

Arithmetic Circuits

CS207 Chapter 8

James YU

yujq3@sustech.edu.cn

Department of Computer Science and Engineering
Southern University of Science and Technology

Jul. 13, 2021



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arithmetic circuits

- One important aspect of digital design not dealt with in earlier lectures is the design and implementation of arithmetic circuits.
 - Various information-processing jobs are carried out by digital computers.
 - Arithmetic operations are among the basic functions of a digital computer.

Addition



- Addition of two binary digits is the most basic arithmetic operation.
 - $0 + 0 = 0$,
 - $0 + 1 = 1$,
 - $1 + 0 = 1$,
 - $1 + 1 = 10$.
 - The higher significant bit of this result is called the *carry*.
- A combinational circuit that performs the addition of two bits as described above is called a *half-adder*.
- The addition operation involves three bits — the *augend bit*, *addend bit*, and the *carry bit* and produces a sum result as well as carry.
- The combinational circuit performing this type of addition operation is called a *full-adder*.

Half-adder

- As described above, a half-adder has two inputs and two outputs.
- Let the input variables augend and addend be designated as A and B , and output functions be designated as S for sum and C for carry.

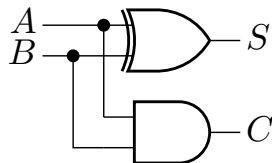
Input variables		Output variables	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- It can be seen that the outputs S and C functions are similar to Exclusive-OR and AND functions, respectively.



Half-adder

- $S = A \oplus B$,
- $C = AB$.



Full-adder



- A combinational circuit of full-adder performs the operation of addition of three bits — the augend, addend, and previous carry X , and produces the outputs sum and carry.

Input variables			Output variables	
X	A	B	S	C
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

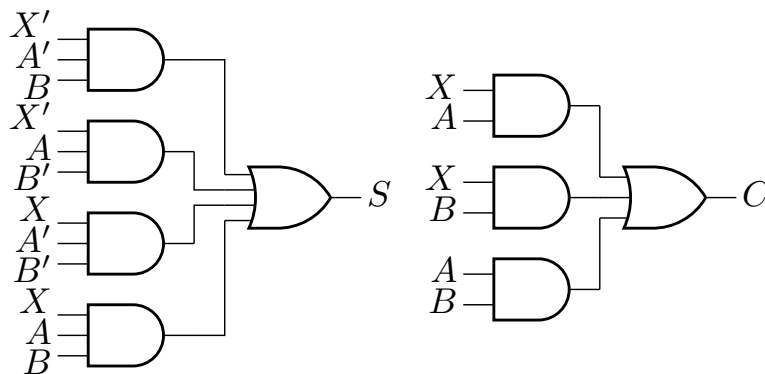
Full-adder



$X \backslash AB$	00	01	11	10
0		1		1
1	1		1	

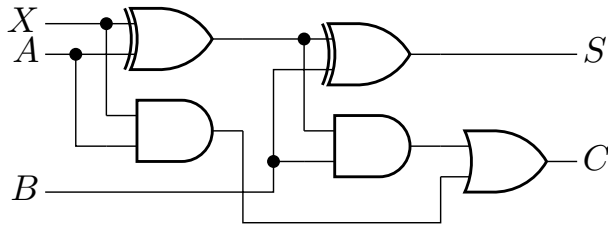
$X \backslash AB$	00	01	11	10
0			1	
1		1	1	1

- $S = X'A'B + X'AB' + XA'B' + XAB,$
- $C = AB + BX + AX.$



Full-adder

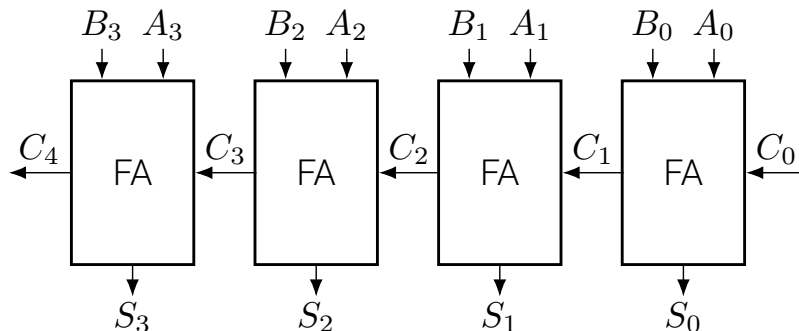
- It can also be implemented with two half adders and one OR gate, as shown below.



$$\begin{aligned} S &= B \oplus (X \oplus A) = B'(XA' + X'A) + B(XA' + X'A)' \\ &= B'(XA' + X'A) + B(XA + X'A') = XA'B' + X'AB' + XAB + X'A'B. \\ C &= B(XA' + X'A) + XA = XA'B + X'AB + XA \end{aligned}$$

Binary adder

- A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.
- It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder in the chain.
- Addition of n -bit numbers requires a chain of n full adders or a chain of one-half adder and $n - 1$ full adders.
 - Below shows the interconnection of four full-adder (FA) circuits to provide a four-bit binary ripple carry adder.

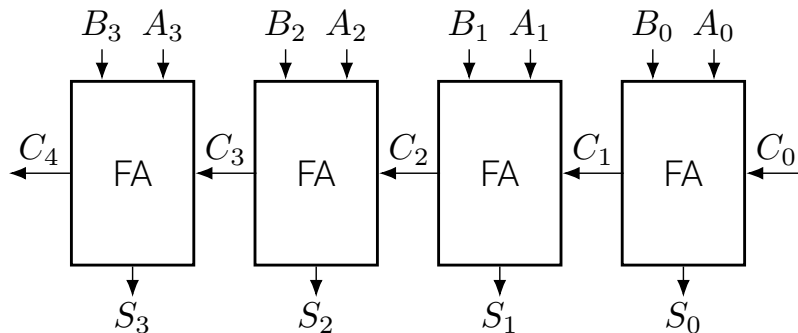


Binary adder



- $1011 + 0011 = 1110$.

Subscript i	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}



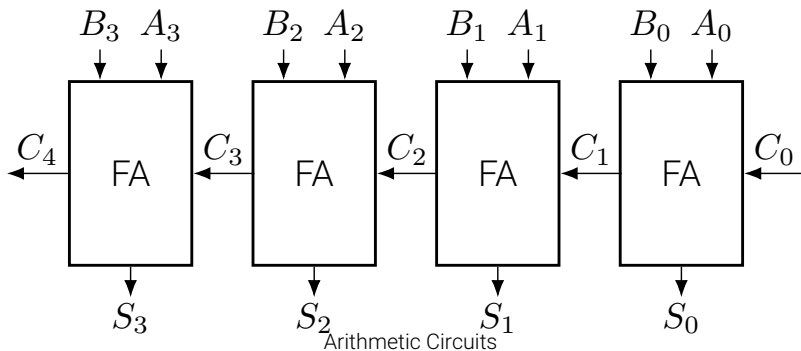
Binary adder

- The four-bit adder is a typical example of a standard component.
- It can be used in many applications involving arithmetic operations.
- Observe that the design of this circuit by the classical method would require a truth table with $2^9 = 512$ entries, since there are nine inputs to the circuit.
- By using an iterative method of cascading a standard function, it is possible to obtain a simple and straightforward implementation.

Carry propagation



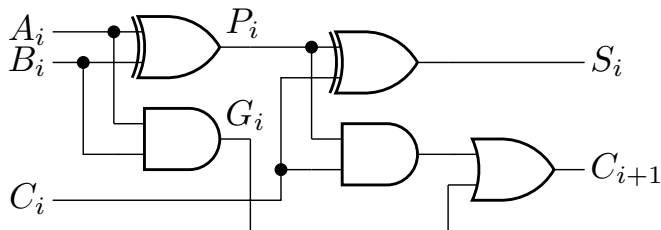
- The addition of two binary numbers in parallel implies that all the bits of the augend and addend are available for computation at the same time.
- As in any combinational circuit, the signal must propagate through the gates before the correct output sum is available in the output terminals.
 - The total propagation time is equal to the propagation delay of a typical gate, times the number of gate levels in the circuit.
 - In this regard, consider output S_3 . Inputs A_3 and B_3 are available as soon as input signals are applied to the adder.
 - However, input carry C_3 does not settle to its final value until C_2 is available from the previous stage.



Carry propagation

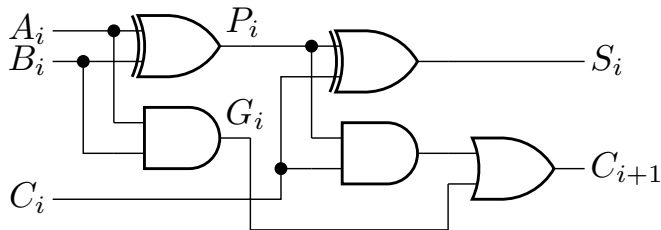
- The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added.
 - Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is critical.
- A solution is to increase the complexity of the equipment in such a way that the carry delay time is reduced.
- The most widely used technique employs the principle of *carry lookahead logic*.

Carry propagation



- Consider this full-adder circuit:
 - $P_i = A_i \oplus B_i$,
 - $G_i = A_i B_i$.
- The output sum and carry can respectively be expressed as
 - $S_i = P_i \oplus C_i$,
 - $C_{i+1} = G_i + P_i C_i$.
- G_i is called a *carry generator*. P_i is called a *carry propagator*.

Carry propagation



- We now write the Boolean functions for the carry outputs of each stage and substitute the value of each C_i from the previous equations.

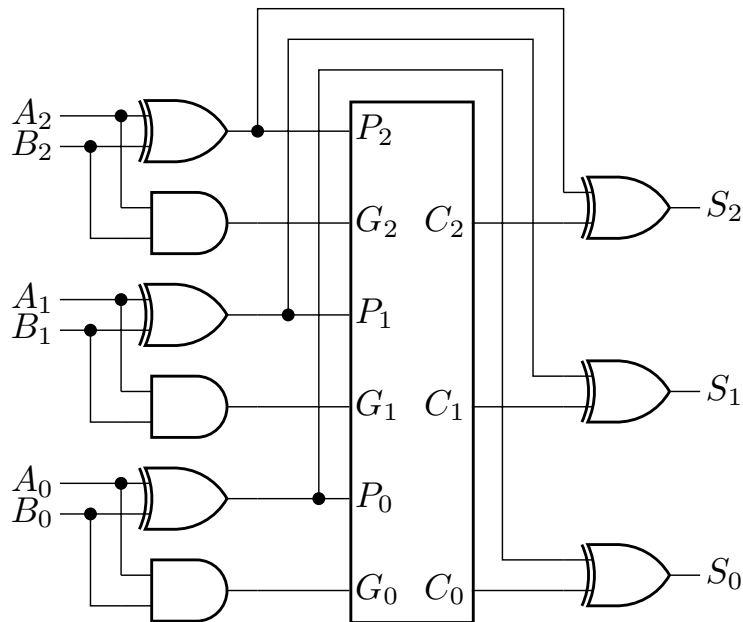
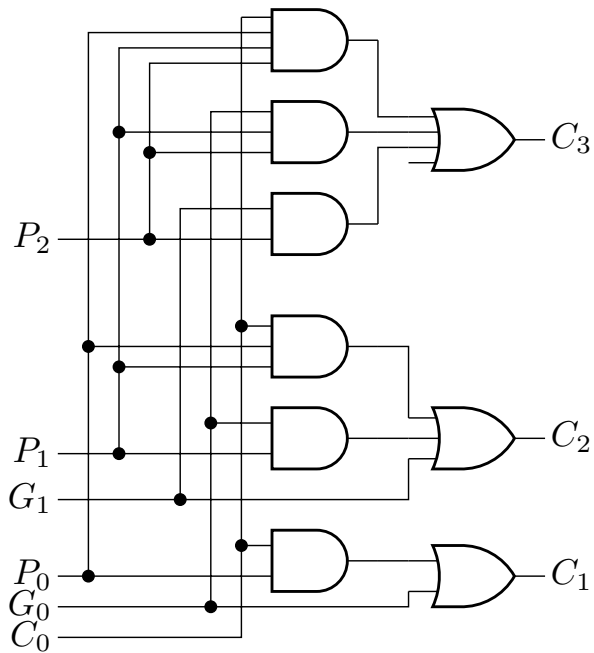
C_0 = input carry,

$$C_1 = G_0 + P_0C_0,$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0,$$

$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0.$$

Carry propagation



Subtraction



- Subtraction is the other basic function of arithmetic operations of information-processing tasks of digital computers.
 - $0 - 0 = 0$,
 - $0 - 1 = 1$ with borrow of 1,
 - $1 - 0 = 1$,
 - $1 - 1 = 0$.
 - The first, third, and fourth operations produce a subtraction of one digit, but the second operation produces a difference bit as well as a *borrow* bit.
- A combinational circuit that performs the subtraction of two bits as described above is called a *half-subtractor*.
- the subtraction operation involves three bits — the *minuend* bit, *subtrahend* bit, and the *borrow* bit, and produces a different result as well as a borrow.
- The combinational circuit that performs this type of subtraction operation is called a *full-subtractor*.

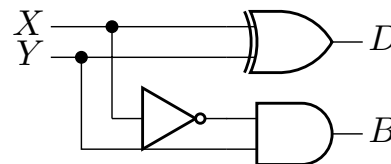


Half-subtractor

- As described above, a half-subtractor has two inputs and two outputs.
- Let the input variables minuend and subtrahend be designated as X and Y , and output functions be designated as D for difference and B for borrow.

Input variables		Output variables	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- $D = X \oplus Y$,
- $B = X'Y$.



Full-subtractor

- A combinational circuit of full-subtractor performs the operation of subtraction of three bits — the minuend, subtrahend, and borrow Z generated from the subtraction operation of previous significant digits and produces the output difference and borrow.

Input variables			Output variables	
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

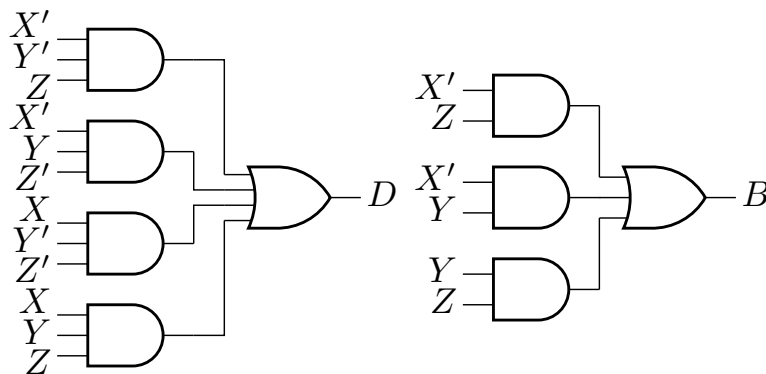
Full-subtractor



$X \backslash AB$	00	01	11	10
0		1		1
1	1		1	

$X \backslash AB$	00	01	11	10
0		1	1	1
1			1	

- $D = X'Y'Z + X'YZ' + XY'Z' + XYZ$,
- $B = X'Z + X'Y + YZ$.



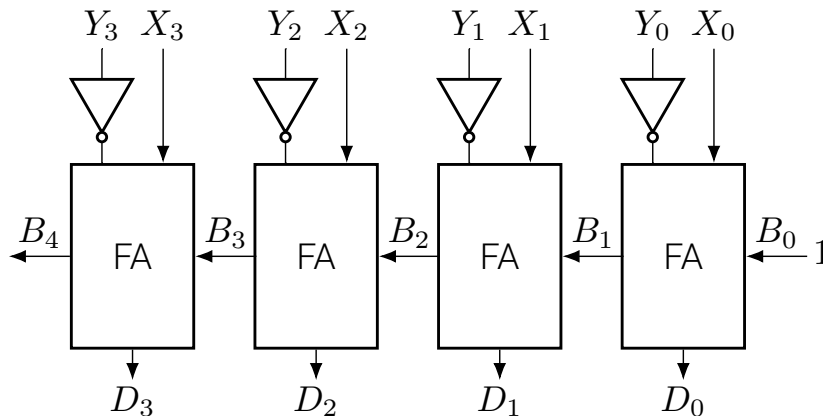
Binary subtractor

- The subtraction of unsigned binary numbers can be done most conveniently by means of complements.
 - The subtraction $A - B$ can be done by taking the 2's complement of B and adding it to A .
 - The 2's complement can be obtained by taking the 1's complement and adding 1 to the least significant pair of bits.
 - The 1's complement can be implemented with inverters, and a 1 can be added to the sum through the input carry.

Binary subtractor

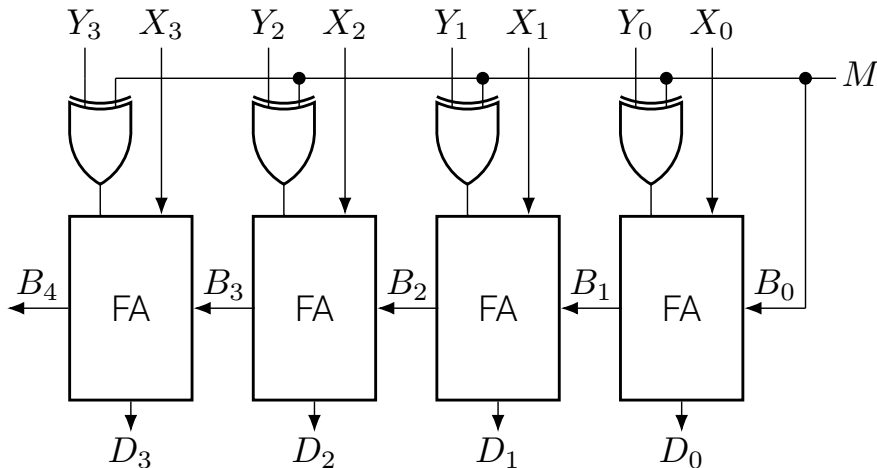


- The circuit for subtracting $A - B$ consists of an adder with inverters placed between each data input B and the corresponding input of the full adder.



Binary subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder.



Binary subtractor

- It is worth noting that binary numbers in the signed-complement system are added and subtracted by the same basic addition and subtraction rules as are unsigned numbers.
- Therefore, computers need only one common hardware circuit to handle both types of arithmetic.

Overflow

- When two numbers with n digits each are added and the sum is a number occupying $n + 1$ digits, we say that an overflow occurred.
 - When the addition is performed with paper and pencil, an overflow is not a problem, since there is no limit by the width of the page to write down the sum.
- Overflow is a problem in digital computers because the number of bits that hold the number is finite and a result that contains $n + 1$ bits cannot be accommodated by an n -bit word.
 - For this reason, many computers detect the occurrence of an overflow, and when it occurs, a corresponding flip-flop is set that can then be checked by the user.
- 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.
 - When two signed numbers are added, the sign bit is treated as part of the number and the end carry does not indicate an overflow.

Overflow



- An overflow may occur if the two numbers are both positive or negative.

carries:	0	1
+70	0	1000110
+80	0	1010000
+150	1	0010110
<hr/>		
carries:	1	0
-70	1	0111010
-80	1	0110000
-150	0	1101010

- If the carry out of the sign bit position is taken as the sign bit of the result, then the nine-bit answer so obtained will be correct.
- But since the answer cannot be accommodated within eight bits, we say that an overflow has occurred.

Overflow

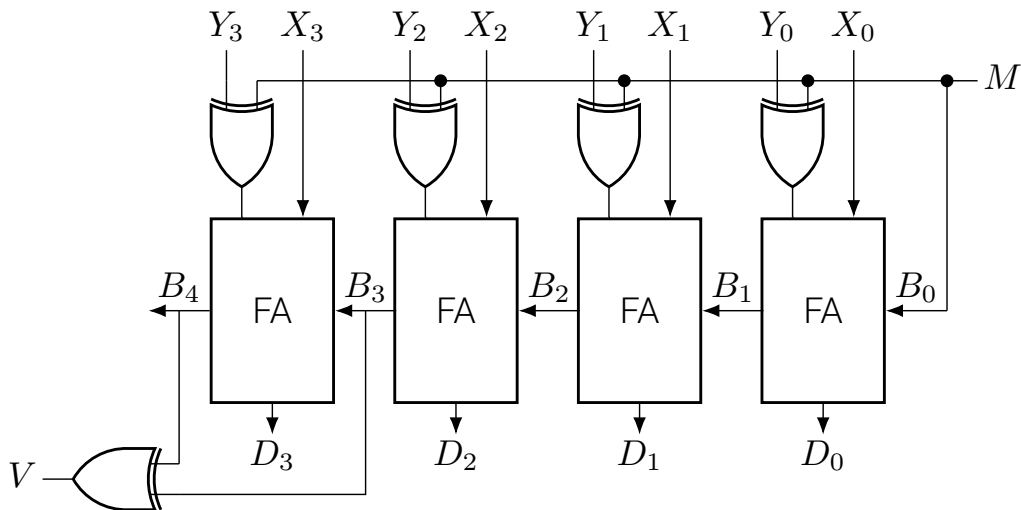


- An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position.
- **If these two carries are not equal, an overflow has occurred.**

carries:	0	1	
+70	0	1000110	
+80	0	1010000	
+150	1	0010110	
<hr/>			
carries:	1	0	
-70	1	0111010	
-80	1	0110000	
-150	0	1101010	

- If the two carries are applied to an exclusive-OR gate, an overflow is detected when the output of the gate is equal to 1.

Overflow



Decimal adder

- Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary coded form.
- An adder for such a computer must employ arithmetic circuits that accept coded decimal numbers and present results in the same code.
- Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage.
 - Since each input digit does not exceed 9, the output sum cannot be greater than $9 + 9 + 1 = 19$, the 1 in the sum being an input carry.
 - Suppose we apply two BCD digits to a four-bit binary adder. The adder will form the sum in binary and produce a result that ranges from 0 through 19.



Decimal adder

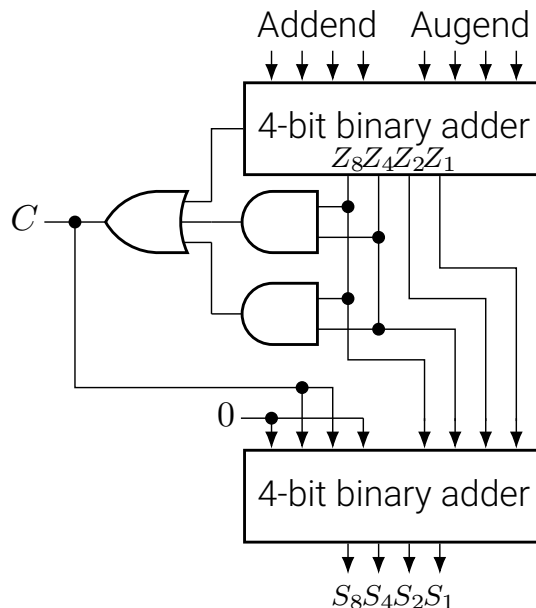
- When the binary sum is equal to or less than 1001 , the result is a valid BCD code.
- When the binary sum is greater than 1001 , we obtain an invalid BCD representation. The addition of binary 6 (0110) to the binary sum converts it to the correct BCD representation and also produces an output carry as required.

K	$Z8$	$Z4$	$Z2$	$Z1$	C	$S8$	$S4$	$S2$	$S1$
0	1	0	1	0	1	0	0	0	0
0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	1	0	0	1	0
0	1	1	0	1	1	0	0	1	1
0	1	1	1	0	1	0	1	0	0
0	1	1	1	1	1	0	1	0	1
1	0	0	0	0	1	0	1	1	0
1	0	0	0	1	1	0	1	1	1
1	0	0	1	0	1	1	0	0	0
1	0	0	1	1	1	1	0	0	1



Decimal adder

- $C = K + Z_8Z_4 + Z_8Z_2$.
- When $C = 1$, it is necessary to add 0110 to the binary sum and provide an output carry for the next stage.



Binary multiplier



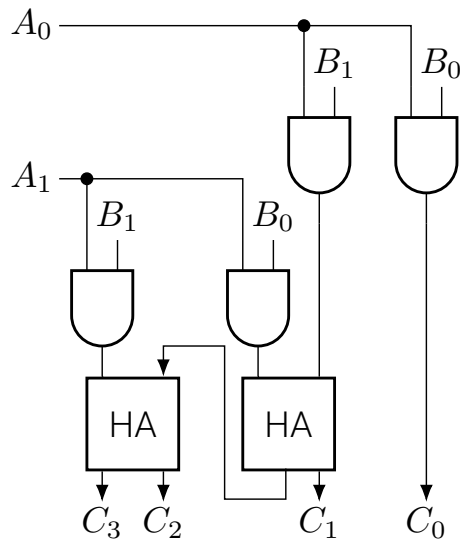
- Multiplication of binary numbers is performed in the same way as multiplication of decimal numbers.
- The multiplicand is multiplied by each bit of the multiplier, starting from the least significant bit.
- Each such multiplication forms a partial product. Successive partial products are shifted one position to the left.

$$\begin{array}{rcccc} & & B_1 & B_0 & \\ & & A_1 & A_0 & \\ & & \hline & & A_0 B_1 & A_0 B_0 & \\ & A_1 B_1 & A_1 B_0 & & \\ & \hline C_3 & C_2 & C_1 & C_0 & \end{array}$$

Binary multiplier



$$\begin{array}{r} B_1 B_0 \\ A_1 A_0 \\ \hline A_0 B_1 \\ A_1 B_1 \\ \hline C_3 C_2 C_0 \end{array}$$



Binary multiplier



- A combinational circuit binary multiplier with more bits can be constructed in a similar fashion.
 - A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier.
 - The binary output in each level of AND gates is added with the partial product of the previous level to form a new partial product.