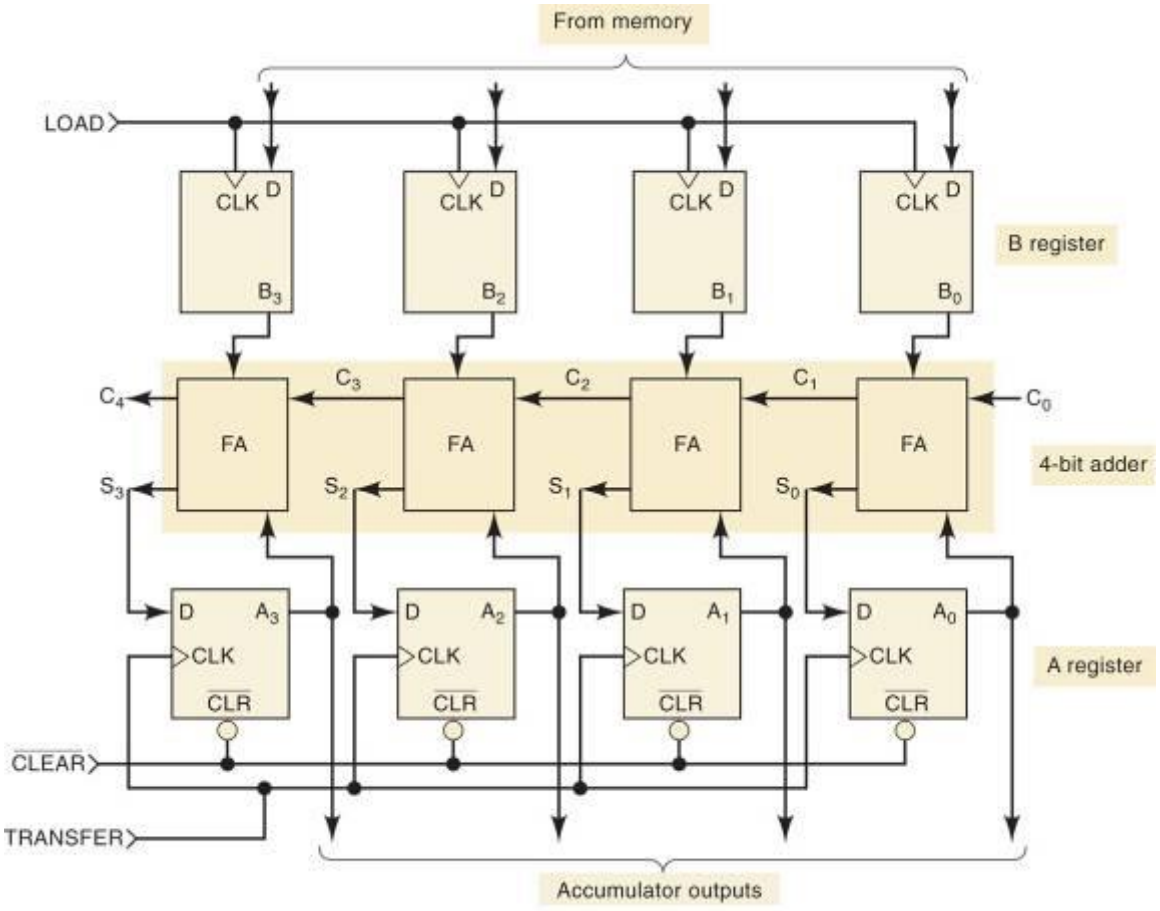
- Use algebraic methods or K-maps to simplify the resulting SOP form.
  - The result is the logic circuit shown here.

Augend bit input	Addend bit input	Carry bit input		Sum bit output	Carry bit output
A	B	C_{IN}		S	C_{OUT}
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



6-12 Complete Parallel Adder With Registers

Four-bit parallel adder circuit, including the storage registers.



6-12 Complete Parallel Adder With Registers

- Adding binary 1001 and 0101 using the circuit:
  - A CLR pulse is applied at t_1 .
  - The first binary number 1001 is transferred from memory to the B register at t_2 .
  - The sum of 1001 and 0000 is transferred to the A register at t_3 .
  - 0101 is transferred from memory to B register at t_4 .
  - Sum outputs are transferred to the A register at t_5.
  - The sum of the two numbers is now present in the accumulator.

6-12 Complete Parallel Adder With Registers

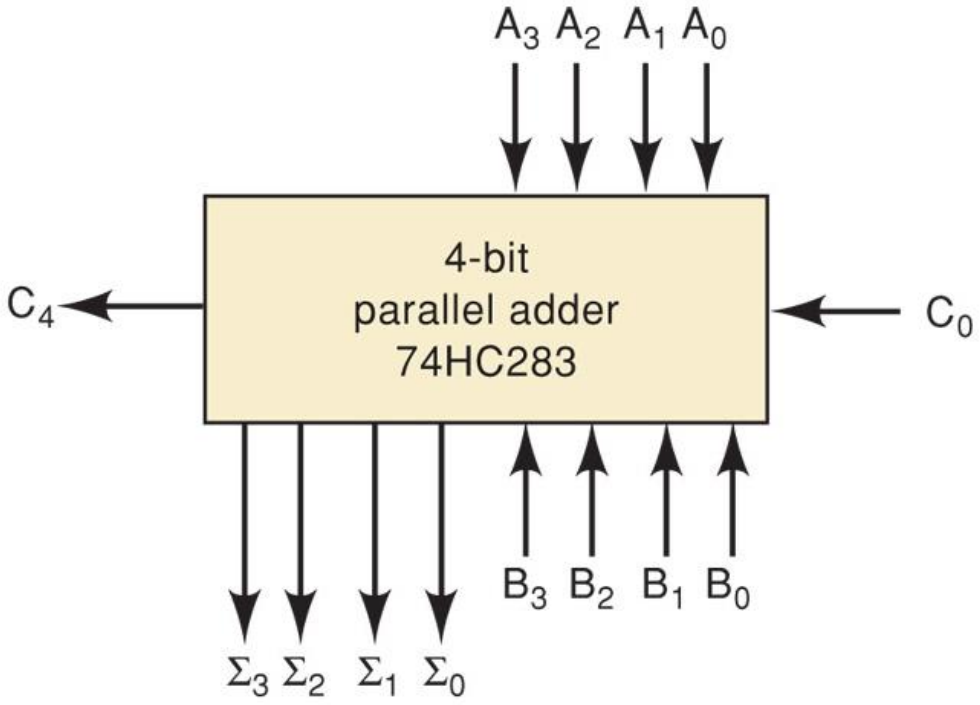
- Brackets indicate the contents of a register:
  - **[A]=1011** is the same as $\mathbf{A_3=1, A_2=0, A_1=1, A_0=1}$
    - The contents of register *A.*
- Transfer of data to or from a register is indicated with an arrow [B]→[A].
  - *"…the contents of register B have been transferred to register A."*

6-13 Carry Propagation

- Parallel adder speed is limited by carry propagation—also called carry ripple.
  - Results from having to wait for the carry bits to "ripple" through the device.
    - Additional bits will introduce more delay.
- The **look-ahead carry** scheme is commonly used in high speed devices to reduce the delay.

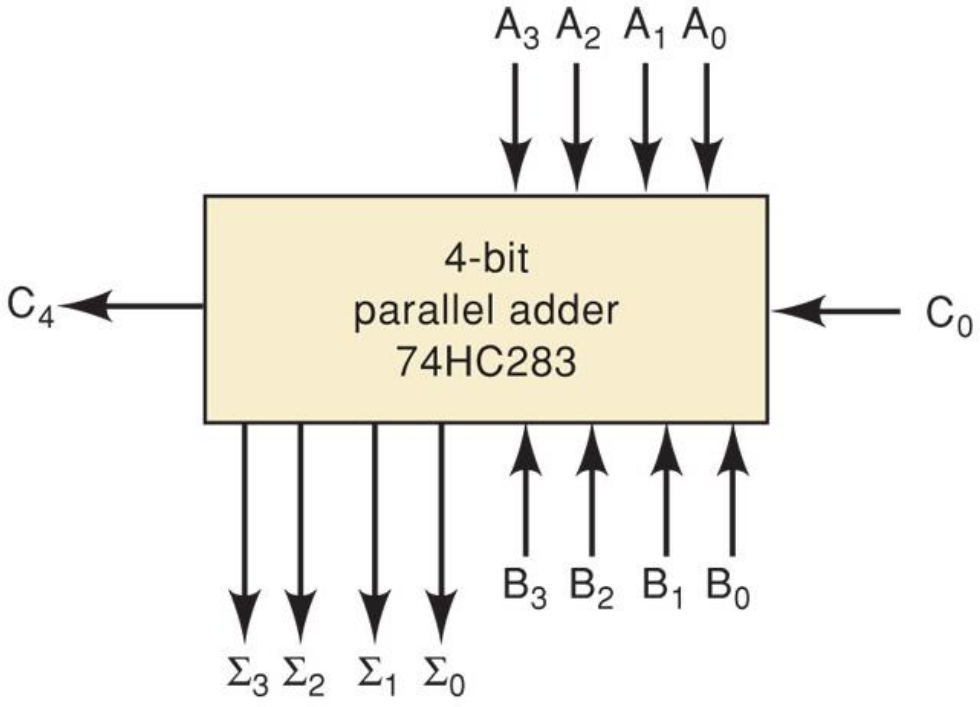
6-14 Integrated Circuit Parallel Adder

- The most common parallel adder is a 4 bit device.
  - 4 interconnected FAs, and look-ahead carry circuits.
    - 7483A, 74LS83A, 74LS283, and 74HC283 are all four-bit parallel-adder chips.



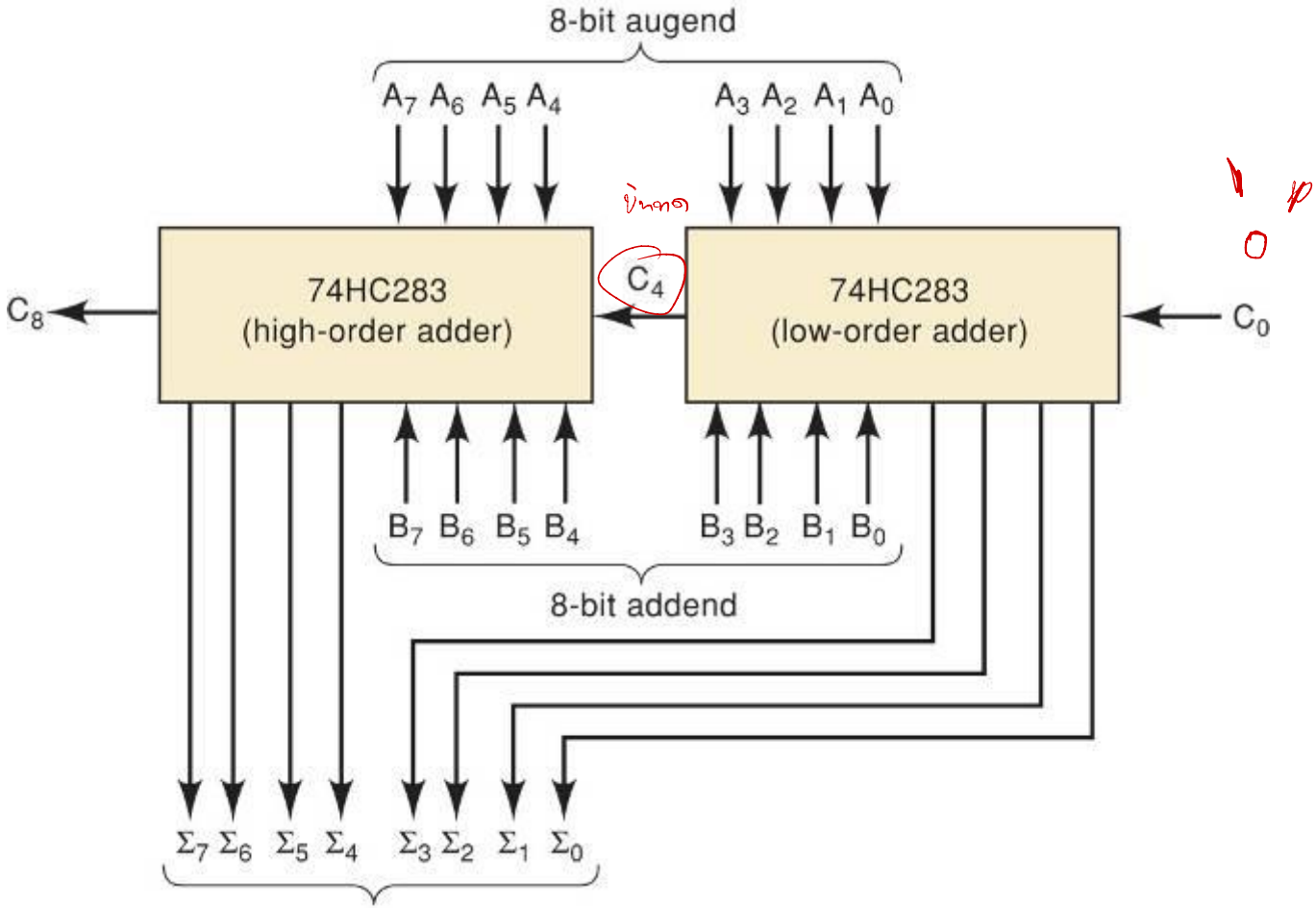
6-14 Integrated Circuit Parallel Adder

- The A and B lines each represent 4 bit numbers to be added.
  - C_0 is the carry-in, C_4 is the carry-out, Σ is the sum.



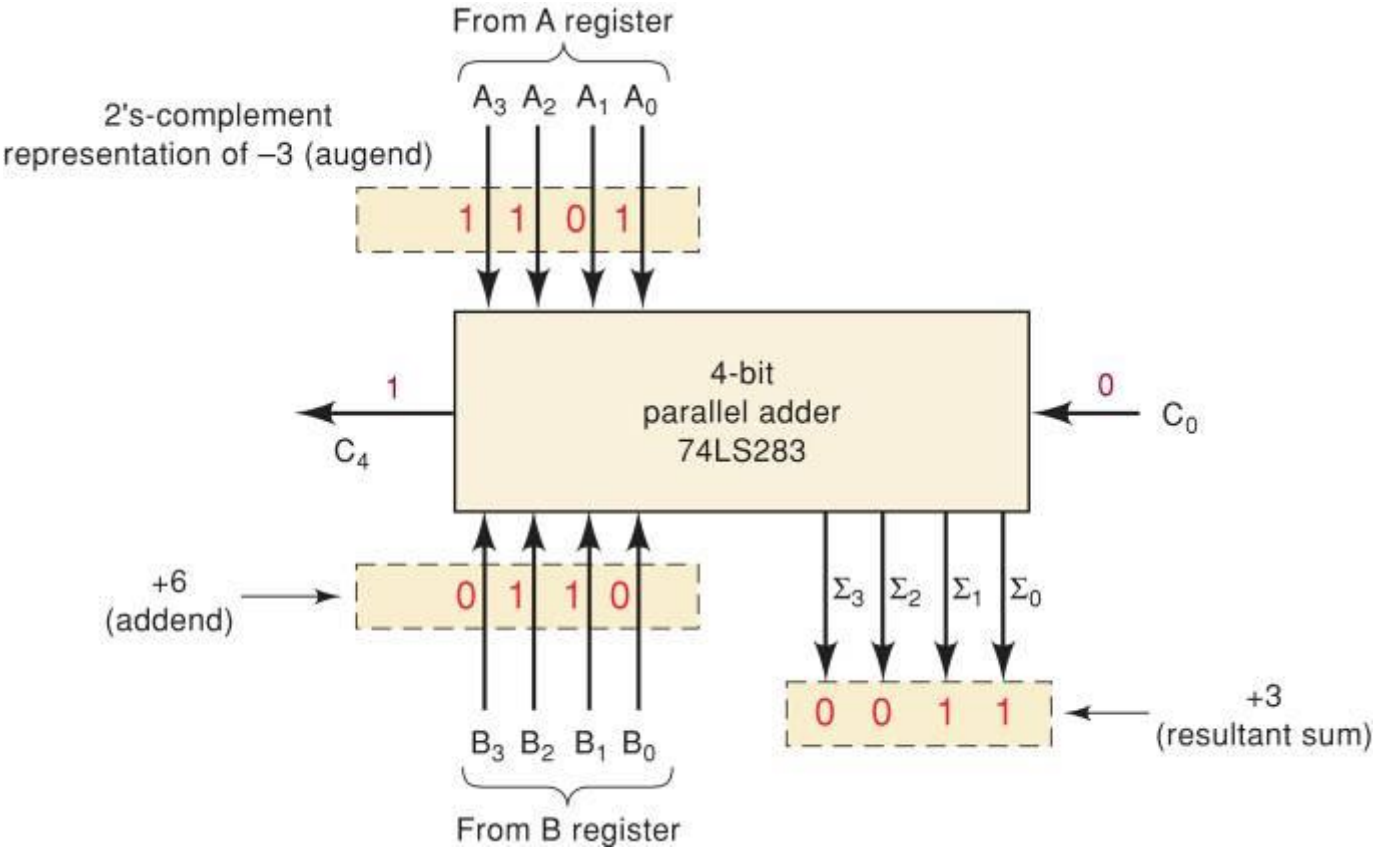
6-14 Integrated Circuit Parallel Adder

Parallel adders may be cascaded together to add larger numbers, in this case two 8 bit numbers.



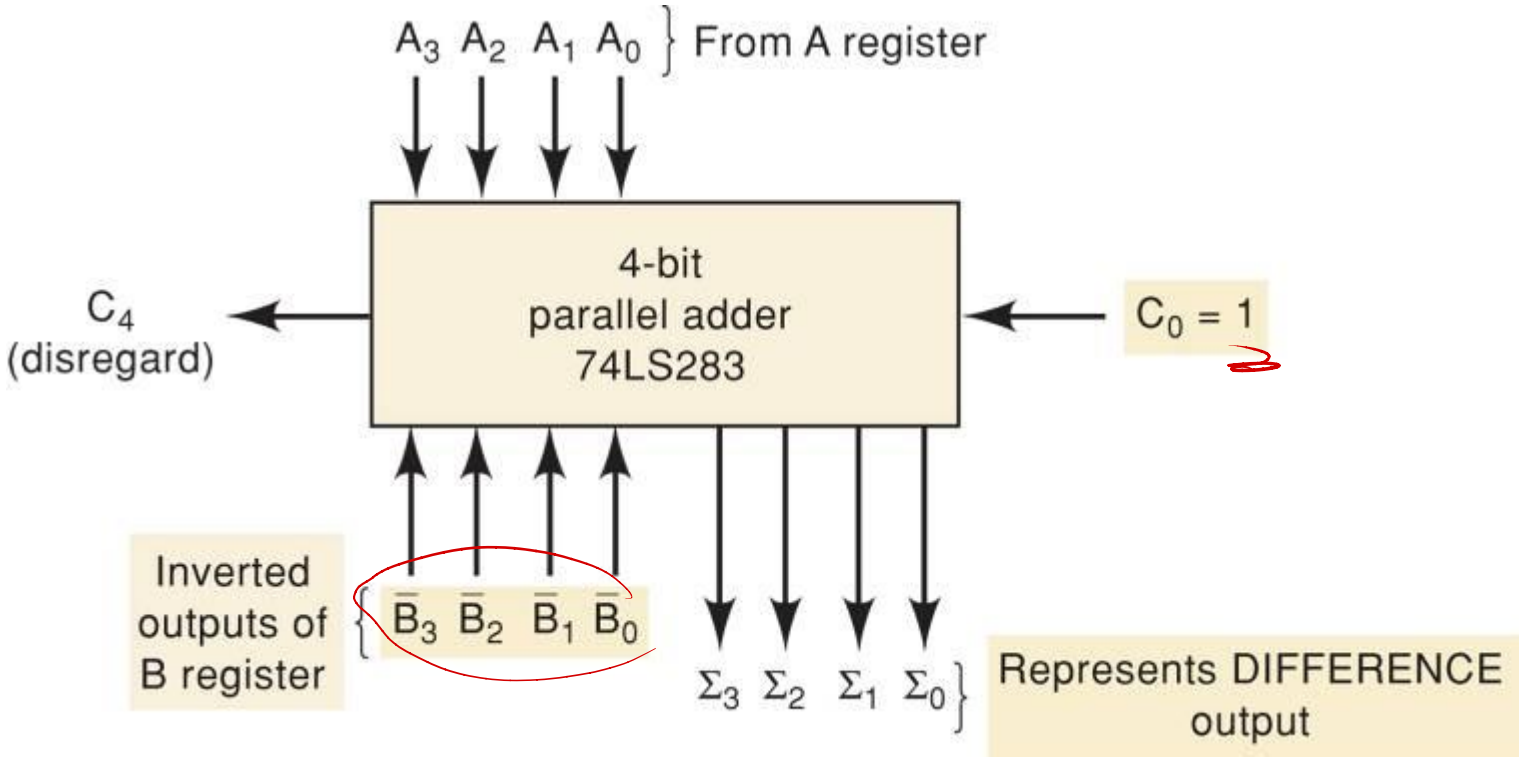
6-15 2's Complement System

- Addition of negative & positive numbers with adders is done by placing the negative number into 2's complement form, then normal addition.



6-15 2's Complement System

- An adder can be used to perform subtraction by designing a way to take the 2's complement for subtraction—as shown.

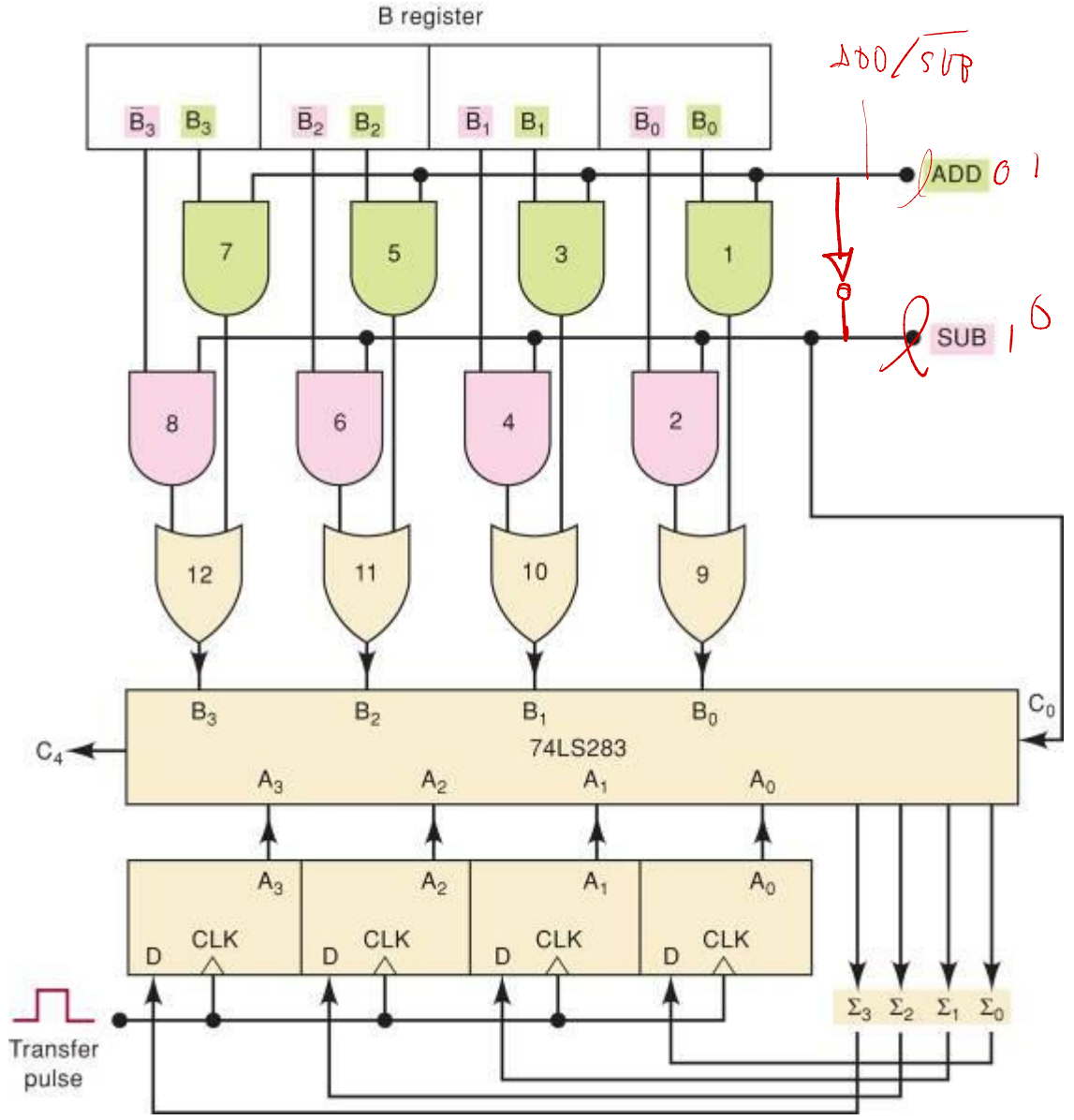


6-15 2's Complement System

- The **adder/subtractor** circuit is controlled by the two control signals ADD and SUB.
  - When ADD is HIGH, the circuit performs addition of numbers stored in the *A* and *B* registers.
  - When SUB is HIGH, the circuit subtracts the B-register number from the A-register number.

6-15 2's Complement System

Parallel adder/subtractor using the 2's-complement system.

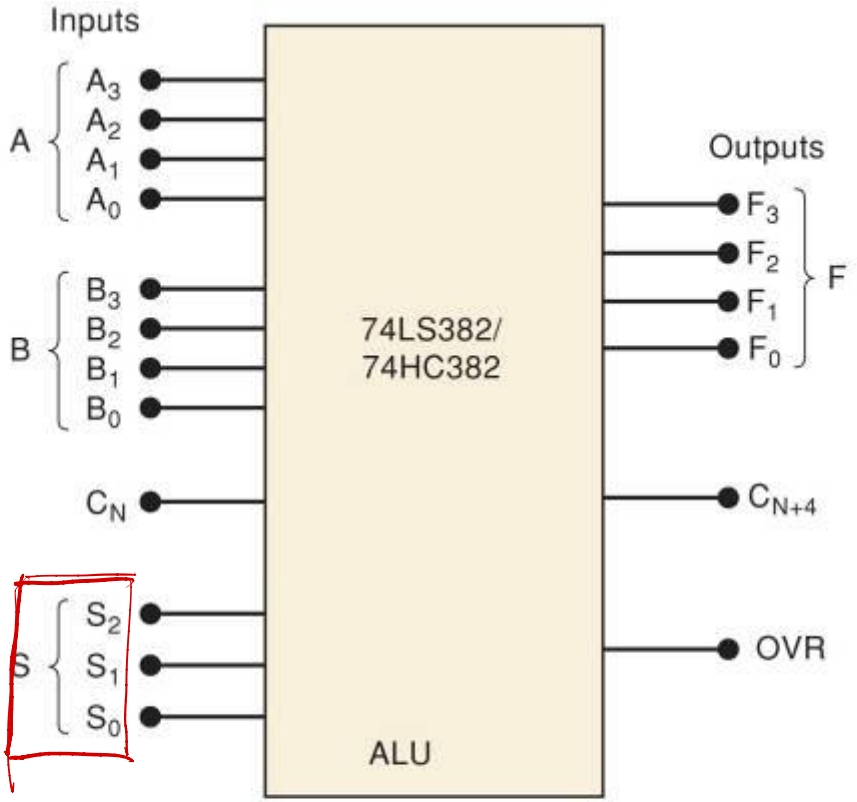


6-16 ALU Integrated Circuits

- ALUs can perform different arithmetic and logic functions as determined by a binary code on the function select inputs.

6-16 ALU Integrated Circuits

The 74LS382 (TTL) and HC382 (CMOS) is a typical device with 8 functions.



Function Table

S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	Needs $C_N = 1$
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	A ⊕ B	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

Notes: S inputs select operation.
OVR = 1 for signed-number overflow.

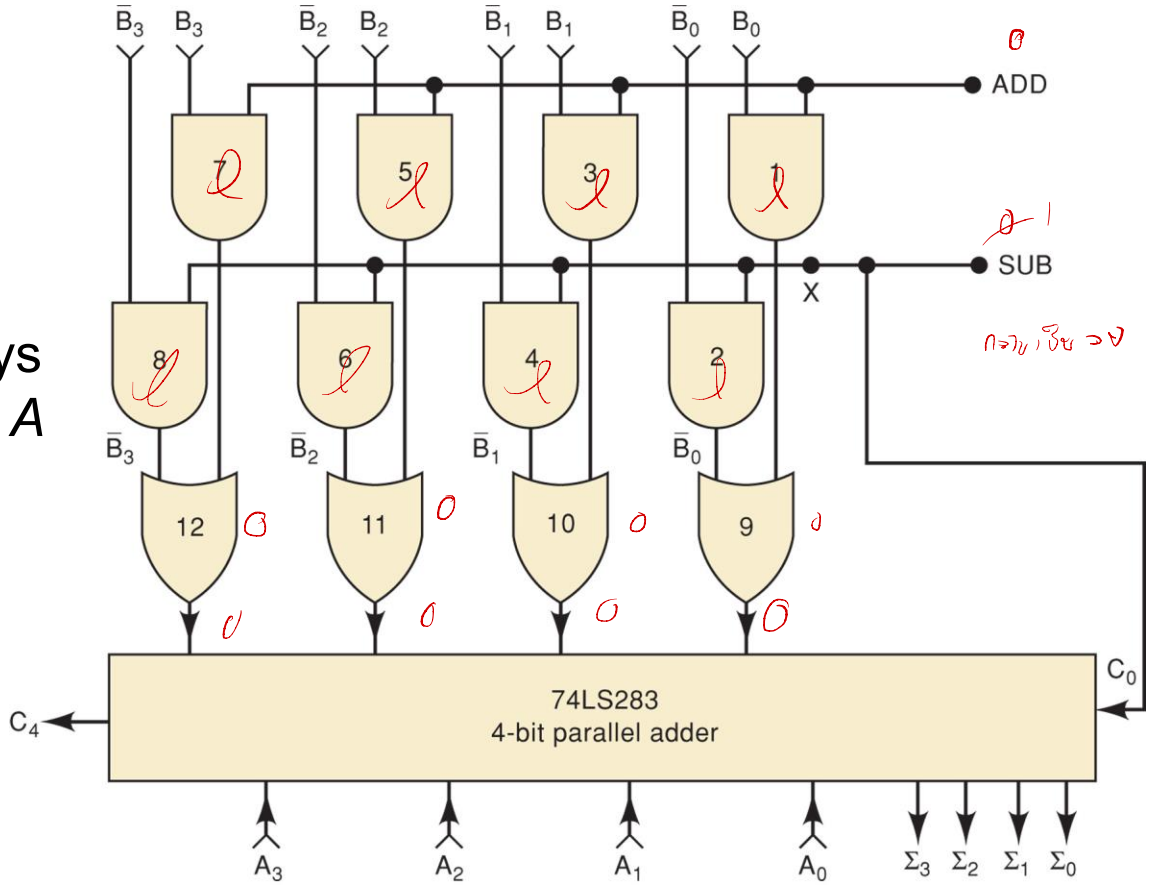
A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

6-17 Troubleshooting Case Study

- Determine the most likely fault…

Mode 1:
ADD = 0, SUB = 0.

The sum outputs are always equal to the number in the *A* register *plus one.*

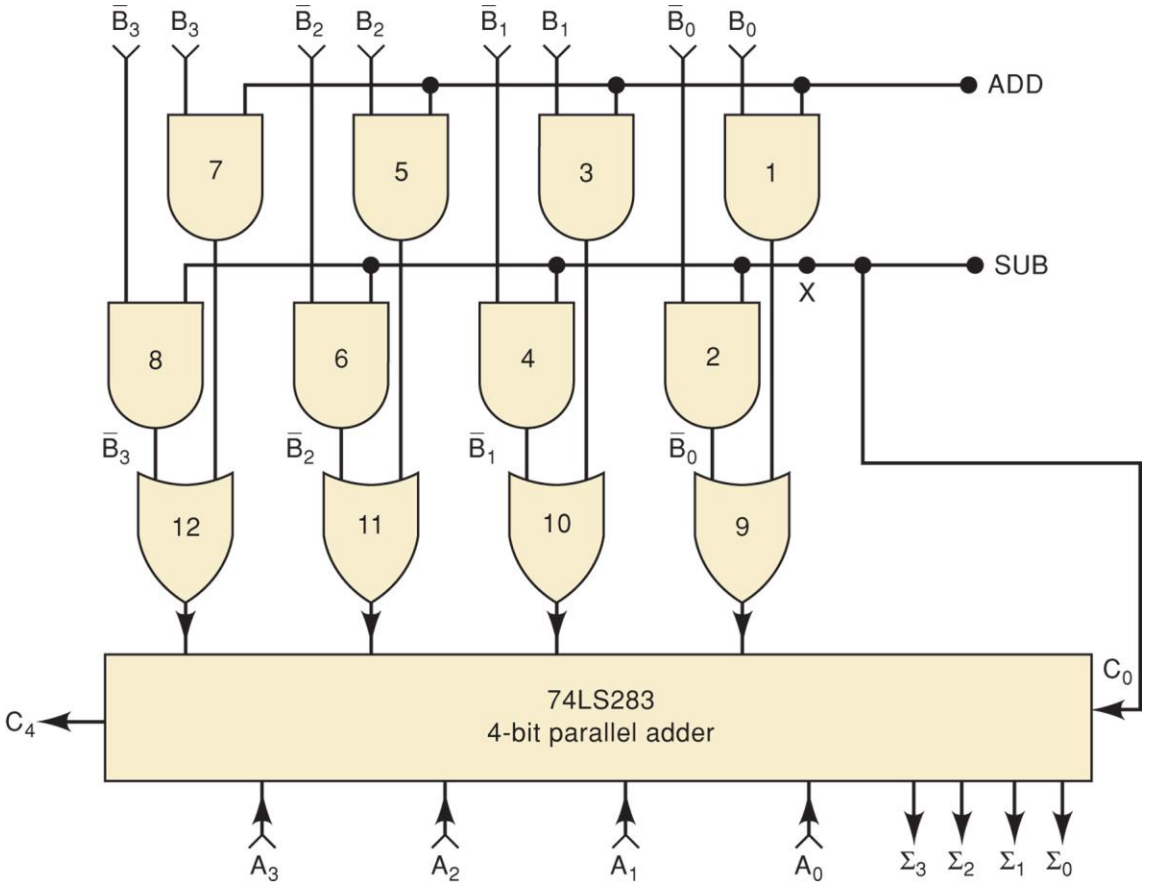


6-17 Troubleshooting Case Study

- Determine the most likely fault…

Mode 2:
ADD = 1, SUB = 0.

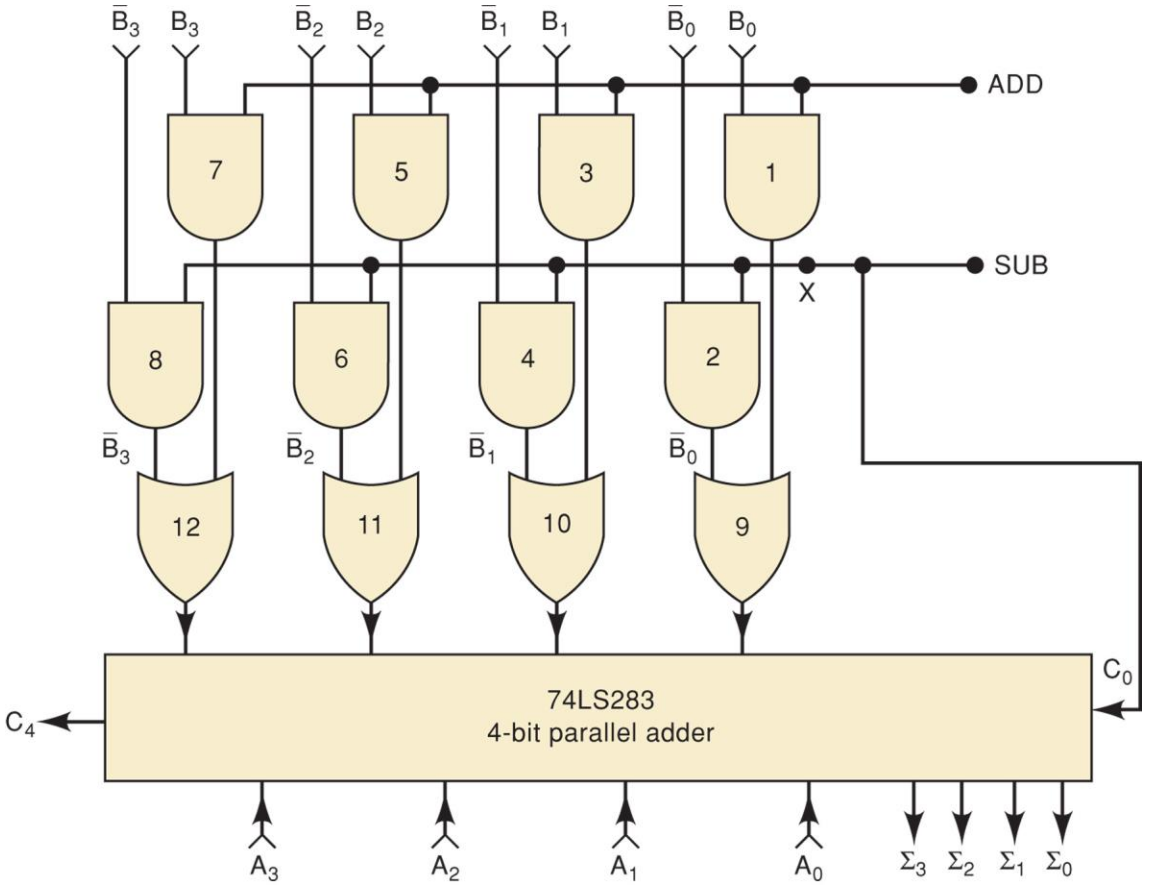
The sum is always 1 more than it should be.



6-17 Troubleshooting Case Study

- Determine the most likely fault…

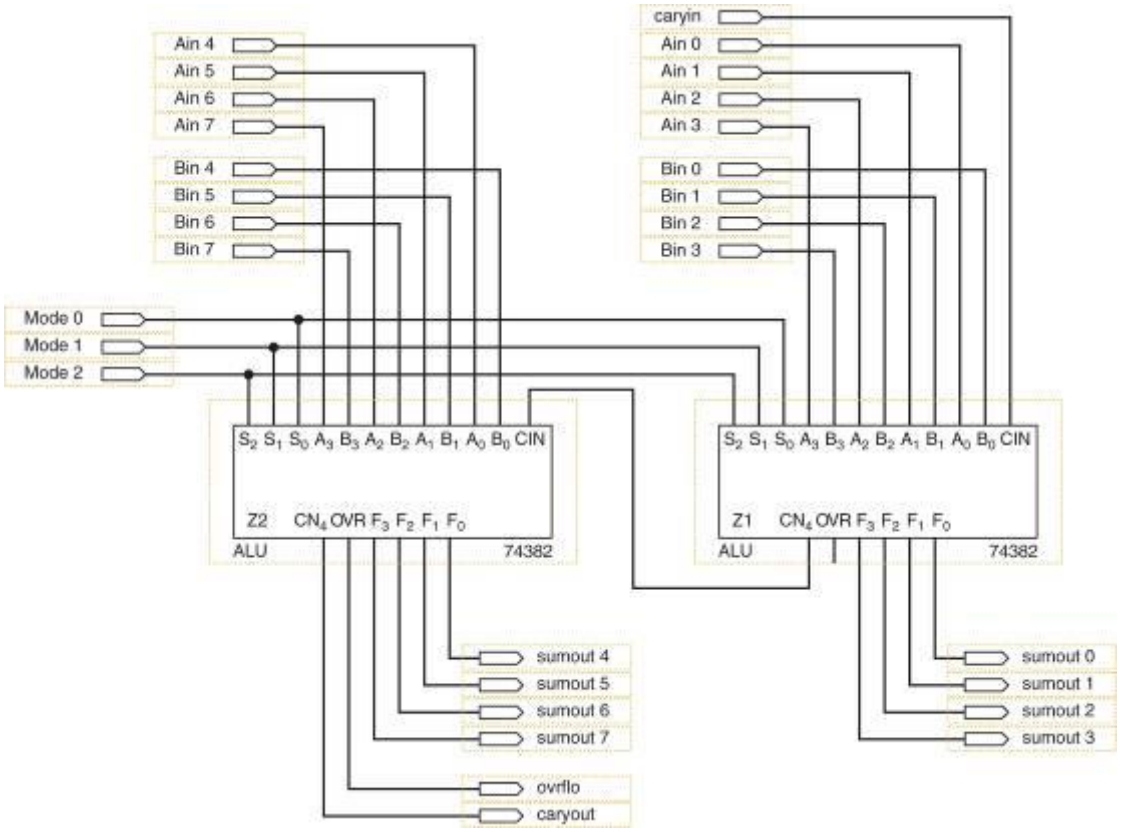
Mode 3:
ADD = 0, SUB = 1.
The Σ outputs are always equal to [A] - [B].



6-18 Using TTL Library Functions with ALTERA

- Altera offers pre-defined logic circuits in macrofunctions.
  - ALUs may be defined graphically.

Graphic description may seem intuitive, but it is often easier to define a device using text and macrofunctions.



6-19 Logical Operations on Bit Arrays

- Two main areas of HDL techniques require understanding:
  - Specifying groups of bits in arrays
  - Using logical operations to combine arrays using Boolean expressions
- HDLs use arrays of bits—a method similar to register notation—to describe signals.
- The four-bit signal named *d* is defined as…
  - AHDL: `VARIABLE d[3..0]  :NODE`
  - VHDL: `SIGNAL d:BIT_VECTOR (3 DOWNTO 1)`

6-20 HDL Adders

- 8-bit parallel adder circuit using HDL languages.
  - Will add 8-bit values *A[8..1]* and *B[8..1]* to produce the 9-bit sum.
- AHDL

```
1   SUBDESIGN fig6_23
2   (
3        a[8..1]        :INPUT;          -- 8-bit augend
4        b[8..1]        :INPUT;          -- 8-bit addend
5        s[9..1]        :OUTPUT;         -- 9-bit sum
6   )
7   VARIABLE
8        aa[9..1]        :NODE;          -- expanded augend
9        bb[9..1]        :NODE;          -- expanded addend
10  BEGIN
11       aa[9..1] = (GND,a[8..1]);       -- concatenate leading zero
12       bb[9..1] = (GND,b[8..1]);       -- to both operands
13       s[9..1] = aa[9..1] + bb[9..1];     -- add expanded operands
14  END;
```

6-20 HDL Adders

- 8-bit parallel adder circuit using HDL languages.
  - Will add 8-bit values *A[8..1]* and *B[8..1]* to produce the 9-bit sum.
- VHDL

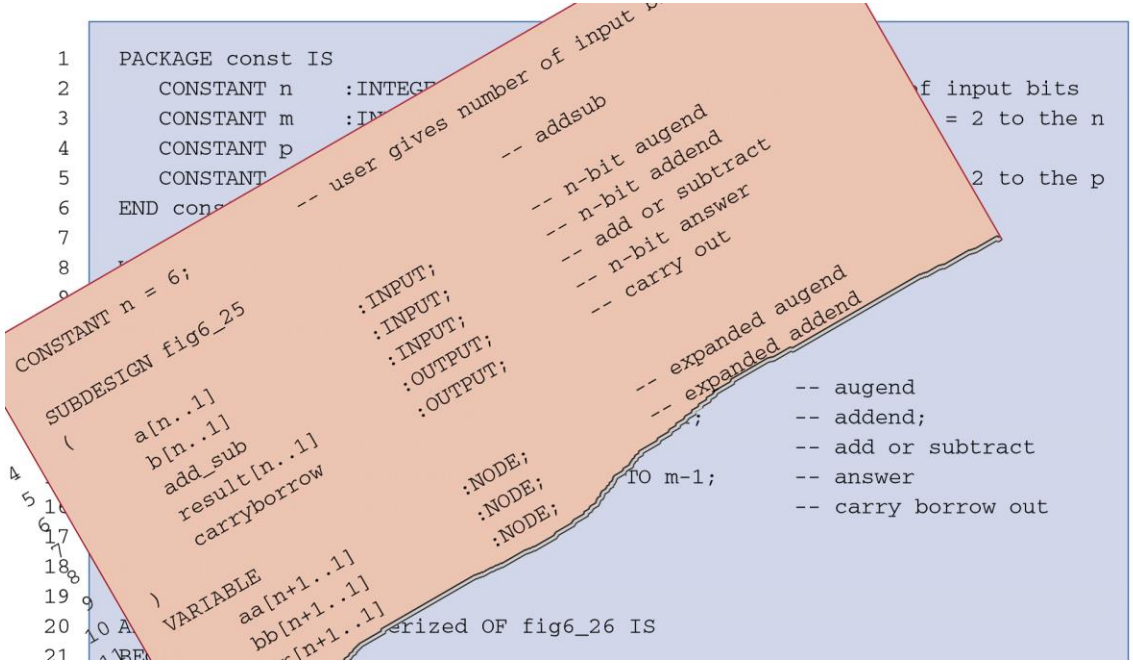
```
1   ENTITY fig6_24 IS
2   PORT (
3        a     :IN INTEGER RANGE 0 TO 255;     -- 8-bit augend
4        b     :IN INTEGER RANGE 0 TO 255;     -- 8-bit addend
5        s     :OUT INTEGER RANGE 0 TO 511     -- 9-bit sum
6   );
7   END fig6_24;
8
9   ARCHITECTURE parallel OF fig6_24 IS
10
11  BEGIN
12       s <= a + b;                           -- add operands
13  END parallel;
```

6-21 Parameterizing the Bit Capacity of a Circuit

- **Constants** are fixed numbers represented by a name (symbol).
  - Define a symbol (name) at the top of the source code.
    - Assign the value for the total number of bits.
  - Use this symbol (name) throughout the code.
- To modify or expand the circuit, only *one* line of code needs to be changed.

6-21 Parameterizing the Bit Capacity of a Circuit

- Add a constant feature to the HDL code for an adder/subtractor circuit.
  - A single input bit named *add_sub* will control the adder/subtractor's function.



```
PACKAGE const IS
   CONSTANT n    :INTEGE                    f input bits
   CONSTANT m    :IN                        = 2 to the n
   CONSTANT p                               2 to the p
END cons
-- user gives number of input b
CONSTANT n = 6;
SUBDESIGN fig6_25
(   a[n..1]                  :INPUT;     -- n-bit augend
    b[n..1]                  :INPUT;     -- n-bit addend
    add_sub                  :INPUT;     -- add or subtract
    result[n..1]             :OUTPUT;    -- n-bit answer
    carryborrow              :OUTPUT;    -- carry out
)
VARIABLE
    aa[n+1..1]               :NODE;      -- expanded augend
    bb[n+1..1]               :NODE;      -- expanded addend
                                         ;
                             TO m-1;     -- augend
                                         -- addend;
                                         -- add or subtract
                                         -- answer
                                         -- carry borrow out
erized OF fig6_26 IS
```

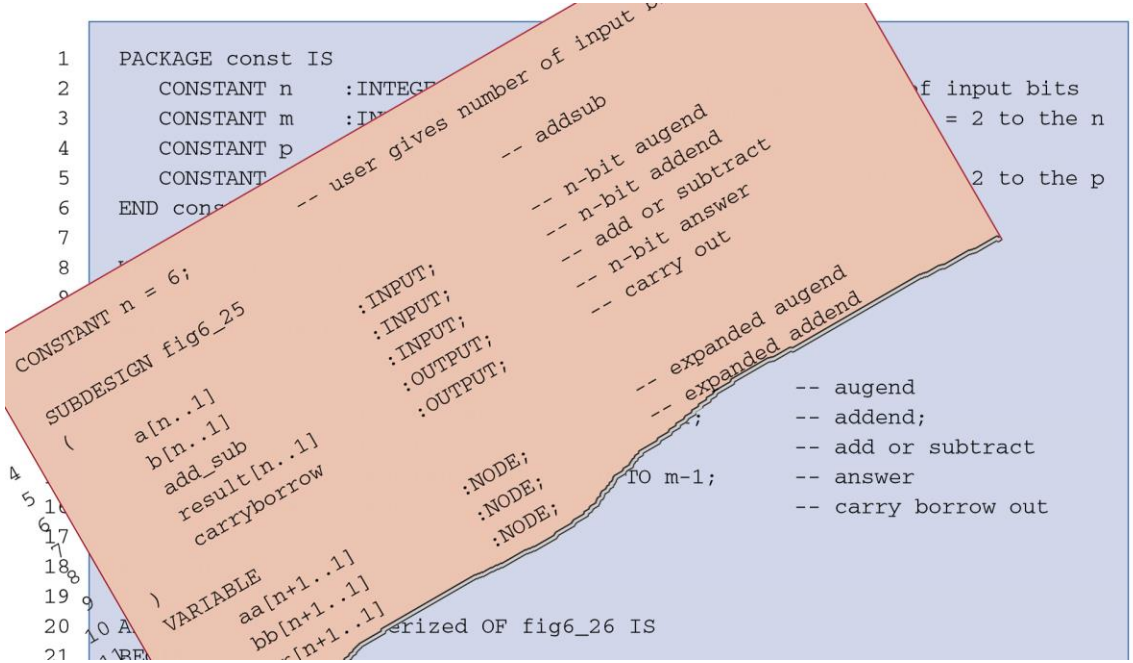
Add the two operands when *add_sub* = 0.

Subtract *b* from *a* when *add_sub* = 1.

See complete HDL code on pgs. 360 - 361

6-21 Parameterizing the Bit Capacity of a Circuit

- Add a constant feature to the HDL code for an adder/subtractor circuit.
  - AHDL—Keyword CONSTANT is followed by the symbolic name and value assigned.

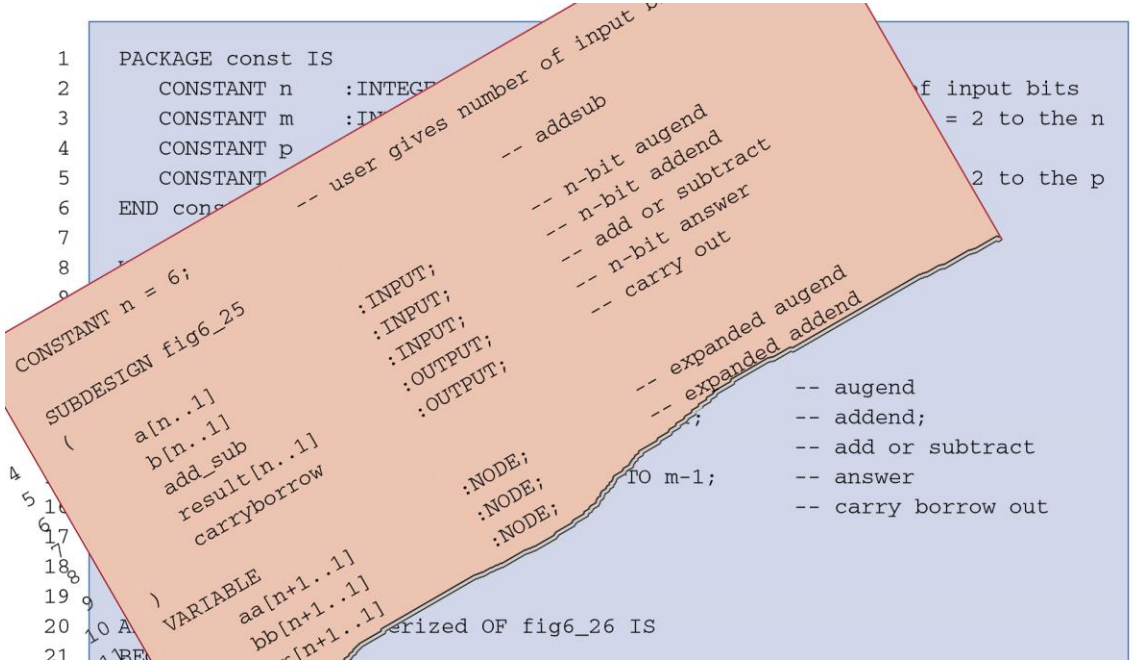


Check for overflow for a signed number operation.

See complete HDL code on pgs. 360 - 361

6-21 Parameterizing the Bit Capacity of a Circuit

- Add a constant feature to the HDL code for an adder/subtractor circuit.
  - VHDL—Keyword CONSTANT is followed by the symbolic name, type, and the value to be assigned.



The VHDL generate statement can be used to concisely replicate several components that are cascaded together.

See complete HDL code on pgs. 360 - 361

6-21 Parameterizing the Bit Capacity of a Circuit

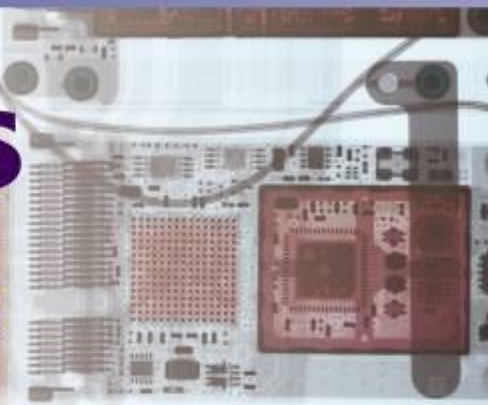
- Altera offers a library of parameterized modules (LPMs) which offer generic solutions for the various logic circuits used in digital systems.

END

ELEVENTH EDITION

Digital Systems

Principles and Applications



Ronald J. Tocci
Monroe Community College

Neal S. Widmer
Purdue University

Gregory L. Moss
Purdue University

PEARSON