

EECE 2322: Fundamentals of Digital Design and Computer Organization

Lecture 5_1: ALU

Xiaolin Xu
Department of ECE
Northeastern University

Binary Multiplier

- ❖ Design
- ❖ Optimization
- ❖ What else?

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

	2	3	4	5
x	9	8	7	6
<hr/>				

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

A multiplication table for the number 5. The top row contains 2, 3, 4, and 5. The left column contains an 'x' symbol and the number 5. A vertical line separates the top row from the bottom row. The bottom row contains 9, 8, 7, and 6. Below the table is a horizontal line. To the right of the table, the number 30 is written in orange, with a blue arrow pointing down to the zero.

	2	3	4	5
x	9	8	7	6
				30

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

A hand-drawn multiplication table on a textured, light brown background. The table consists of four columns and four rows of numbers. The first column contains 'x' at the top and '5' at the bottom. The second column contains '2' at the top and '9' at the bottom. The third column contains '3' at the top and '8' at the bottom. The fourth column contains '4' at the top and '7' at the bottom. The fifth column contains '5' at the top and '6' at the bottom. Arrows point from the '4' in the first column to the '5' in the fifth column, and from the '5' in the fifth column to the '0' in the result row. The result row contains '2' at the tens place and '0' at the ones place, with a small orange '3' written above the tens digit. A horizontal line separates the table from the result row.

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

$$\begin{array}{r} 5 \\ \times 6 \\ \hline 30 \end{array}$$

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

A hand-drawn multiplication diagram on a textured yellow background. On the left, there is a blue 'x' symbol. Above it are the numbers 5 and 6, with blue arrows pointing from each number to the corresponding digits in the product. The product is written below a horizontal line: 1 4 2 0 2 7 3 0. The tens column has orange digits (4, 0, 2, 7), and the ones column has black digits (2, 7, 3, 0).

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

A diagram illustrating a multiplication process. At the top, there are two rows of numbers: 2, 3, 4, 5 on the top row and 9, 8, 7, 6 on the bottom row. A diagonal line connects the 5 and 6. Below this, there is a multiplication table with an 'x' symbol. The result of the multiplication is shown as 1, 4, 2, 0, 2, 7, 3, 0. An arrow points from the second digit of the result (4) to the second digit of the multiplier (2). The second digit of the result (4) is highlighted in orange.

An Example with Decimal Numbers

- ❖ Review this process
 - ❖ We need some look-up-table, i.e., $5 * 6 = 30$

The diagram illustrates a multiplication problem and a corresponding look-up table. The multiplication is shown as x followed by a horizontal line, then the result $1\ 1\ 4\ 2\ 0\ 2\ 7\ 3\ 0$. Above the horizontal line is a 4x4 grid of numbers. Blue arrows indicate the mapping from the factors 5 and 6 to the grid: one arrow from 5 to the row 9, 8, 7, 6, and another from 6 to the column 2, 3, 4, 5. The result 1 1 4 2 0 2 7 3 0 has its digits 4, 0, 7, and 3 highlighted in orange, while the other digits are in black. A blue arrow points from the orange digit 4 in the result back to the digit 5 in the grid.

- ❖ There will be many more such operation!

An Example with Decimal Numbers

- ❖ Review this process

- ❖ We need some look-up-table

The image shows a yellow notepad with two multiplication problems written on it.

The first problem is:

$$\begin{array}{r} 2345 \\ \times 9876 \\ \hline 14202730 \end{array}$$

The digits 2, 3, 4, 5 in the top number and 9, 8, 7, 6 in the bottom number are colored blue. The digits 1, 4, 2, 0, 2, 7, 3, 0 in the result are black, except for the tens digit 4 which is orange. There are question marks in the empty boxes of the multiplication grid.

The second problem is:

$$\begin{array}{r} \times \\ \hline ? ? ? ? ? \\ ? ? ? ? ? \\ \hline ? ? ? ? ? ? ? \end{array}$$

The first digit of the multiplier is blue, while the others and the entire multiplicand are black. The result digits are also black, except for the first one which is blue.

- ❖ There will be many more such operation!

Any Quick, Straightforward Solution?

- ❖ A simpler example

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline \end{array}$$

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline \end{array}$$

Any Quick, Straightforward Solution?

- ❖ A simpler example

A binary multiplication diagram. On the left, there is a blue 'x' symbol. To its right is a vertical column of four binary digits: 1, 0, 0, 0. Below this column is a horizontal line. To the right of the line is another vertical column of four binary digits: 1, 0, 0, 0. The bottom row of the multiplication result is 1 0 0 0. The top digit of the second column (0) has three blue arrows pointing to the bottom digit of the first column (1), indicating that it is being multiplied by 1.

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}$$

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \end{array}$$

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} \times \\ \begin{array}{r} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \end{array}$$

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Any Quick, Straightforward Solution?

- ❖ A simpler example

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

- ❖ What we learned from there?

Summary from the Previous Example

Summary from the Previous Example

$$\begin{array}{r} & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \times & \begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ & \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \\ & \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \\ & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \hline & \begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \end{array}$$

Summary from the Previous Example

- ❖ Multiplier = 1

$$\begin{array}{r} & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \times & \begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ & \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \\ & \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \\ & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \hline & \begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \end{array}$$

Summary from the Previous Example

- ❖ Multiplier = 1

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The diagram shows a multiplication of two binary numbers. The multiplicand is 1000 and the multiplier is 1001. The result is 1001000. A blue 'x' symbol is placed next to the multiplicand. The digit 1 in the multiplier is circled in orange. The product 1000 is highlighted with an orange underline. The final result 1001000 is shown in red.

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand
- ❖ Multiplier = 0

$$\begin{array}{r} & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \times & \begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \\ \hline & \begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \end{array}$$

The diagram shows a multiplication process. The multiplicand is 1000, and the multiplier is 1001. The result is 1001000. A blue 'x' symbol is placed next to the first multiplicand. The digit 1 in the multiplier is circled in orange. The product 1000 is highlighted with an orange underline. The final result 1001000 is shown in red.

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand
- ❖ Multiplier = 0

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The diagram shows a multiplication process. The multiplicand is 1000, and the multiplier is 1001. The result is 1001000. A blue circle highlights the digit 0 in the multiplier, and a red underline highlights the 000 in the partial product row, indicating that the multiplicand was directly applied. The final result is shown in red.

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand
- ❖ Multiplier = 0
 - ❖ Directly apply all-0

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The diagram shows a binary multiplication process. The multiplicand is 1000 and the multiplier is 1001. The result is 1001000. The middle column of the multiplier (0) is circled in orange, and the corresponding row of the partial product (000) is also circled in orange, illustrating the direct application of the multiplicand when the multiplier bit is 1. The bottom row of zeros represents the result of multiplying by the least significant bit of the multiplier, which is 0.

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand
- ❖ Multiplier = 0
 - ❖ Directly apply all-0
- ❖ No more look-up-table checking

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The diagram shows a multiplication process. The multiplicand is 1000 and the multiplier is 1001. The result is 1001000. The digit 0 in the multiplier is highlighted with a yellow circle, and the corresponding row of partial products (000) is also highlighted with a yellow underline. The final result 1001000 is shown in red.

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand
- ❖ Multiplier = 0
 - ❖ Directly apply all-0
- ❖ No more look-up-table checking
- ❖ This example is applicable to binary!

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The diagram shows a binary multiplication process. The multiplicand is 1000 and the multiplier is 1001. The result is 1001000. The middle column of the multiplier (0) is circled in orange, and the middle row of the partial products (000) is also circled in orange, illustrating the direct application of the multiplicand. The final result is shown in red.

Summary from the Previous Example

- ❖ Multiplier = 1
 - ❖ Directly apply multiplicand
- ❖ Multiplier = 0
 - ❖ Directly apply all-0
- ❖ No more look-up-table checking
- ❖ This example is applicable to binary!
- ❖ An important reason why our computer use binary data!

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The diagram shows a binary multiplication process. The multiplicand is 1000 and the multiplier is 1001. The result is 1001000. A blue circle highlights the 1 in the multiplier's second column, and a red circle highlights the 1 in the result's fourth column. Horizontal orange lines highlight the 000 in the multiplier and the 0000 in the result.

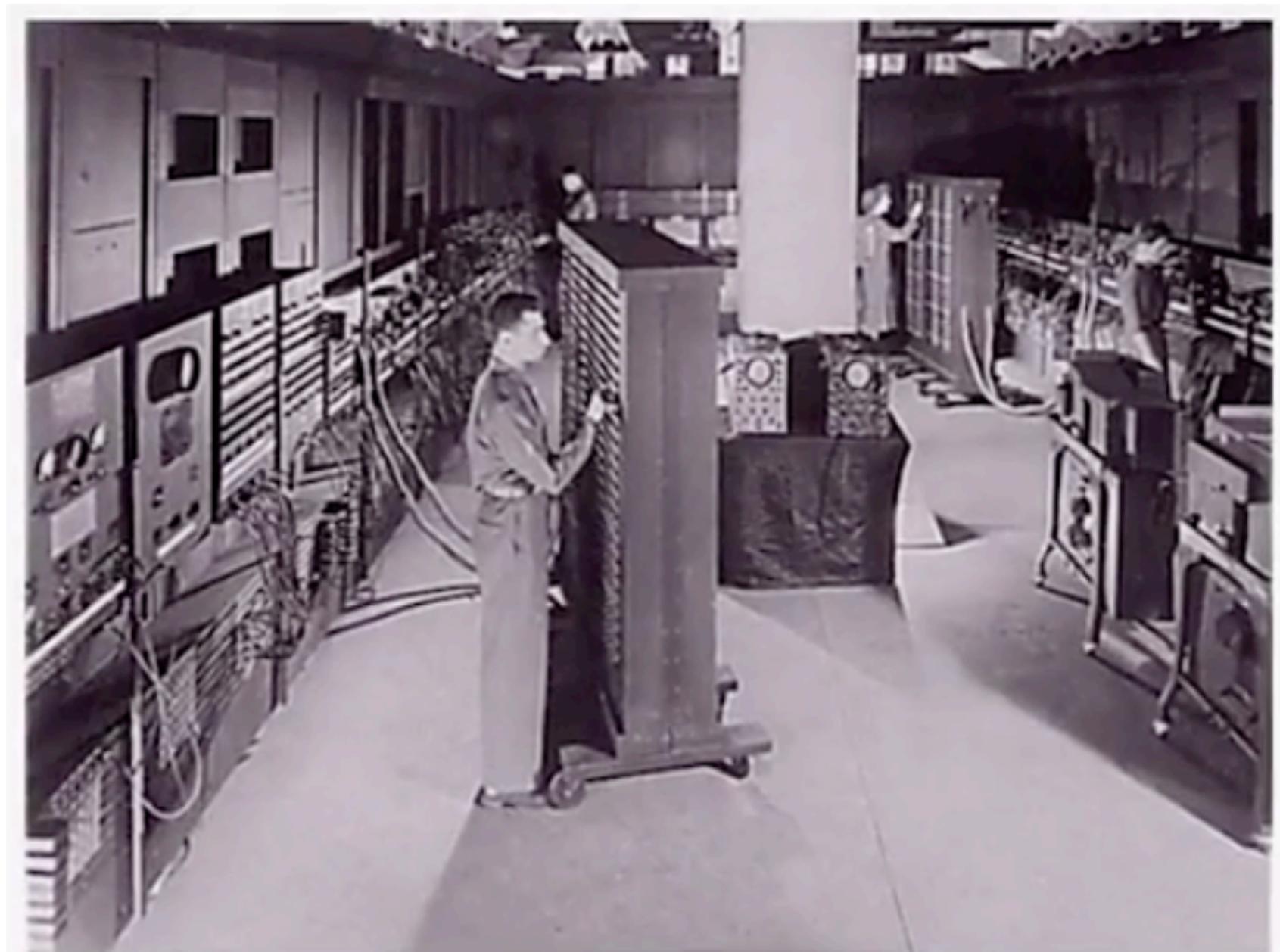
Between Decimal and Binary

Between Decimal and Binary

- ❖ ENIAC, a computer based on decimal

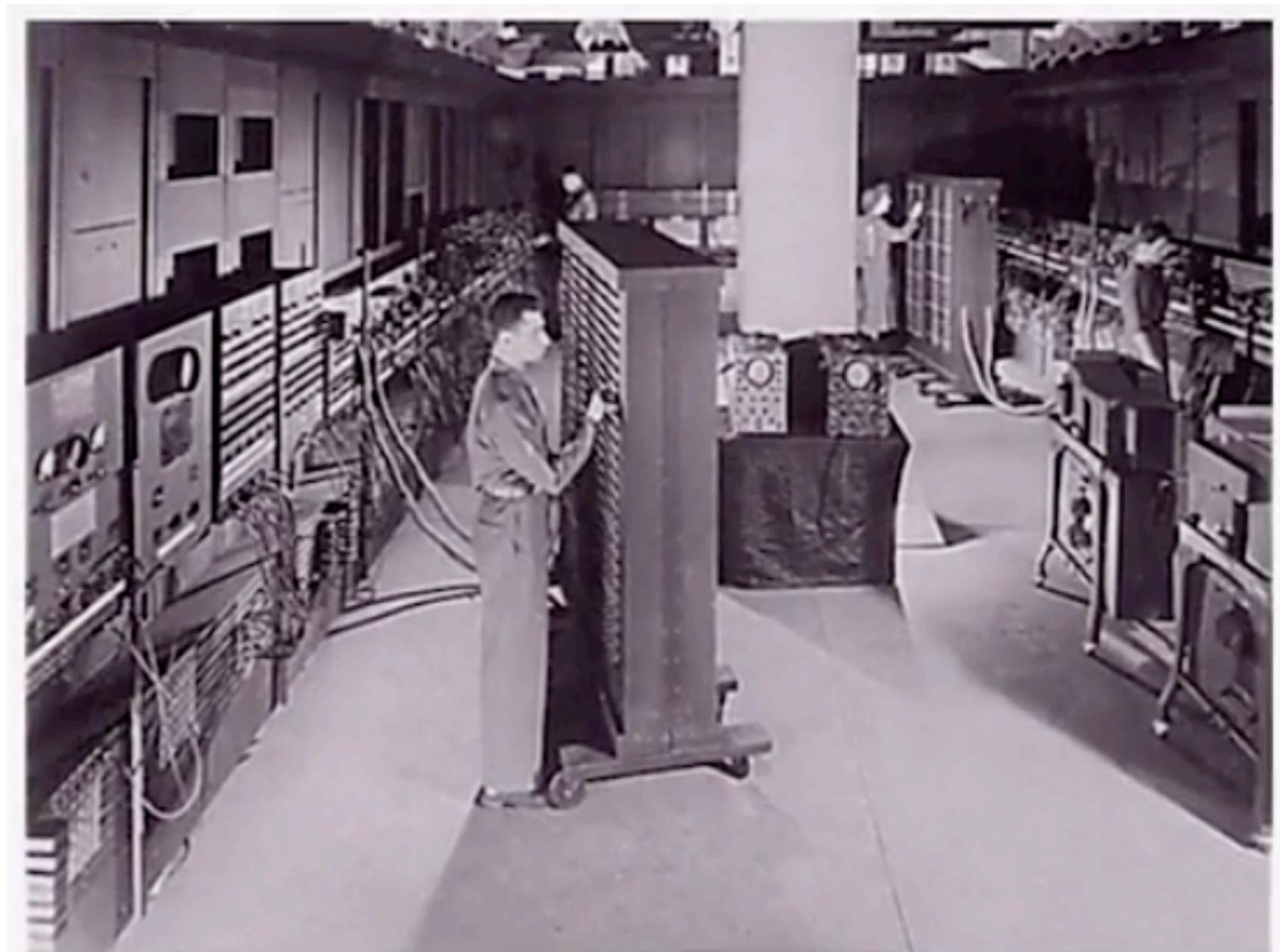
Between Decimal and Binary

- ❖ ENIAC, a computer based on decimal



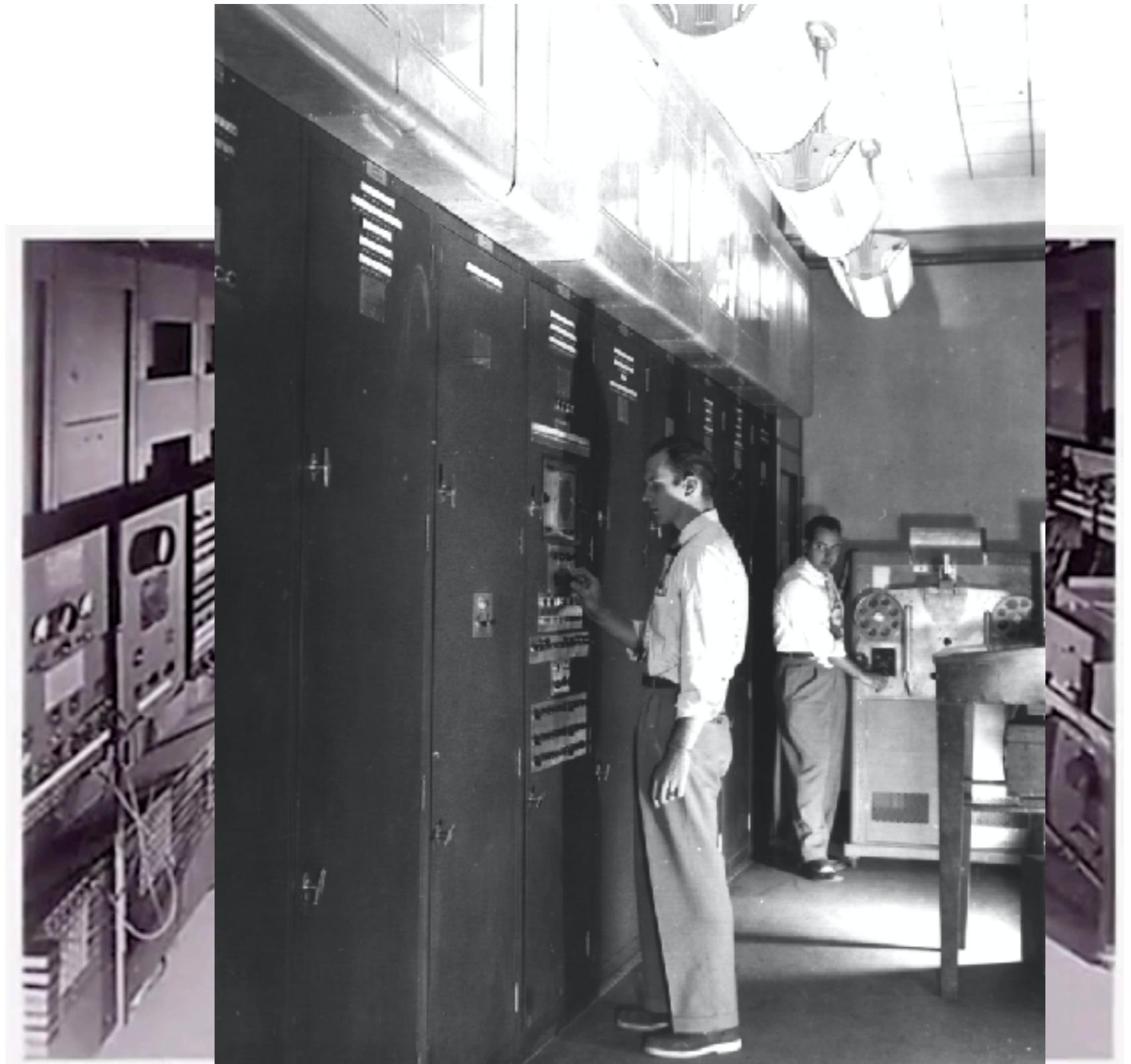
Between Decimal and Binary

- ❖ ENIAC, a computer based on decimal
- ❖ EDVAC, binary computing



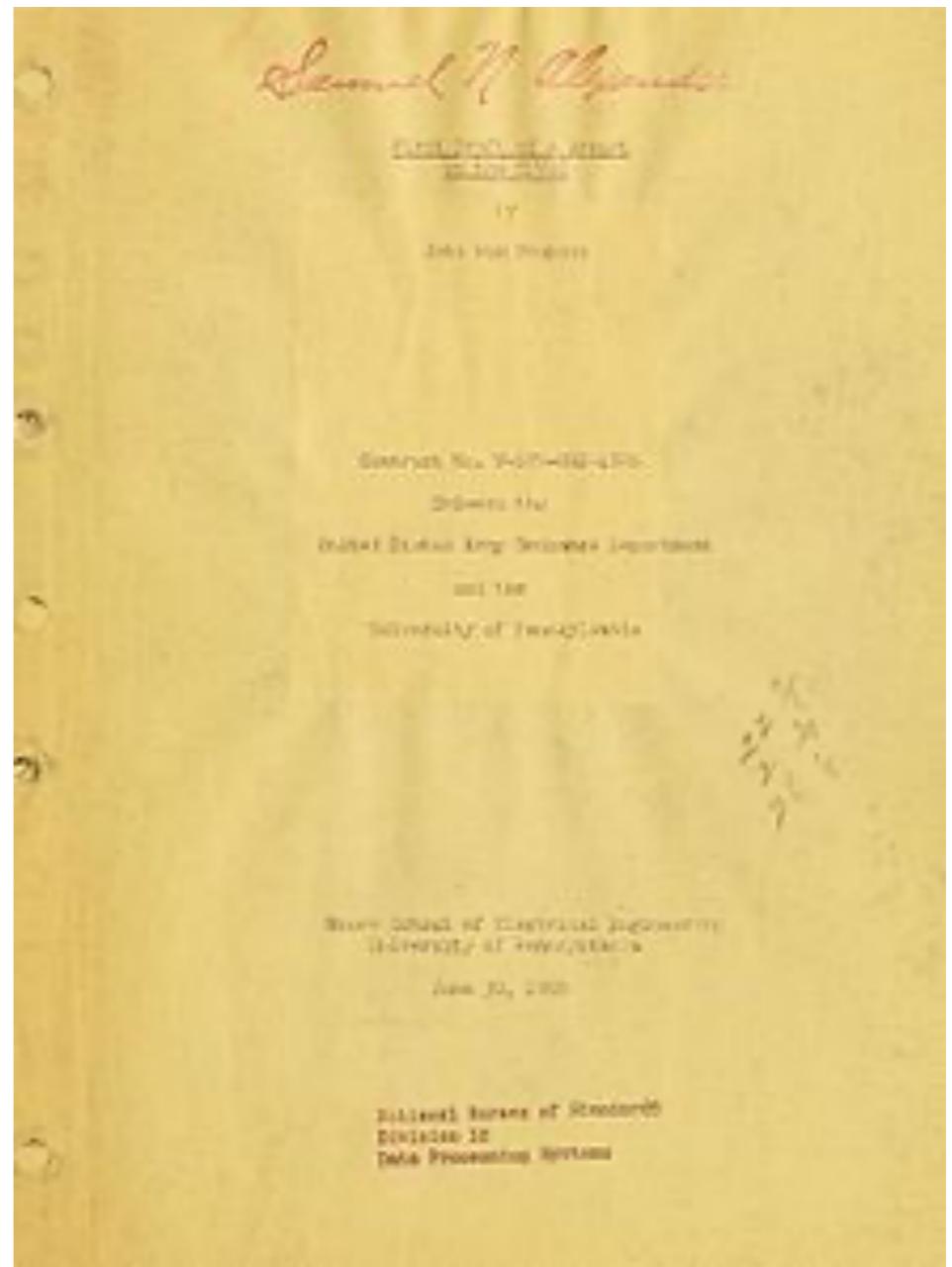
Between Decimal and Binary

- ❖ ENIAC, a computer based on decimal
- ❖ EDVAC, binary computing

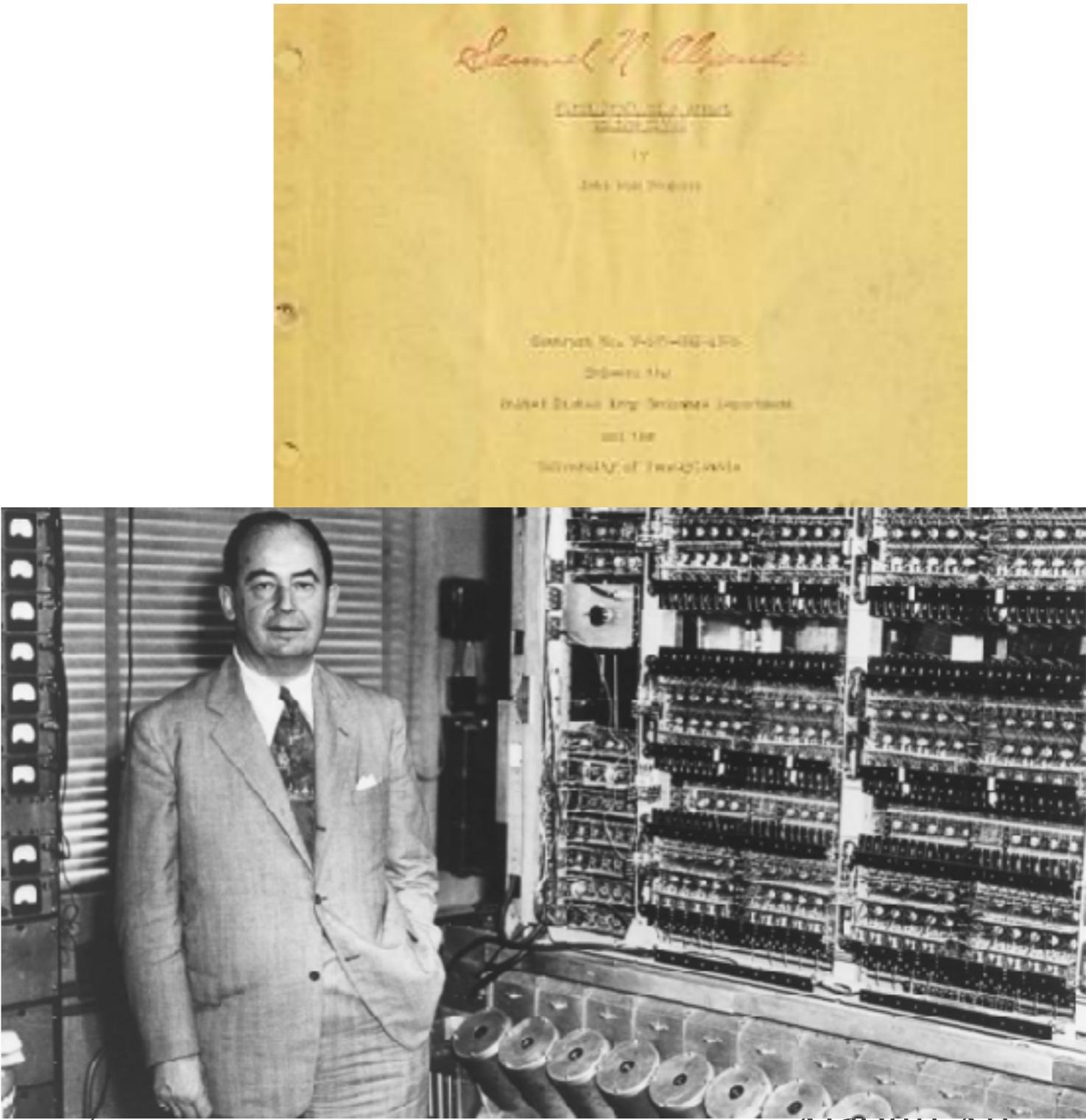


Between Decimal and Binary

Between Decimal and Binary

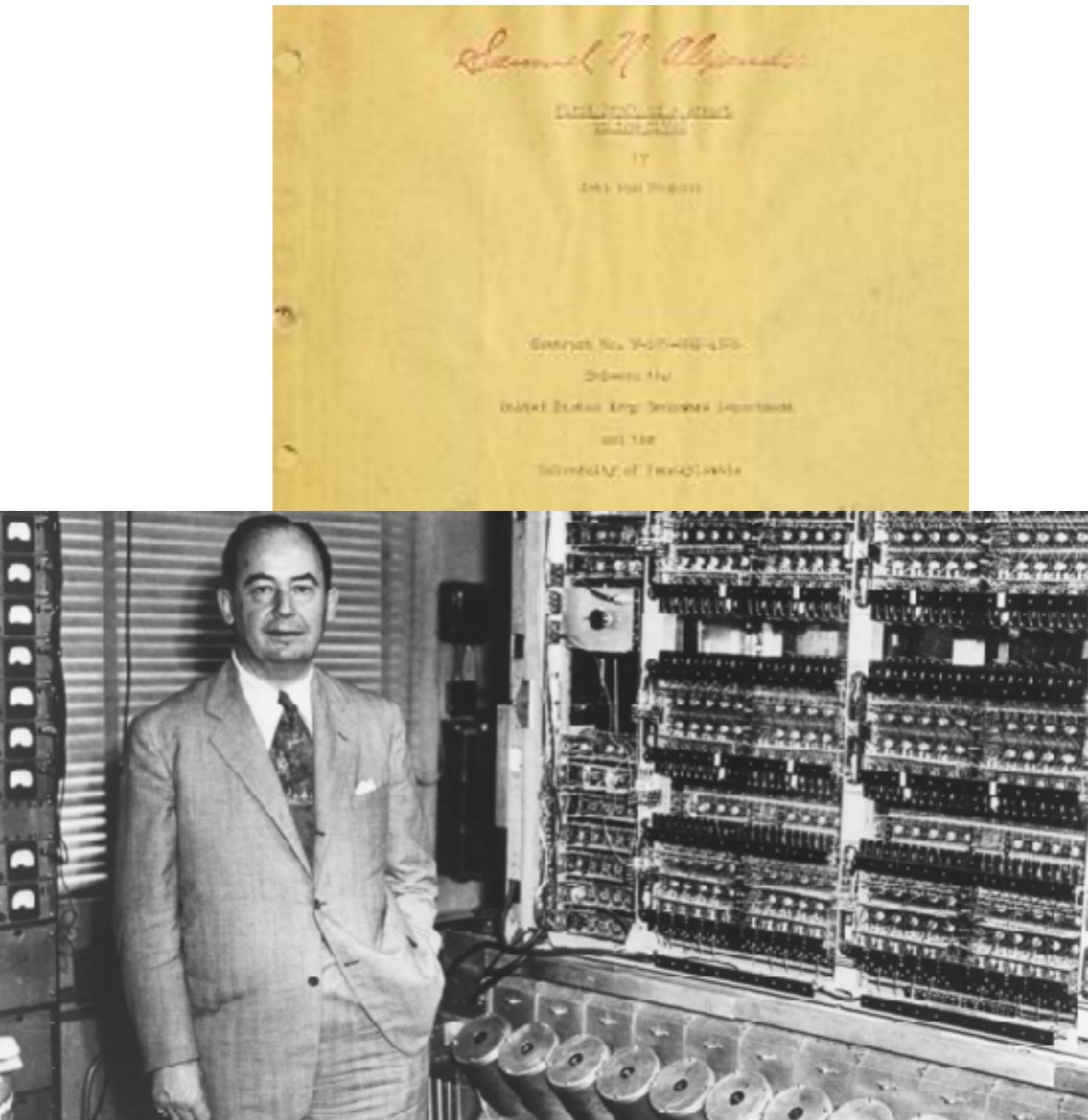


Between Decimal and Binary



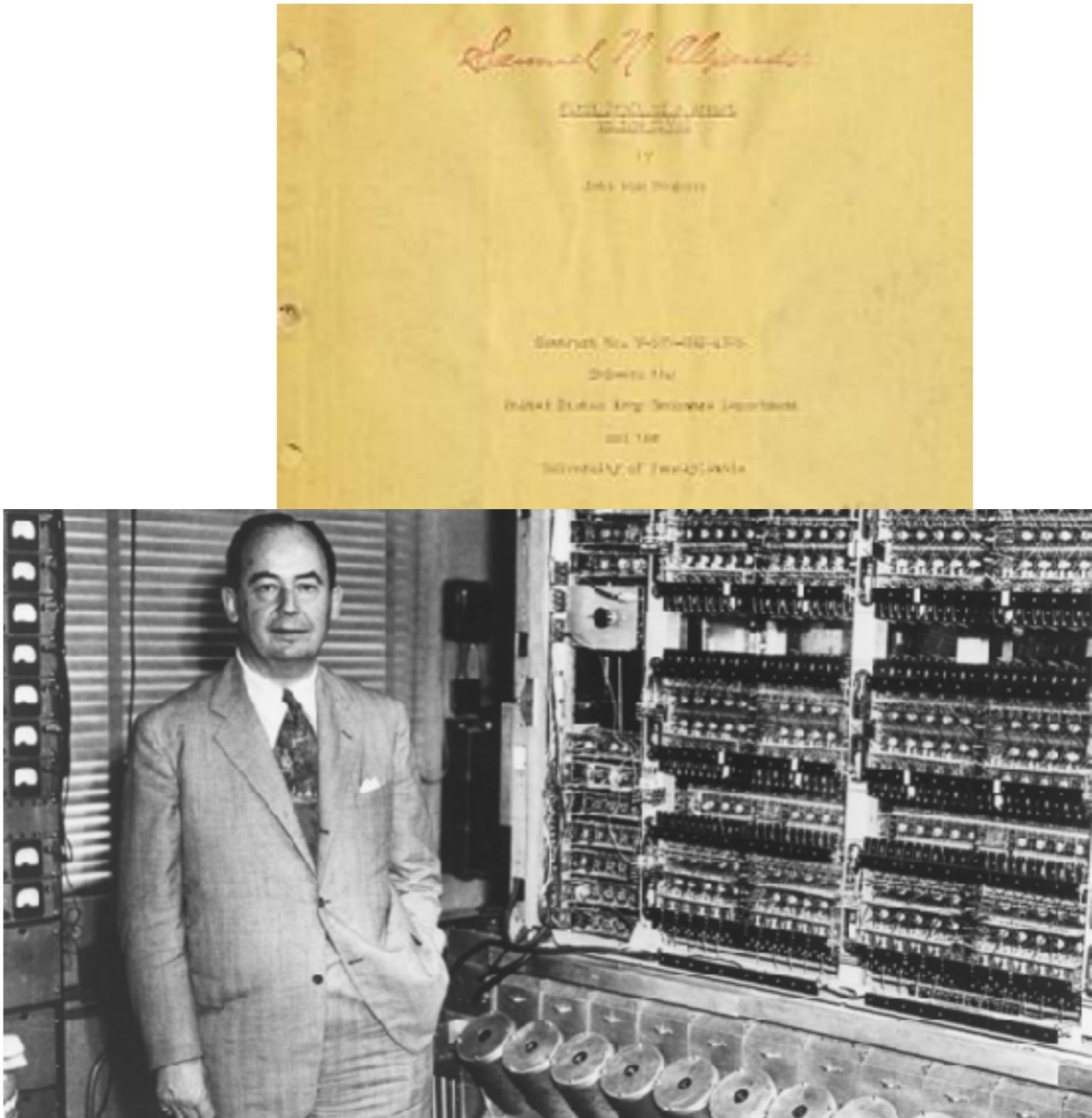
Between Decimal and Binary

- ❖ Transistor is a “all-or-non” device
 - ❖ Thus, only two values are reasonable



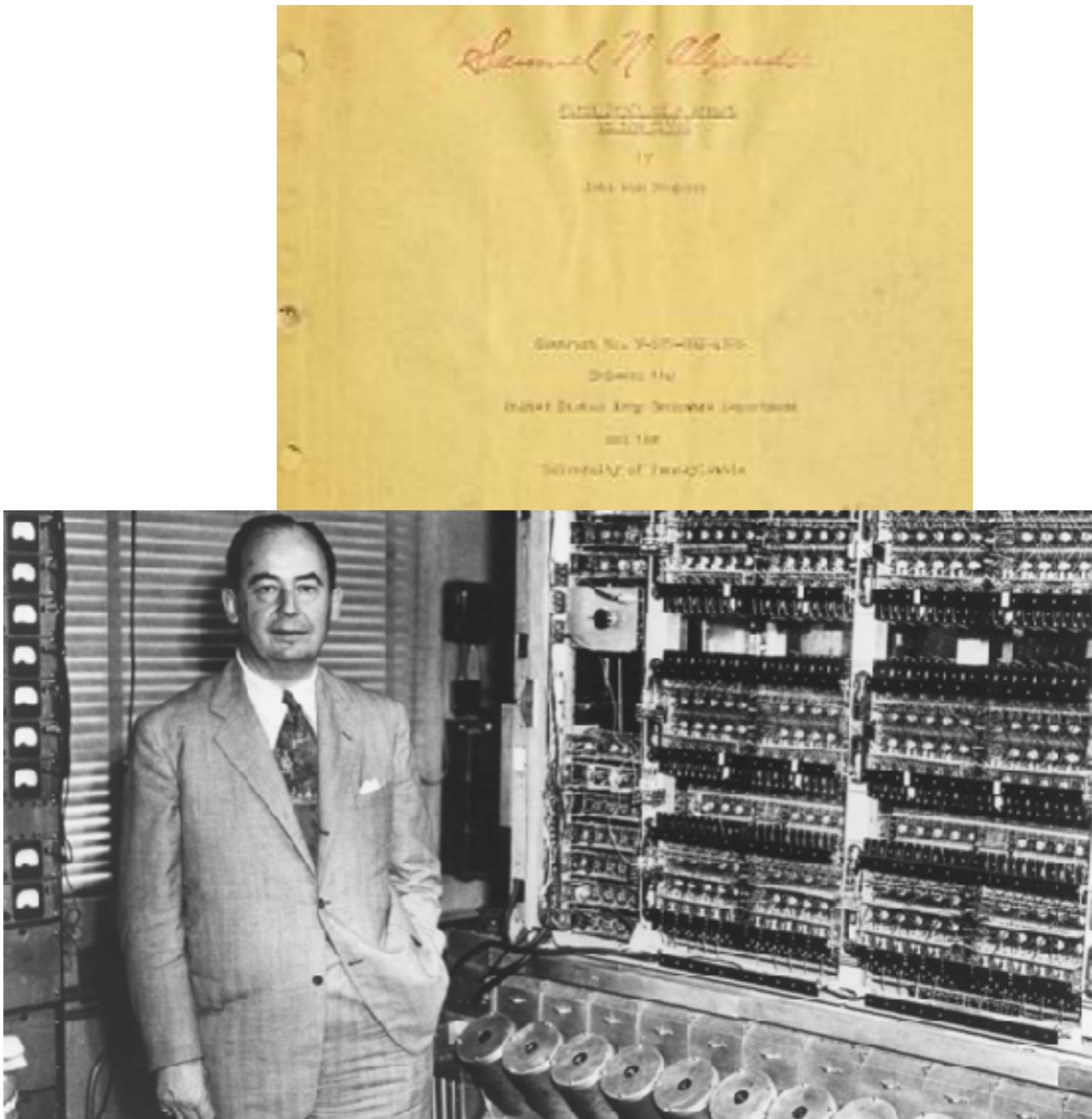
Between Decimal and Binary

- ❖ Transistor is a “all-or-non” device
 - ❖ Thus, only two values are reasonable
- ❖ Binary can significantly simplify the multiplication and division
 - ❖ No more look-up-table
 - ❖ No more intermediate value to be stored



Between Decimal and Binary

- ❖ Transistor is a “all-or-non” device
 - ❖ Thus, only two values are reasonable
- ❖ Binary can significantly simplify the multiplication and division
 - ❖ No more look-up-table
 - ❖ No more intermediate value to be stored
- ❖ But remember, decimal is for human being
 - ❖ We need converter! I.e., keyboard, monitor, etc.



Hardware Implementation

Hardware Implementation

$$\begin{array}{r} & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \times & \begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \end{array}$$

Hardware Implementation

- ❖ Different from what we do on a paper?

$$\begin{array}{r} & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \\ \times & \begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \\ \hline & \begin{array}{cccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \end{array}$$

Hardware Implementation

- ❖ Different from what we do on a paper?

$$\begin{array}{r} & \begin{array}{r} 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline \end{array} \\ & \begin{array}{r} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \end{array}$$

Hardware Implementation

- ❖ Different from what we do on a paper?
- ❖ Paper to computer
 - ❖ “Smaller paper size”
 - ❖ Only has space for
 - ❖ Multiplicand
 - ❖ Multiplier
 - ❖ Product
 - ❖ We have eraser as well!

A multiplication problem on aged, yellowish paper. The problem is set up as follows:

1	0	0	0			
×	1	0	0			
1	0	0	0			
0	0	0	0			
0	0	0	0			
1	0	0	0			
1	0	0	1	0	0	0

Hardware Implementation

- ❖ Different from what we do on a paper?
- ❖ Paper to computer
 - ❖ “Smaller paper size”
 - ❖ Only has space for
 - ❖ Multiplicand
 - ❖ Multiplier
 - ❖ Product
 - ❖ We have eraser as well!

$$\begin{array}{r} & 1 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Hardware Implementation

Hardware Implementation

$$\begin{array}{r} \times \\ \begin{array}{r} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array} \\ \hline \begin{array}{r} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

A binary multiplication diagram showing the multiplication of two 4-bit binary numbers. The top number is 1000 and the bottom number is 1001. The result is 0000000. A circled '1' in the multiplier is highlighted, and a horizontal bar highlights the first row of the partial product column.

Hardware Implementation

$$\begin{array}{r} \times \\ \begin{array}{r} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array} \\ \hline \begin{array}{r} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

A binary multiplication diagram showing the multiplication of two 4-bit binary numbers. The top number is 1000 and the bottom number is 1001. The result is 0000000. A circled '1' in the multiplier is highlighted, and a horizontal bar highlights the first row of the partial product column.

Hardware Implementation

- ❖ 1, Product marked as all-0

		1	0	0	0	
	x	1	0	0	1	
		1	0	0	0	
		0	0	0	0	
		0	0	0	0	
		1	0	0	0	
		0	0	0	0	0

Hardware Implementation

- ❖ 1, Product marked as all-0
- ❖ 2, Multiplier = 1
 - ❖ Directly apply multiplicand
 - ❖ However, no space for it!
 - ❖ Use eraser, store it as product (temporary!)

		1	0	0	0	0
	x	1	0	0	1	
		1	0	0	0	
		0	0	0	0	
		0	0	0	0	
		1	0	0	0	
		0	0	0	0	0

Hardware Implementation

- ❖ 1, Product marked as all-0
- ❖ 2, Multiplier = 1
 - ❖ Directly apply multiplicand
 - ❖ However, no space for it!
 - ❖ Use eraser, store it as product (temporary!)
- ❖ 3, We need shift-to-left now
 - ❖ No space for it anymore...
 - ❖ How to proceed?

		1	0	0	0	0
	x	1	0	0	1	
		1	0	0	0	
		0	0	0	0	
		0	0	0	0	
		1	0	0	0	
		0	0	0	0	0

Hardware Implementation

- ❖ 1, Product marked as all-0
- ❖ 2, Multiplier = 1
 - ❖ Directly apply multiplicand
 - ❖ However, no space for it!
 - ❖ Use eraser, store it as product (temporary!)
- ❖ 3, We need shift-to-left now
 - ❖ No space for it anymore...
 - ❖ How to proceed?

×	1	0	0	0	0	0
	1	0	0	1		
		1	0	0	0	0
		0	0	0	0	0
		0	0	0	0	0
		1	0	0	0	0
	0	0	0	1	0	0

Hardware Implementation

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product

	1	0	0	0
x	1	0	0	1
	1	0	0	0
	0	0	0	0
	0	0	0	0
	1	0	0	0
	0	0	0	1
	0	0	0	0

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

	1	0	0	0
x	1	0	0	1
	1	0	0	0
	0	0	0	0
	0	0	0	0
	1	0	0	0
	0	0	0	1
	0	0	0	0

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

A binary multiplication diagram on a light brown background. It shows the multiplication of two 4-bit binary numbers: 1000 and 1001. The multiplicand 1000 is at the top, followed by a multiplication sign (x), and the multiplier 1001. A horizontal line separates the factors from the partial products. Below the line, there are four rows of partial products: 1000, 0000 (underlined in orange), 0000, and 1000. The sum of these partial products is 00010000, which is the final result.

	1	0	0	0			
x	1	0	0	1			
<hr/>							
	1	0	0	0			
	0	0	0	0			
	0	0	0	0			
	1	0	0	0			
	0	0	0	1	0	0	0

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

A binary multiplication diagram on a yellow background. At the top, the multiplicand 1000 is written above a blue multiplication sign. To its right, the multiplier 1001 is written. Below the multiplication sign is a horizontal line. Underneath the multiplicand, there are four rows of zeros, each aligned with one of the digits of the multiplicand. To the right of these rows is another horizontal line. Below the first horizontal line, the partial product 1000 is written. Below the second horizontal line, the partial product 0000 is written. Below the third horizontal line, the partial product 0000 is written. Below the fourth horizontal line, the partial product 1000 is written. At the bottom, the final product 0001000 is written in red.

$$\begin{array}{r} 1000 \\ \times 1001 \\ \hline 1000 \\ 0000 \\ 0000 \\ 1000 \\ \hline 0001000 \end{array}$$

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

A binary multiplication diagram on a yellow background. It shows the multiplication of two 4-bit binary numbers: 1000 and 1001. The multiplication symbol (×) is between them. A horizontal line separates the factors from the partial products. There are four partial products: 0000, 0000, 0000, and 1000, stacked vertically. The 1000 is underlined in orange. Another orange circle highlights the 0 in the tens column of the 1001 multiplier. The final result at the bottom is 0001000, where the first three digits are in red.

	1	0	0	0			
×	1	0	0	1			
<hr/>							
	1	0	0	0			
	0	0	0	0			
	0	0	0	0			
	1	0	0	0			
<hr/>							
	0	0	0	1	0	0	0

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

A binary multiplication diagram. At the top, the multiplicand 1000 is written above a multiplication sign ×, and the multiplier 1001 is to its right. Below the multiplication sign is a horizontal line. To the left of the line, there are four rows of zeros, each aligned with one of the digits of the multiplicand. To the right of the line, there are four rows of digits, starting with 1000 and followed by three rows of zeros. At the bottom, the result 0001000 is written in red.

1	0	0	0	0		
×	1	0	0	1		
	1	0	0	0		
	0	0	0	0		
	0	0	0	0		
	1	0	0	0		
<hr/>						
0	0	0	1	0	0	0

Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

A binary multiplication diagram on a light brown background. At the top left is the number 1000. To its right is a blue multiplication sign (×). To the right of the sign is the number 1001, with the third bit from the left (0) highlighted with a yellow circle. Below these numbers is a horizontal line. Underneath the line, the first row of digits is 1000. Below that is a row of zeros. Below that is another row of zeros. Below that is a row of ones. A horizontal line underlines the first three digits of this row. Below this underline is a red row of digits: 0001000. The entire diagram is enclosed in a thin black border.

$$\begin{array}{r} 1000 \\ \times 1001 \\ \hline 0001000 \end{array}$$

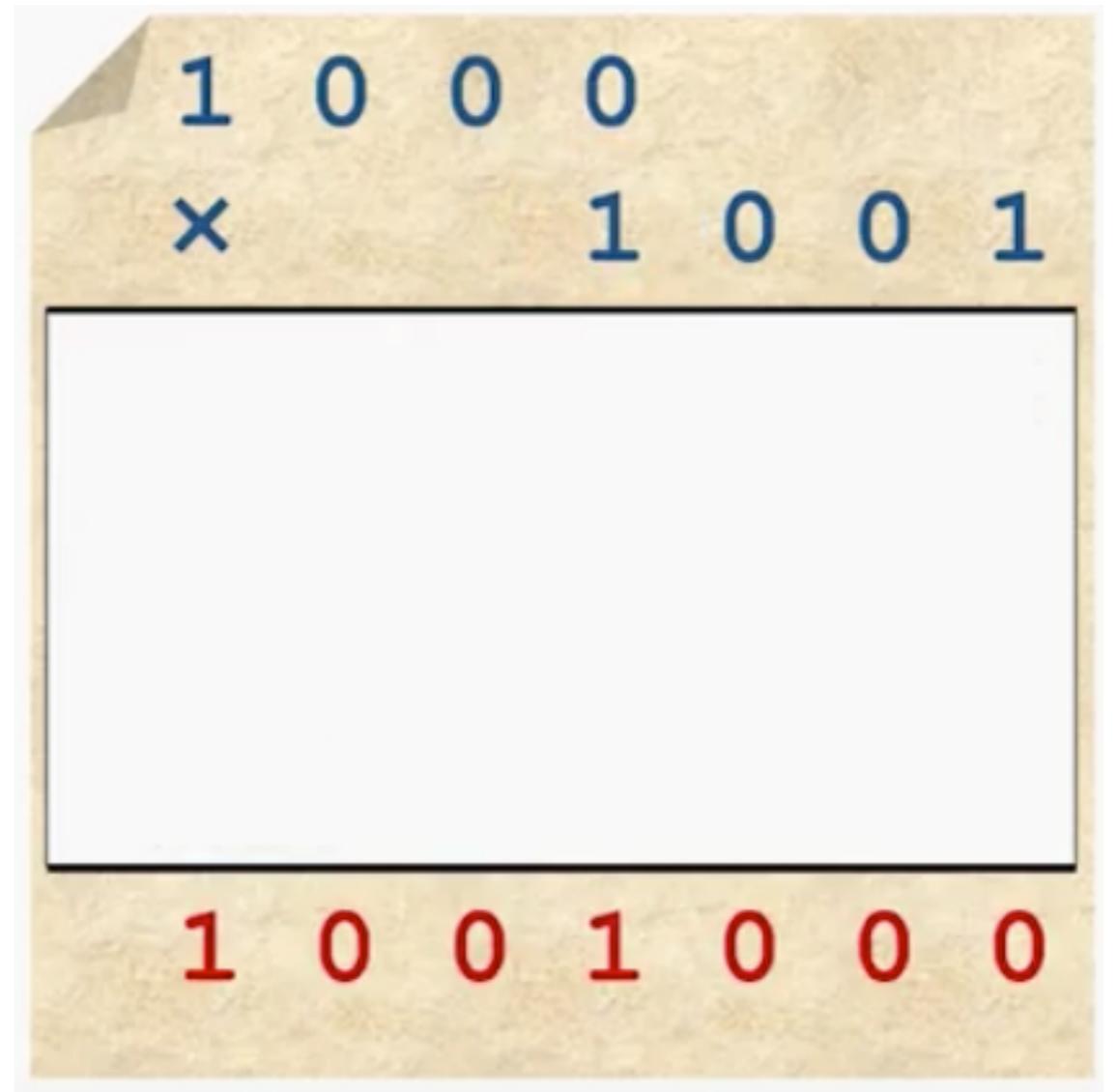
Hardware Implementation

- ❖ 4, Shifting the multiplicand instead
 - ❖ Calculate as step 1-3
 - ❖ If all-0
 - ❖ Just like we already added it to the product, or did nothing
 - ❖ If not all-0
 - ❖ Adding it to the product
- ❖ 5, Repeat step 4

A binary multiplication diagram on a light brown background. It shows the multiplication of two 4-bit binary numbers: 1000 (multiplicand) and 1001 (multiplier). The result is 1001000. The diagram uses a grid for the multiplication steps and a horizontal line for the final sum. The multiplier '1001' has its third bit circled in orange. The partial product row under '1001' and the final sum row are also highlighted with orange lines.

1	0	0	0			
x	1	0	0	1		
<hr/>						
	1	0	0	0		
	0	0	0	0		
	0	0	0	0		
	1	0	0	0		
<hr/>						
1	0	0	1	0	0	0

Hardware Implementation



Hardware Implementation

- ❖ 6, All bit of multiplier checked?

A binary multiplication diagram. At the top, the number 1000 is written above a multiplication sign (×). To the right of the multiplication sign is the number 1001. A large empty rectangular box is positioned below the numbers, intended for the product. Below this box, the result of the multiplication is shown: 1001000. The digits 1, 0, 0, 1, 0, 0, 0 are written in red.

$$\begin{array}{r} 1000 \\ \times 1001 \\ \hline \end{array}$$

1 0 0 1 0 0 0

Hardware Implementation

- ❖ 6, All bit of multiplier checked?
 - ❖ No, repeat step 4

A binary multiplication diagram. At the top, the number 1000 is written above a multiplication sign (×). To the right of the multiplication sign is the number 1001. A large empty rectangular box is positioned below the numbers, intended for the product. Below this box, the result of the multiplication, 1001000, is written in red.

$$\begin{array}{r} 1000 \\ \times 1001 \\ \hline \end{array}$$

1001000

Hardware Implementation

- ❖ 6, All bit of multiplier checked?
 - ❖ No, repeat step 4
 - ❖ Yes, done!

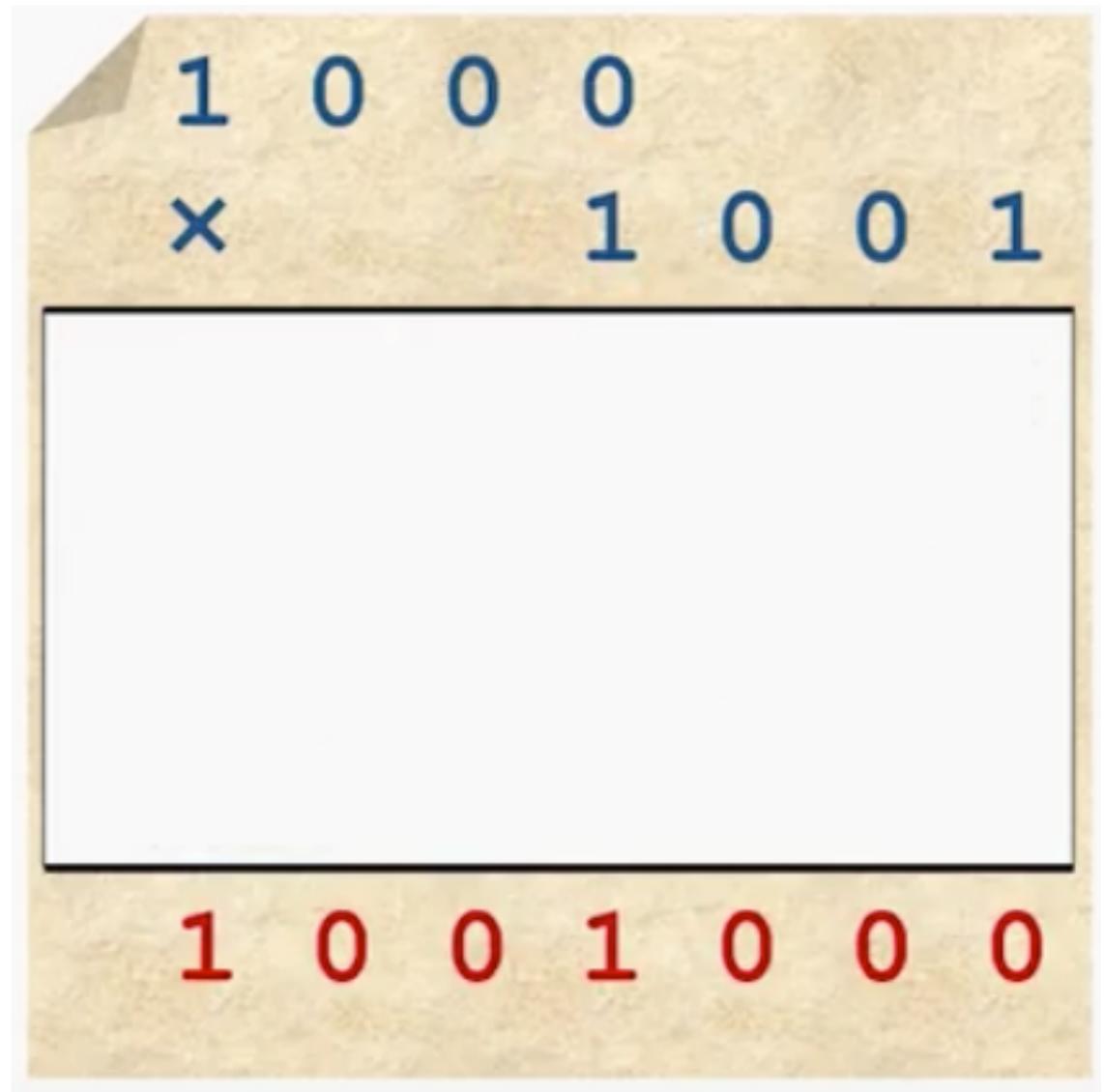
A binary multiplication diagram. At the top, the number 1000 is written above a multiplication sign (×). To the right of the multiplication sign is the number 1001. A large empty rectangular box is positioned below the numbers, intended for the result. Below this box, the product 1001000 is written in red.

$$\begin{array}{r} 1000 \\ \times 1001 \\ \hline \end{array}$$

1001000

Hardware Implementation

- ❖ 6, All bit of multiplier checked?
 - ❖ No, repeat step 4
 - ❖ Yes, done!
- ❖ The proposed method does not need extra space



Hardware Implementation

- ❖ 6, All bit of multiplier checked?
 - ❖ No, repeat step 4
 - ❖ Yes, done!
- ❖ The proposed method does not need extra space
- ❖ Suitable for hardware implementation

$$\begin{array}{r} 1000 \\ \times \quad \quad \quad 1001 \\ \hline \end{array}$$

1 0 0 1 0 0 0

Circuit Implementation

$$\begin{array}{r} 1 0 0 0_{\text{two}} \\ \times 1 0 0 1_{\text{two}} \\ \hline \end{array}$$

Circuit Implementation

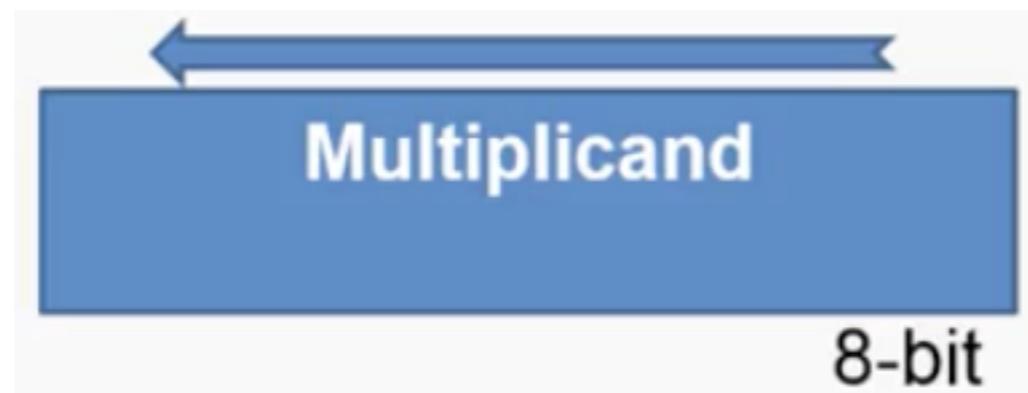
- ❖ 8-bit register for multiplicand

$$\begin{array}{r} 1 0 0 0 \\ \times 1 0 0 1 \\ \hline \end{array}_{\text{two}}$$

Circuit Implementation

- ❖ 8-bit register for multiplicand

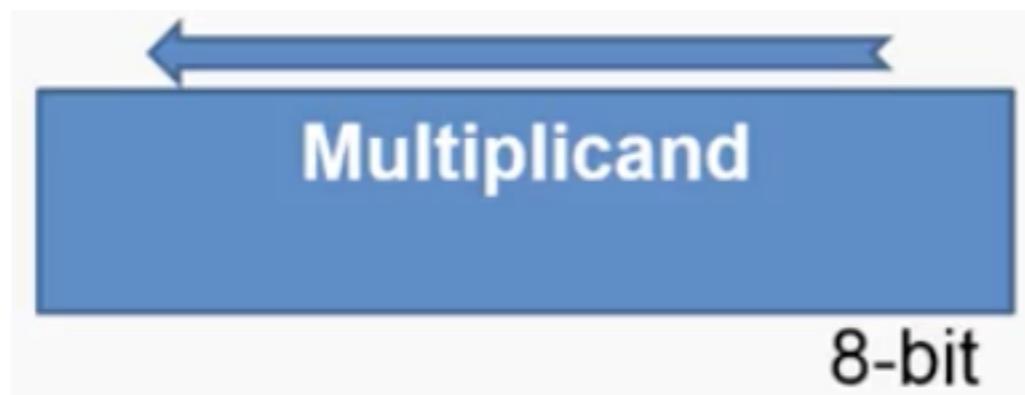
$$\begin{array}{r} 1 0 0 0_2 \\ \times 1 0 0 1_2 \\ \hline \end{array}$$



Circuit Implementation

- ❖ 8-bit register for multiplicand
 - ❖ With left shift function

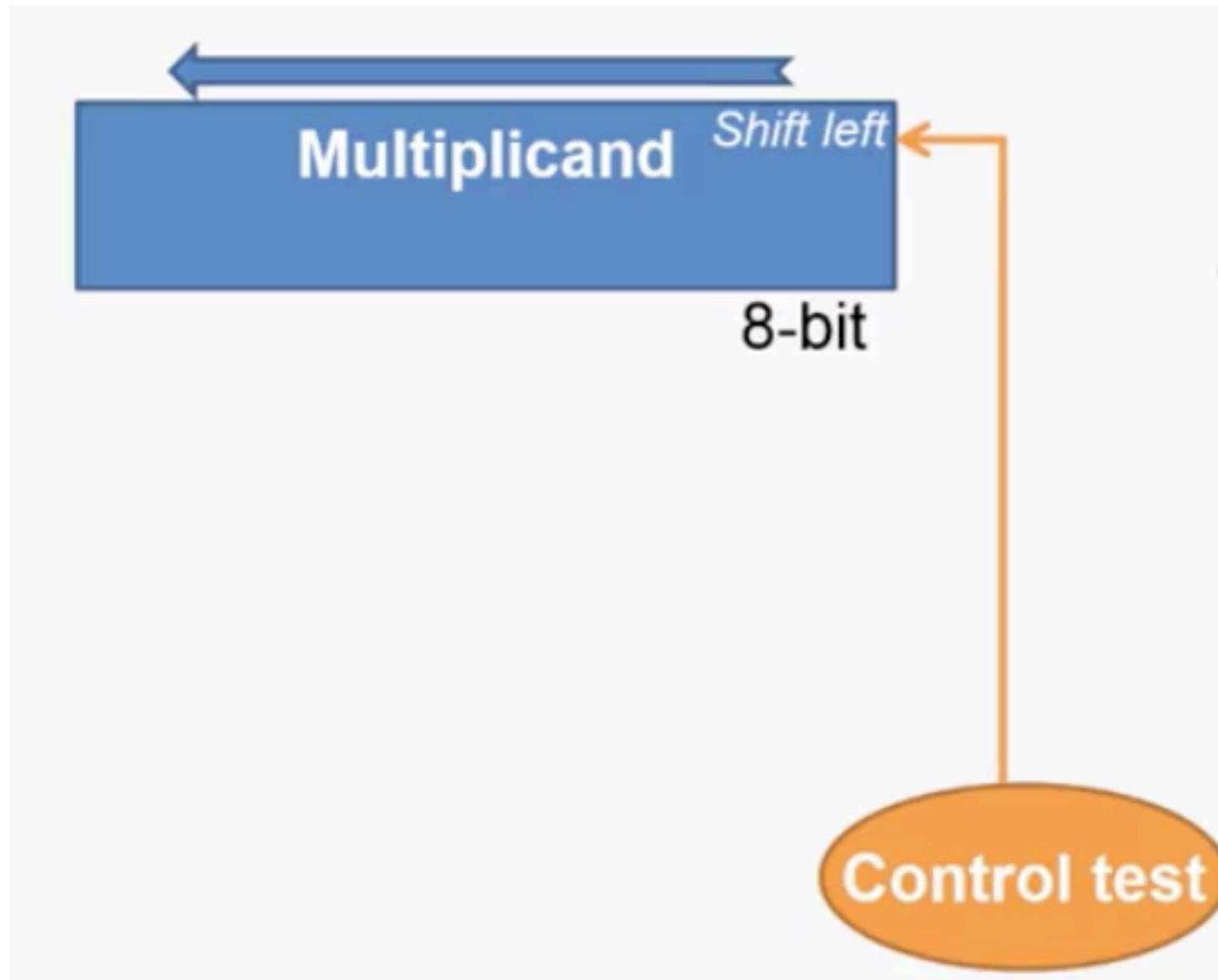
$$\begin{array}{r} 1 0 0 0_2 \\ \times 1 0 0 1_2 \\ \hline \end{array}$$



Circuit Implementation

- ❖ 8-bit register for multiplicand
 - ❖ With left shift function

$$\begin{array}{r} 1 0 0 0_2 \\ \times 1 0 0 1_2 \\ \hline \end{array}$$



Circuit Implementation

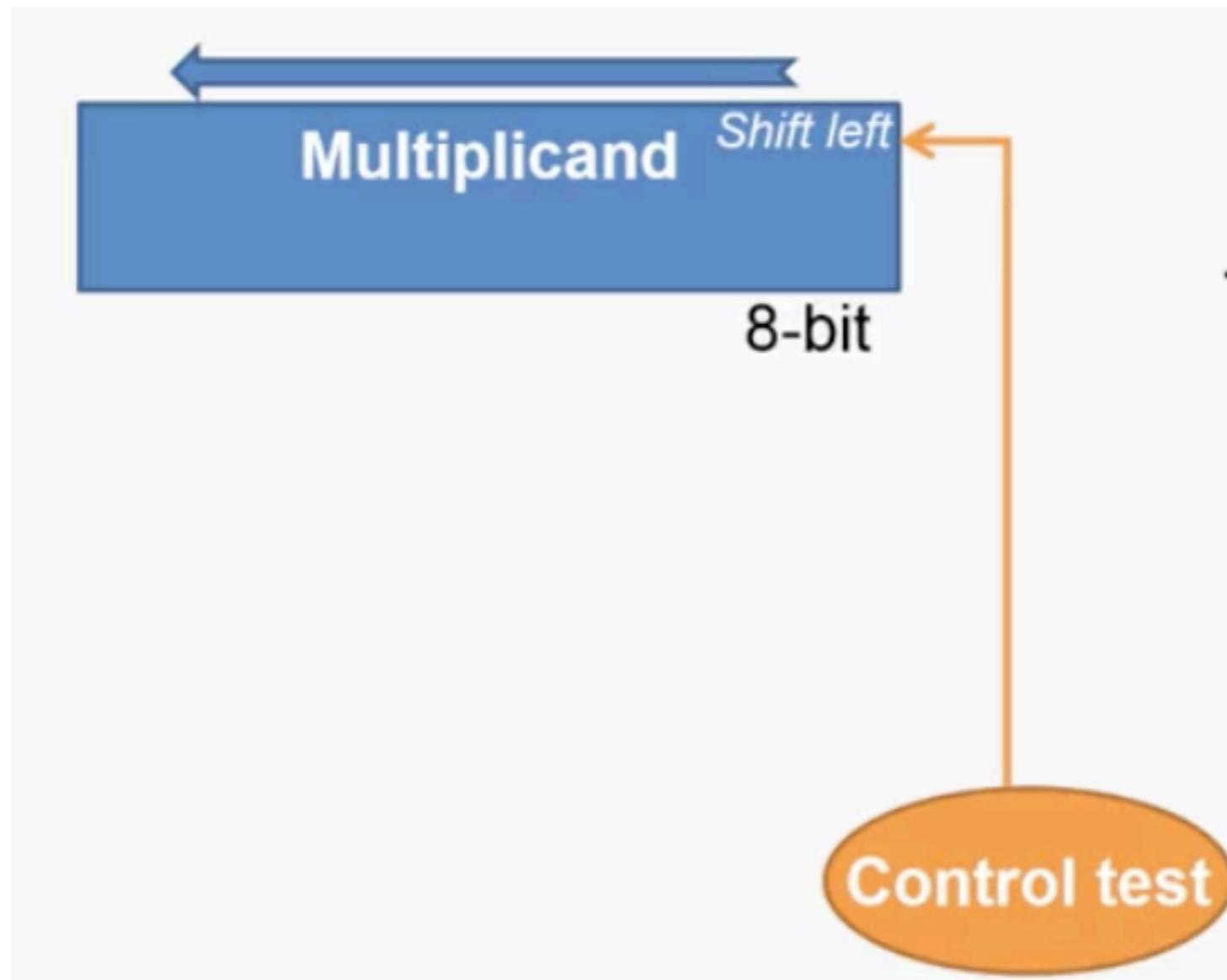
- ❖ 8-bit register for multiplicand
 - ❖ With left shift function

$$\begin{array}{r} 1 0 0 0_2 \\ \times 1 0 0 1_2 \\ \hline \end{array}$$

Circuit Implementation

- ❖ 8-bit register for multiplicand
 - ❖ With left shift function

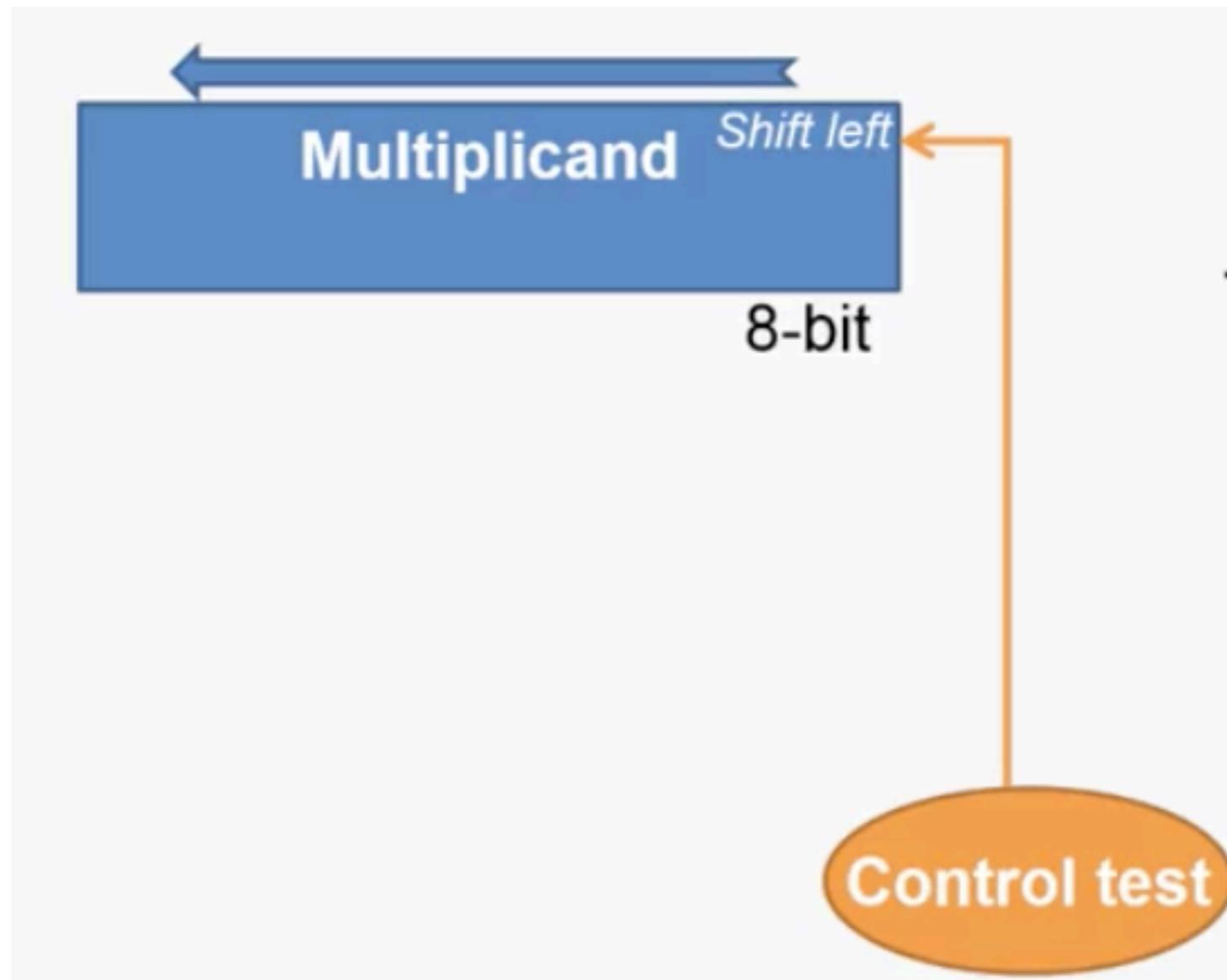
$$\begin{array}{r} 1 \ 0 \ 0 \ 0_{\text{two}} \\ \times 1 \ 0 \ 0 \ 1_{\text{two}} \\ \hline \end{array}$$



Circuit Implementation

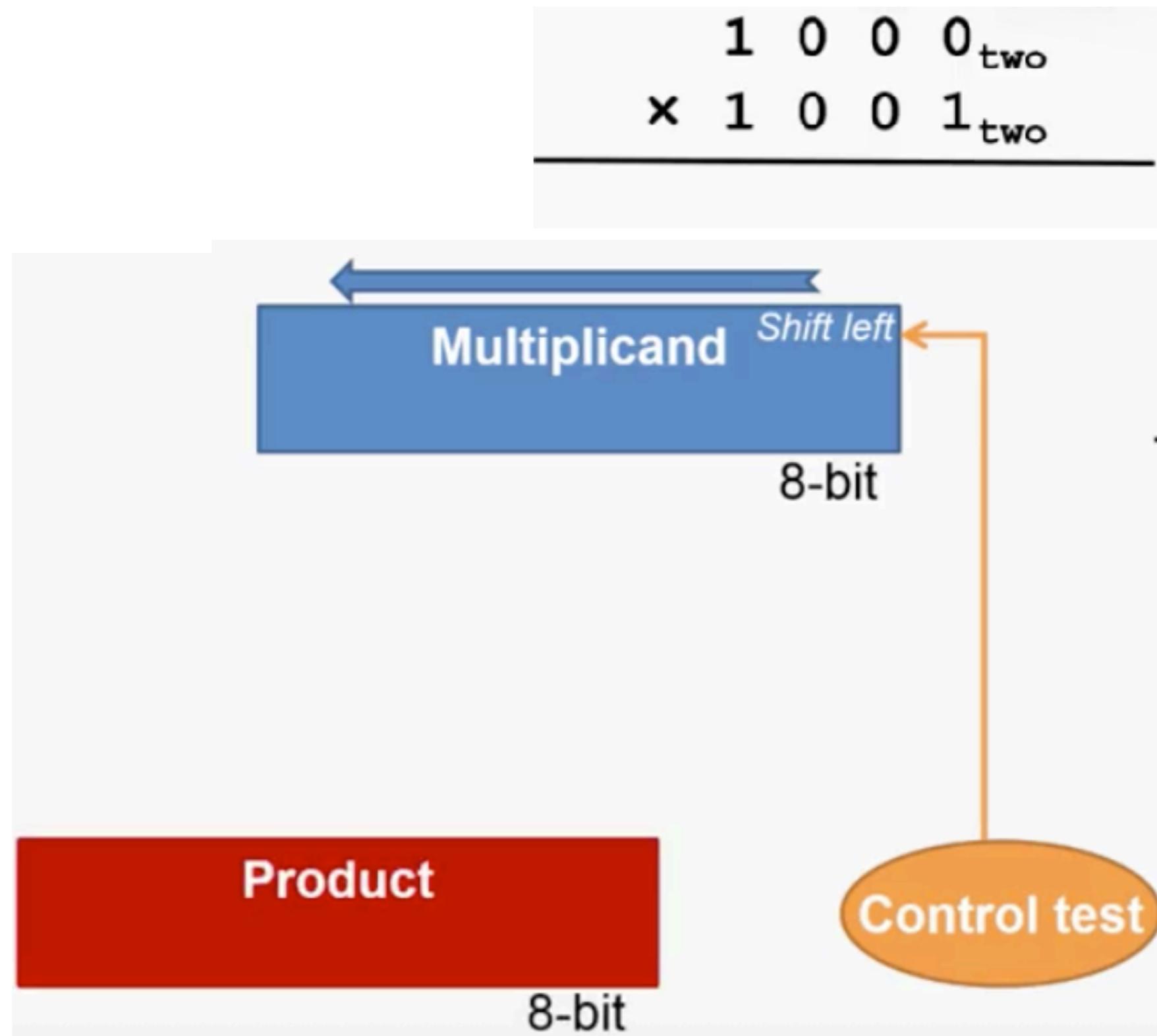
- ❖ 8-bit register for multiplicand
 - ❖ With left shift function
- ❖ 8-bit register for product

$$\begin{array}{r} 1 \ 0 \ 0 \ 0_{\text{two}} \\ \times 1 \ 0 \ 0 \ 1_{\text{two}} \\ \hline \end{array}$$



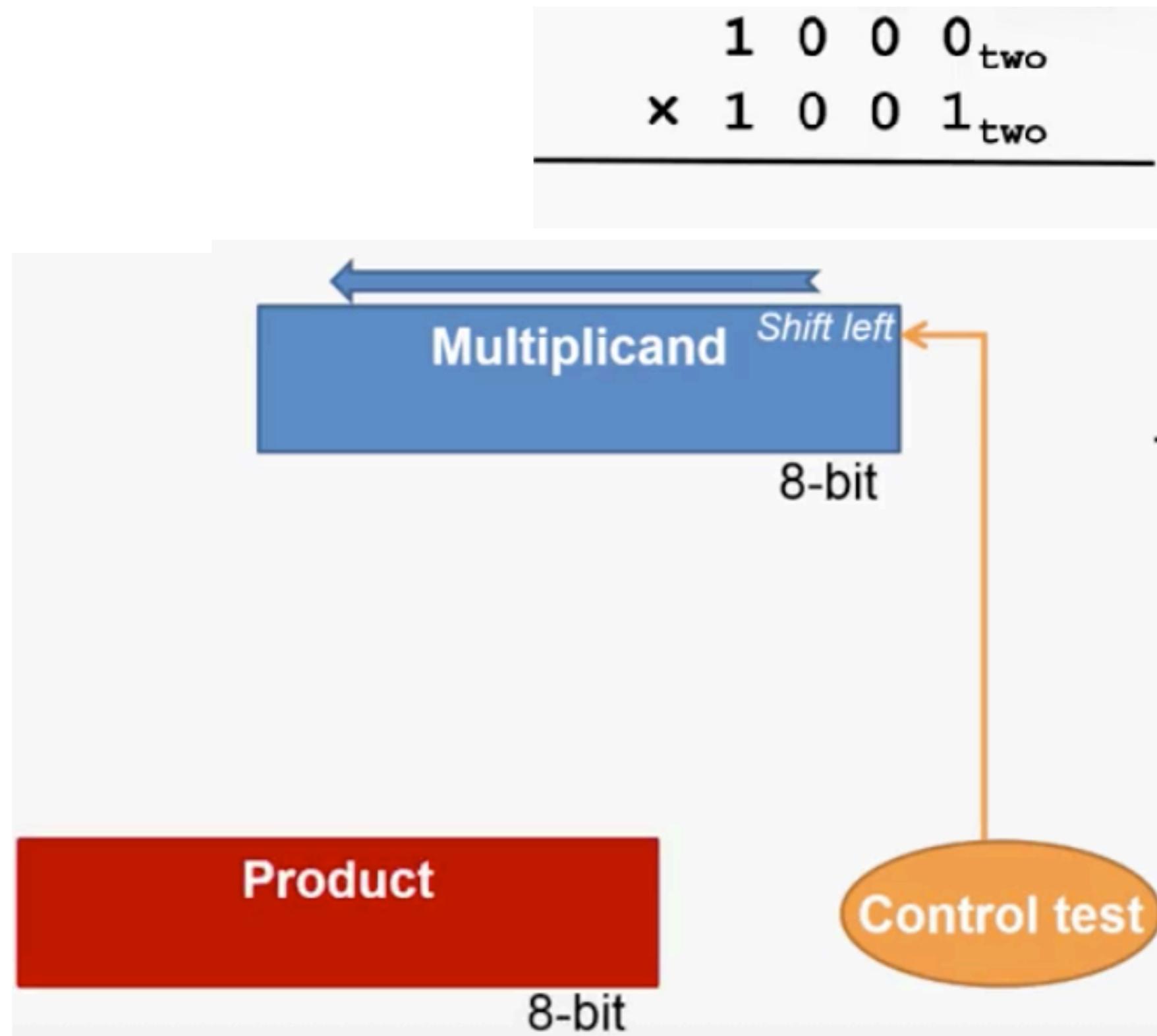
Circuit Implementation

- ❖ 8-bit register for multiplicand
 - ❖ With left shift function
- ❖ 8-bit register for product



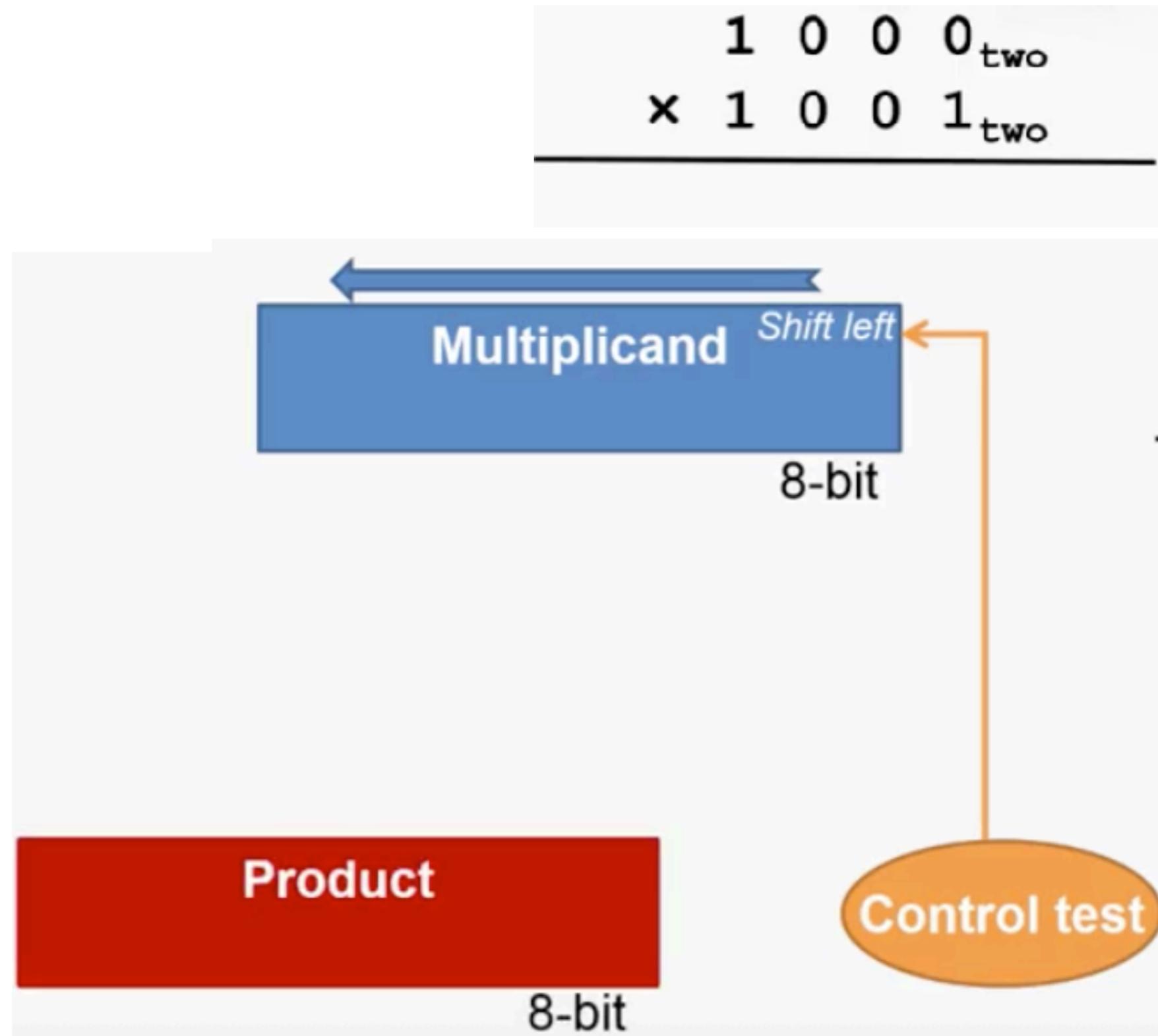
Circuit Implementation

- ❖ 8-bit register for multiplicand
 - ❖ With left shift function
- ❖ 8-bit register for product
- ❖ What else?



Circuit Implementation

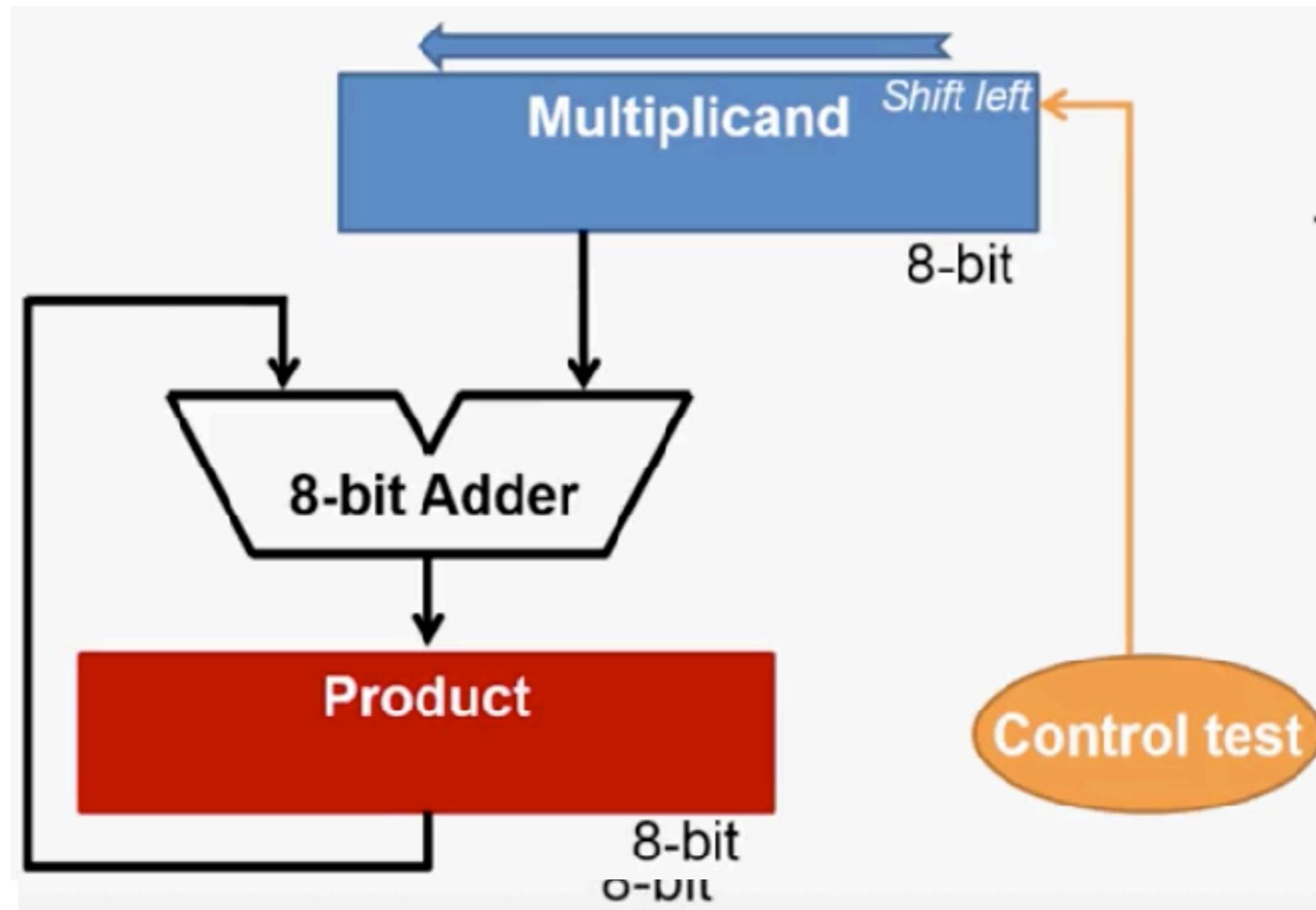
- ❖ 8-bit register for multiplicand
 - ❖ With left shift function
- ❖ 8-bit register for product
- ❖ What else?
- ❖ 8-bit Adder



Circuit Implementation

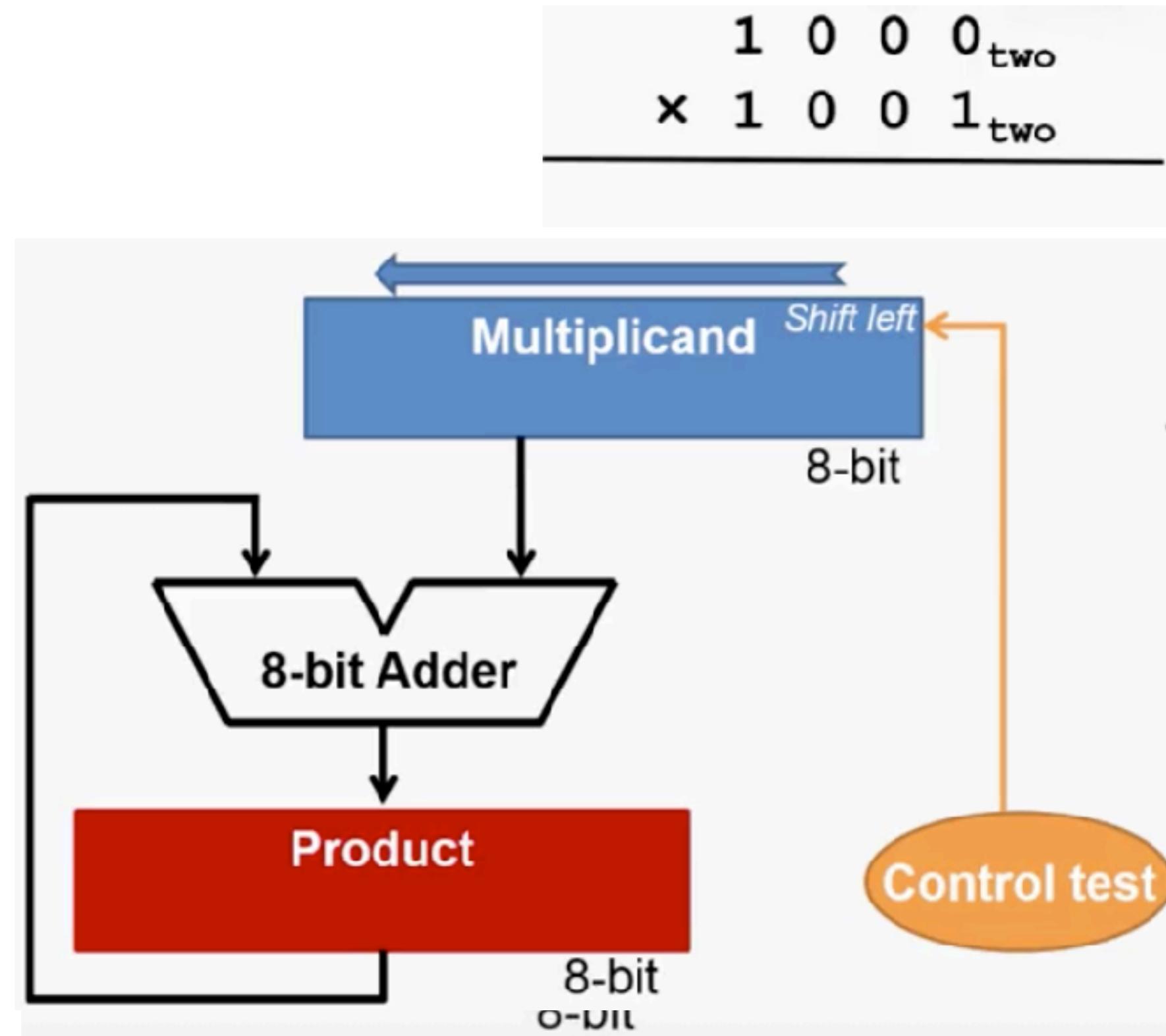
- ❖ 8-bit register for multiplicand
 - ❖ With left shift function
- ❖ 8-bit register for product
- ❖ What else?
- ❖ 8-bit Adder

$$\begin{array}{r} 10000_{\text{two}} \\ \times 1001_{\text{two}} \\ \hline \end{array}$$

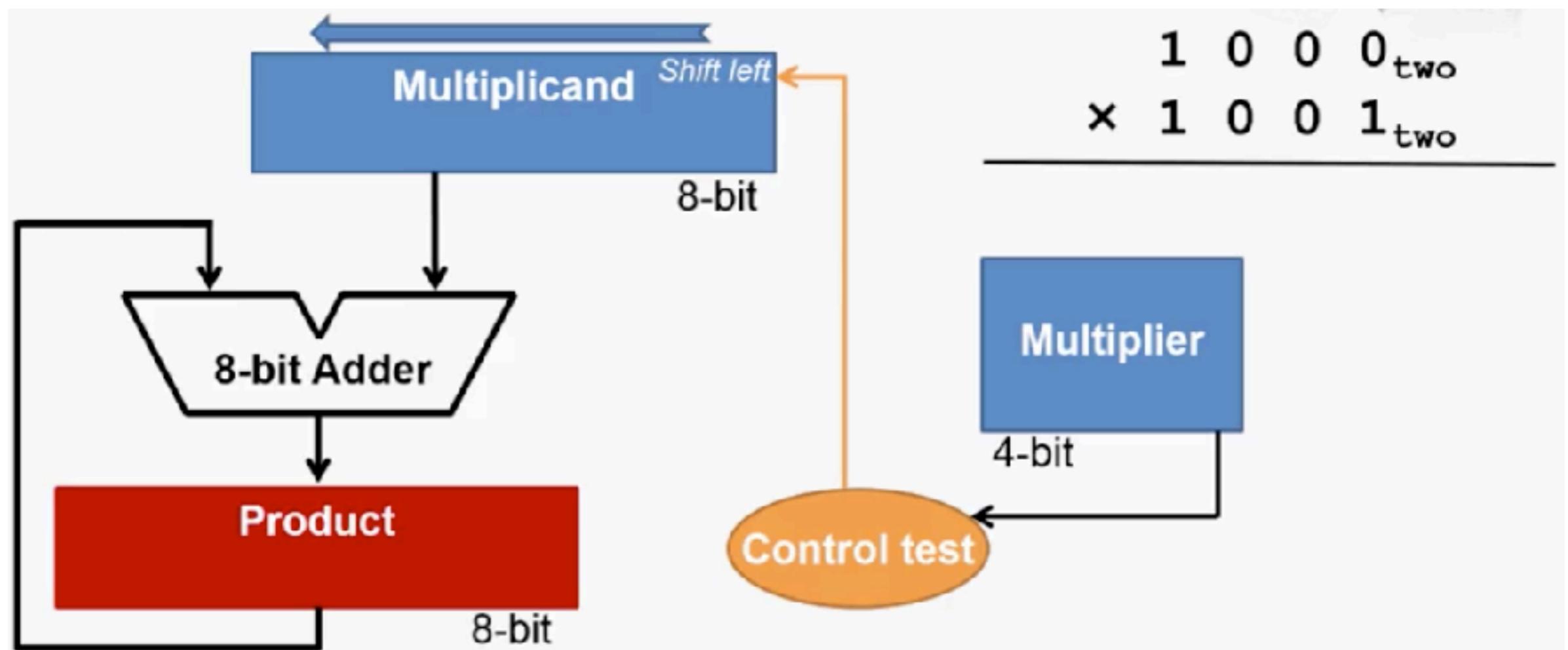


Circuit Implementation

- ❖ 8-bit register for multiplicand
 - ❖ With left shift function
- ❖ 8-bit register for product
- ❖ What else?
- ❖ 8-bit Adder
- ❖ What else?

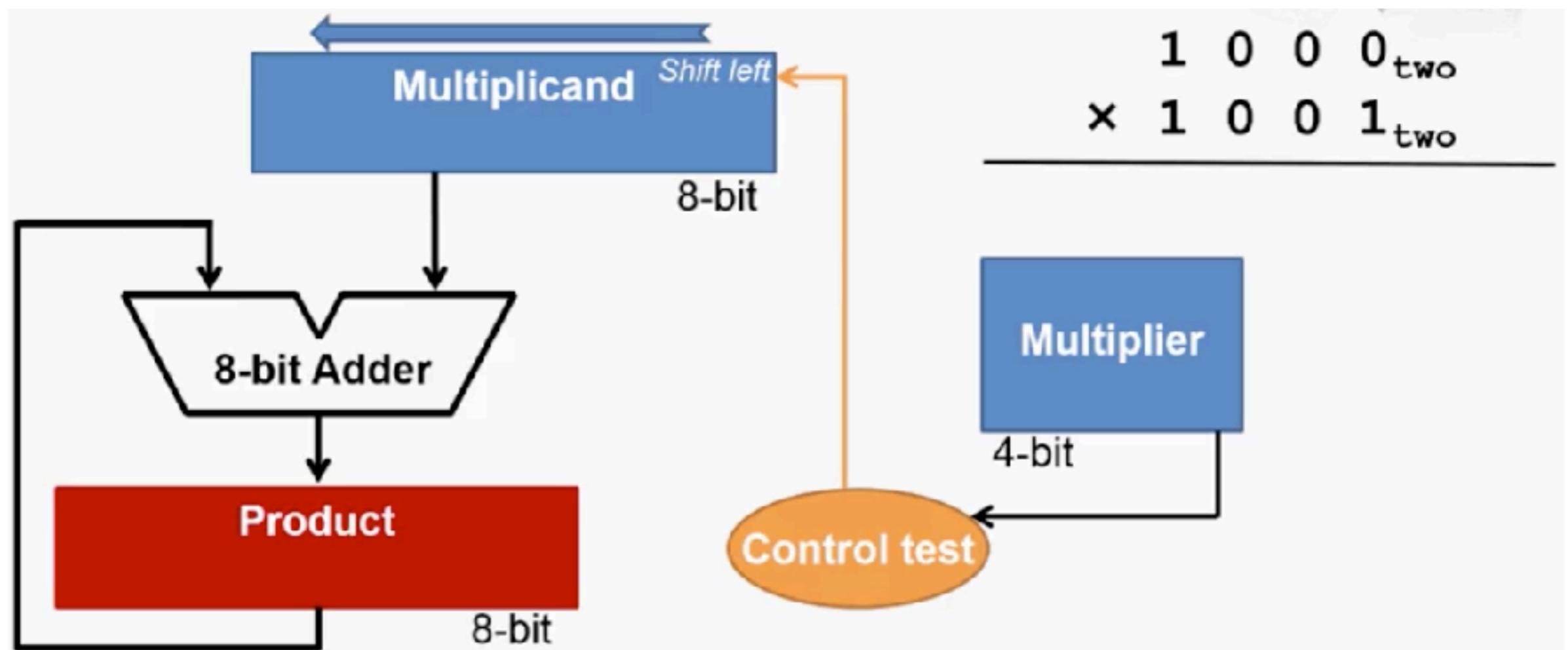


Circuit Implementation



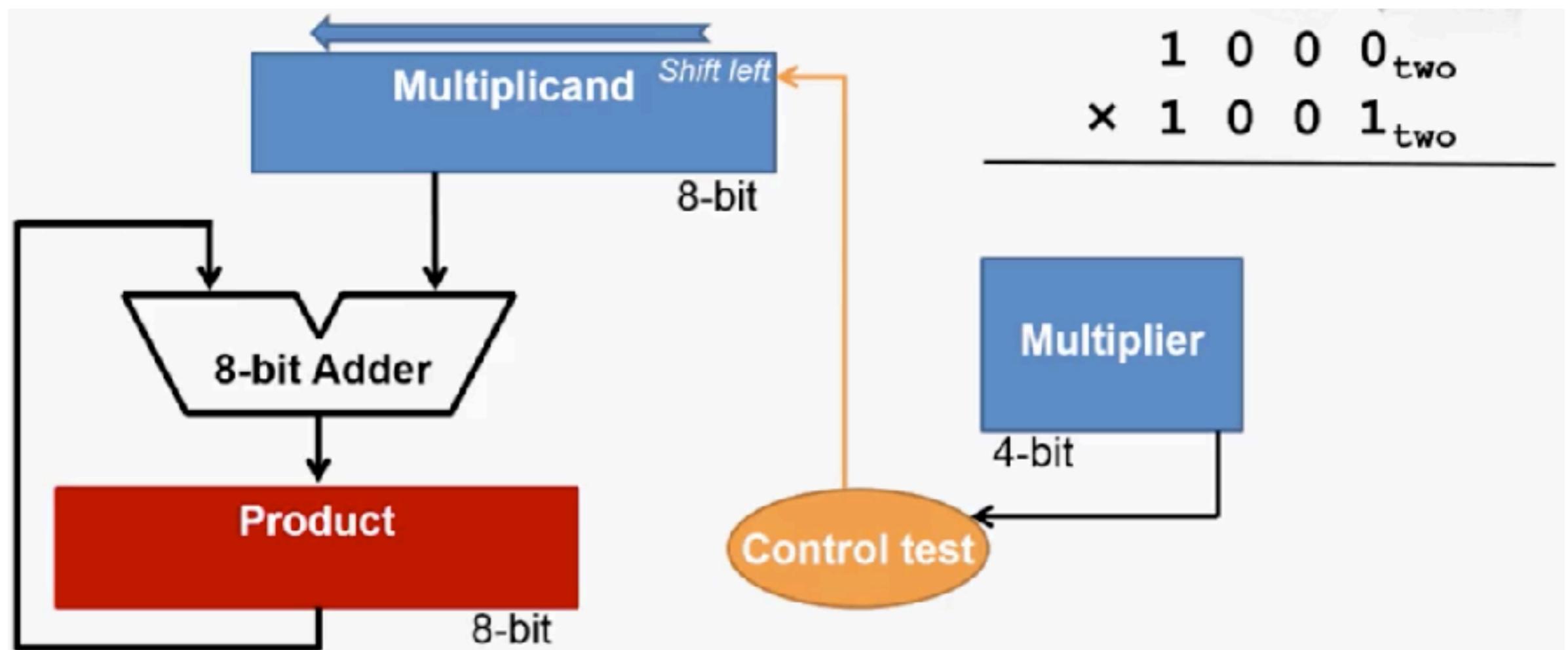
Circuit Implementation

- ❖ 4-bit register for multiplier



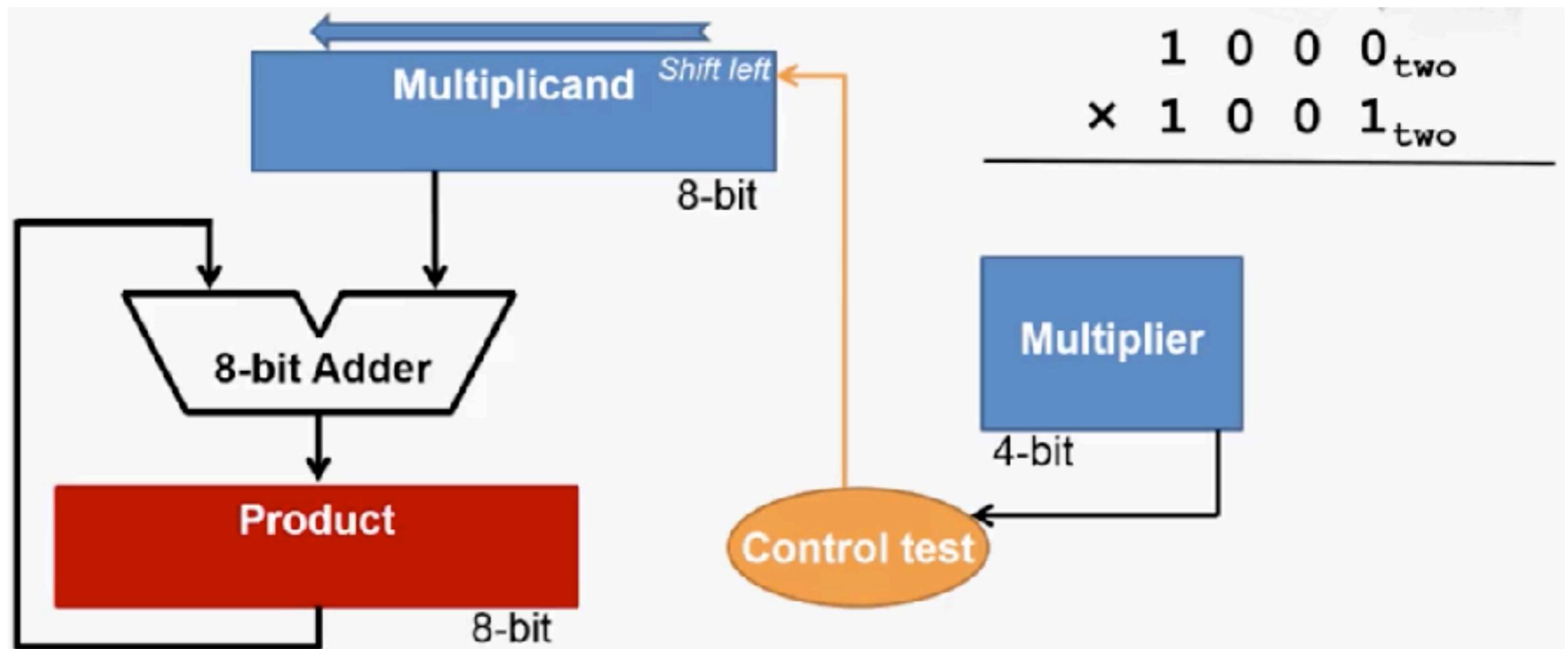
Circuit Implementation

- ❖ 4-bit register for multiplier
 - ❖ What is the data from multiplier to control test?

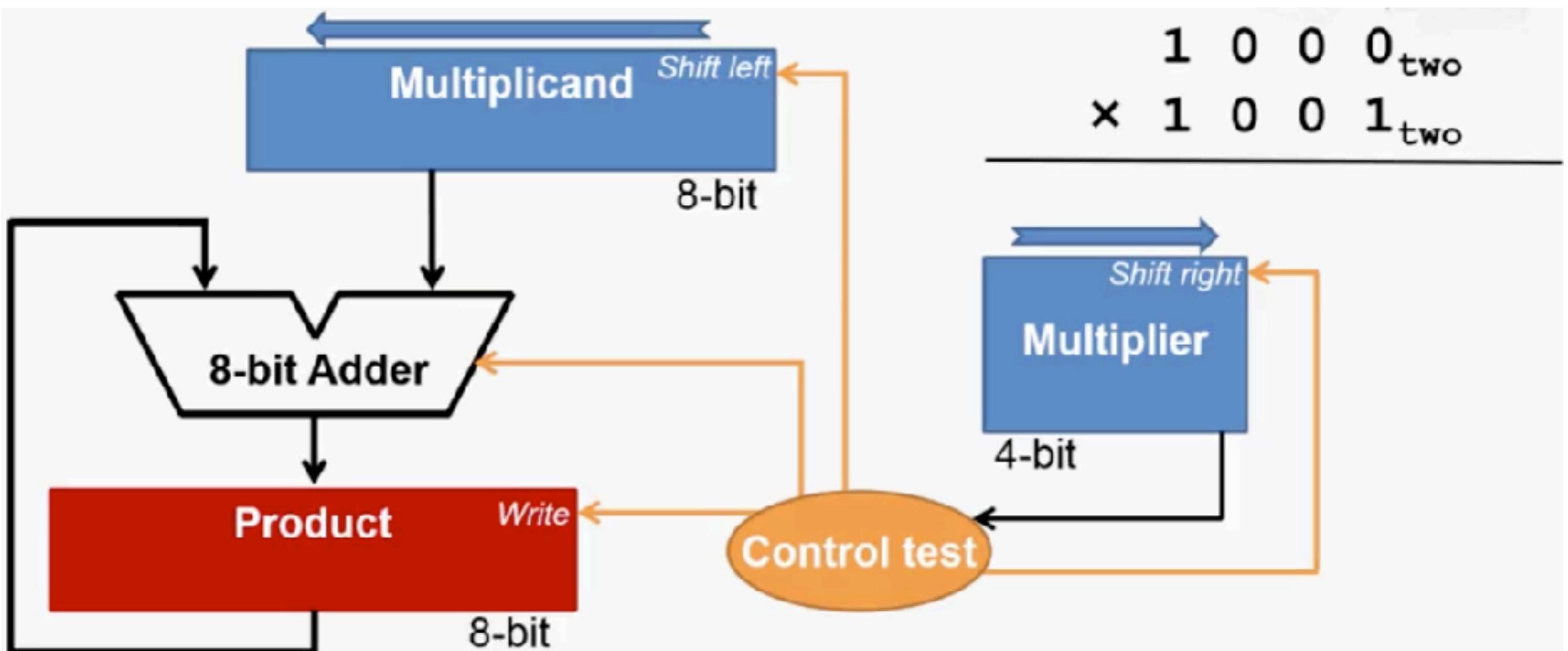


Circuit Implementation

- ❖ 4-bit register for multiplier
 - ❖ What is the data from multiplier to control test?
 - ❖ LSB == 0? Shift to left? Else?

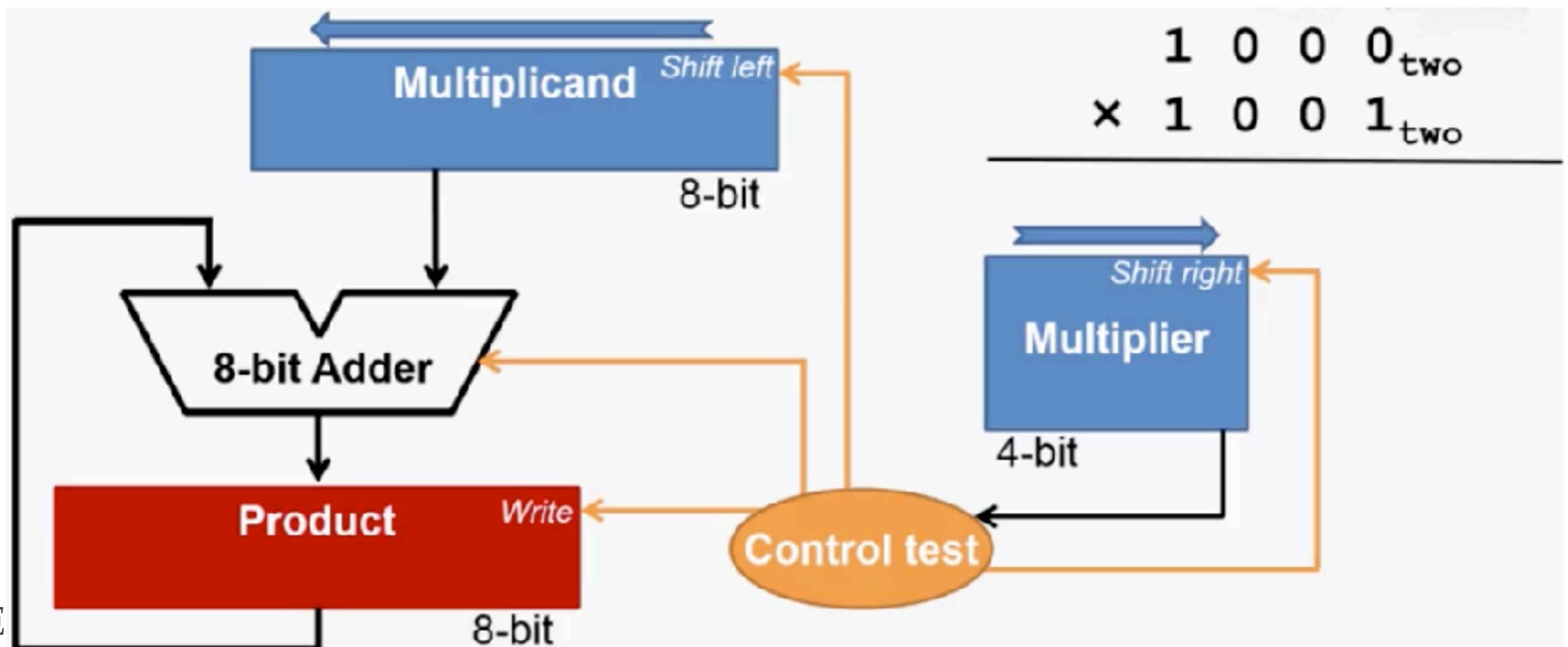


Circuit Implementation



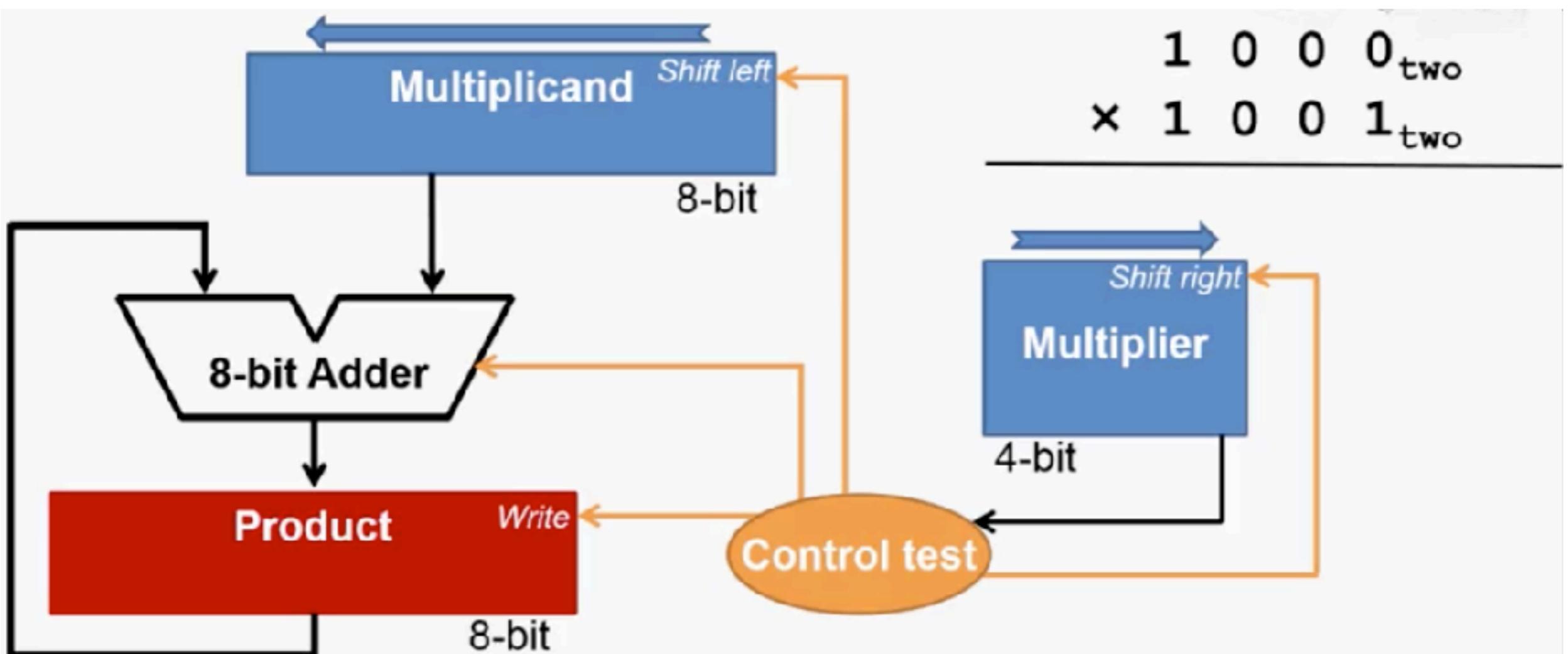
Circuit Implementation

- ❖ Adder executed?



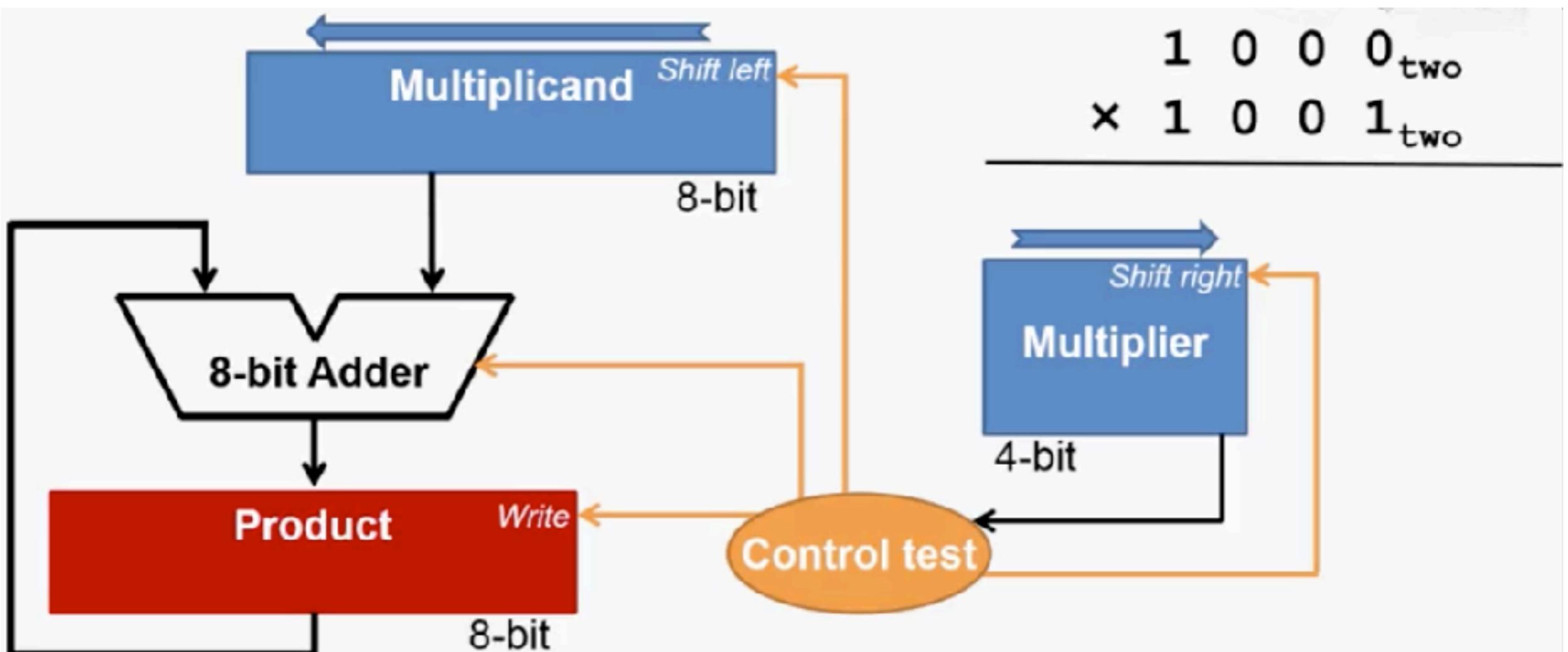
Circuit Implementation

- ❖ Adder executed?
- ❖ Writing operation executed?



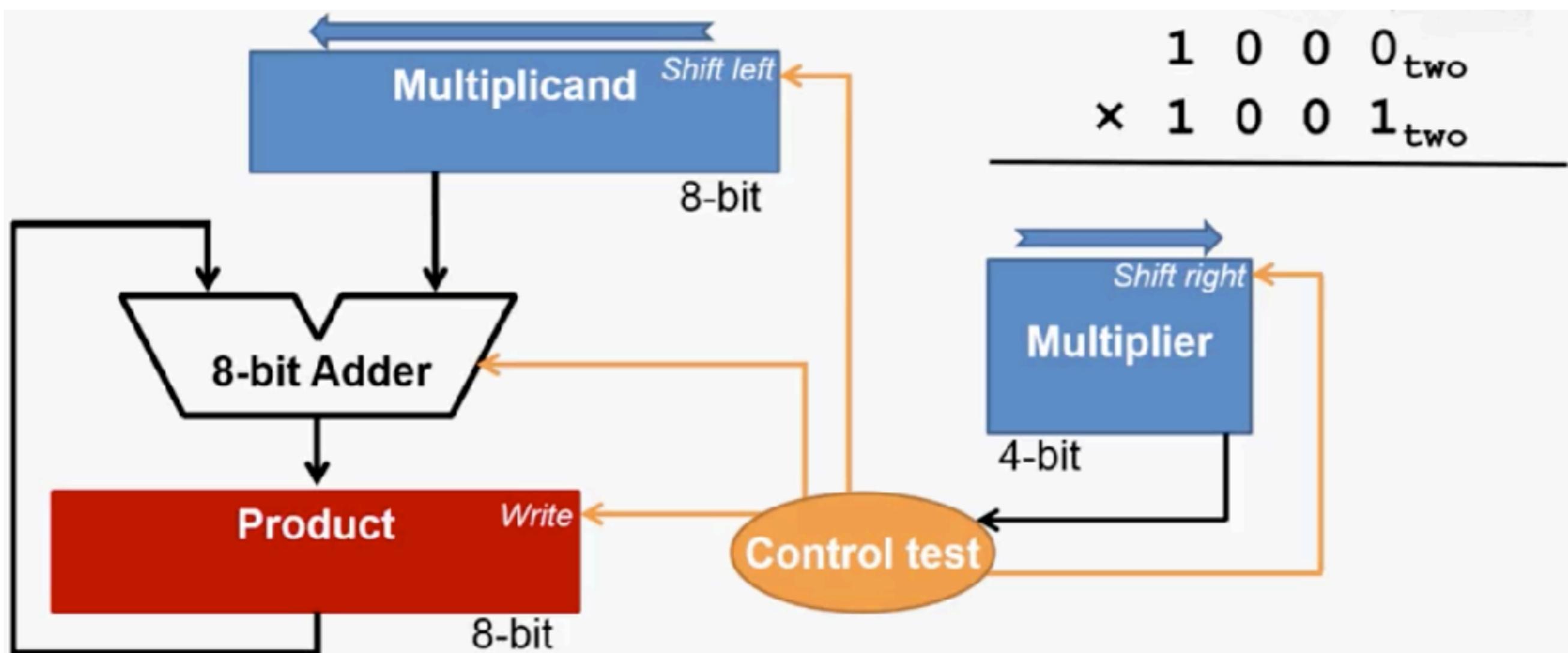
Circuit Implementation

- ❖ Adder executed?
- ❖ Writing operation executed?
- ❖ What else?

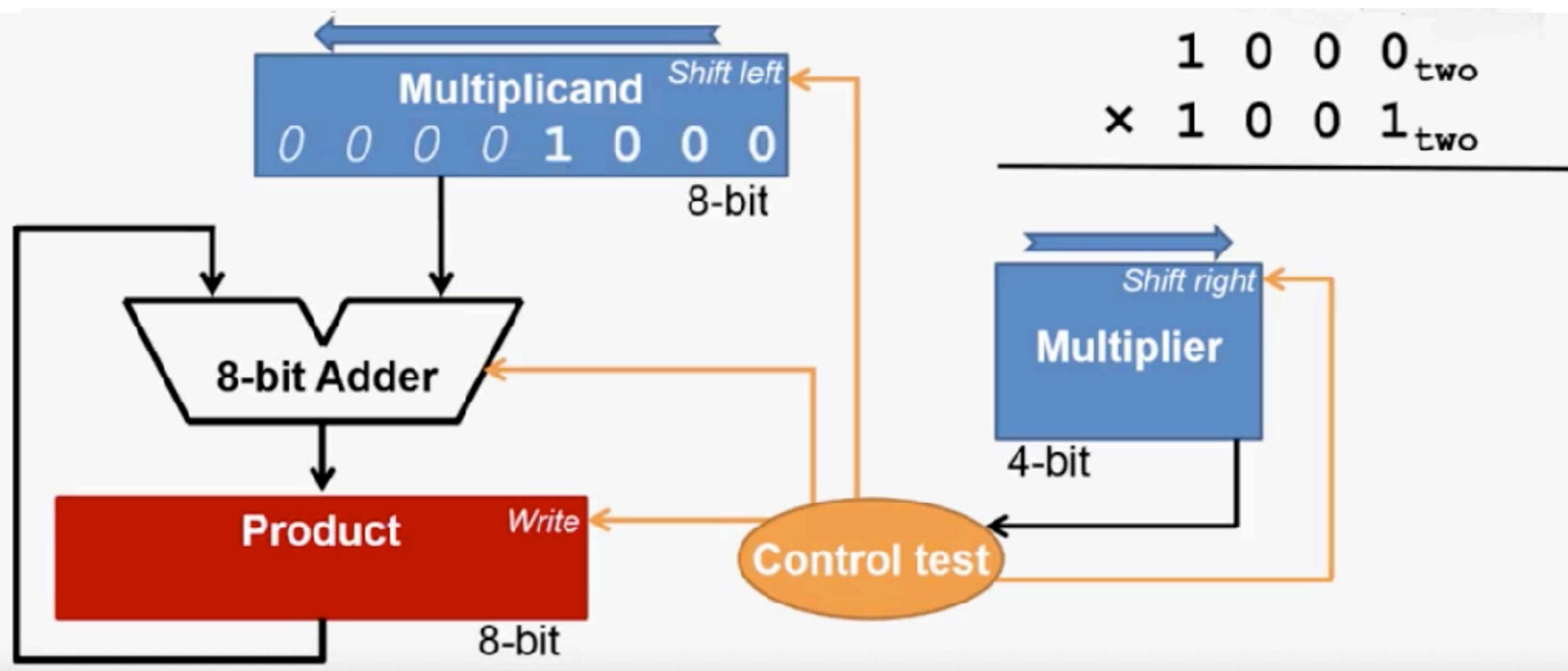


Initialization

Initialization

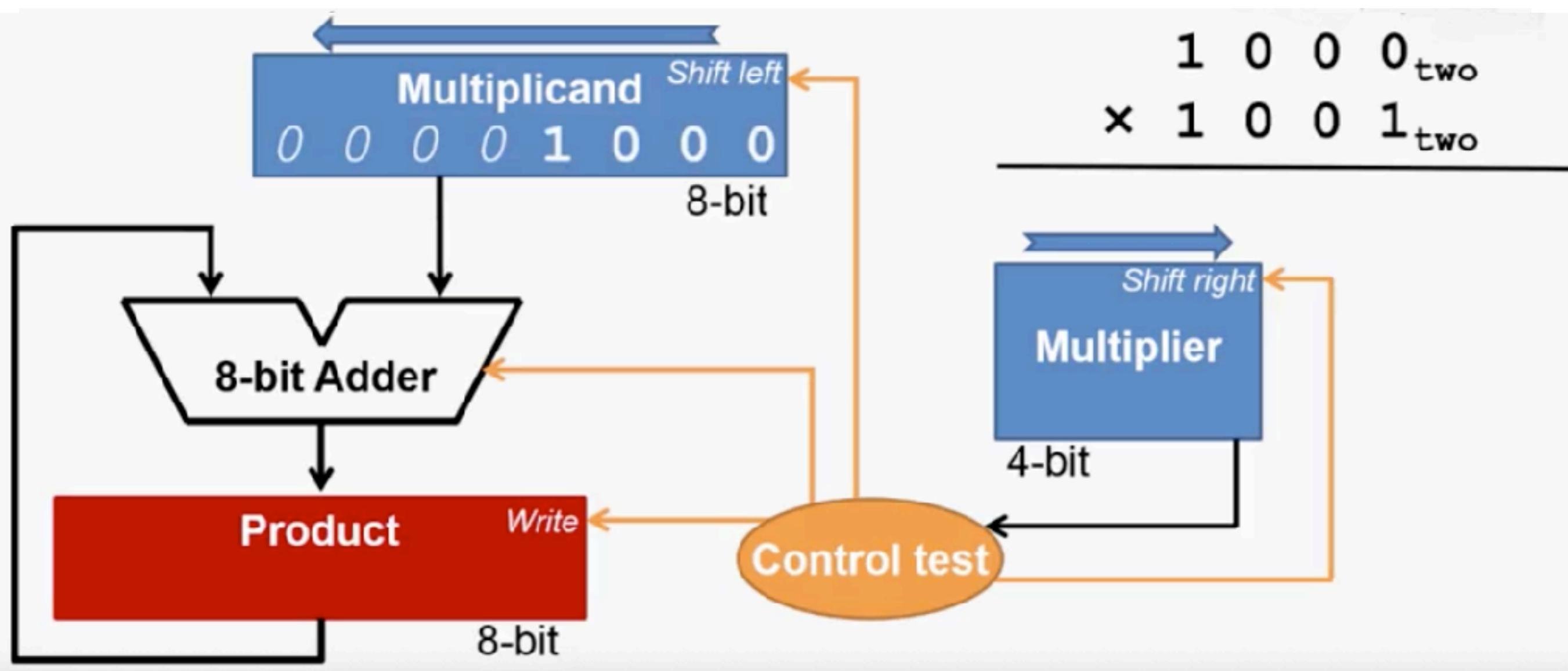


Initialization

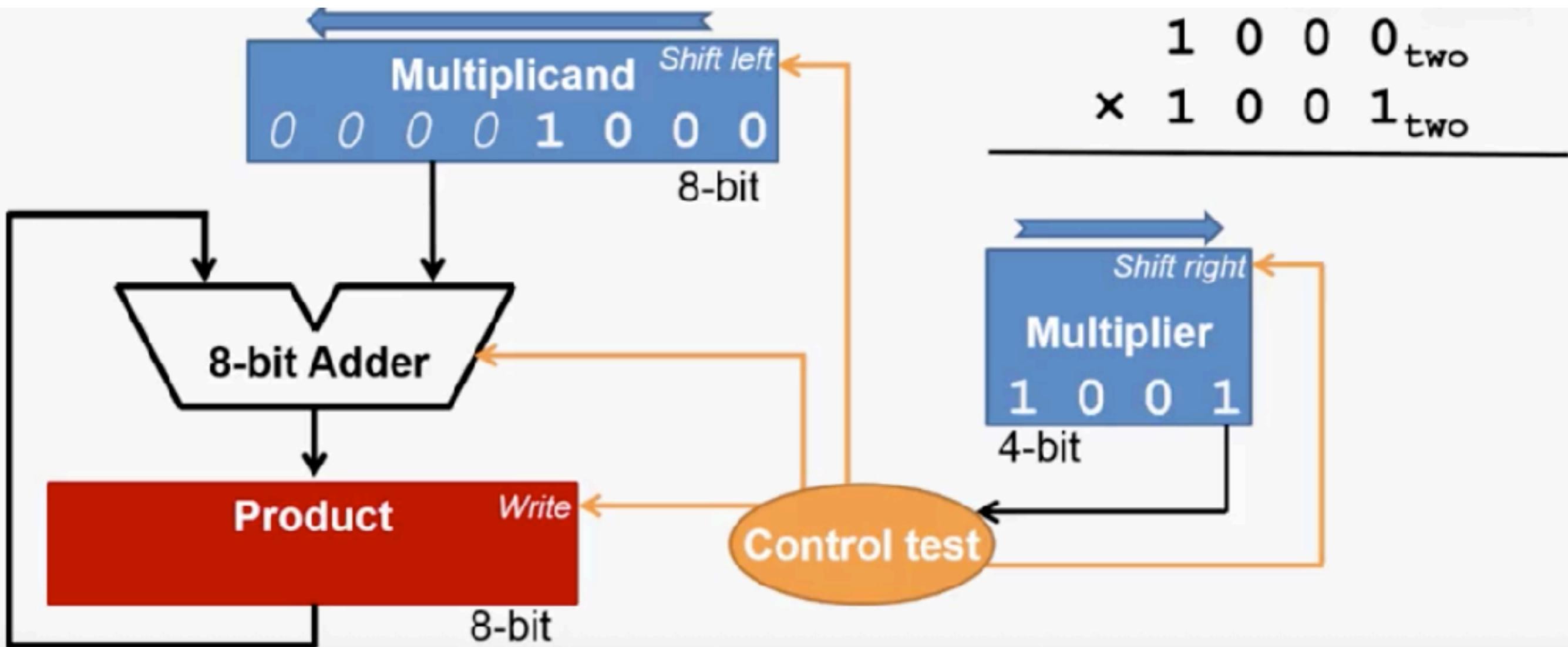


Initialization

- ❖ Load multiplicand (note the difference between 0000 and 1000)

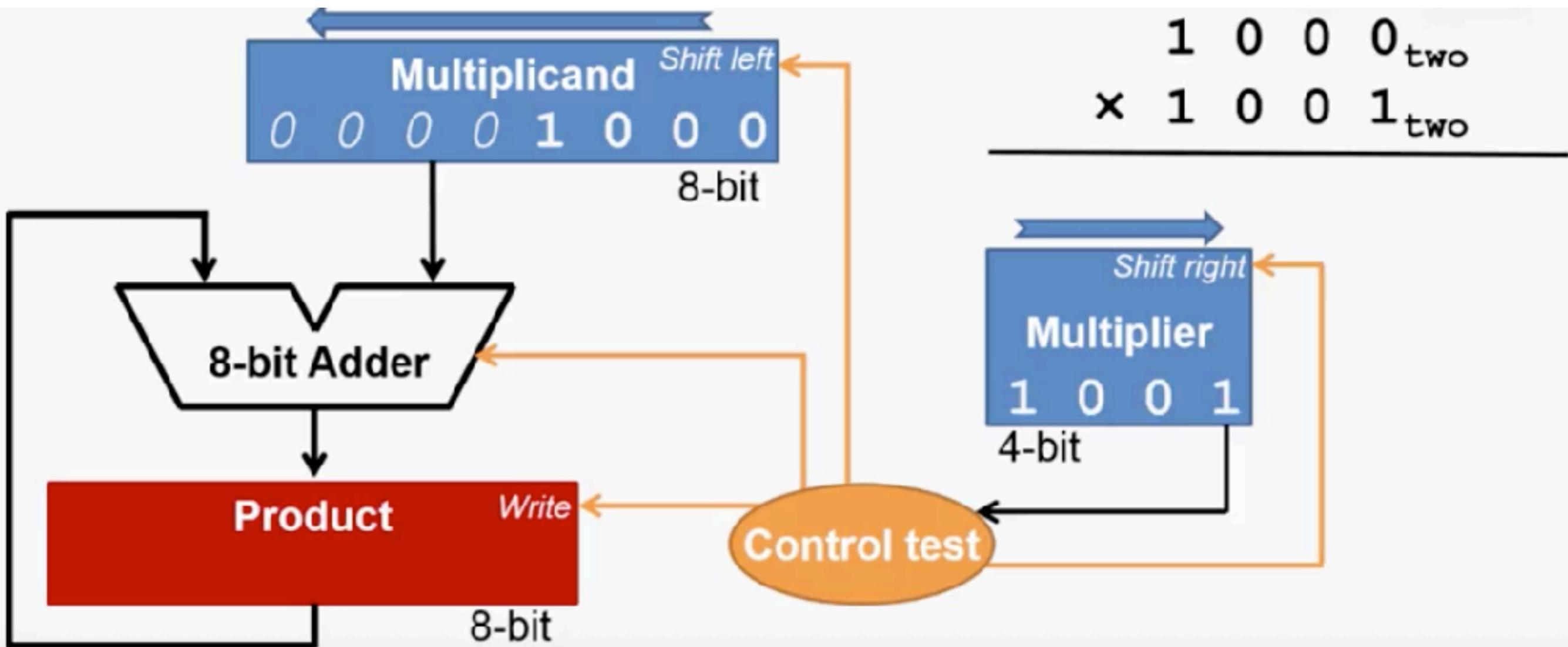


Initialization



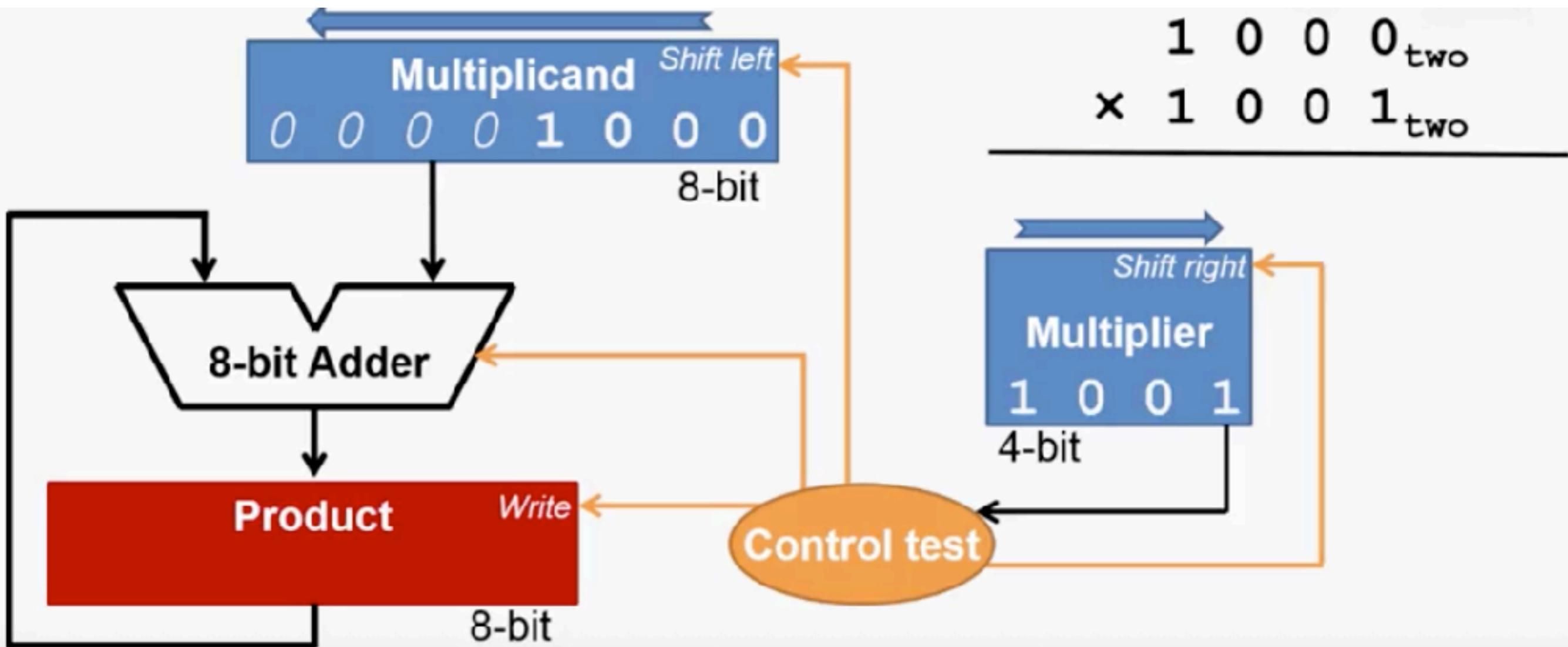
Initialization

- ❖ Load multiplier



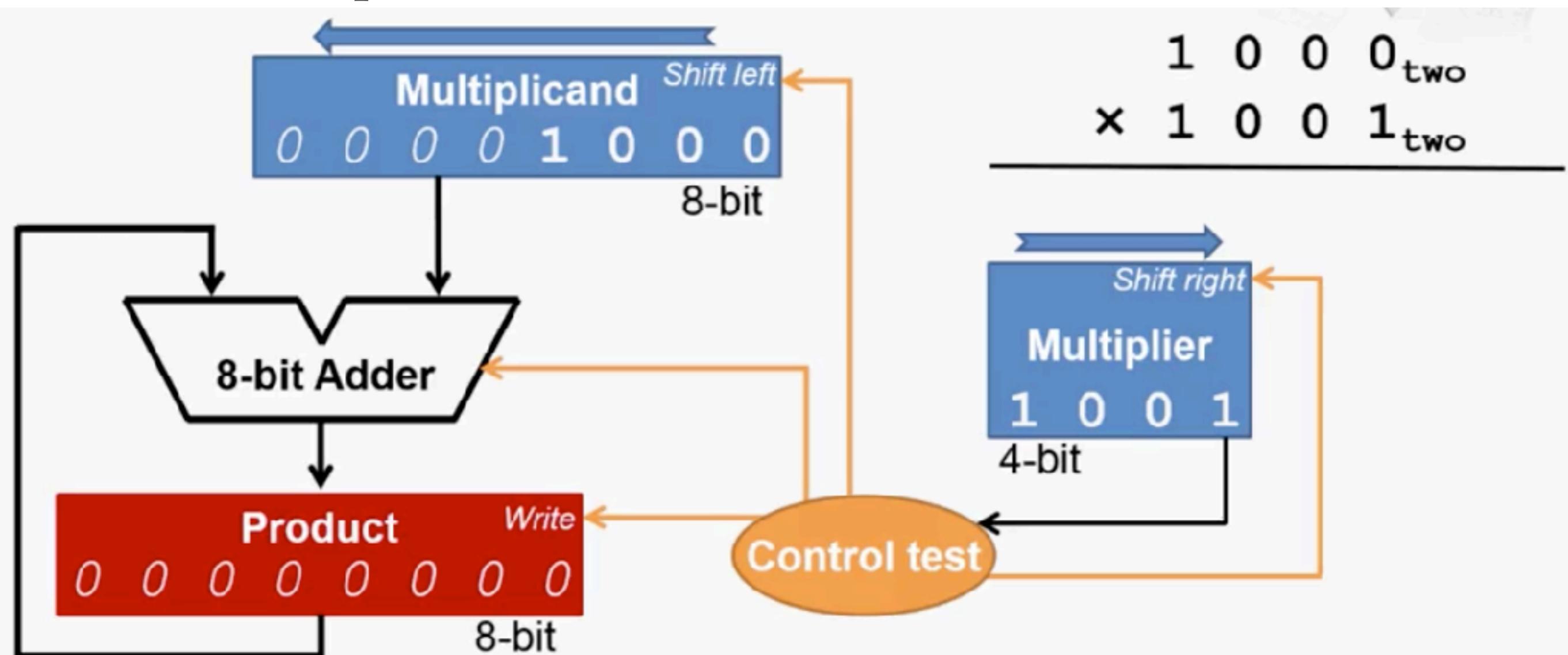
Initialization

- ❖ Load multiplier
- ❖ Initialize product (note the value is *00000000*)



Initialization

- ❖ Load multiplier
- ❖ Initialize product (note the value is *00000000*)

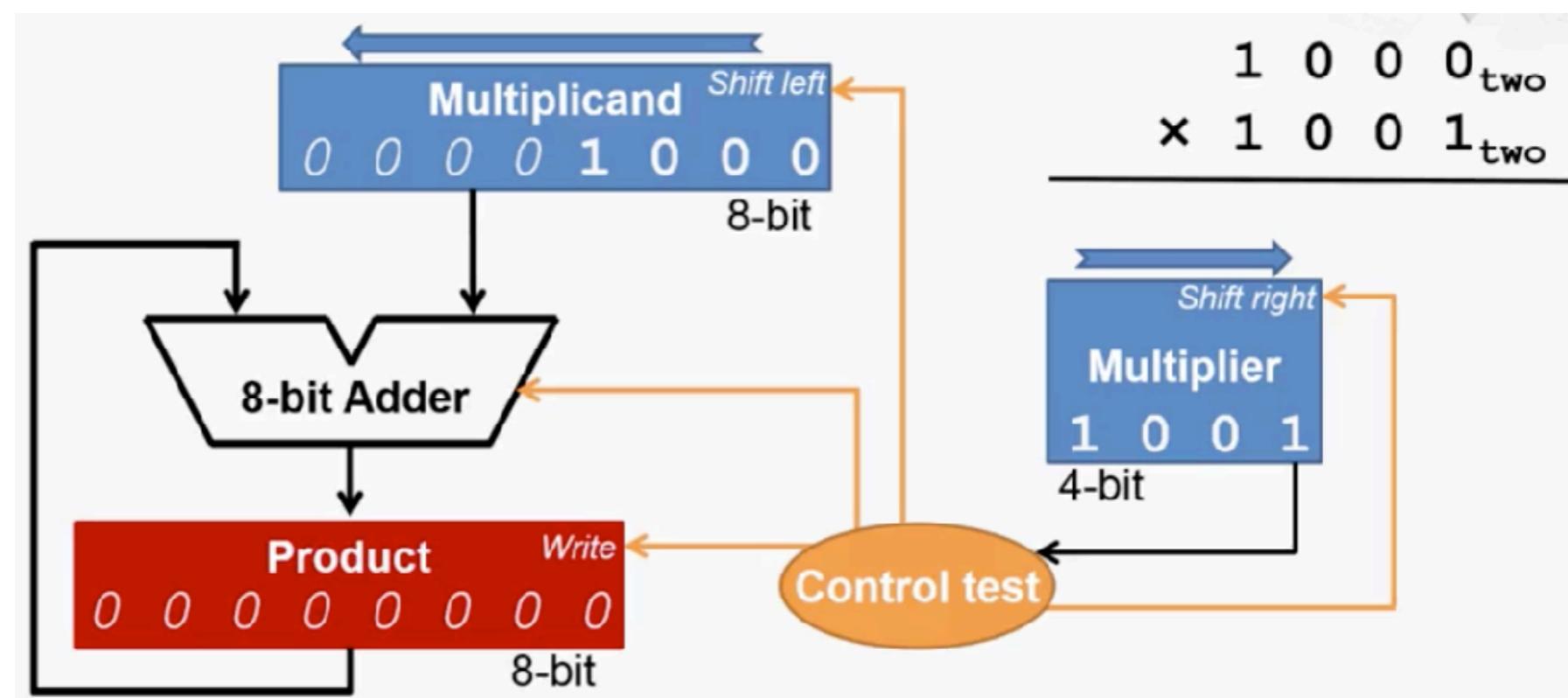


Step 1

- ❖ Recall the operation with paper and pencil

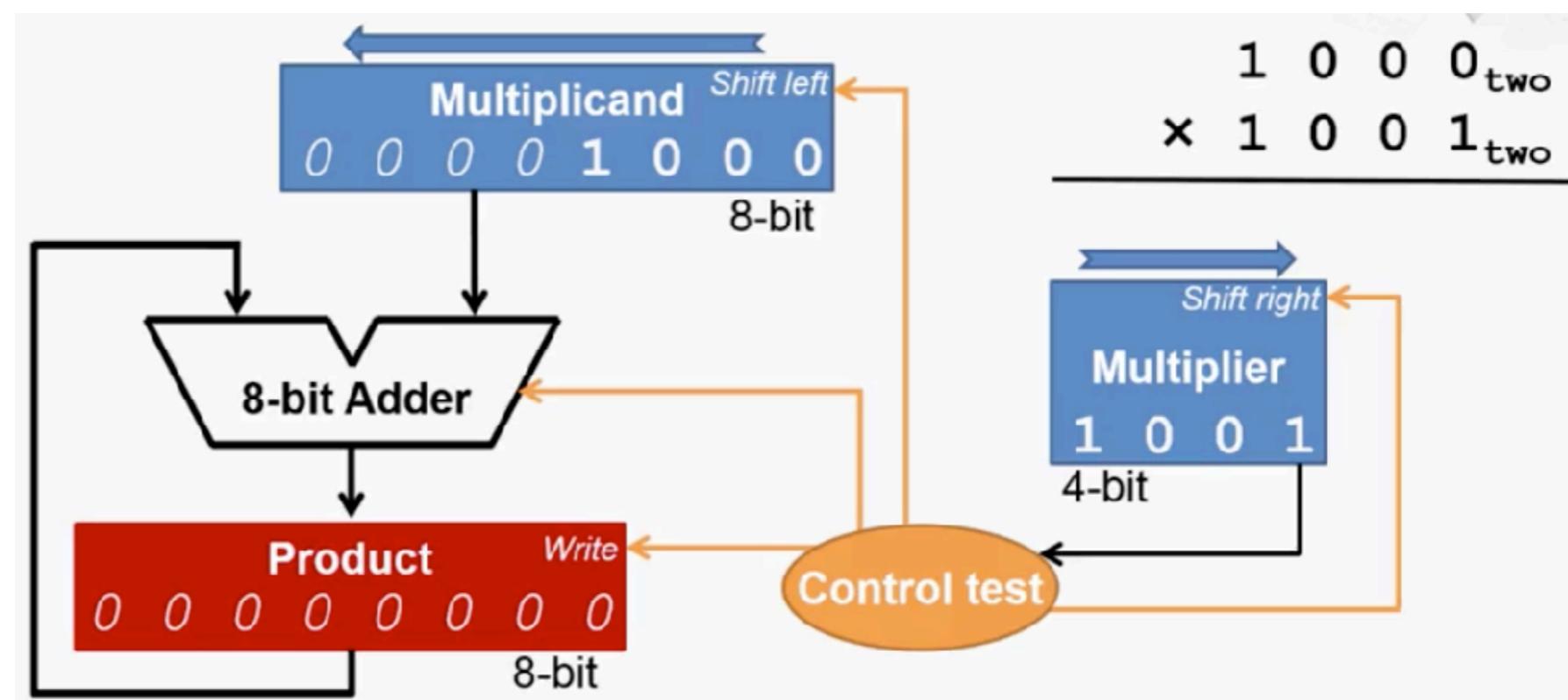
Step 1

- ❖ Recall the operation with paper and pencil



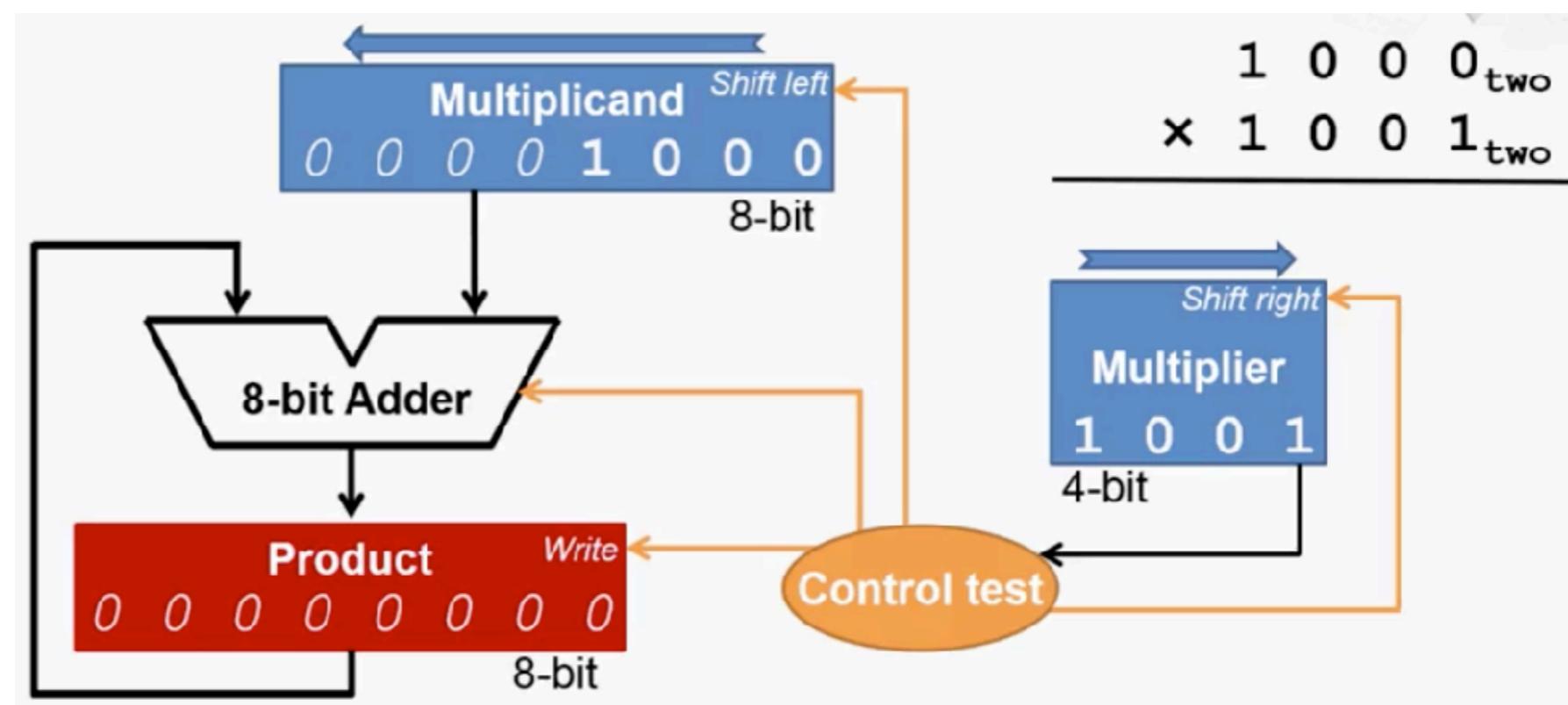
Step 1

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?



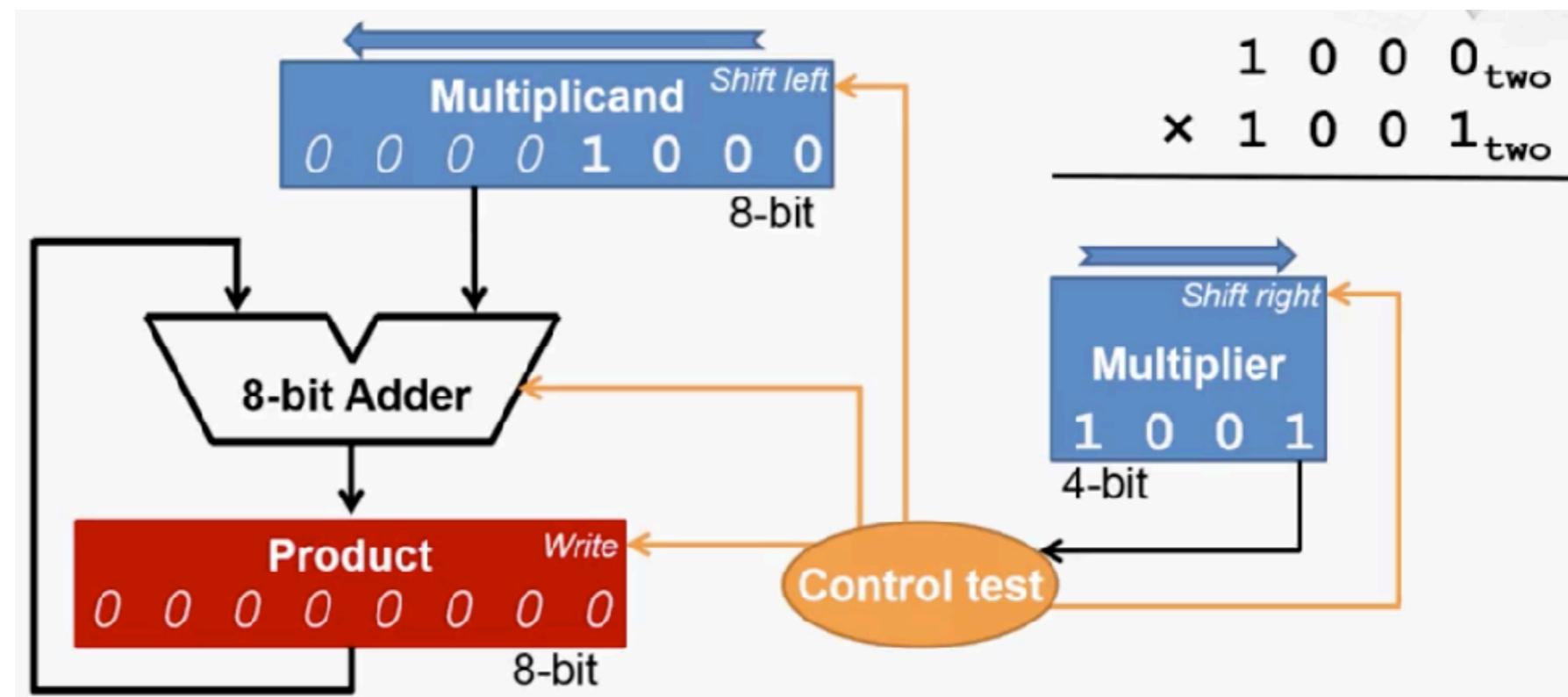
Step 1

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier



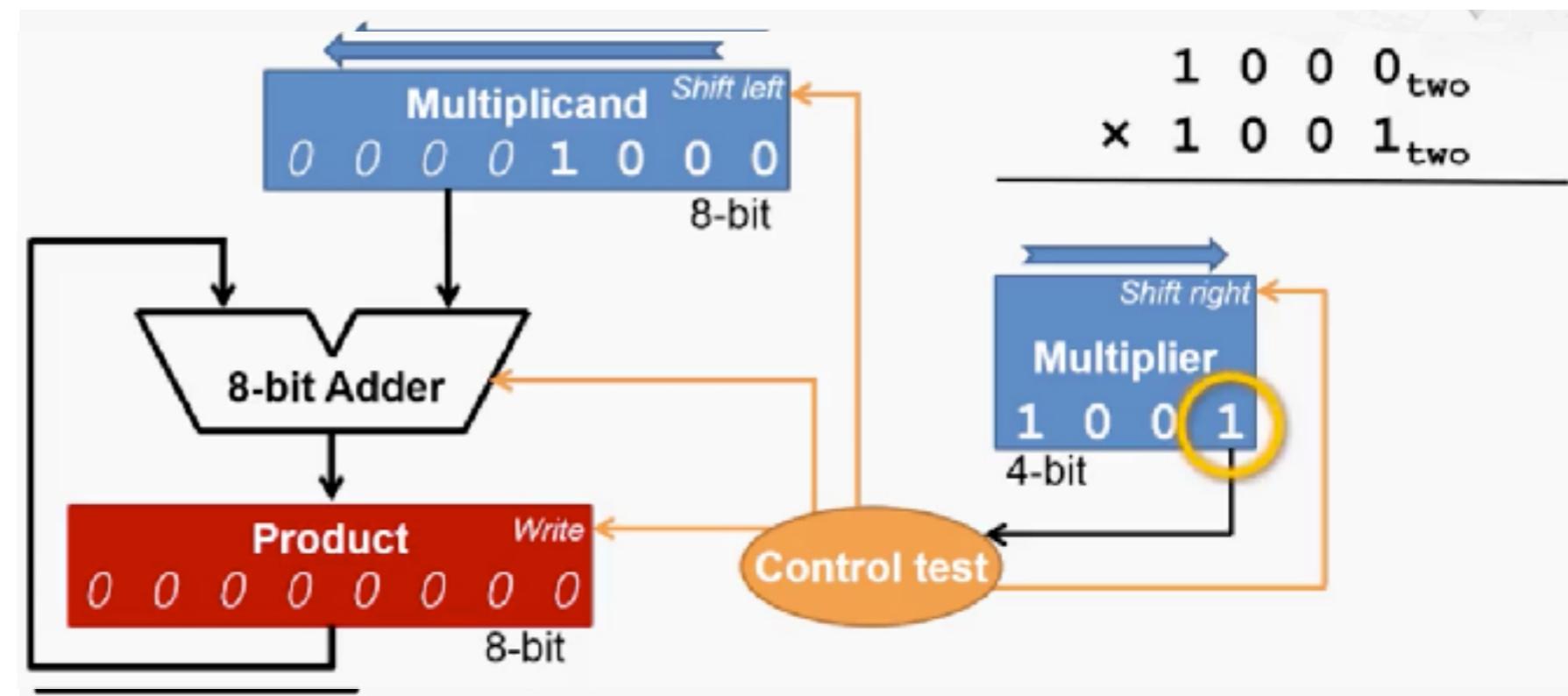
Step 1

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1



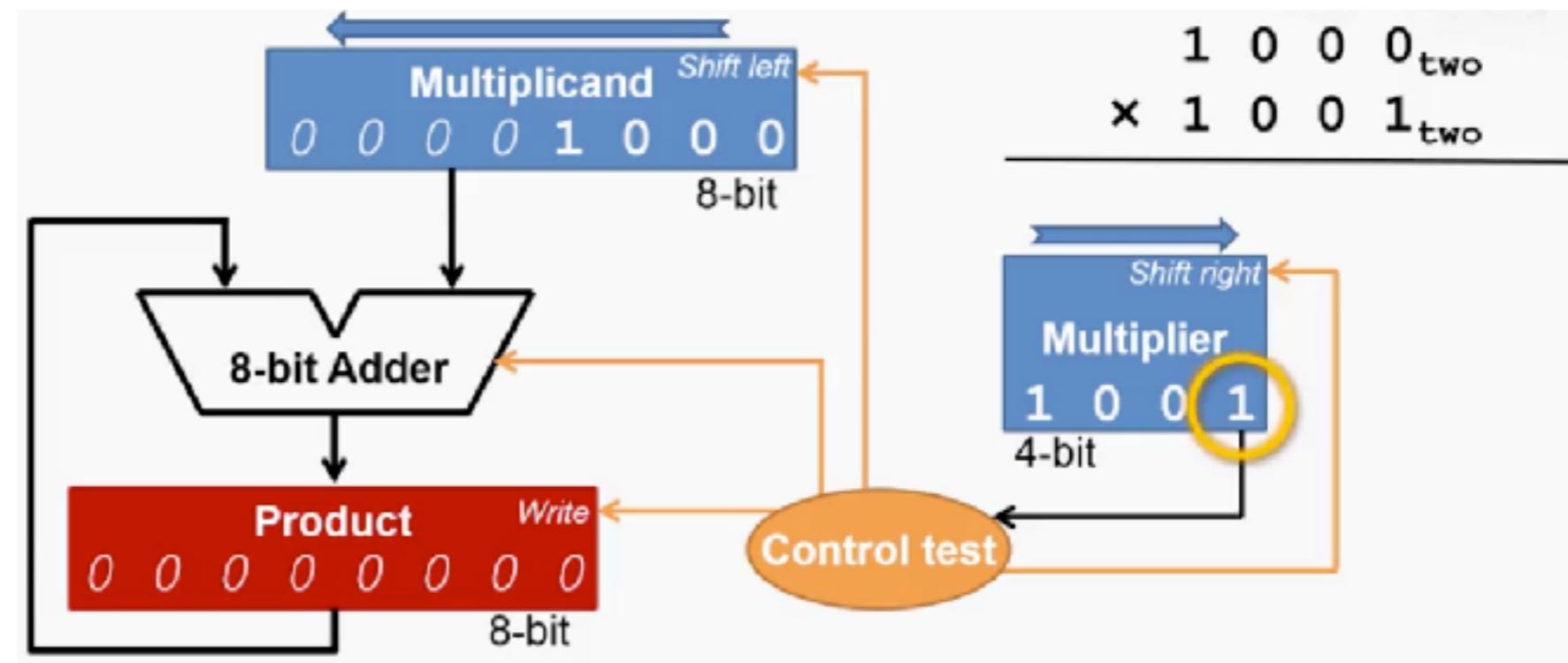
Step 1

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1



Step 1(a)

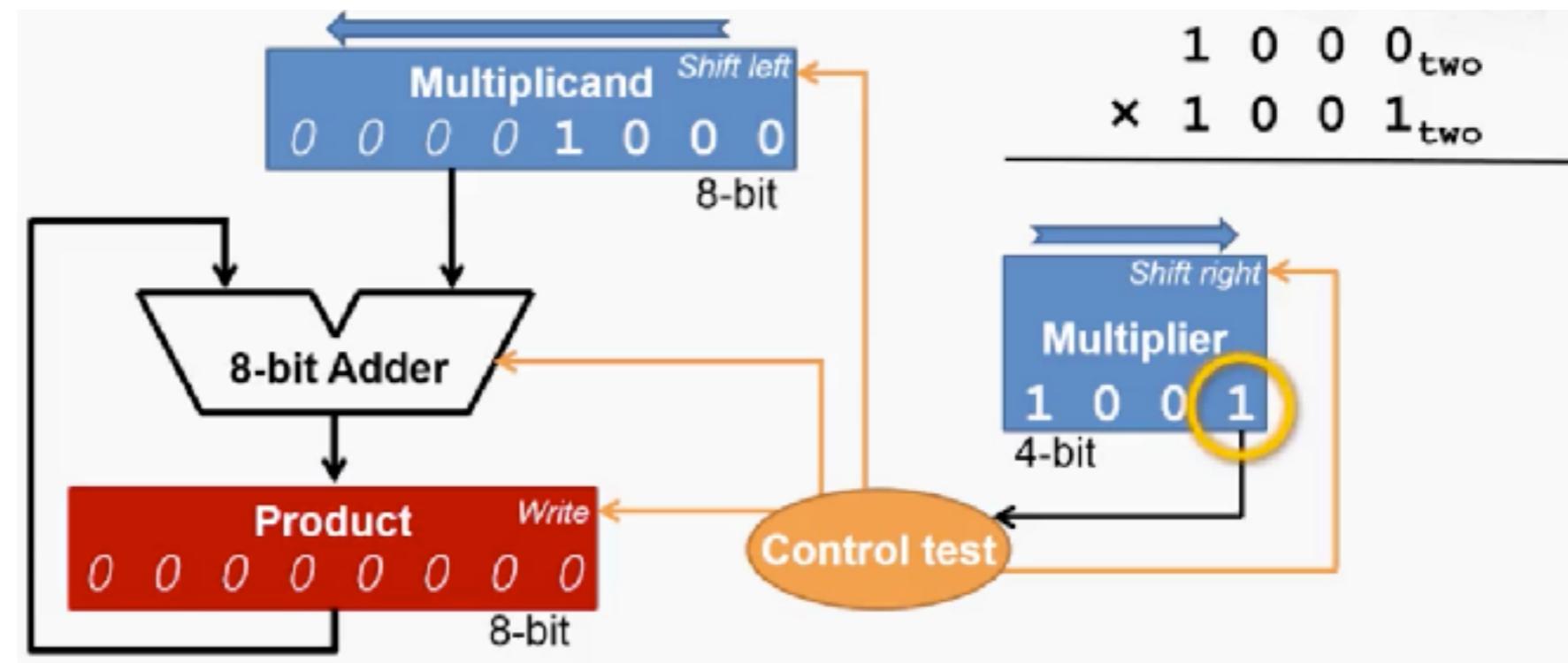
- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier



Step 1(a)

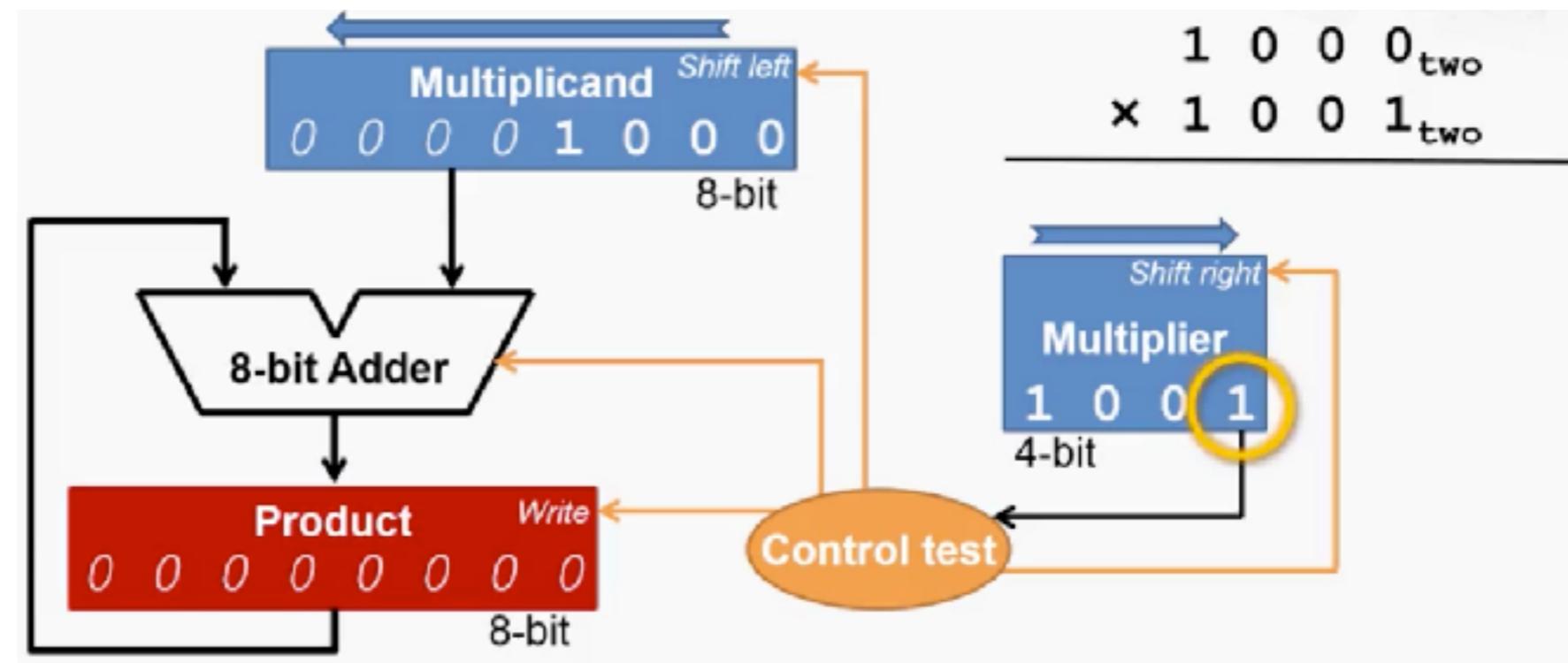
- ❖ Recall the operation with paper and pencil
- ❖ What should we do?

- ❖ Check the LSB of multiplier
 - ❖ If 1



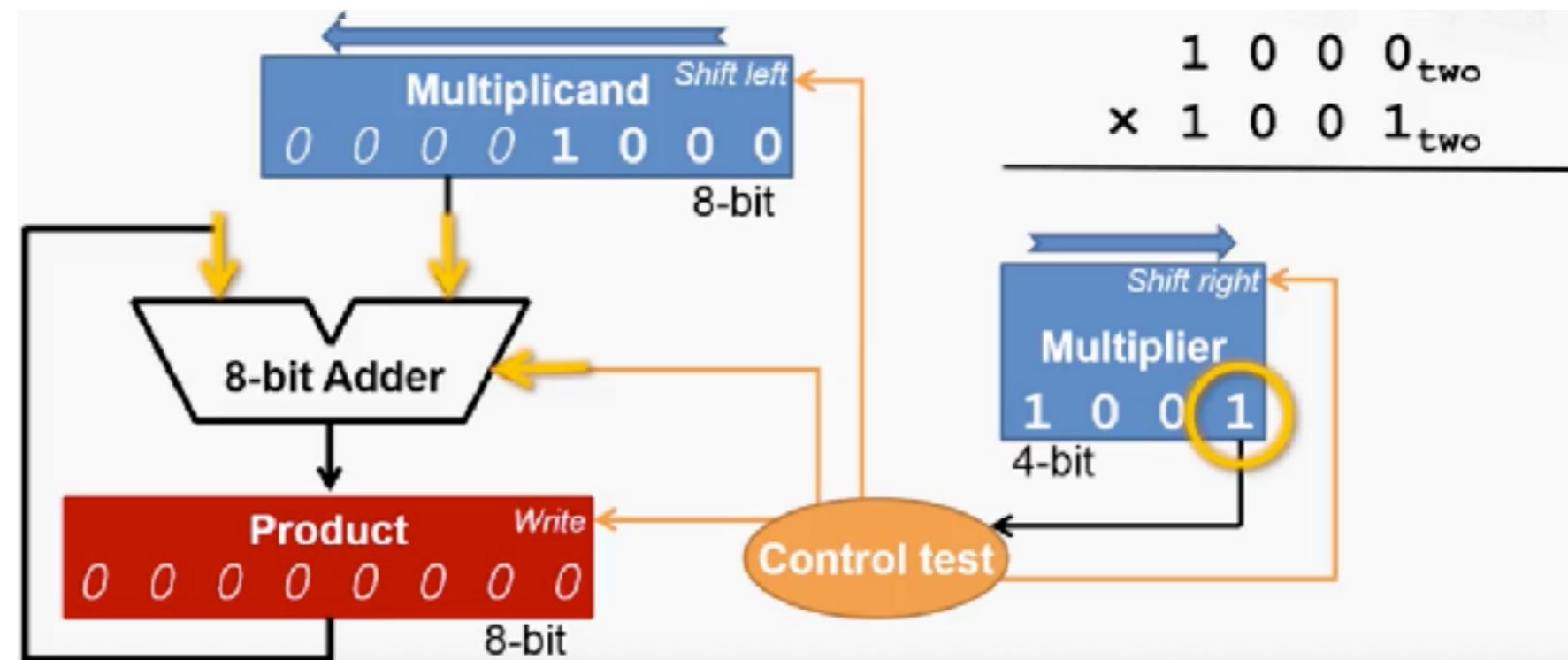
Step 1(a)

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1
 - ❖ Adding multiplicand and product



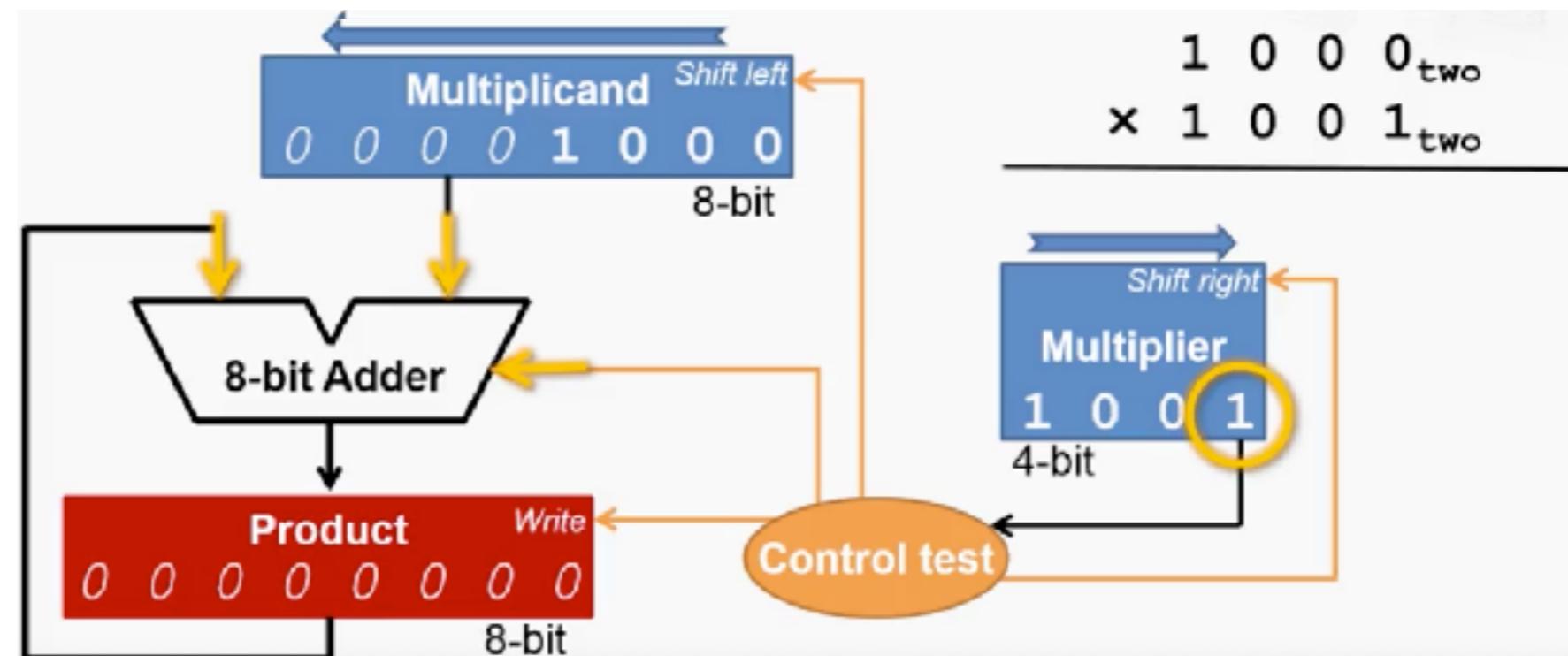
Step 1(a)

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1
 - ❖ Adding multiplicand and product



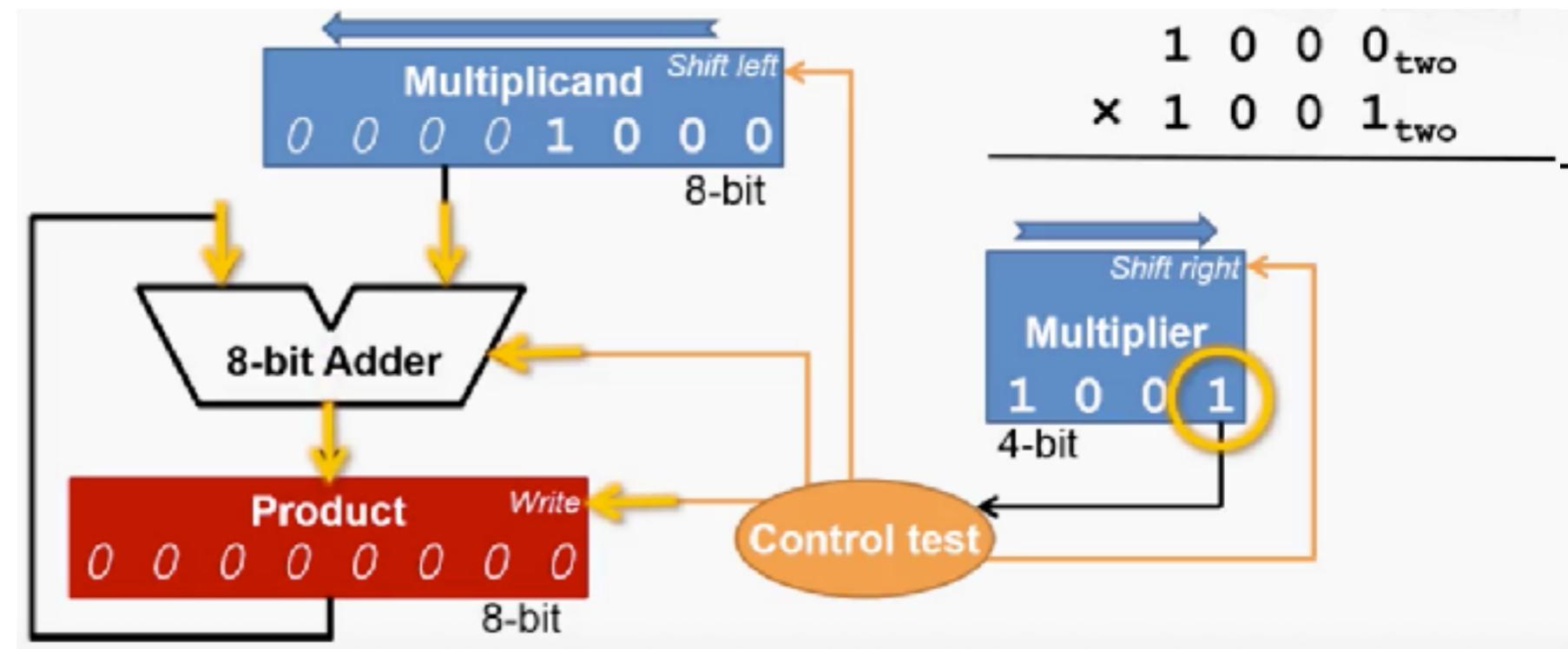
Step 1(a)

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1
 - ❖ Adding multiplicand and product
 - ❖ Store the value into product register



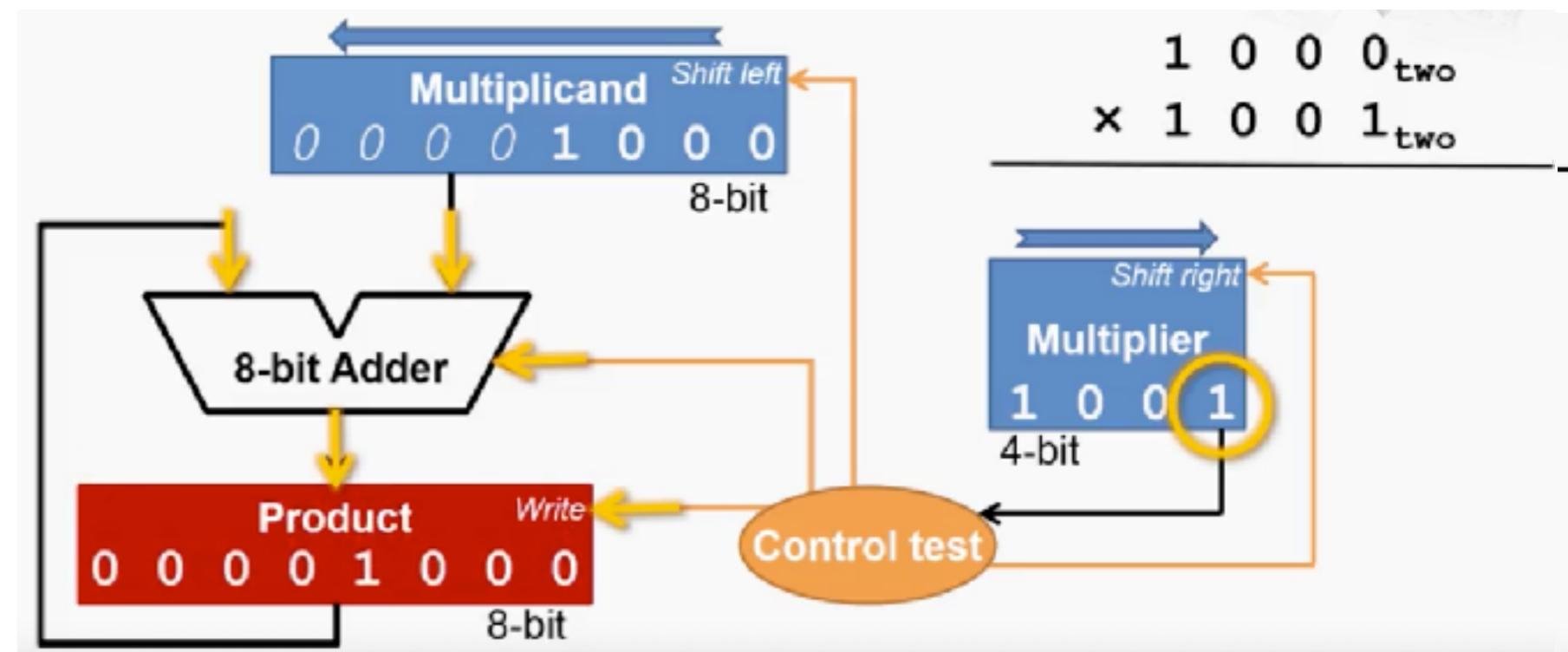
Step 1(a)

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1
 - ❖ Adding multiplicand and product
 - ❖ Store the value into product register

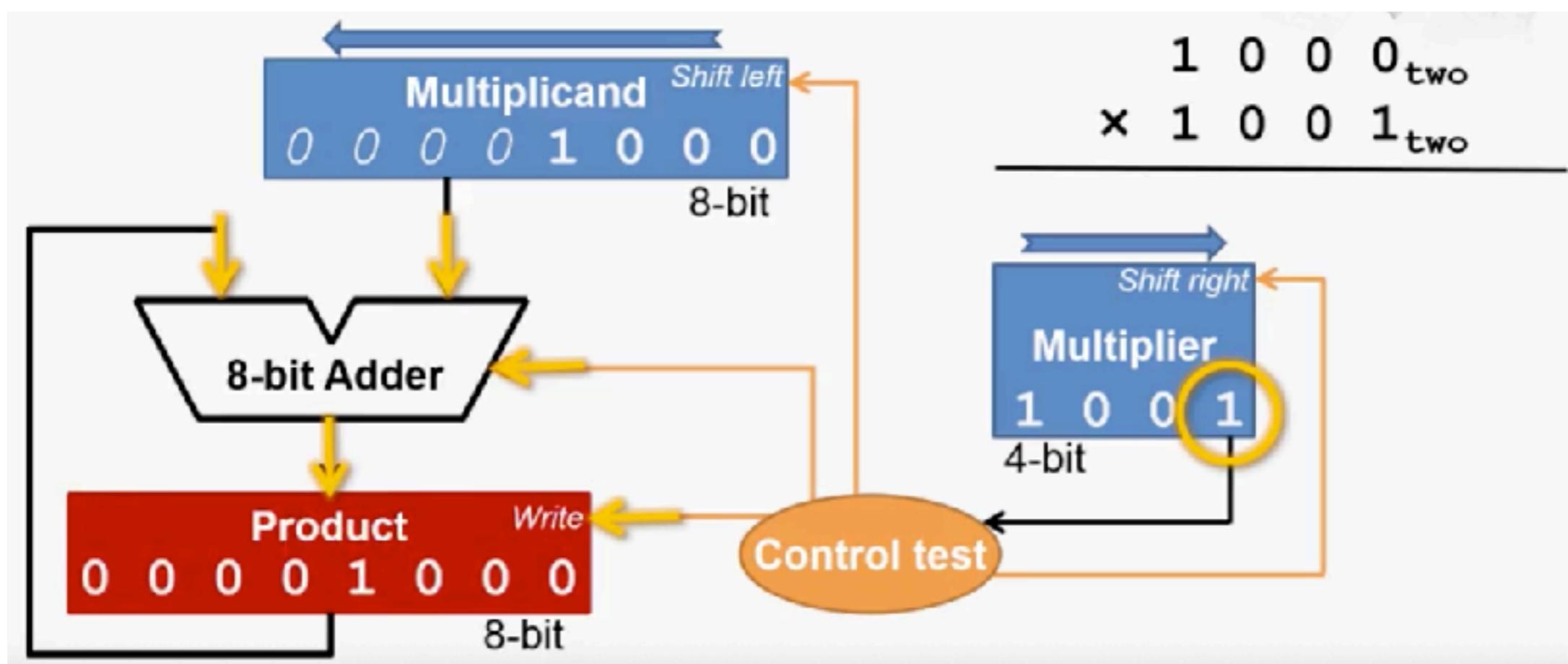


Step 1(a)

- ❖ Recall the operation with paper and pencil
- ❖ What should we do?
 - ❖ Check the LSB of multiplier
 - ❖ If 1
 - ❖ Adding multiplicand and product
 - ❖ Store the value into product register

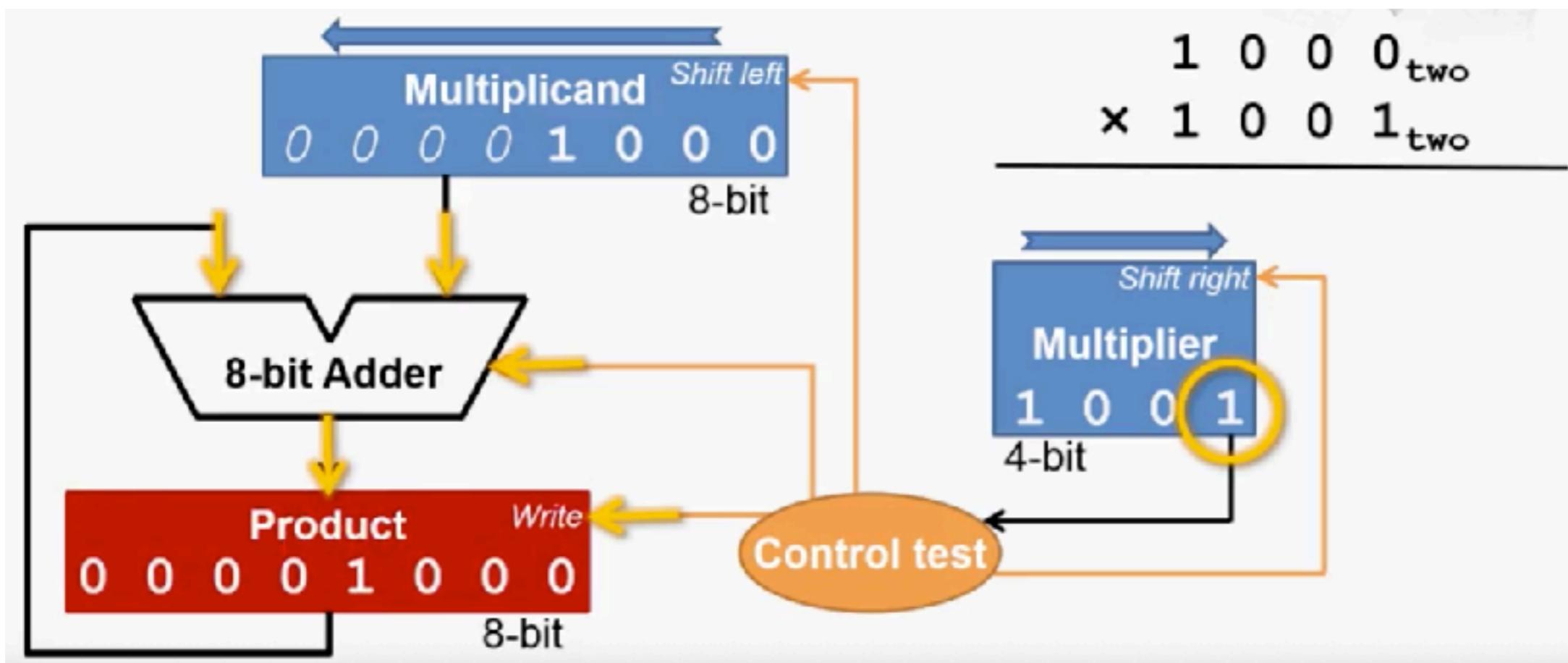


Step 2



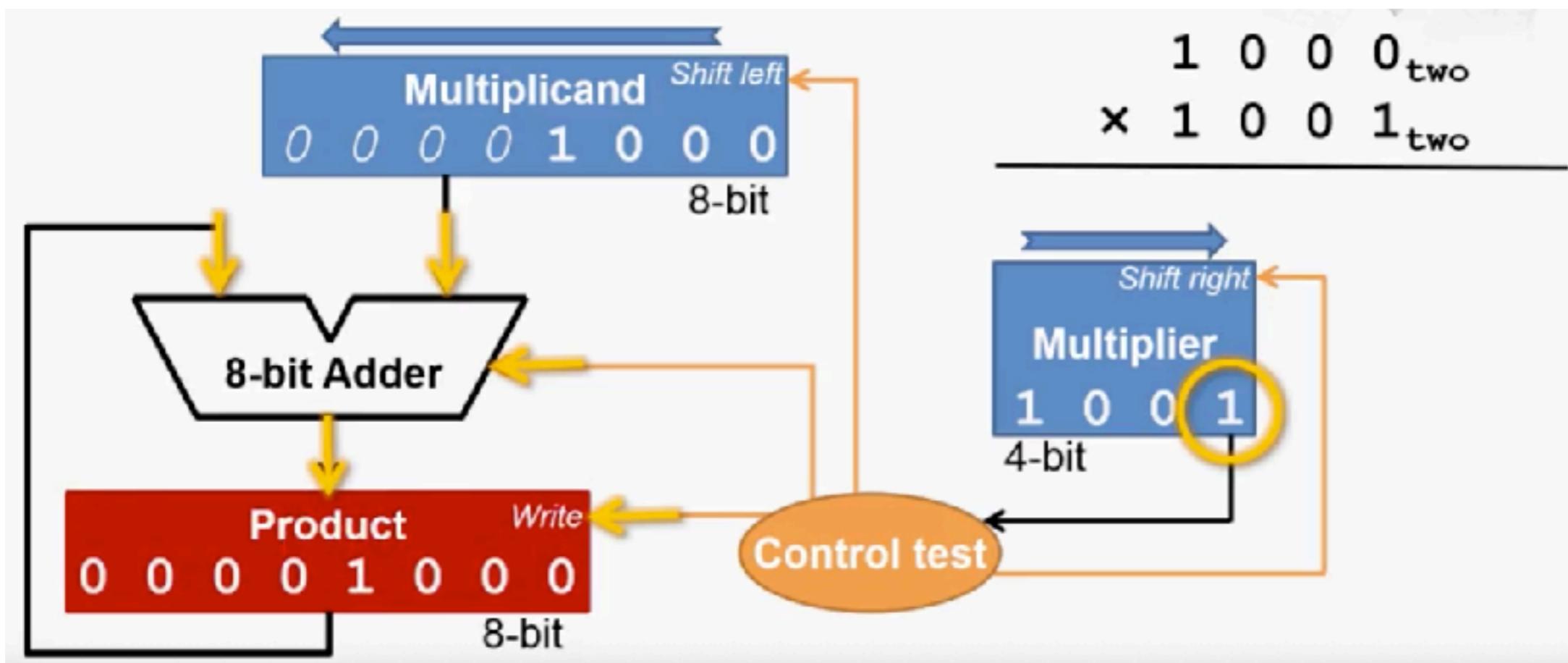
Step 2

- ❖ What should we do as step 2?



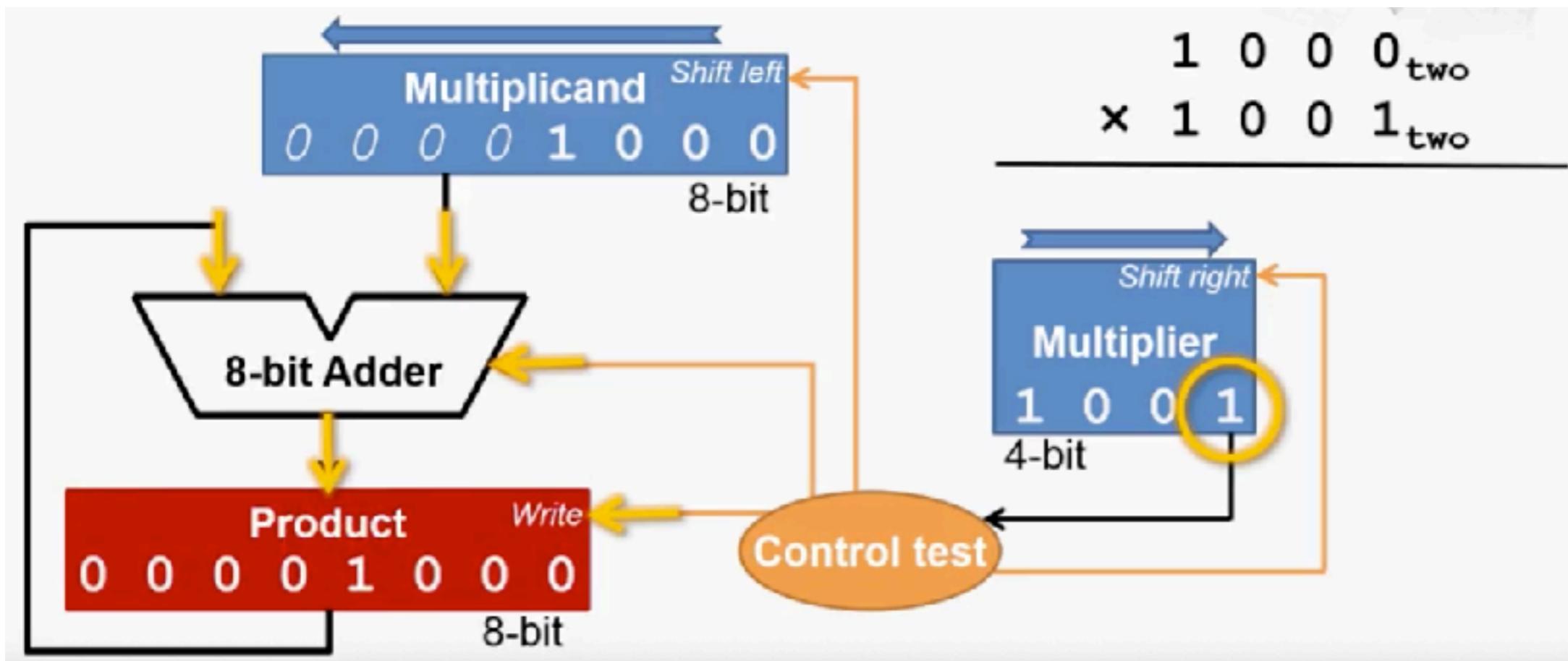
Step 2

- ❖ What should we do as step 2?
 - ❖ Recall what we did with the emulation



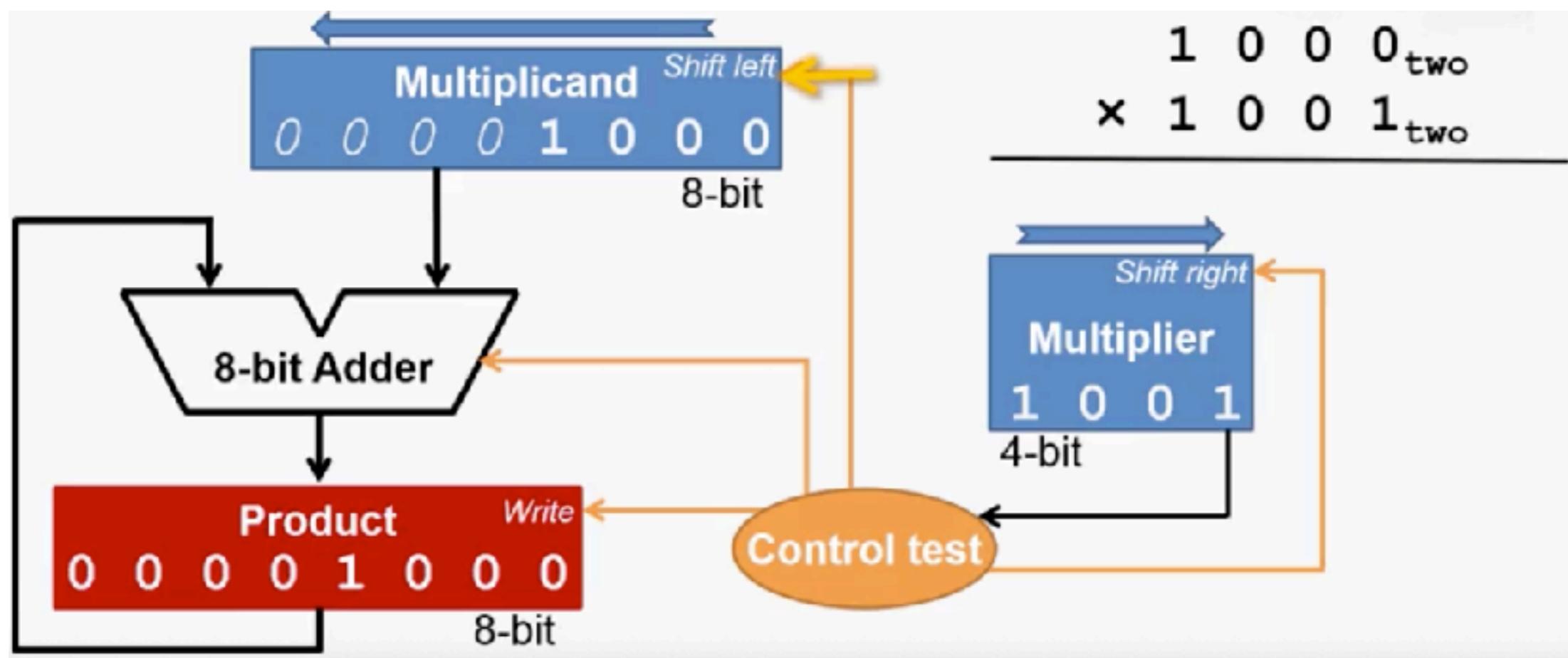
Step 2

- ❖ What should we do as step 2?
 - ❖ Recall what we did with the emulation
 - ❖ Left shift!



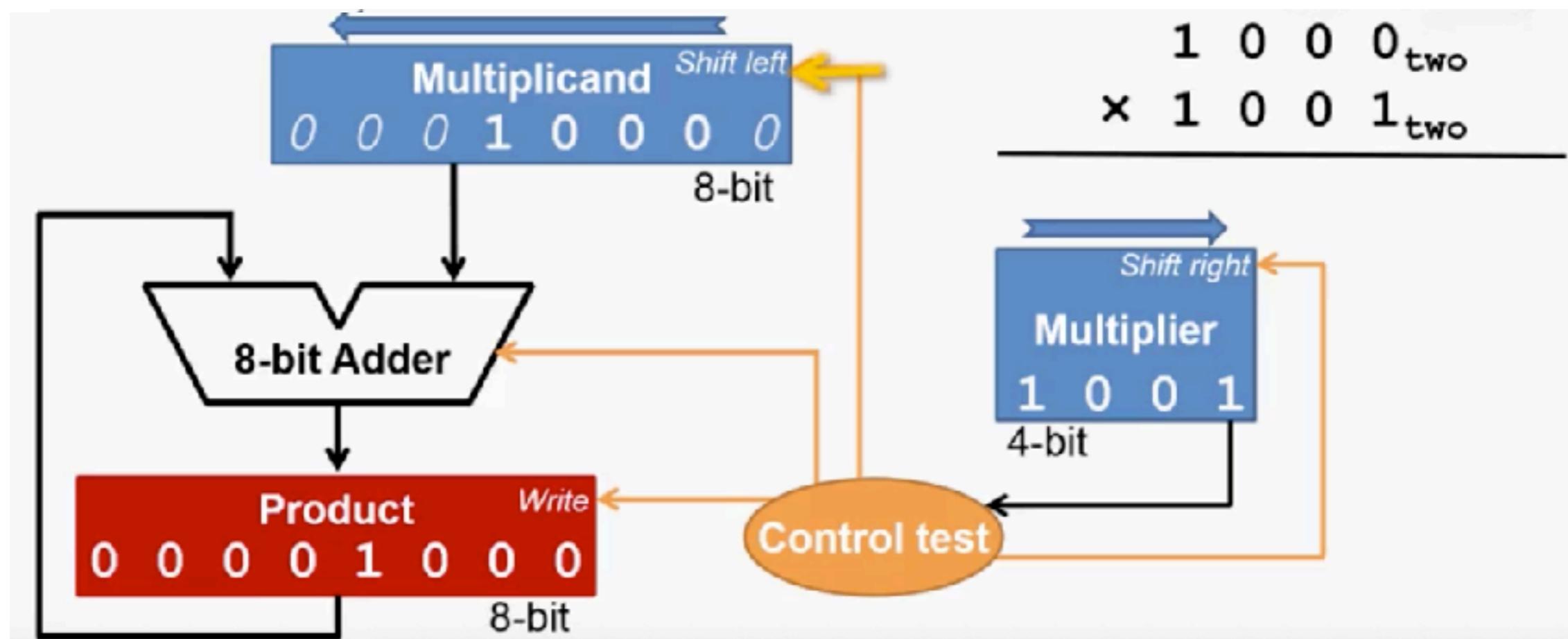
Step 2

- ❖ What should we do as step 2?
 - ❖ Recall what we did with the emulation
 - ❖ Left shift!

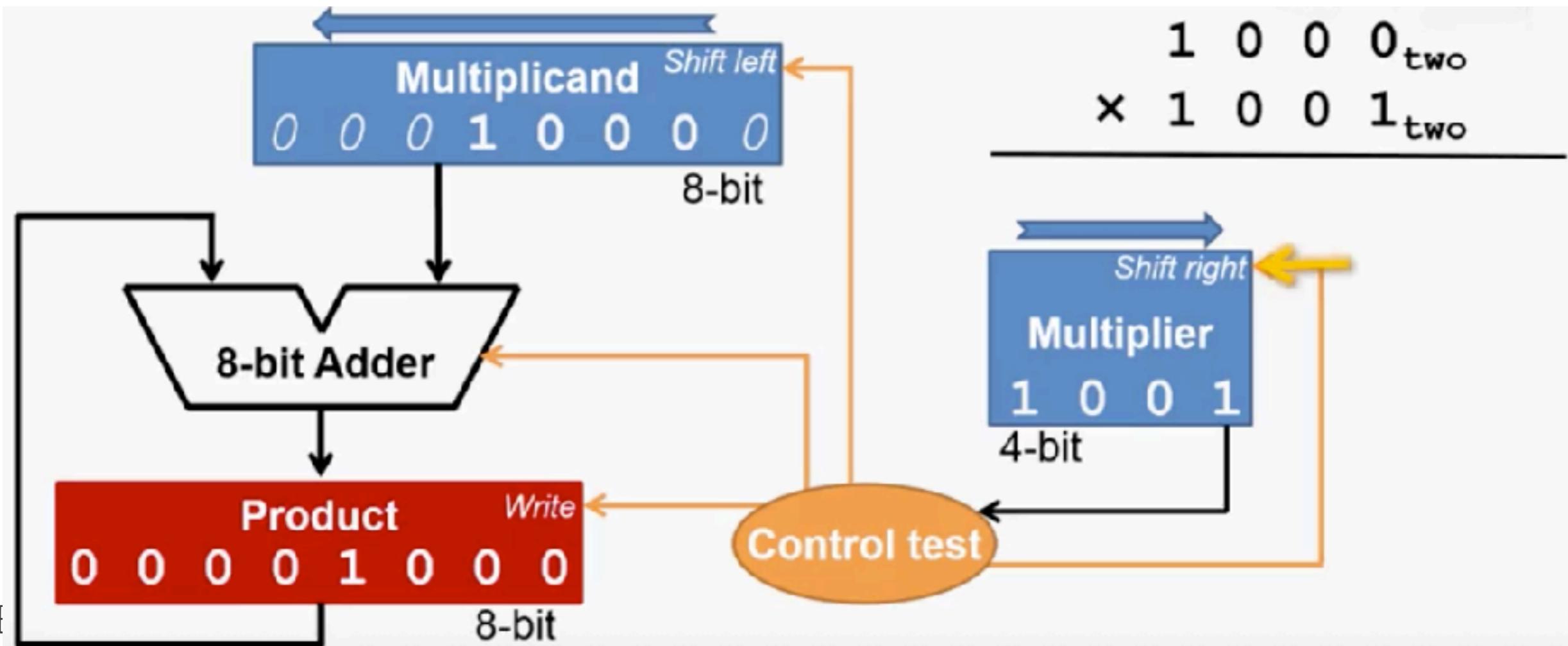


Step 2

- ❖ What should we do as step 2?
 - ❖ Recall what we did with the emulation
 - ❖ Left shift!

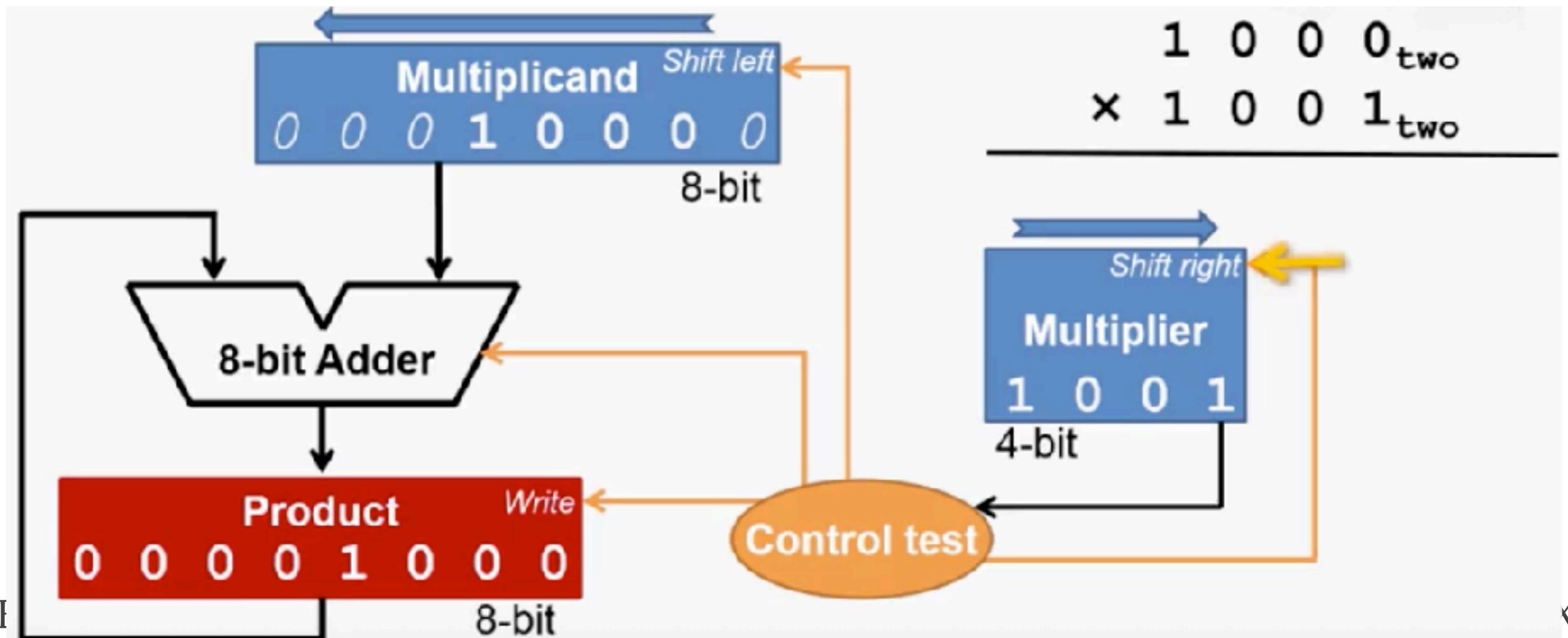


Step 3



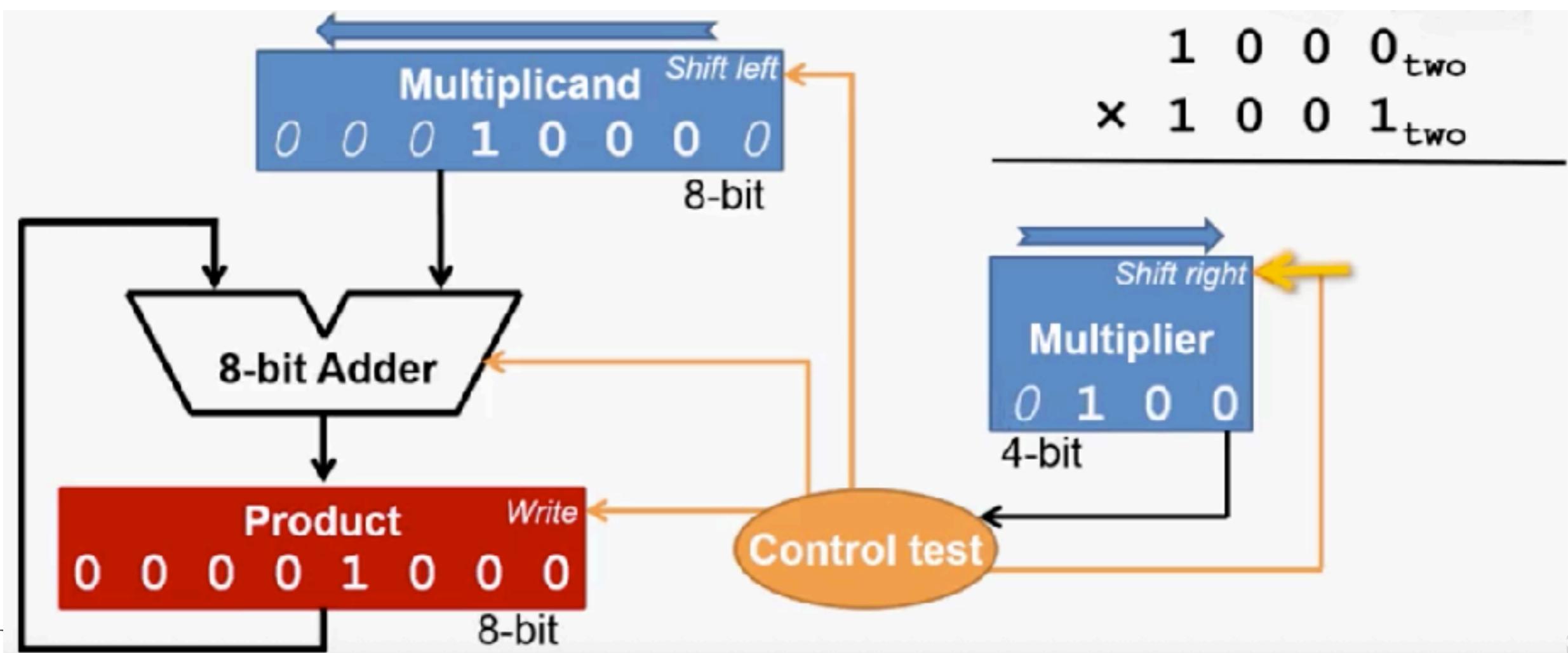
Step 3

- ❖ Right shift of multiplier



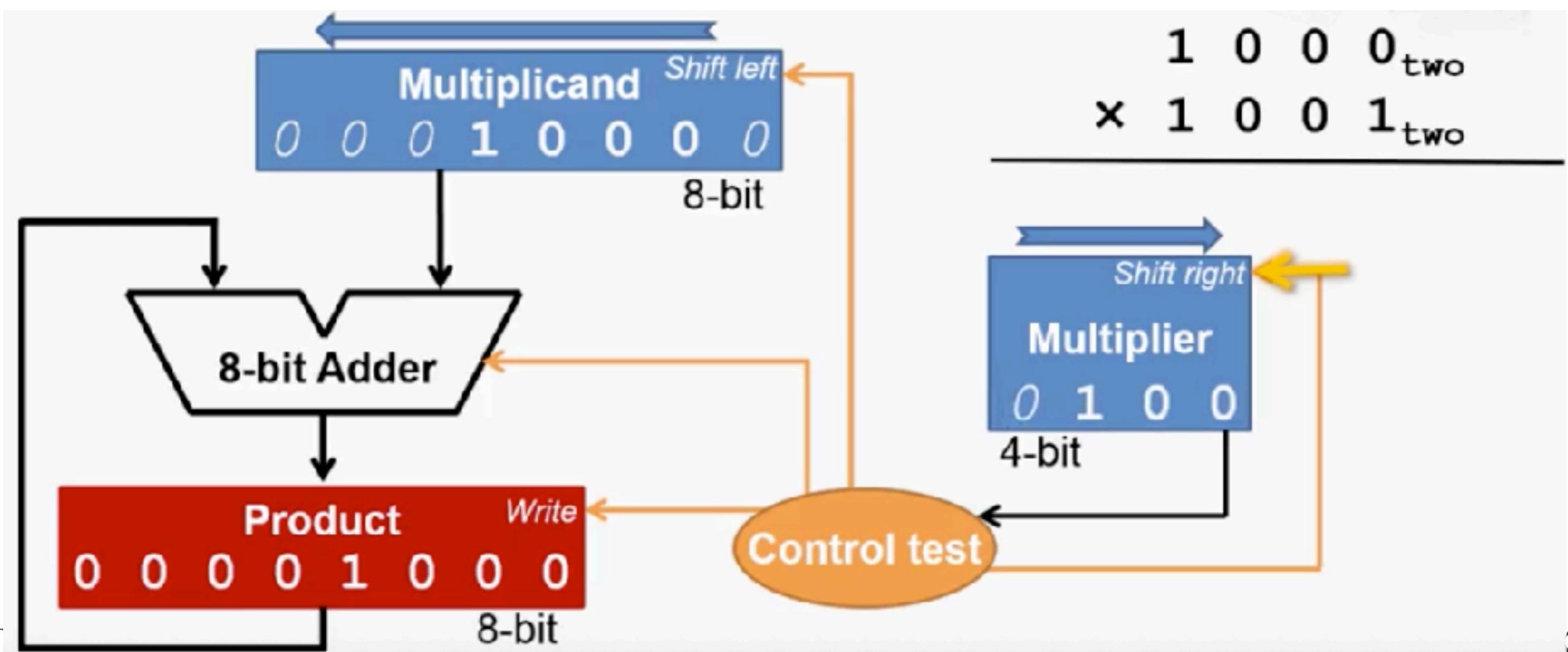
Step 3

- ❖ Right shift of multiplier



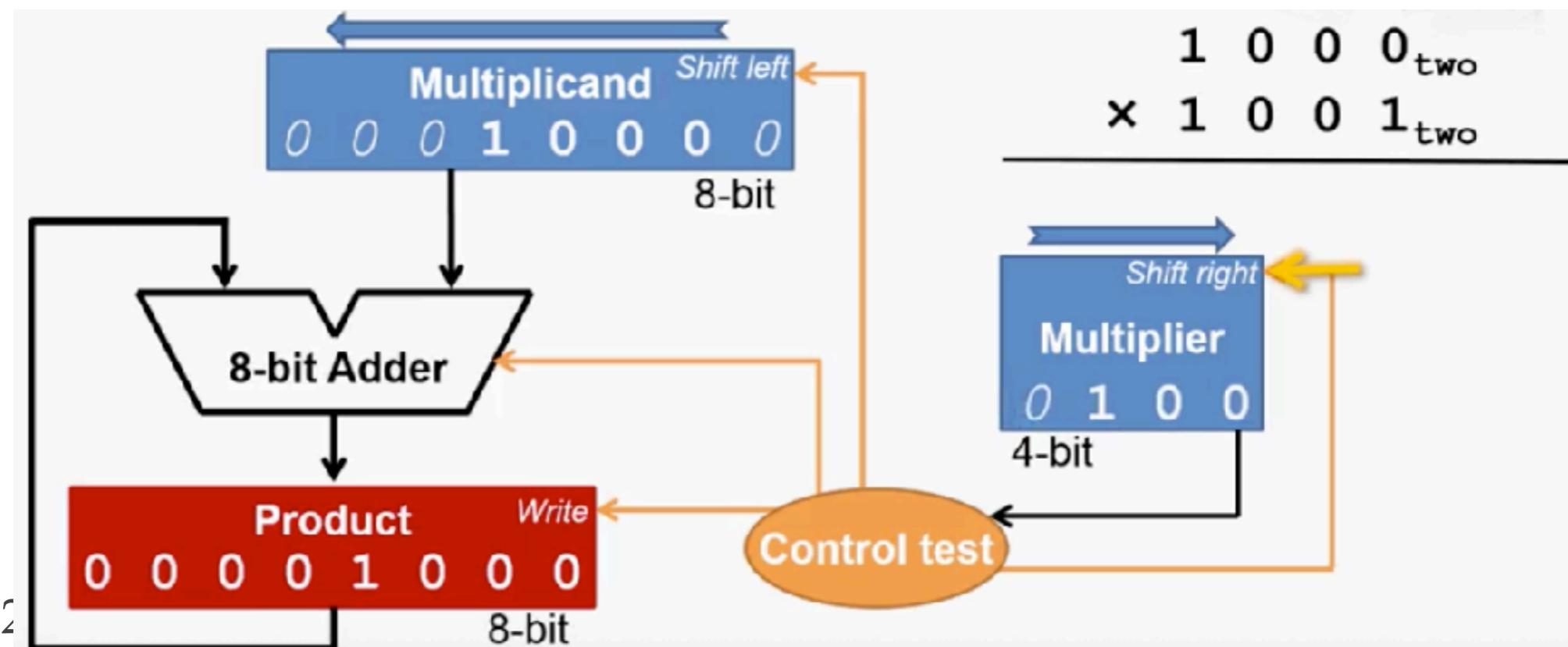
Step 3

- ❖ Right shift of multiplier
 - ❖ Instead of shifting, we can set up indicator to check which data we operate in last round, however, complicated! Now, only LSB!



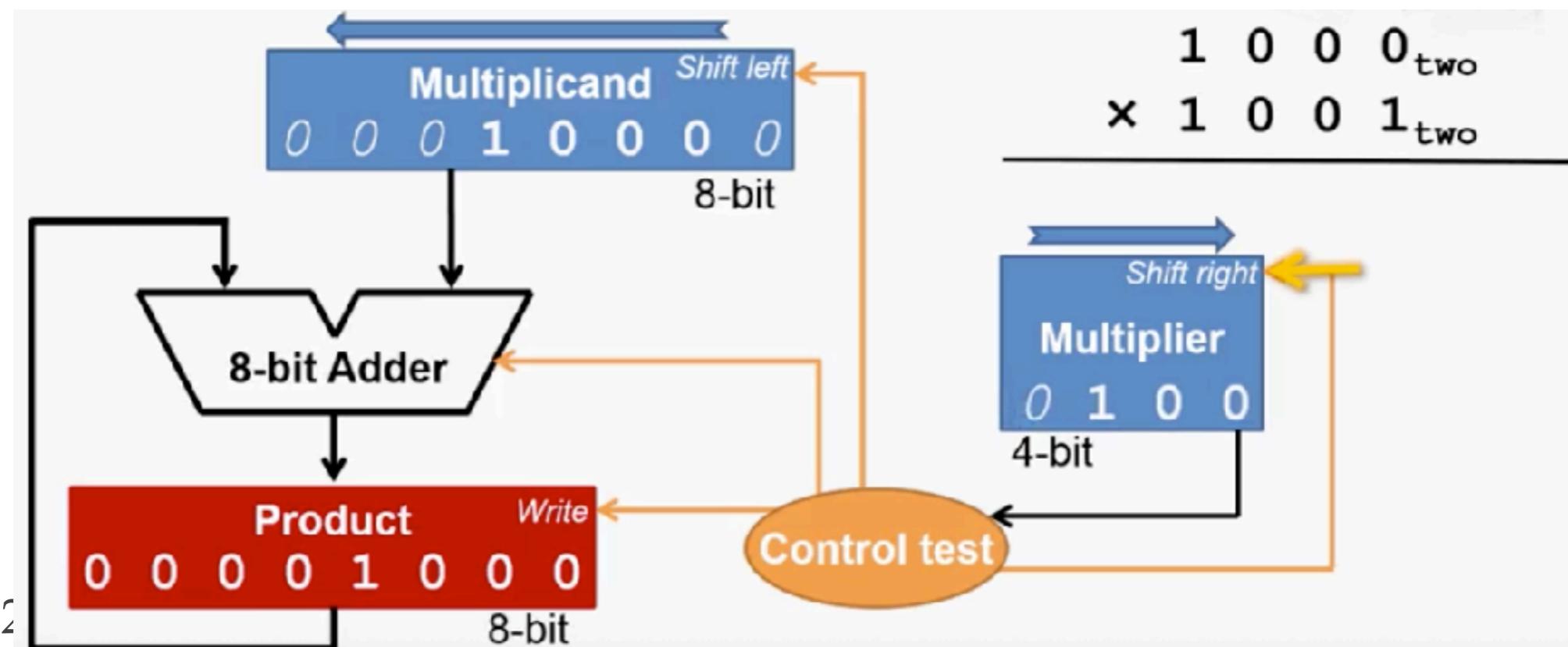
Analysis of Step 2 and 3

- ❖ Why we can do the right shift of multiplier?



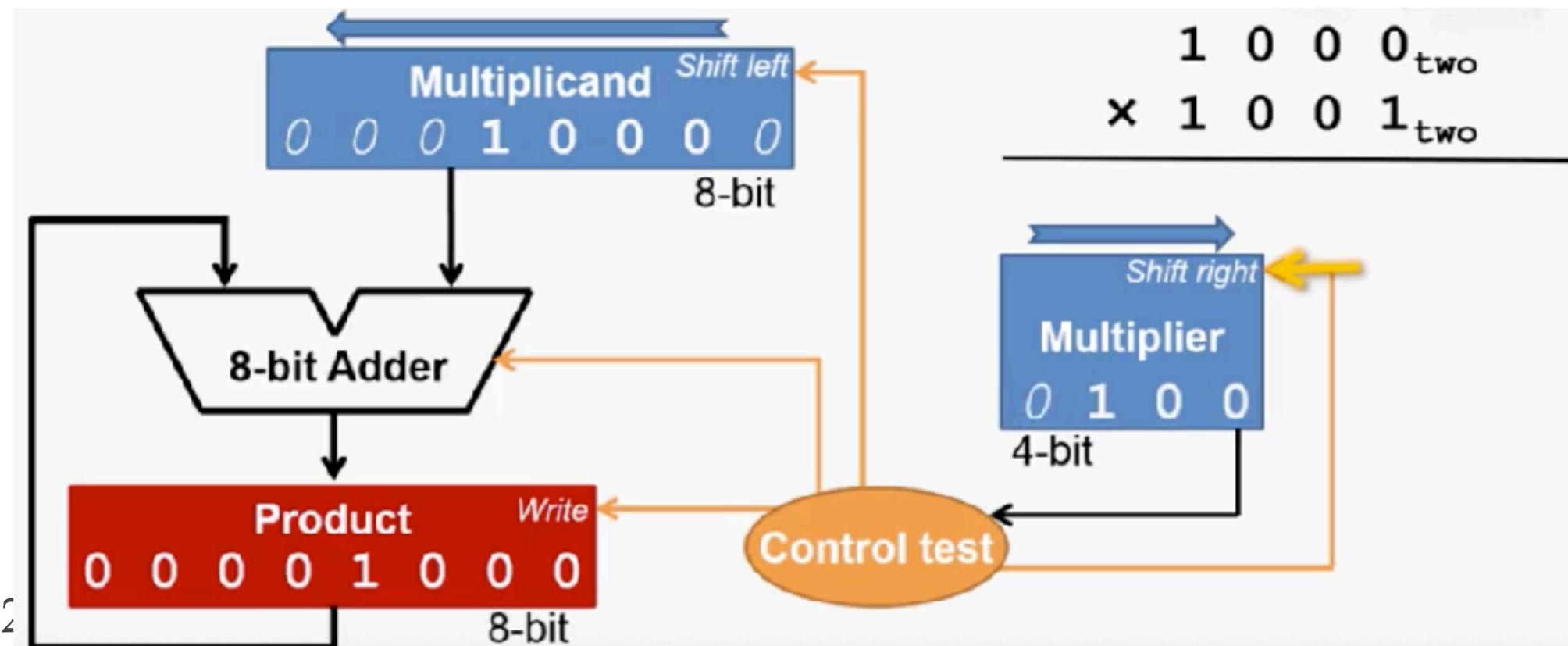
Analysis of Step 2 and 3

- ❖ Why we can do the right shift of multiplier?
- ❖ The data has no use anymore



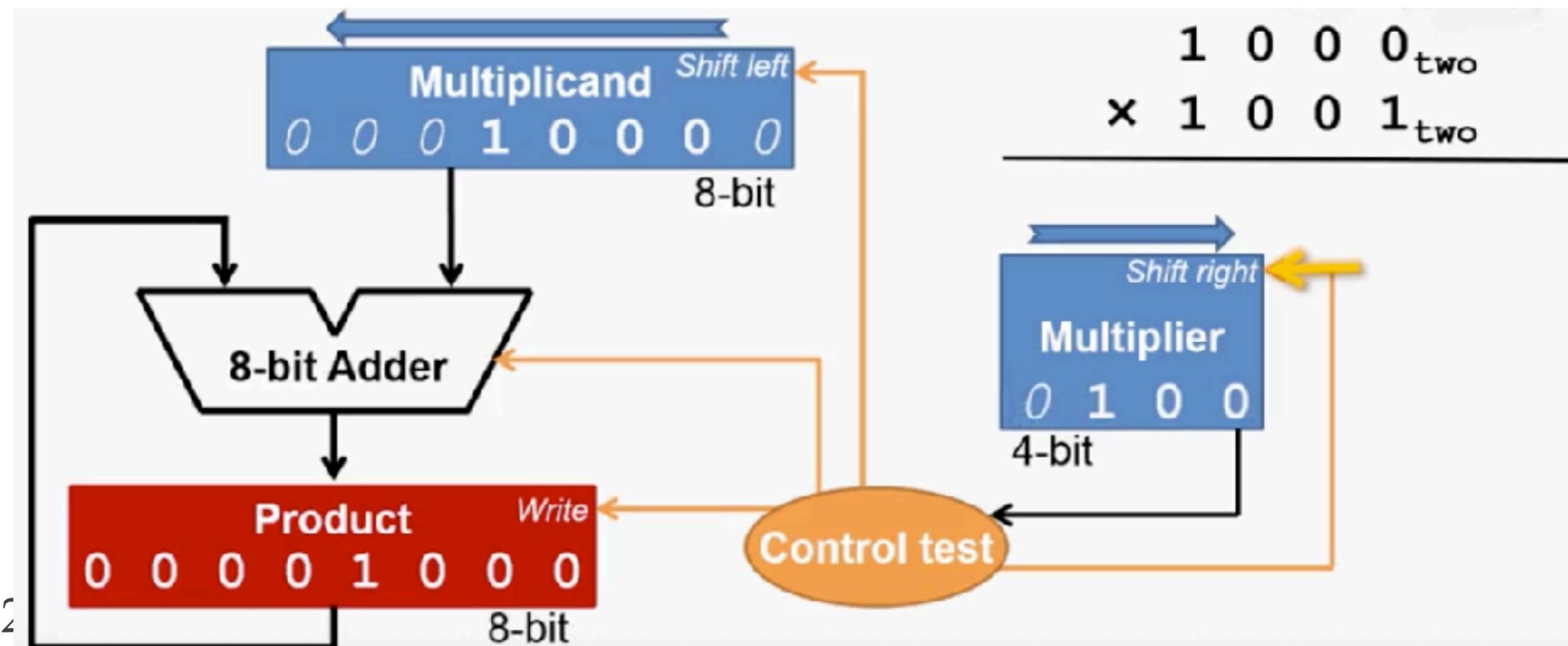
Analysis of Step 2 and 3

- ❖ Why we can do the right shift of multiplier?
 - ❖ The data has no use anymore
- ❖ Step 2 and 3 are making preparation for next round!

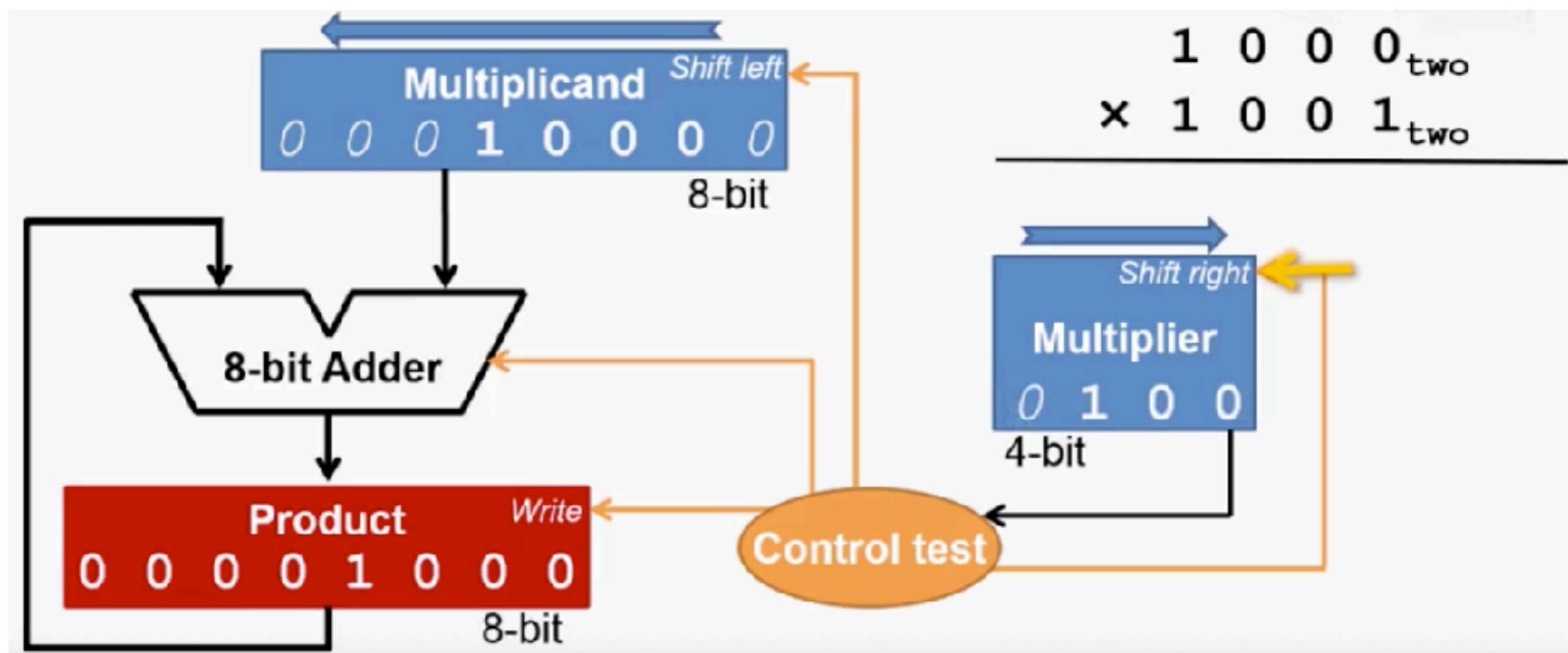


Analysis of Step 2 and 3

- ❖ Why we can do the right shift of multiplier?
- ❖ The data has no use anymore
- ❖ Step 2 and 3 are making preparation for next round!
- ❖ But hold on, do we have next round or not?

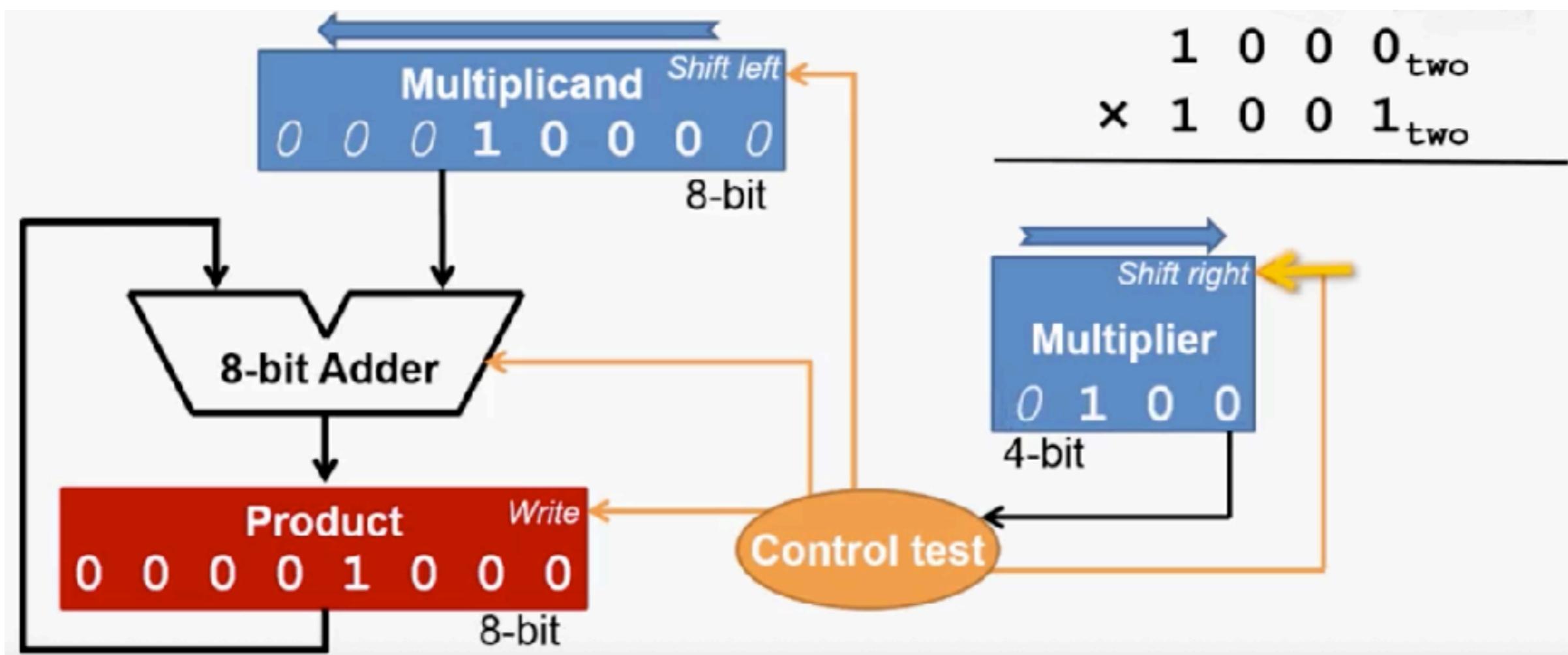


Step 4



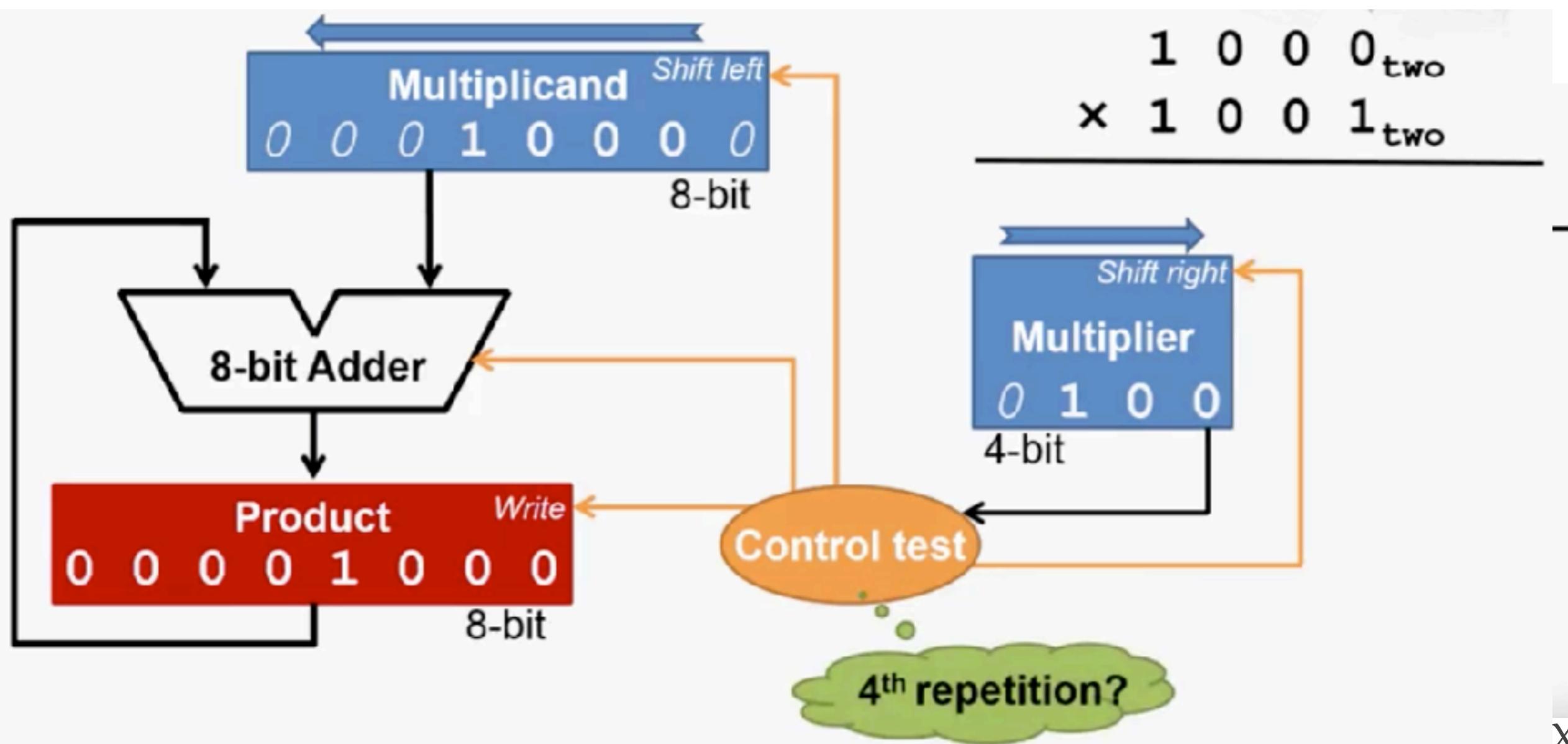
Step 4

- ❖ Control test module decide whether it is already the last round? I.e., the 4th round?



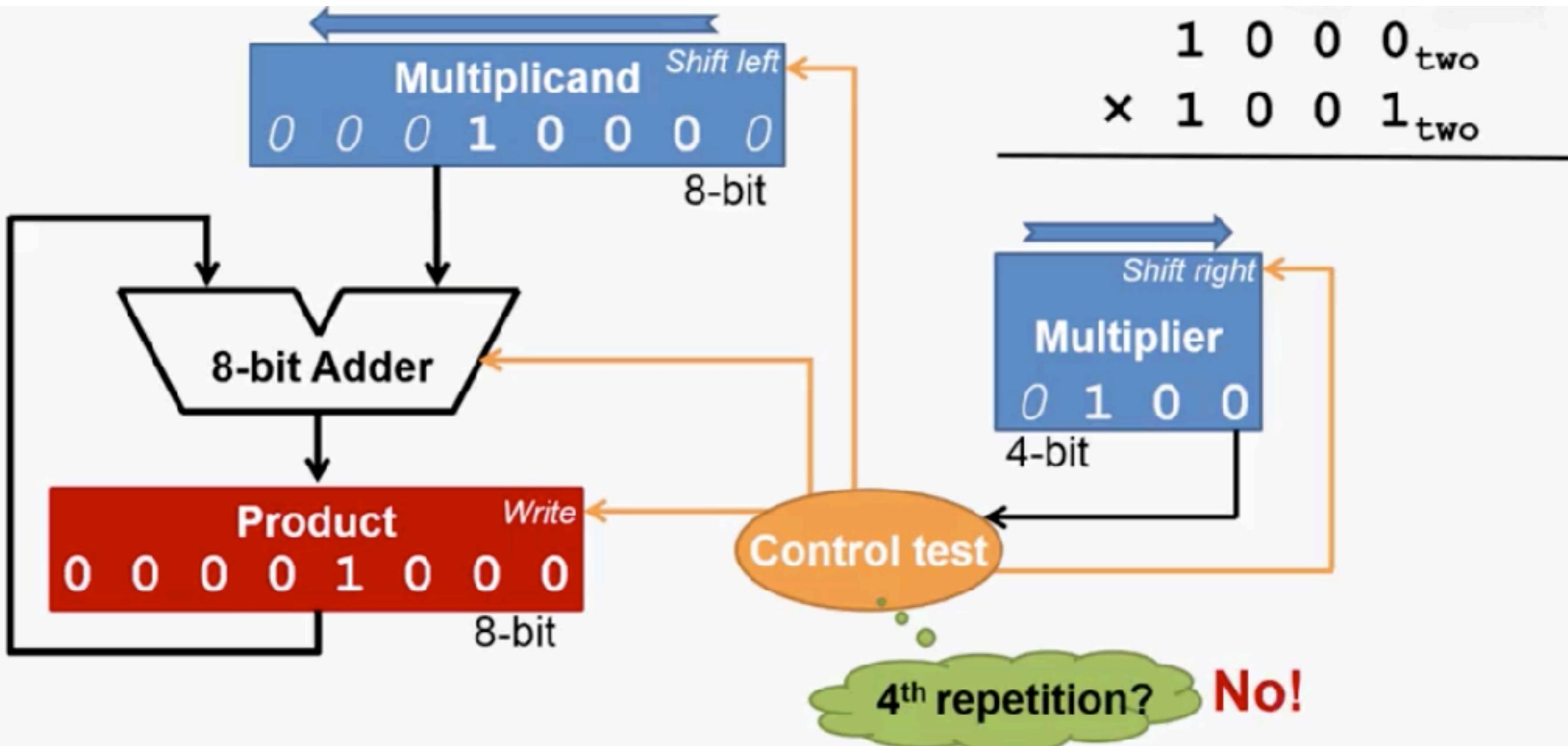
Step 4

- ❖ Control test module decide whether it is already the last round? I.e., the 4th round?



Step 4

- ❖ Control test module decide whether it is already the last round? I.e., the 4th round?



Control Test

- ❖ Another important module
- ❖ Repetition checking
- ❖ Simply counter can do it

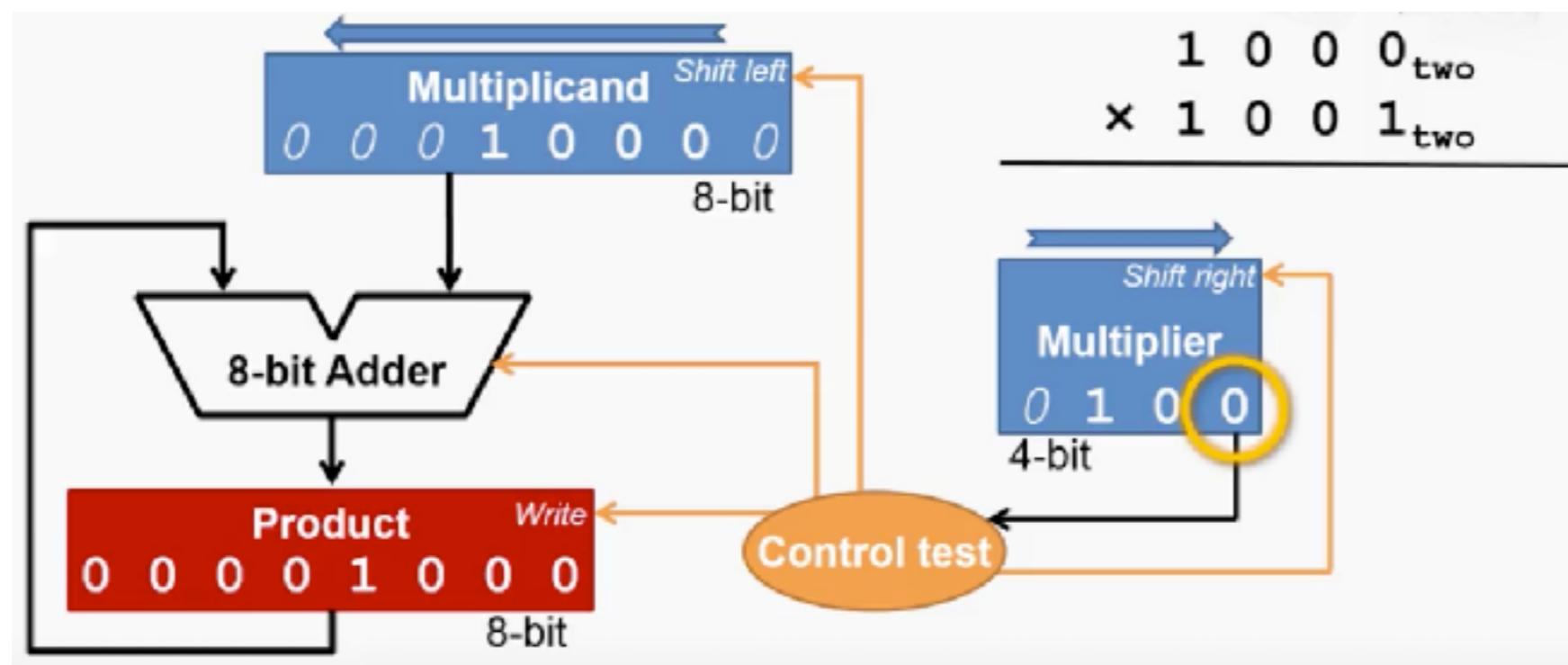
Step 1 of Round 2

Step 1 of Round 2

- ❖ 1, Check the LSB of multiplier

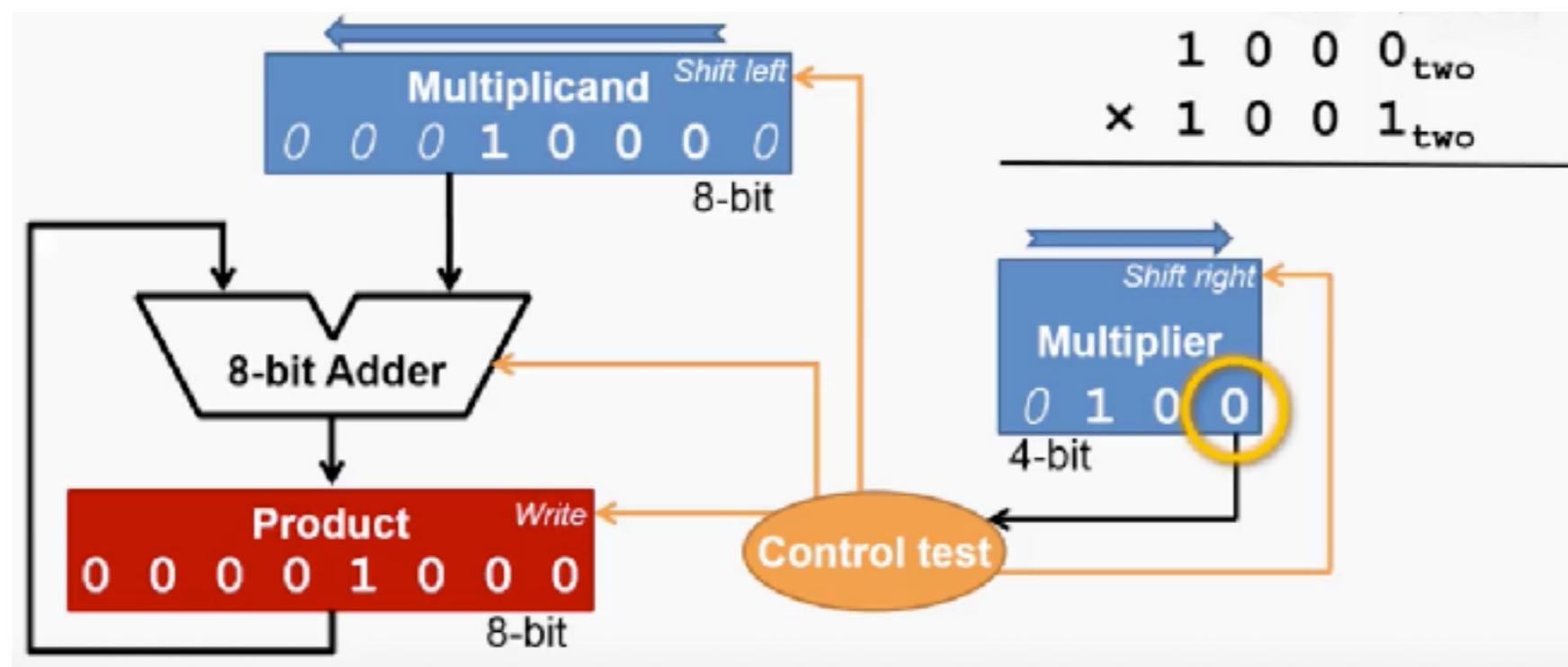
Step 1 of Round 2

- 1, Check the LSB of multiplier



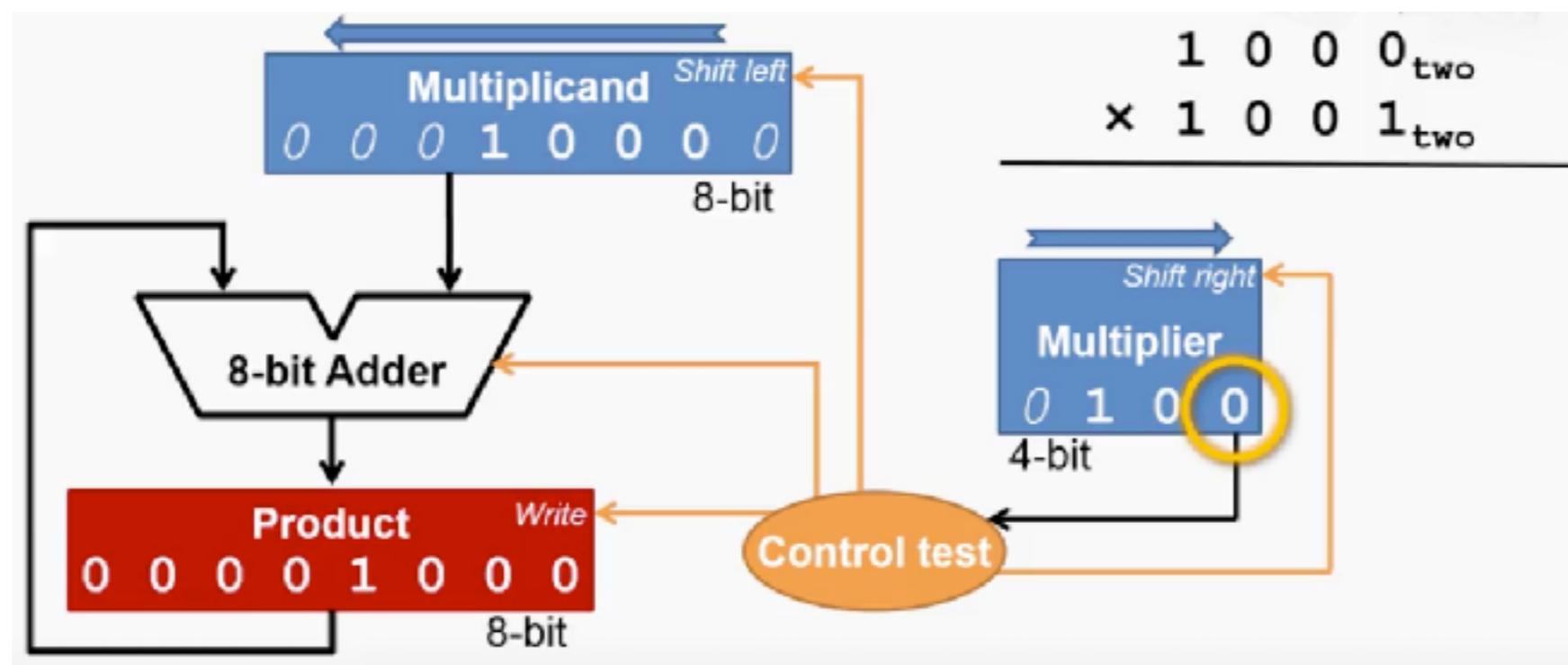
Step 1 of Round 2

- ❖ 1, Check the LSB of multiplier
 - ❖ If 1, do what?



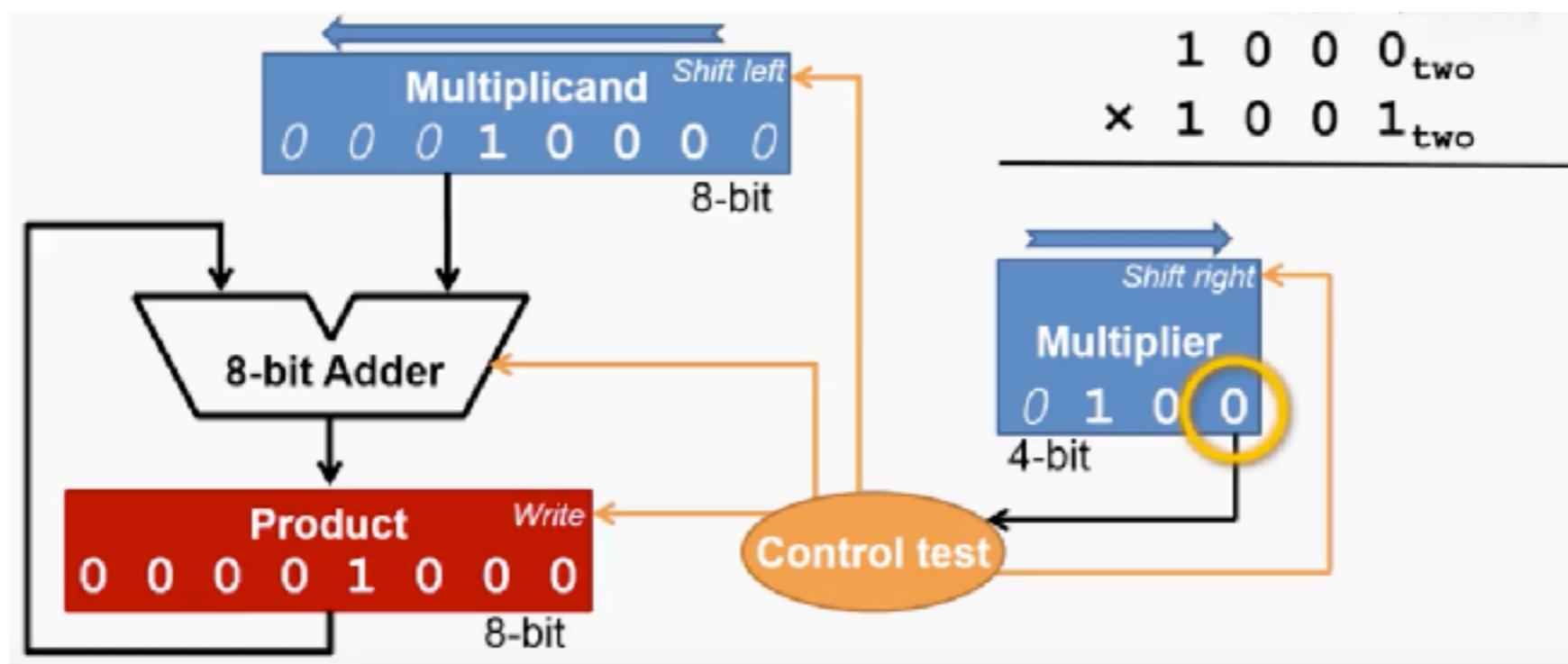
Step 1 of Round 2

- ❖ 1, Check the LSB of multiplier
 - ❖ If 1, do what?
 - ❖ If 0, do what?



Step 1 of Round 2

- ❖ 1, Check the LSB of multiplier
 - ❖ If 1, do what?
 - ❖ If 0, do what?
- ❖ We get 0, do nothing, no step 2 of round 2

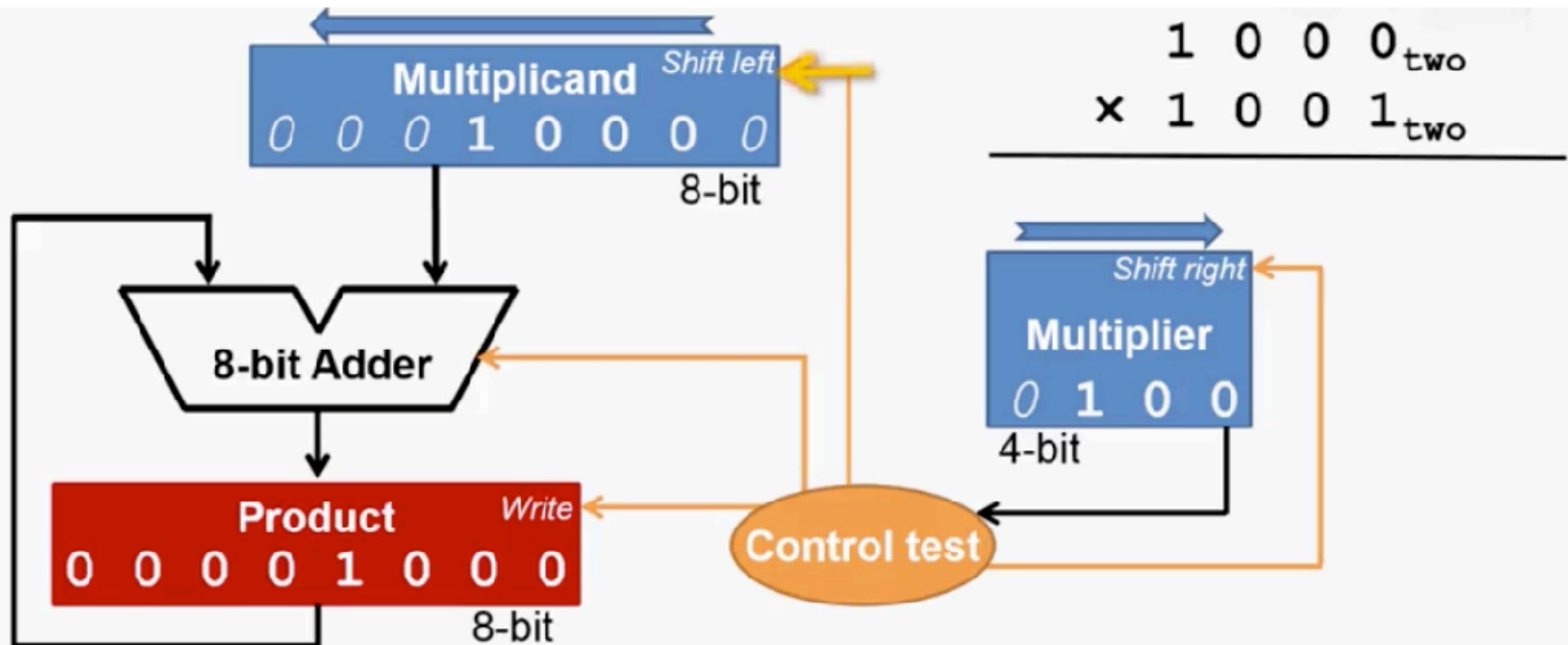


Step 2 of Round 2

- ❖ What does this step do?
 - ❖ Left shift of multiplicand

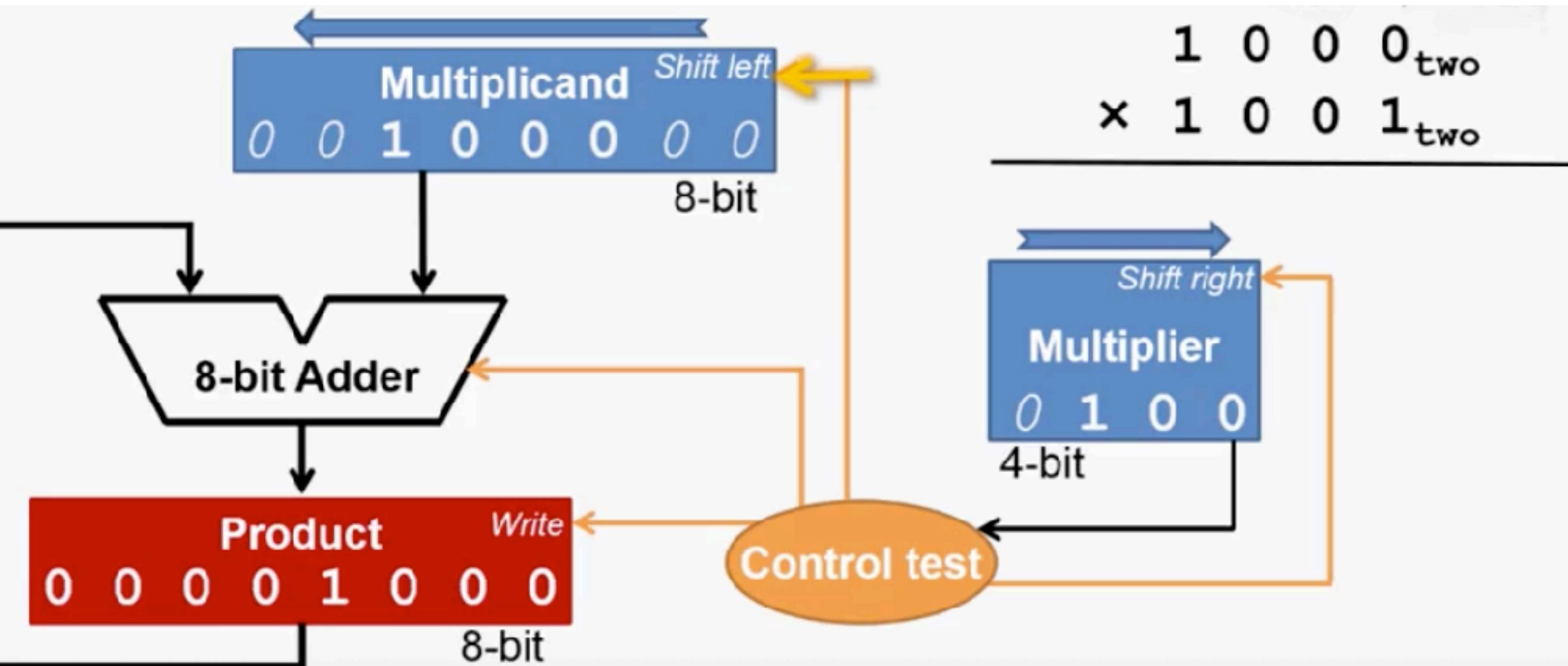
Step 2 of Round 2

- ❖ What does this step do?
 - ❖ Left shift of multiplicand



Step 2 of Round 2

- ❖ What does this step do?
 - ❖ Left shift of multiplicand

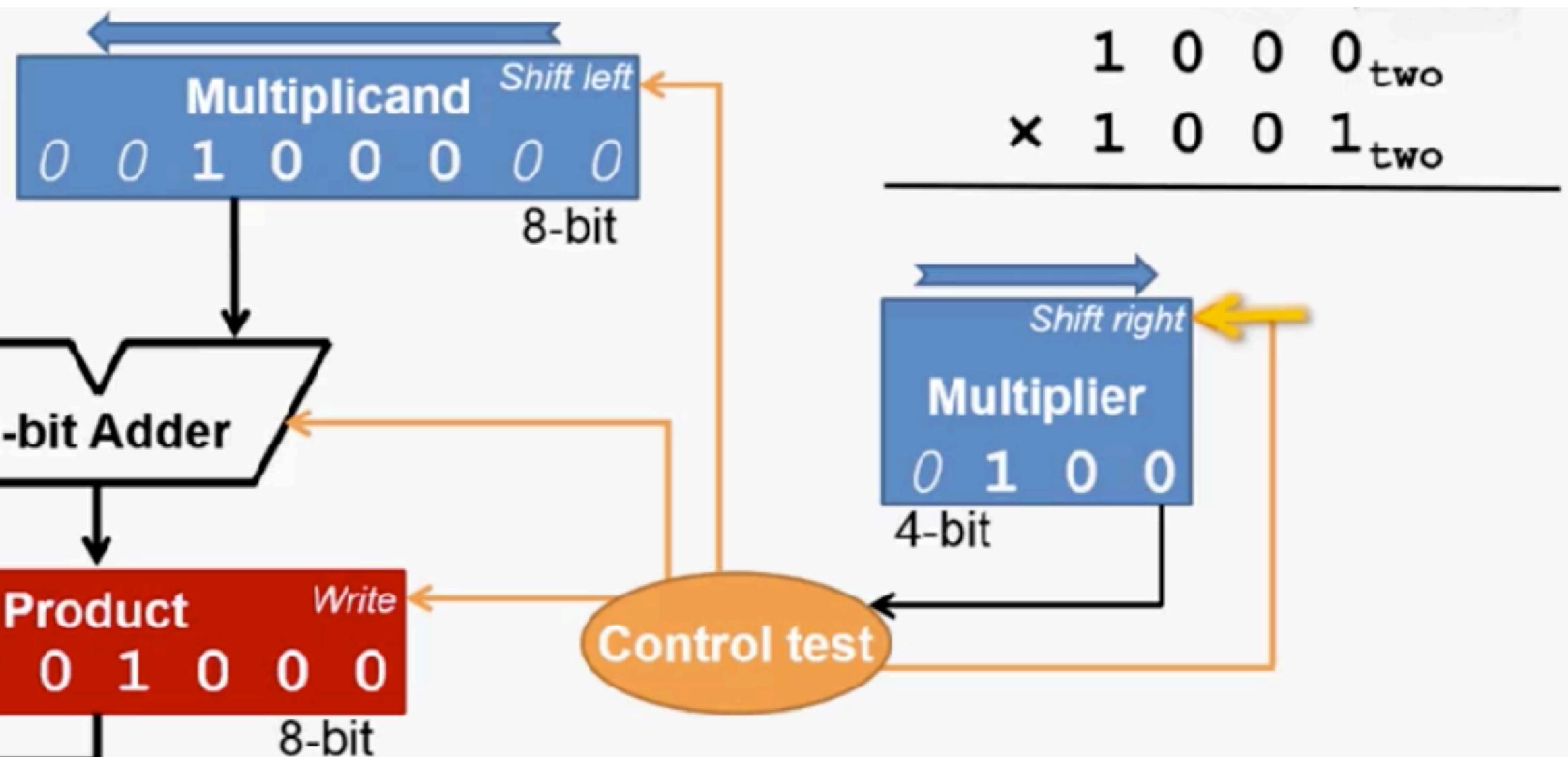


Step 3 of Round 2

- ❖ Shift right of multiplier

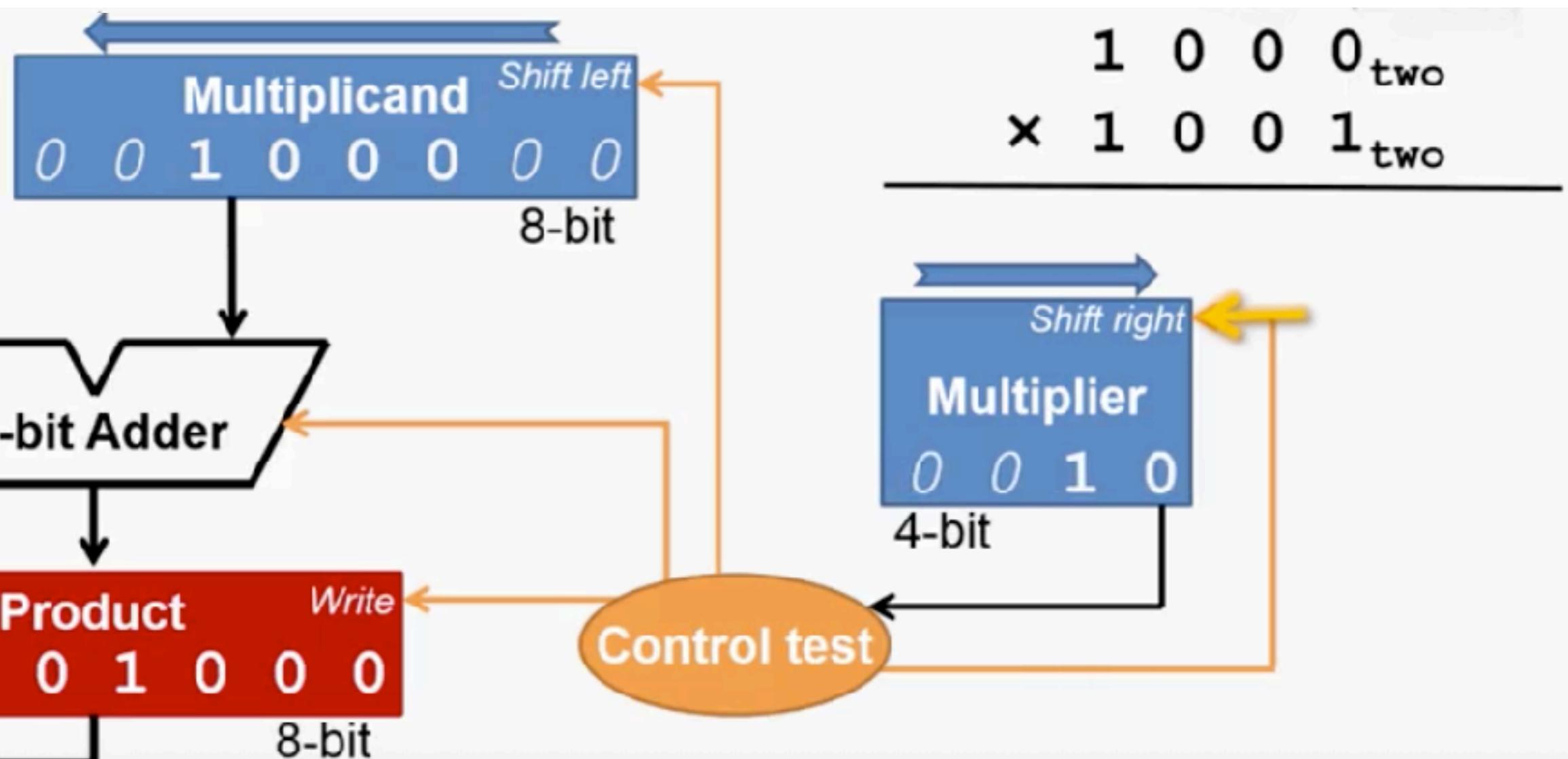
Step 3 of Round 2

- ❖ Shift right of multiplier



Step 3 of Round 2

- ❖ Shift right of multiplier



Step 4 of Round 2

Step 4 of Round 2

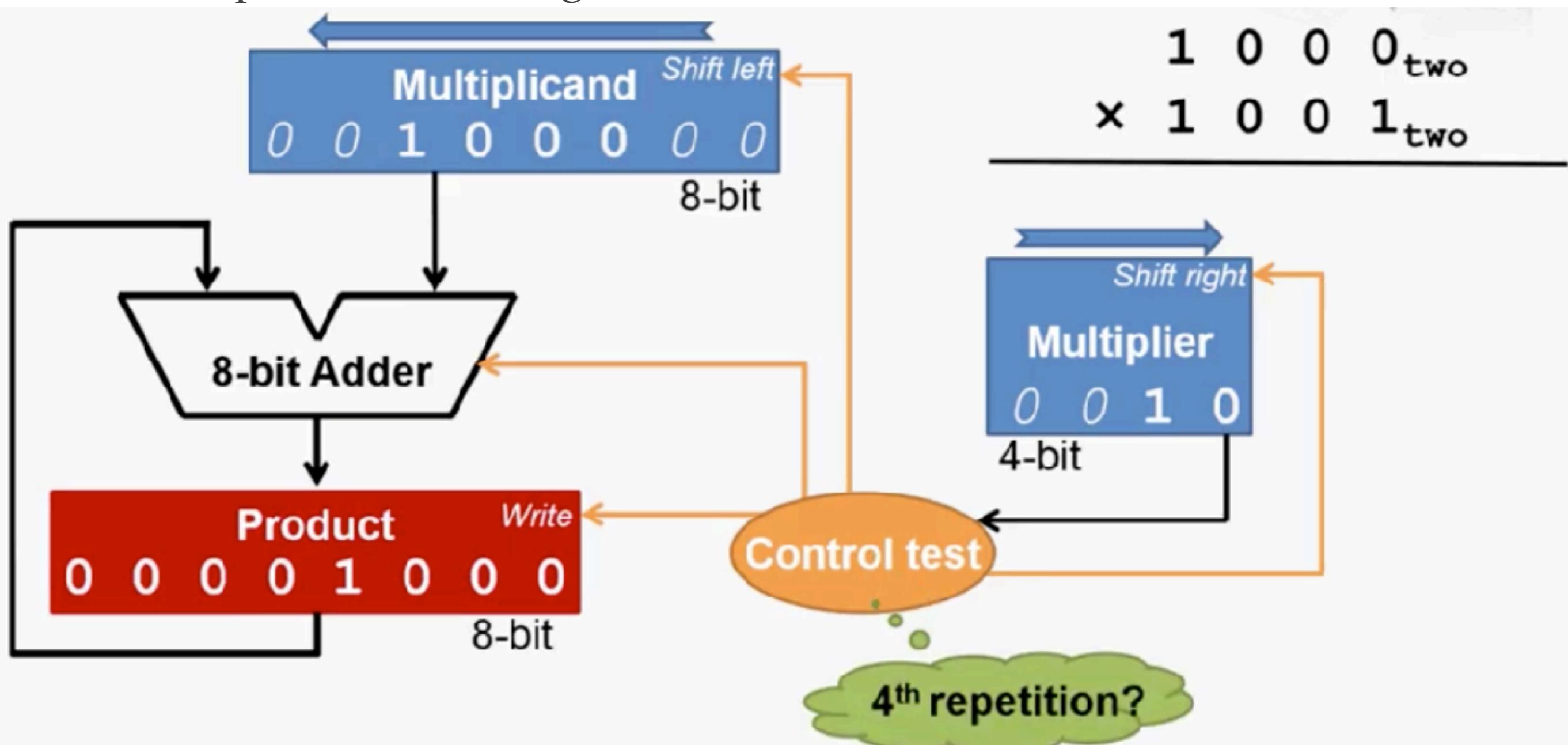
- ❖ What does this step do?

Step 4 of Round 2

- ❖ What does this step do?
 - ❖ Repetition checking!

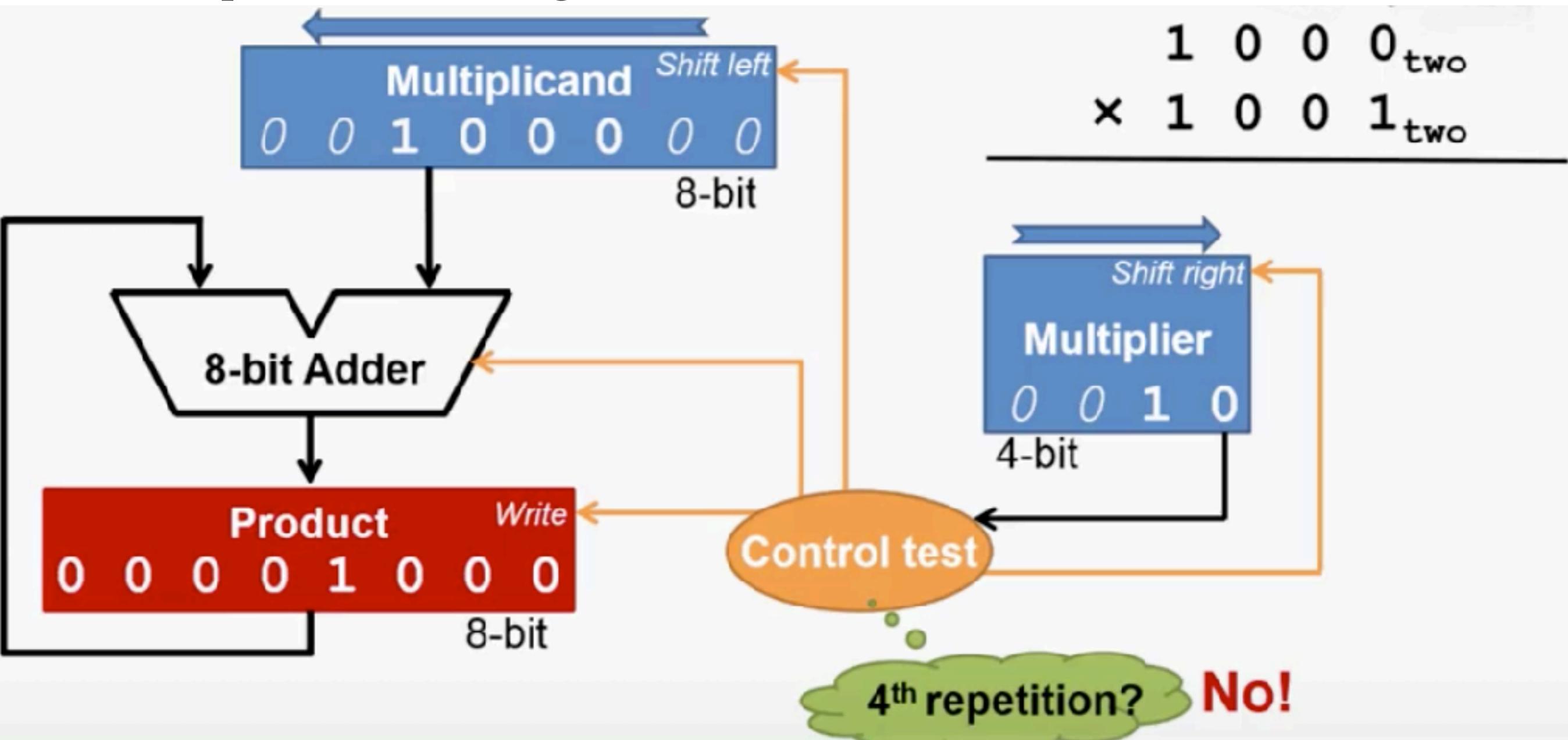
Step 4 of Round 2

- ❖ What does this step do?
 - ❖ Repetition checking!

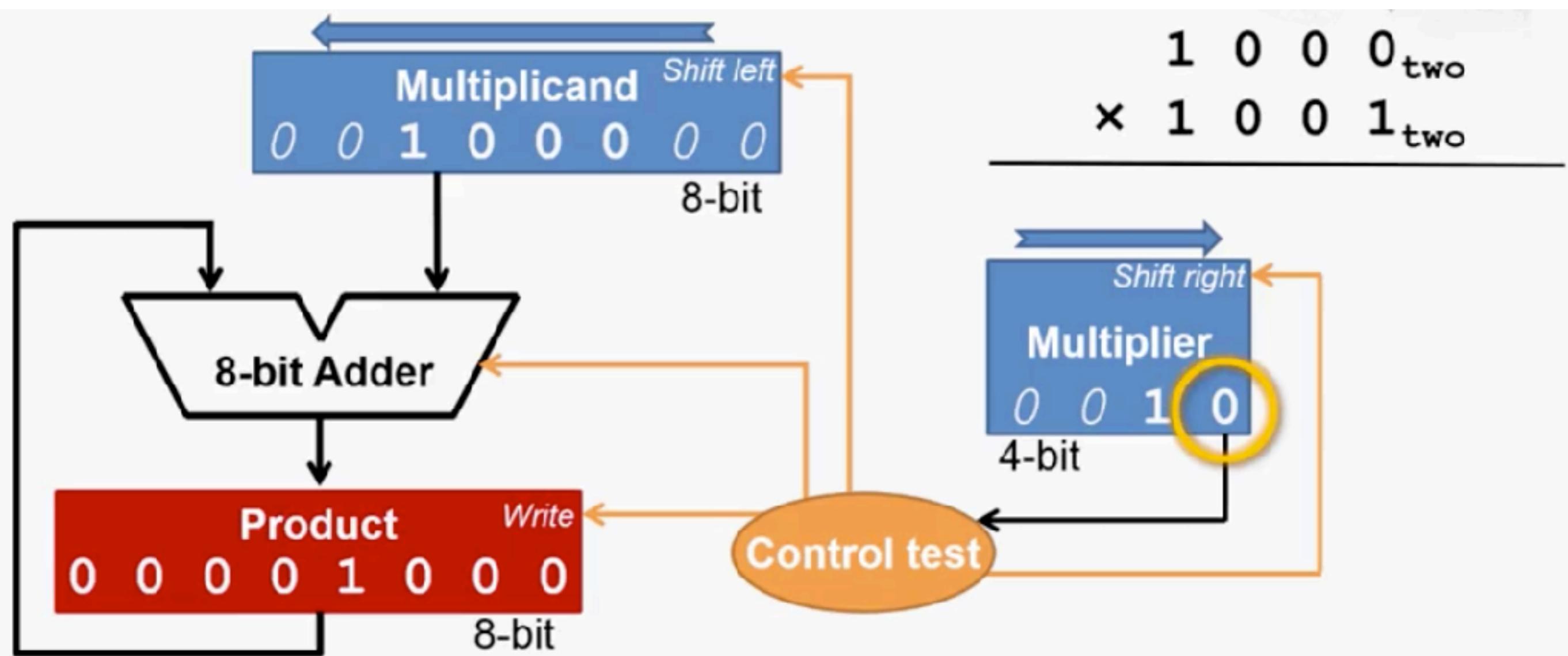


Step 4 of Round 2

- ❖ What does this step do?
 - ❖ Repetition checking!

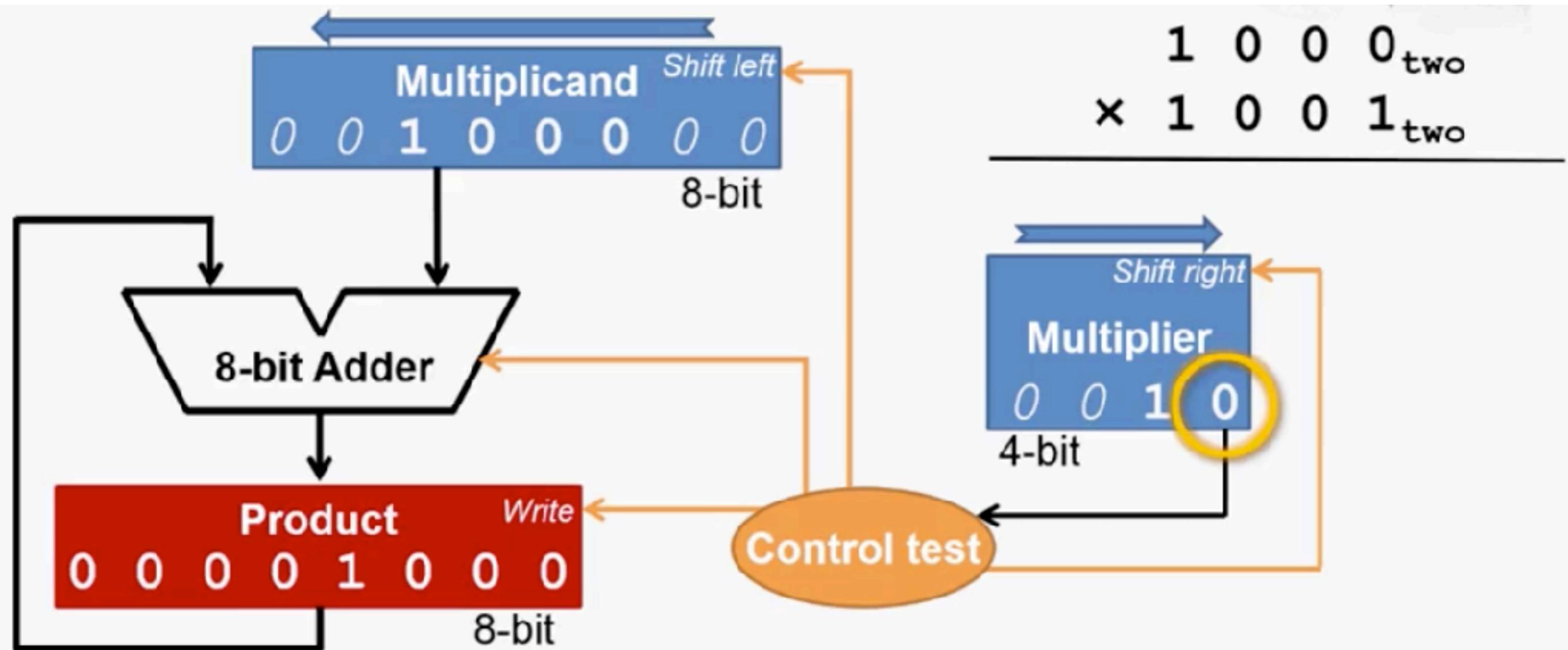


Step 1 of Round 3



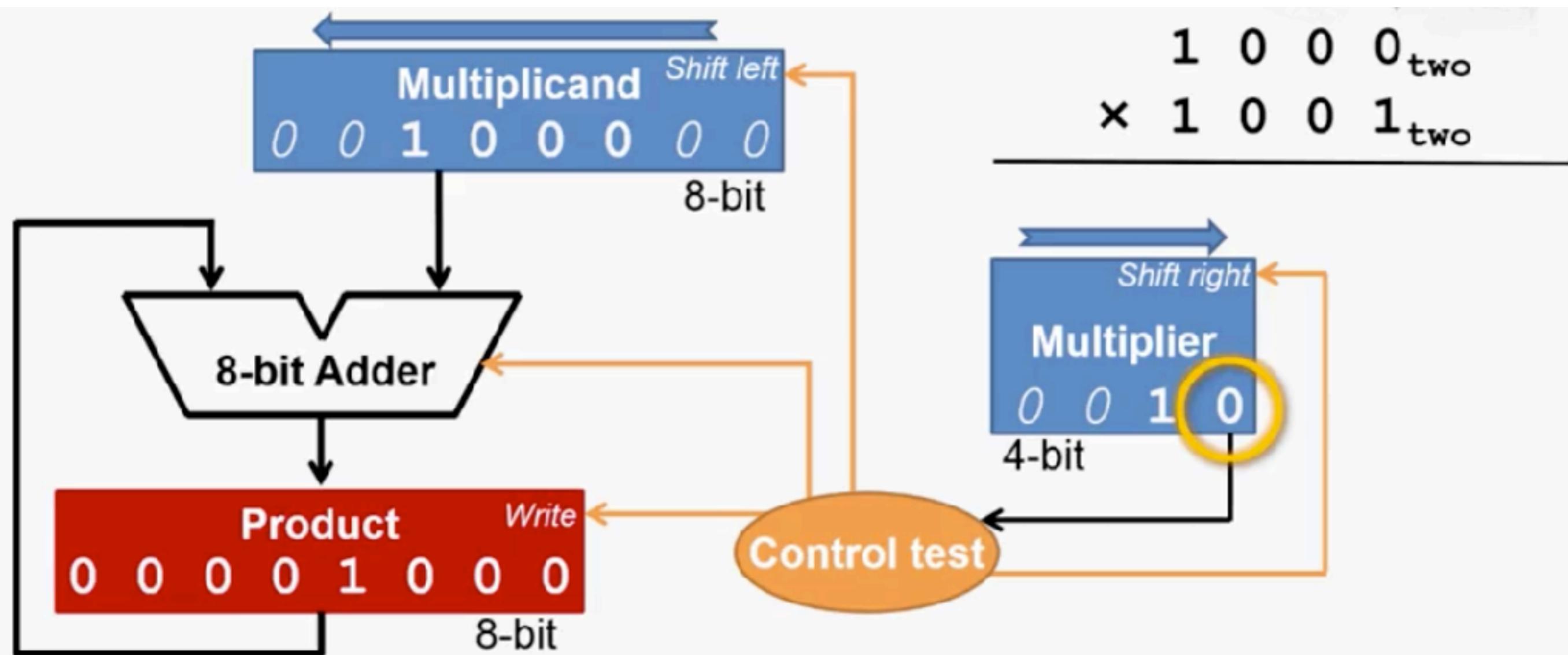
Step 1 of Round 3

- ❖ Checking the LSB of multiplier

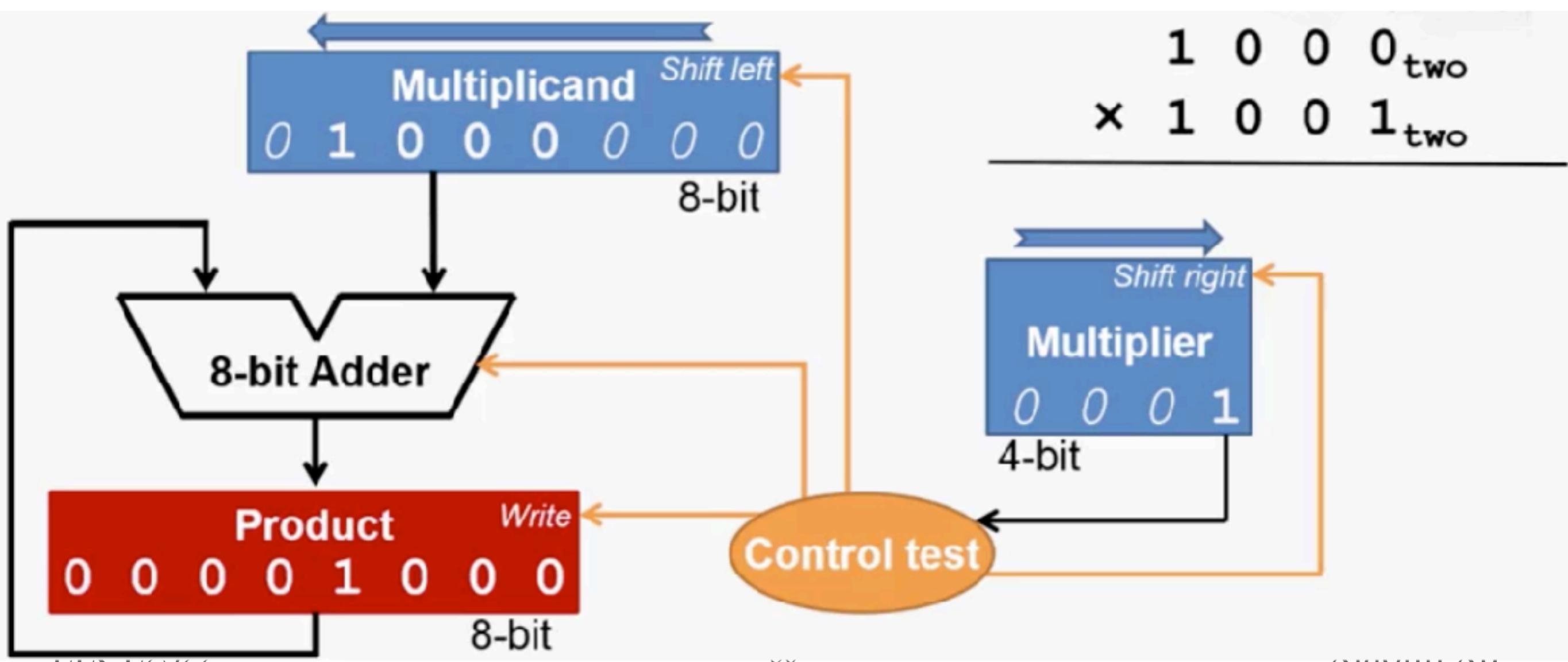


Step 1 of Round 3

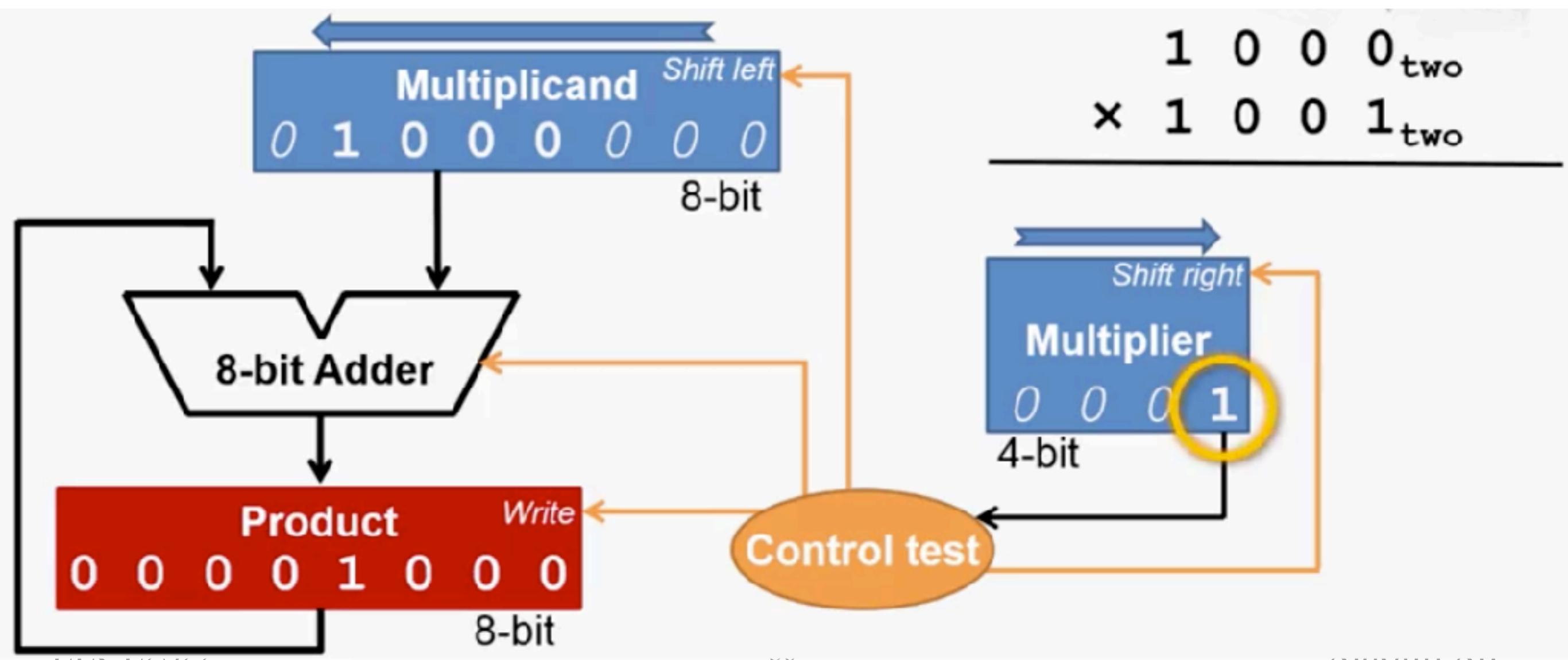
- ❖ Checking the LSB of multiplier
- ❖ Repeat in mind what happens for each step of round 3



Step 1 of Round 4

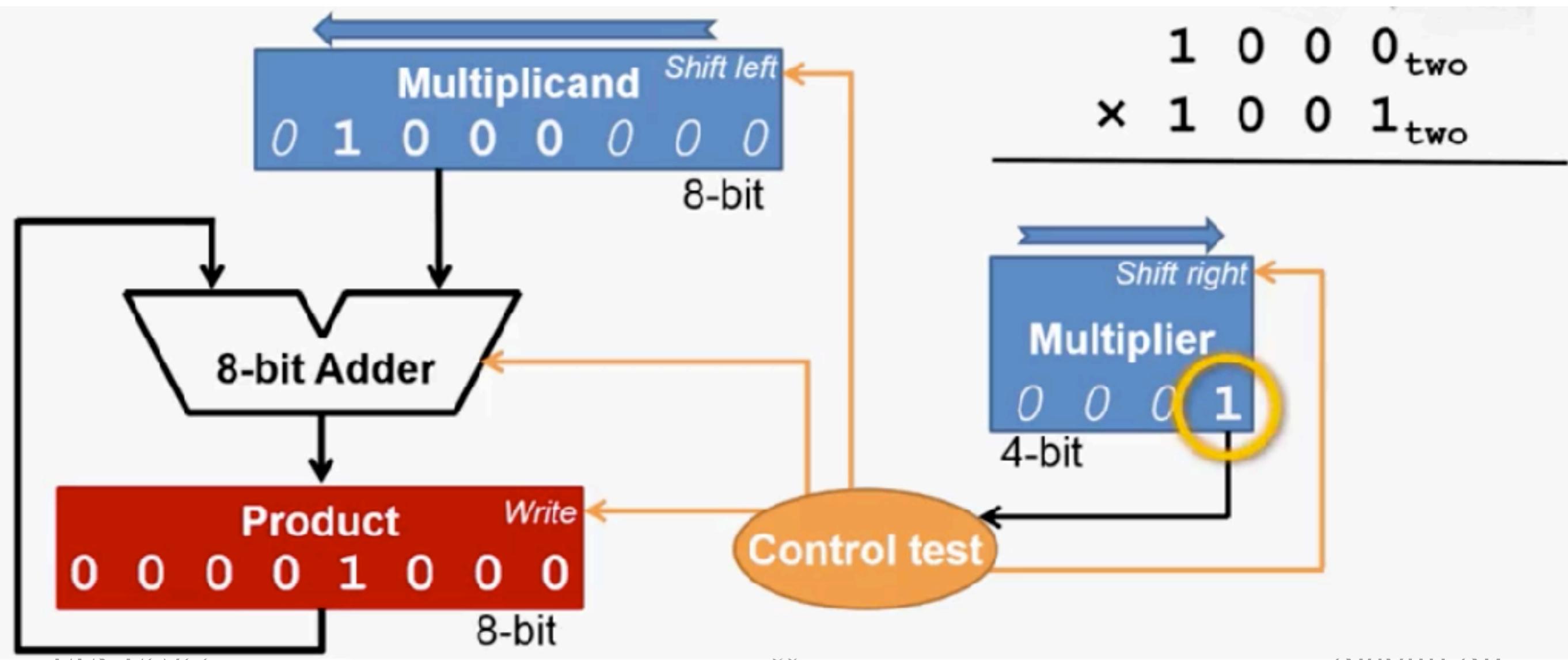


Step 1 of Round 4

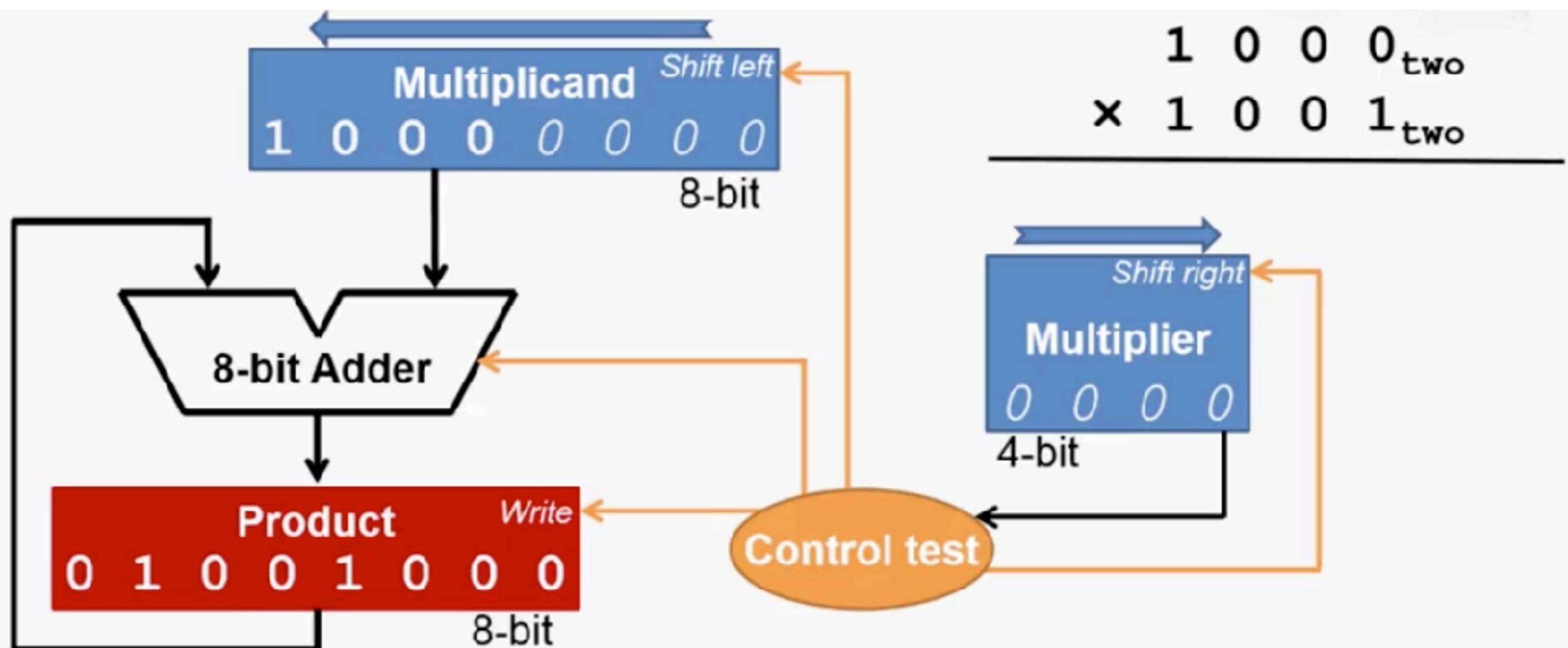


Step 1 of Round 4

- You should remember what is the step 1 now.

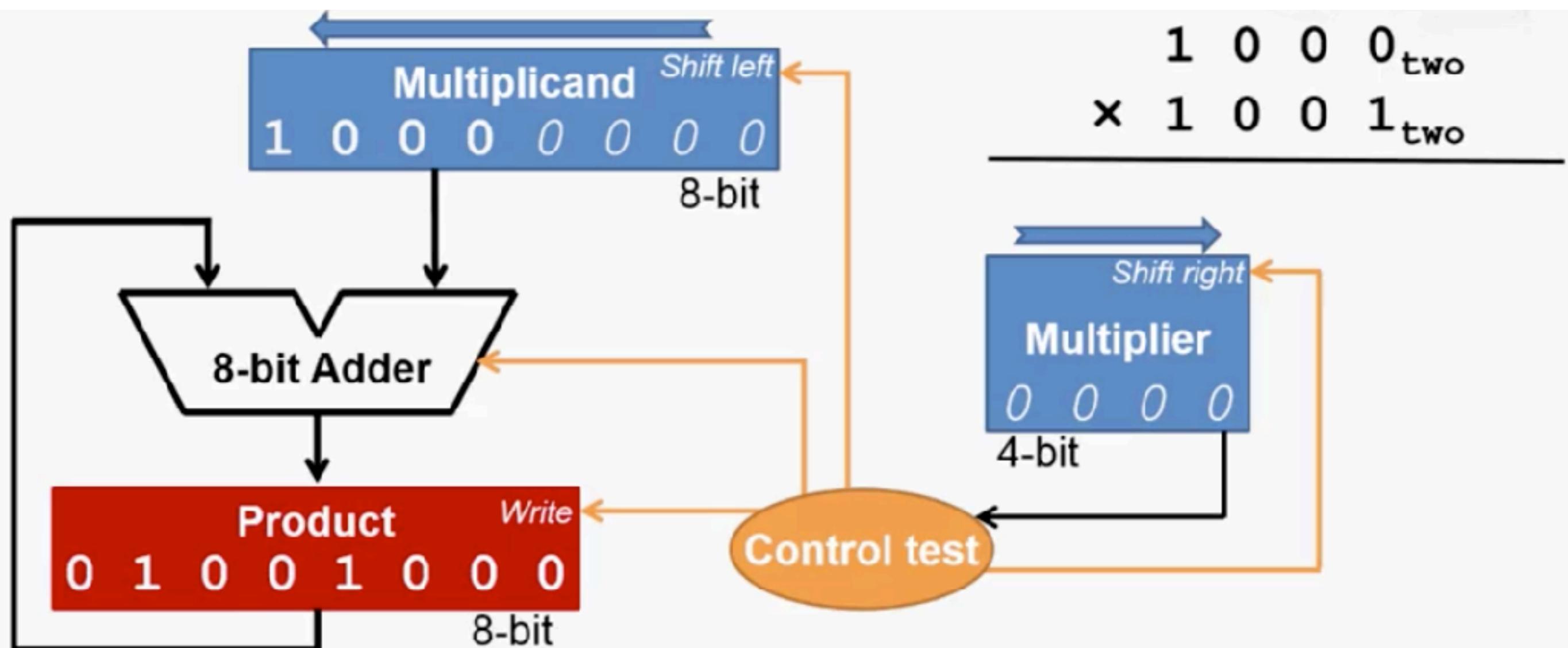


Step 4 of Round 4



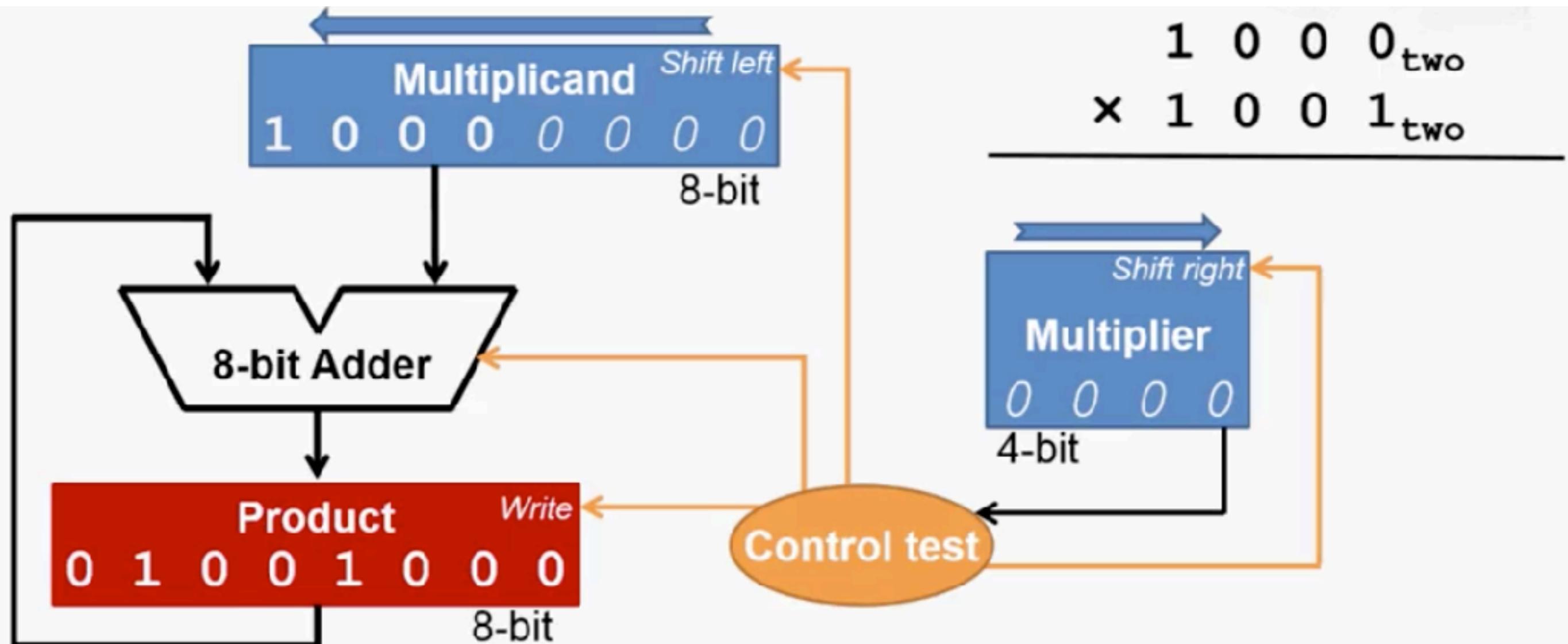
Step 4 of Round 4

- ❖ Multiplicand now has 10000000 (note the difference)



Step 4 of Round 4

- ❖ Multiplicand now has 10000000 (note the difference)
- ❖ All rounds done! However, hardware does not know it!

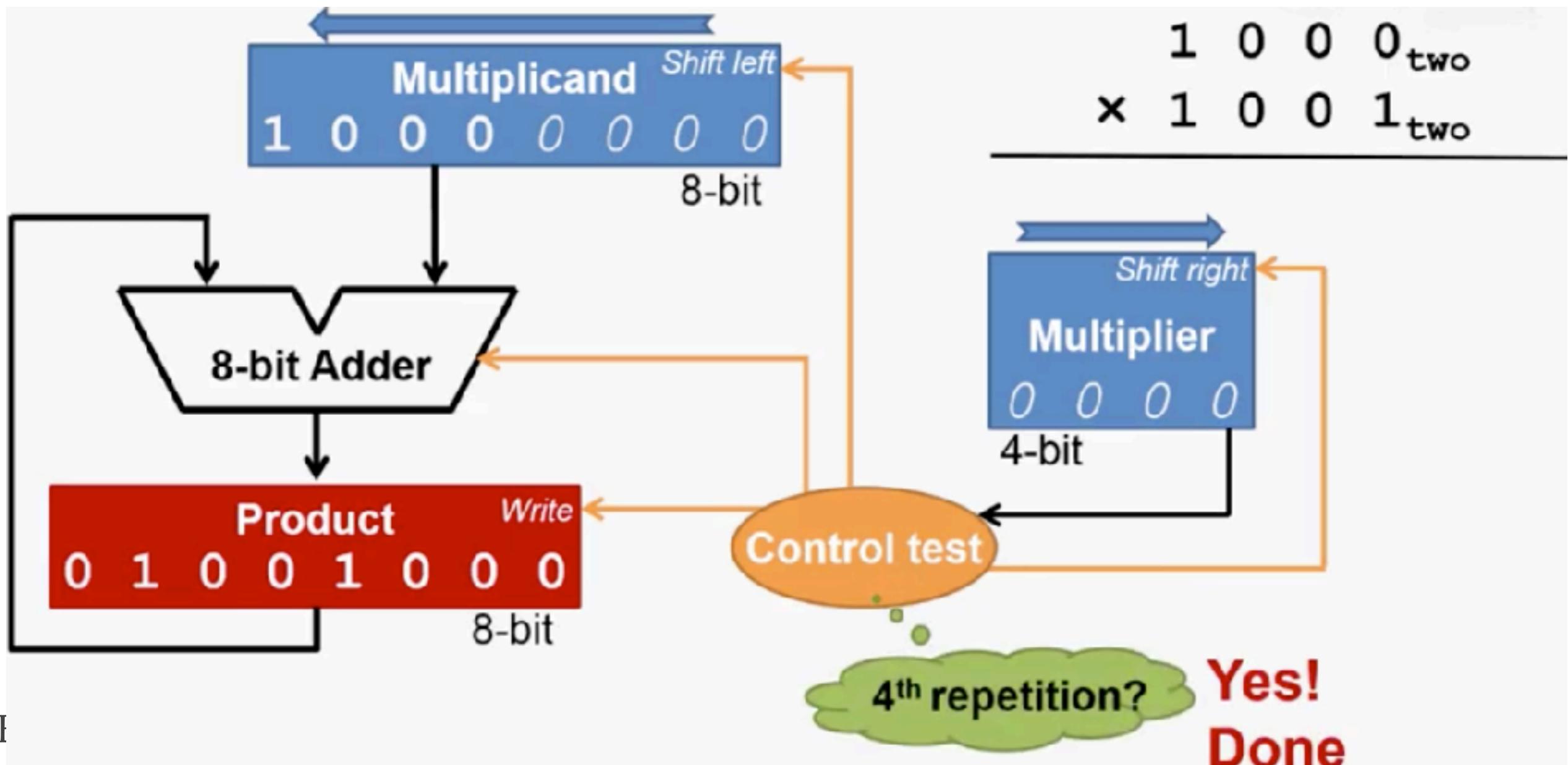


Step 4 of Round 4

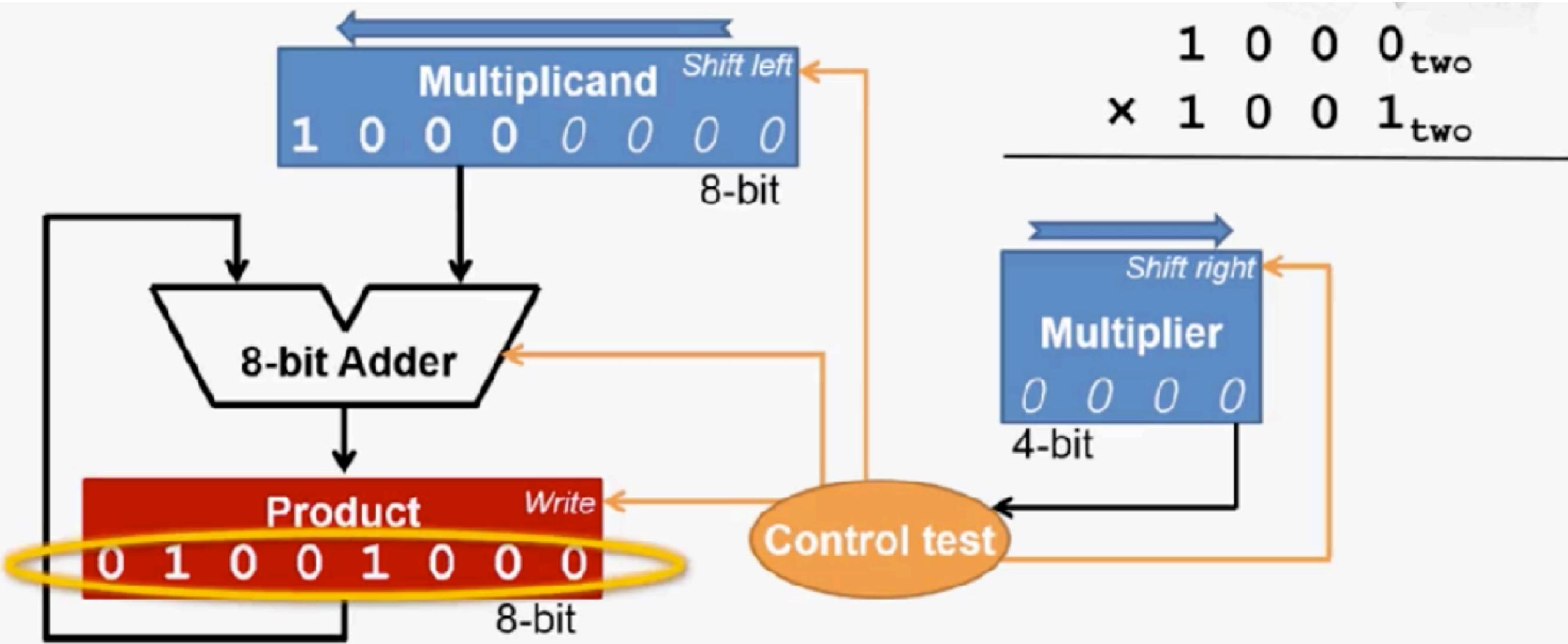
- ❖ Multiplicand now has **10000000** (note the difference)
- ❖ All rounds done! However, hardware does not know it!

Step 4 of Round 4

- ❖ Multiplicand now has 10000000 (note the difference)
- ❖ All rounds done! However, hardware does not know it!

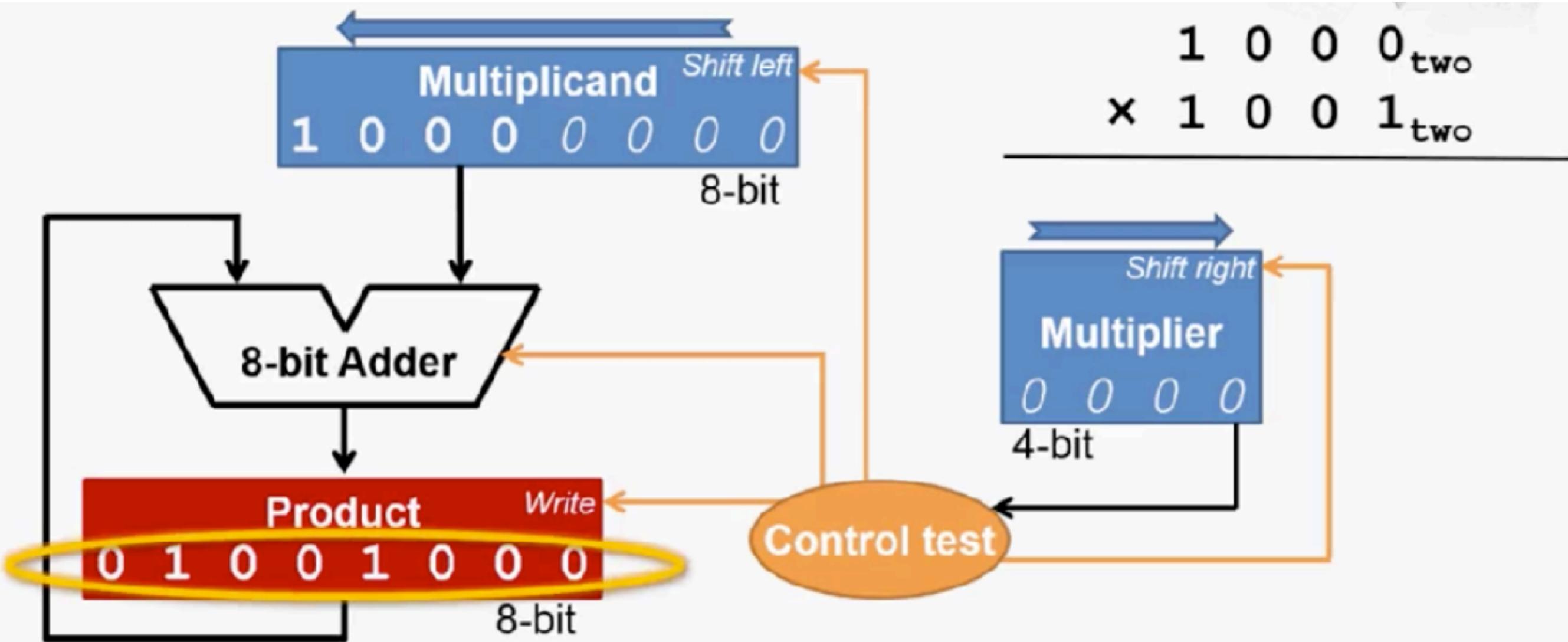


Results from All the 4 Rounds



Results from All the 4 Rounds

- ❖ 01001000, correct!



Summary

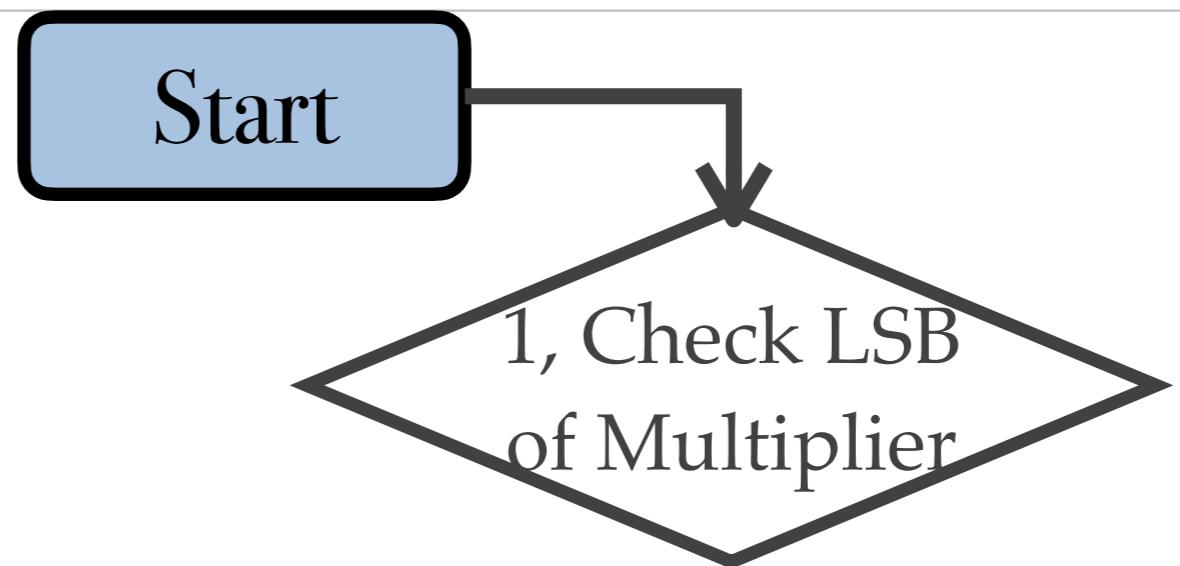
- ❖ Very straightforward hardware implementation
 - ❖ Each round repeated the same operation
 - ❖ No manual change in between
 - ❖ Simply left/ right shifting
 - ❖ Simply checking repetition
 - ❖ Adder does not need to worries about aligning the two inputs

What about N-bit Multiplication?

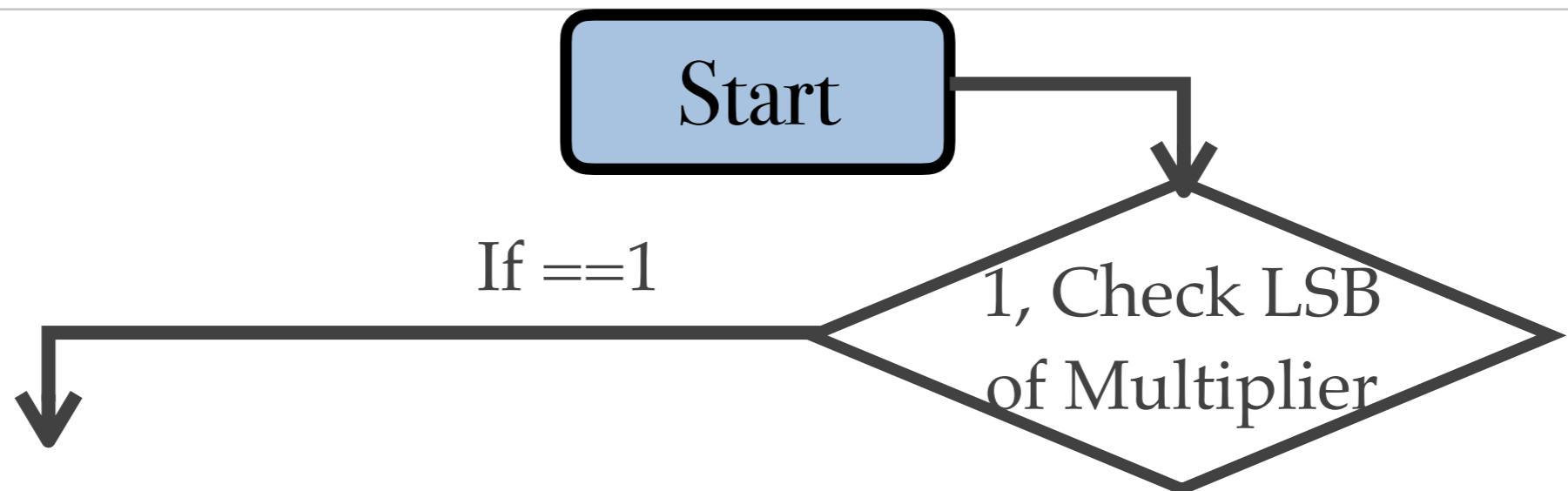
What about N-bit Multiplication?

Start

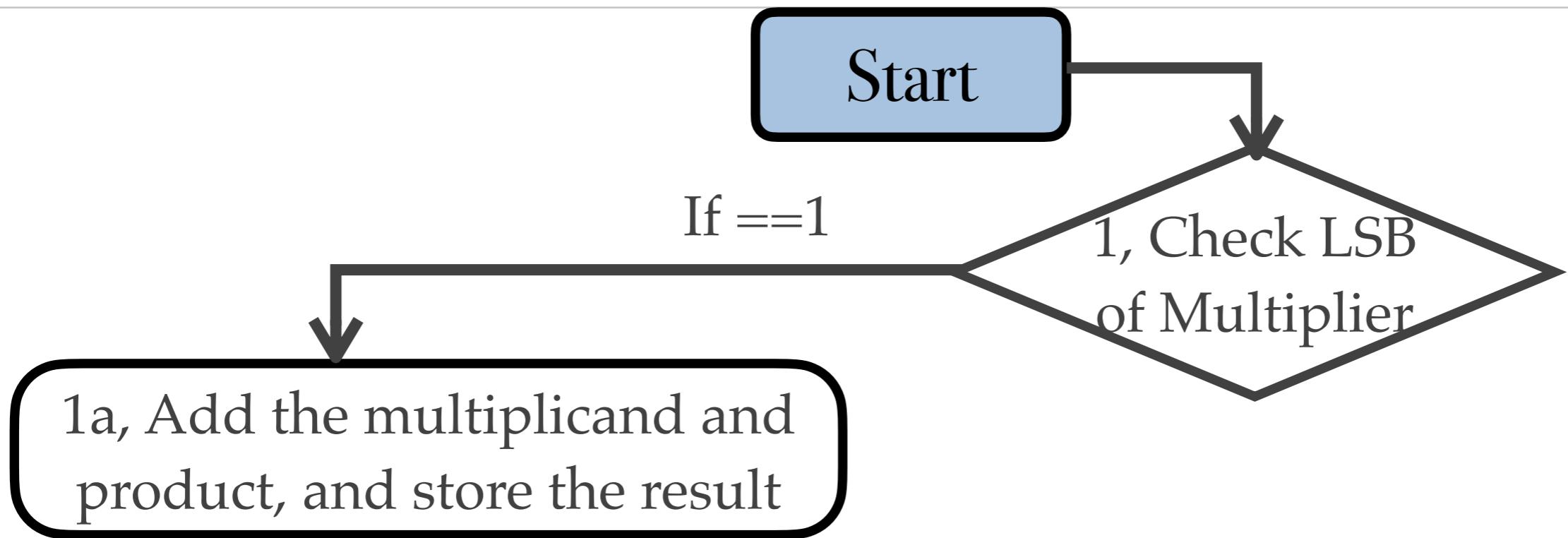
What about N-bit Multiplication?



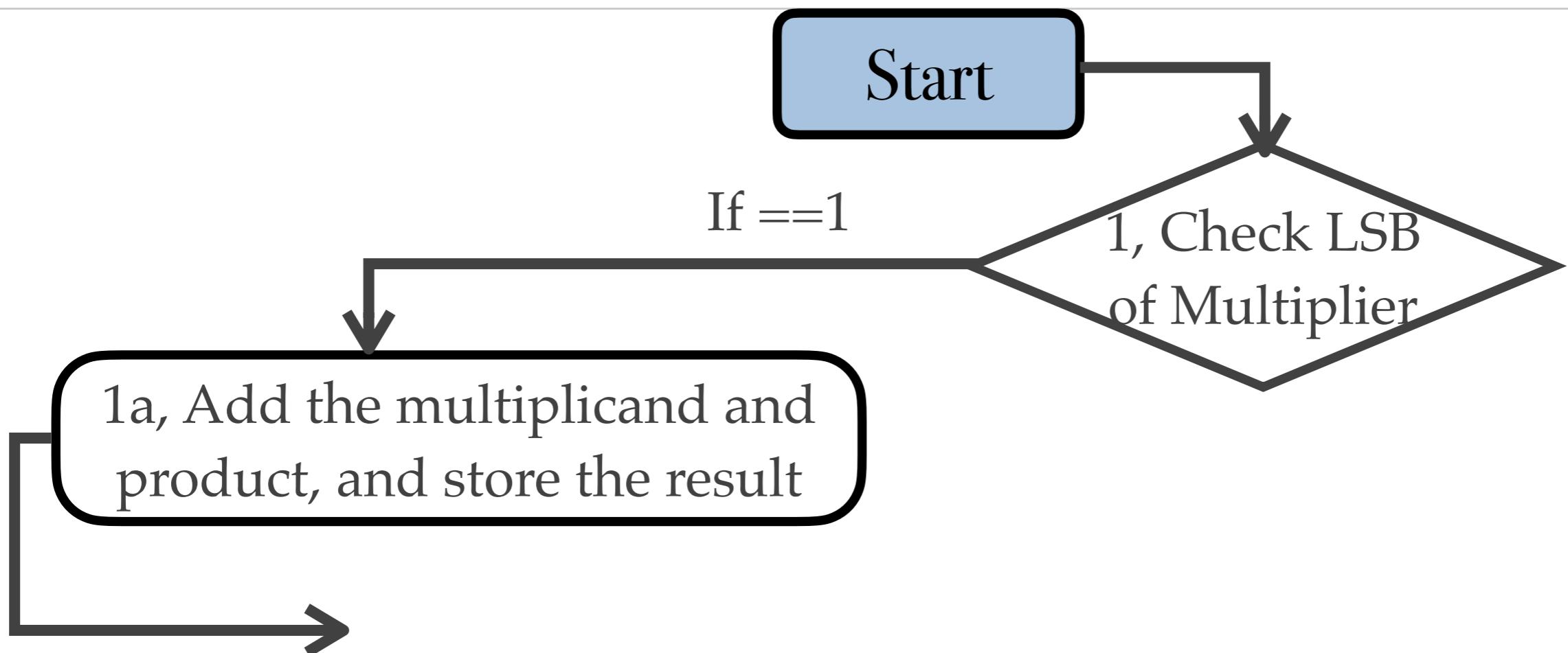
What about N-bit Multiplication?



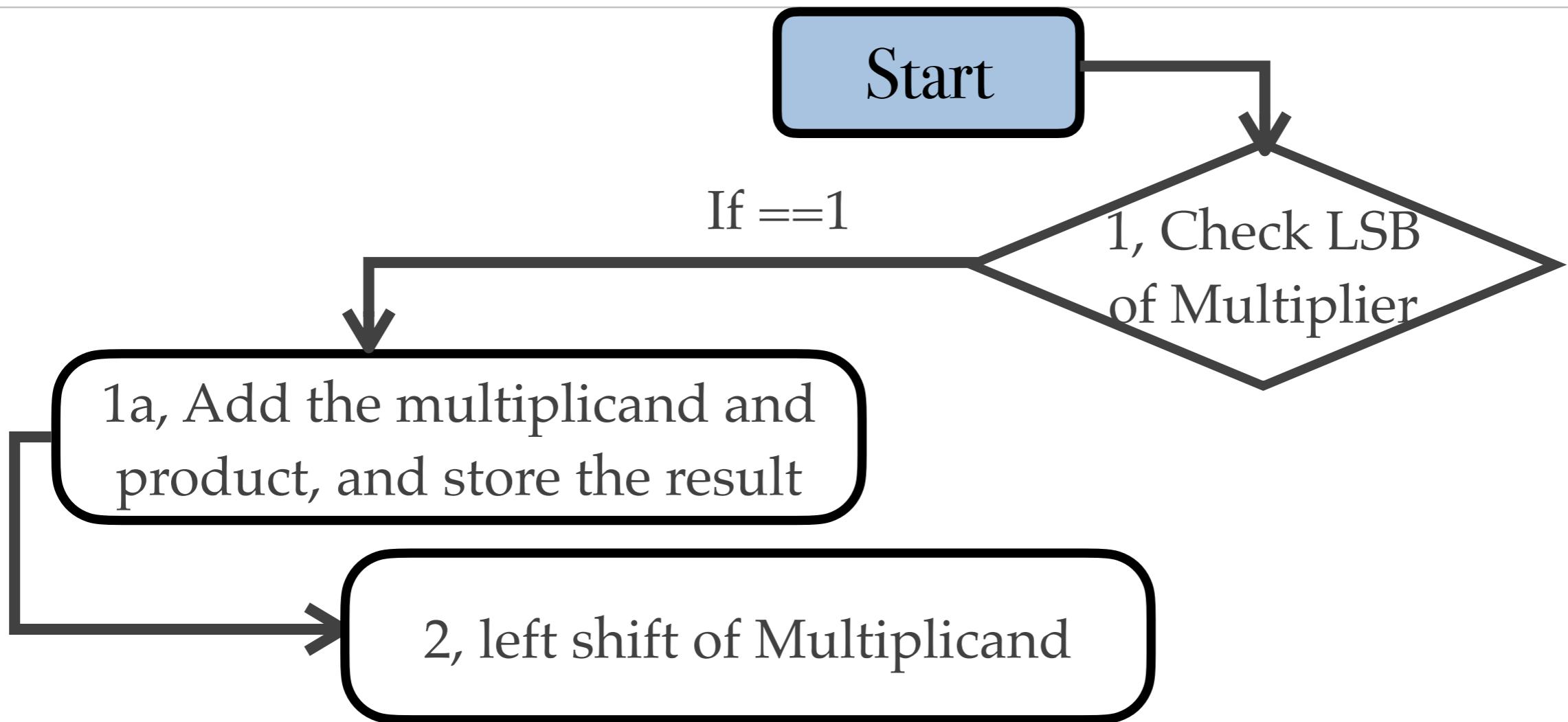
What about N-bit Multiplication?



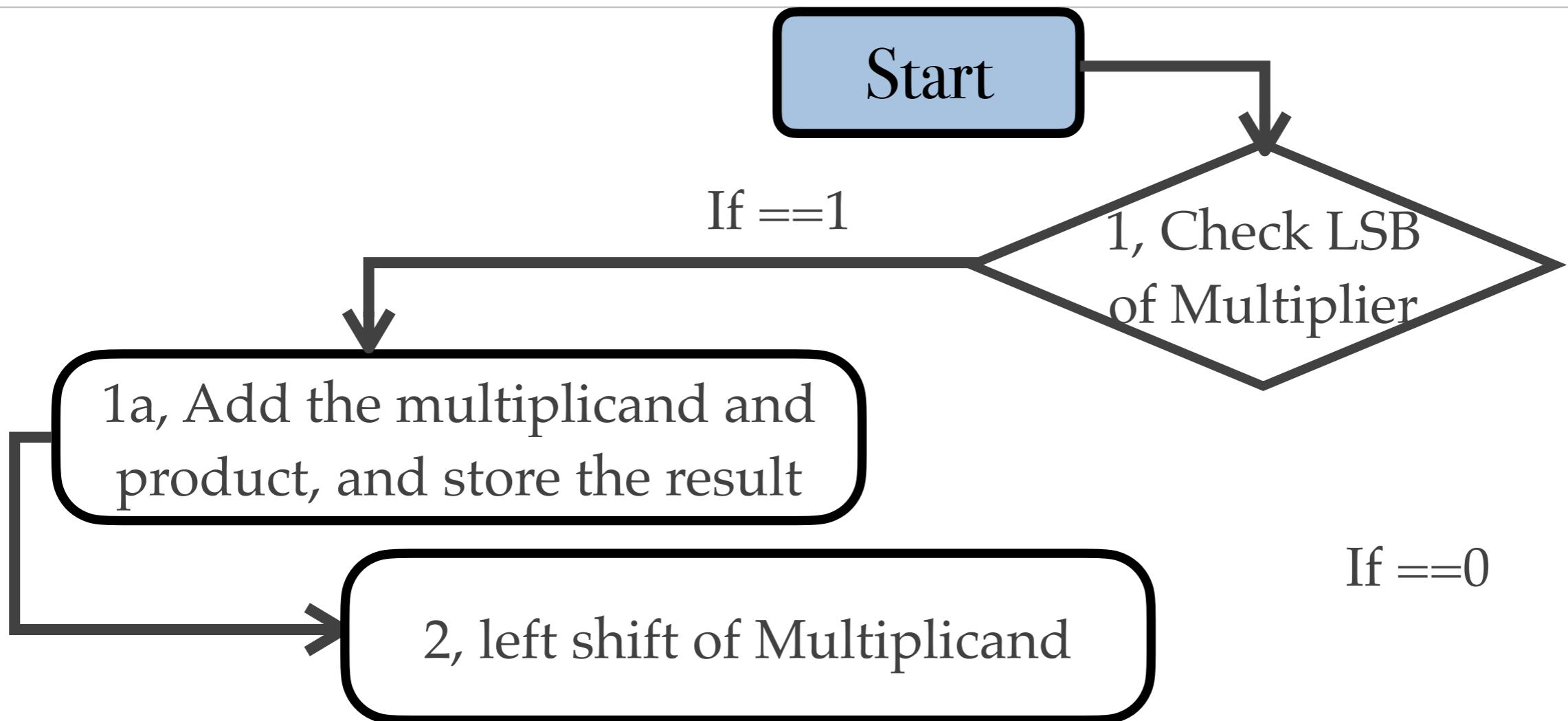
What about N-bit Multiplication?



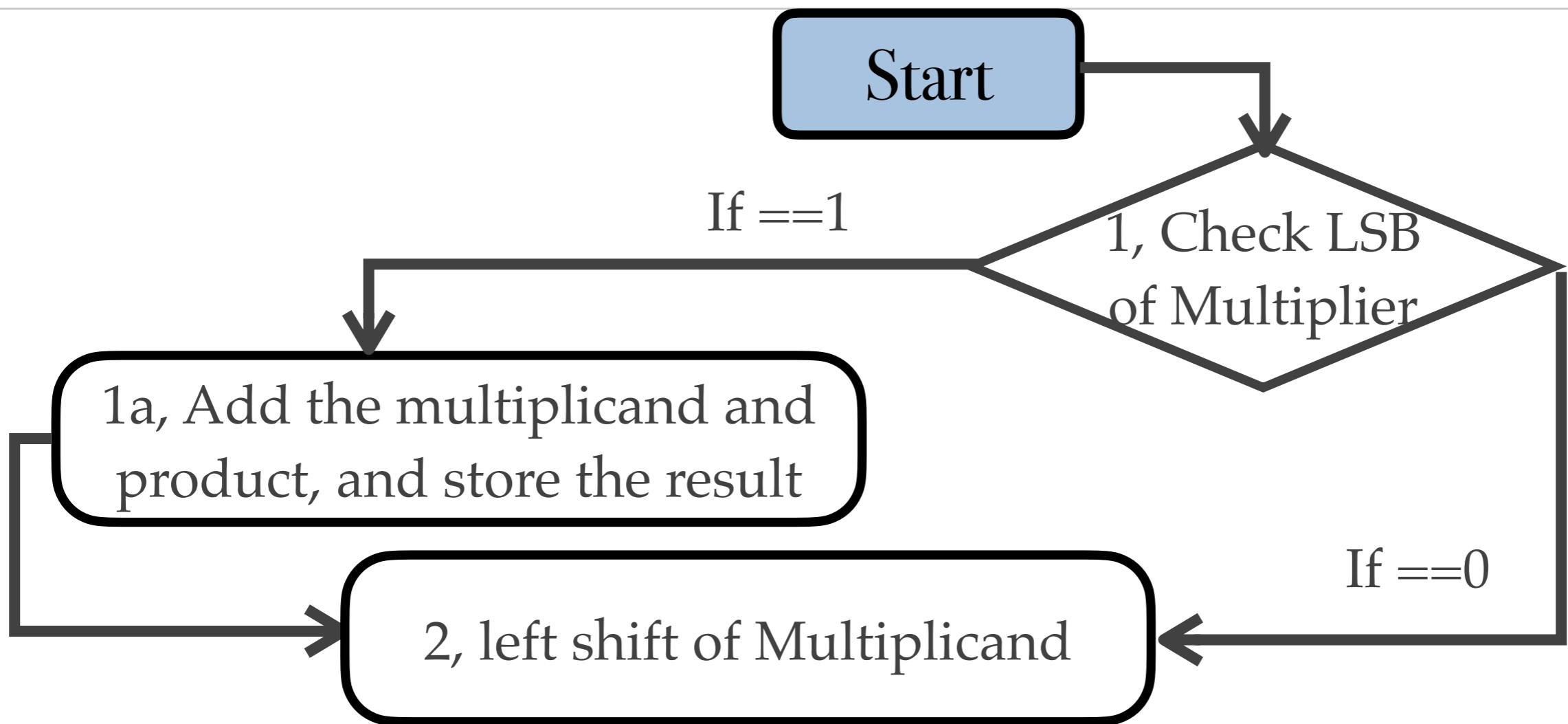
What about N-bit Multiplication?



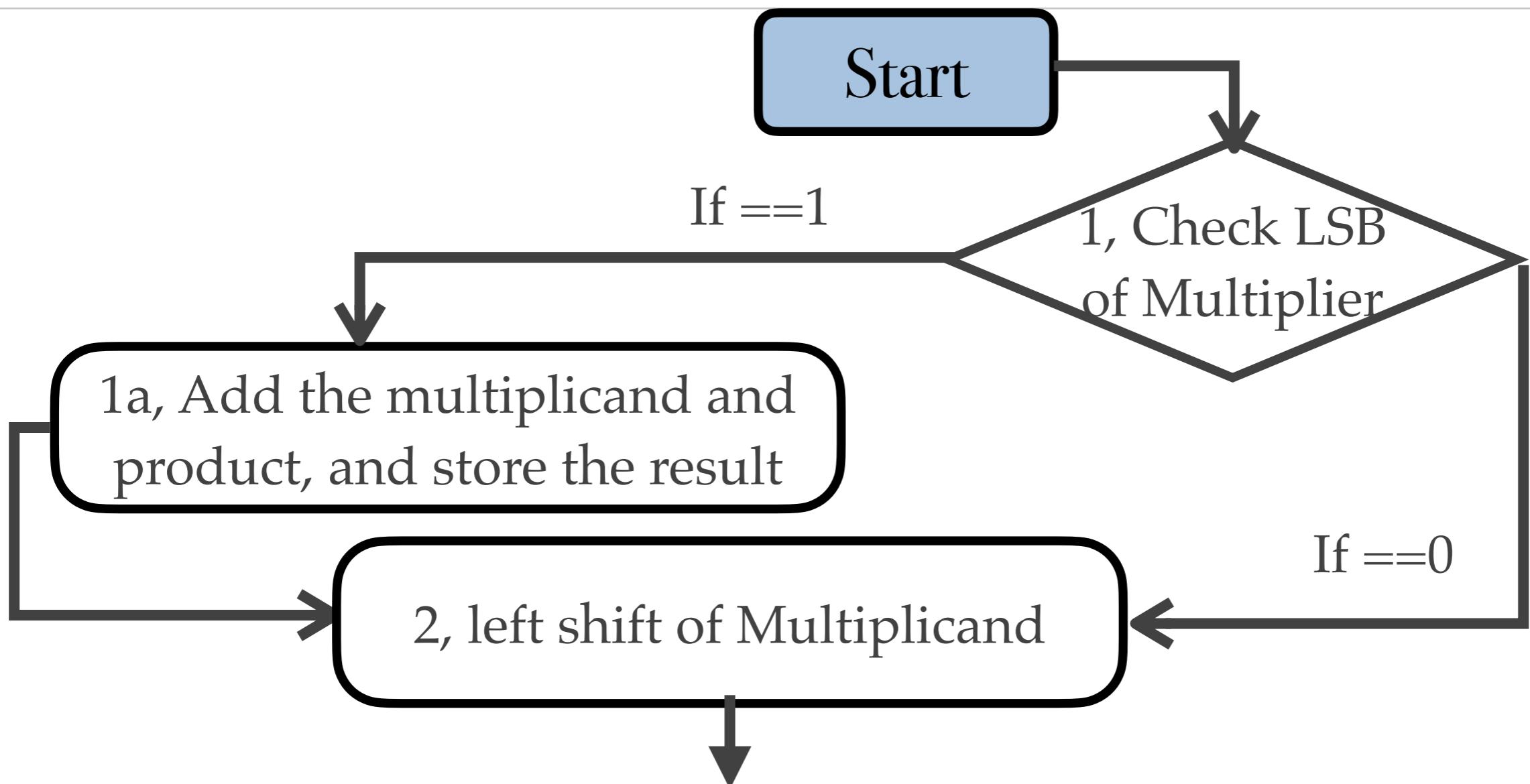
What about N-bit Multiplication?



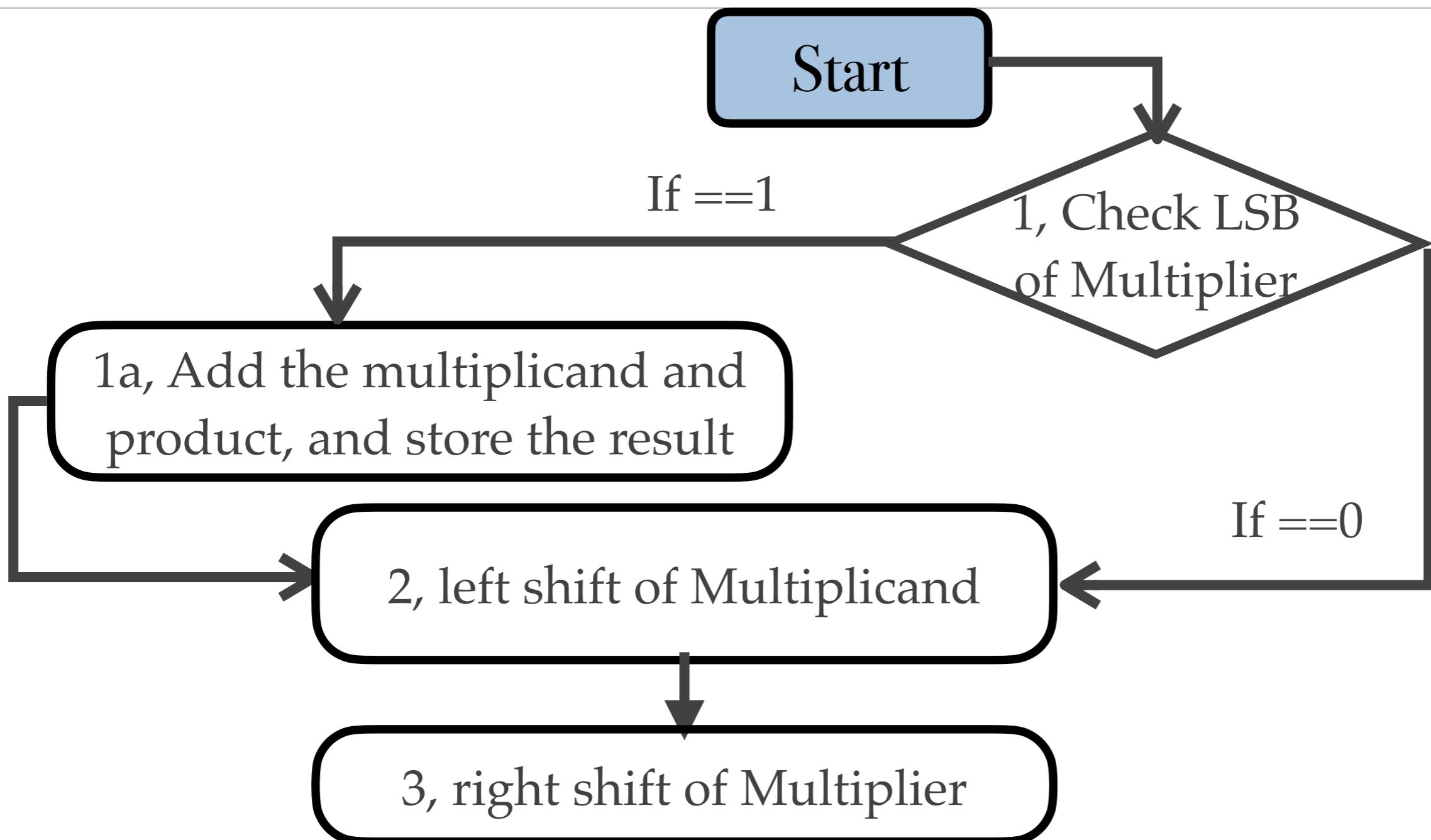
What about N-bit Multiplication?



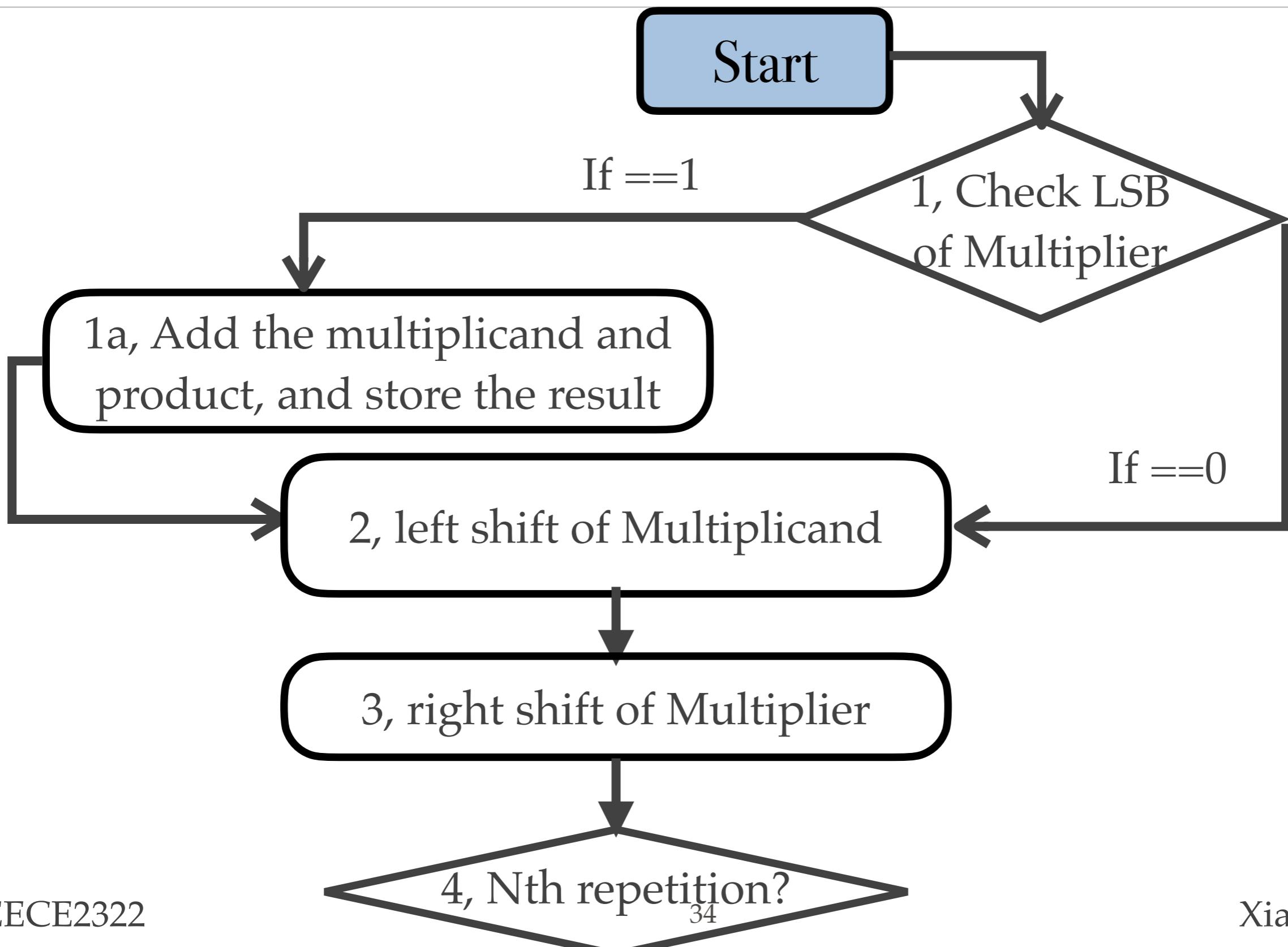
What about N-bit Multiplication?



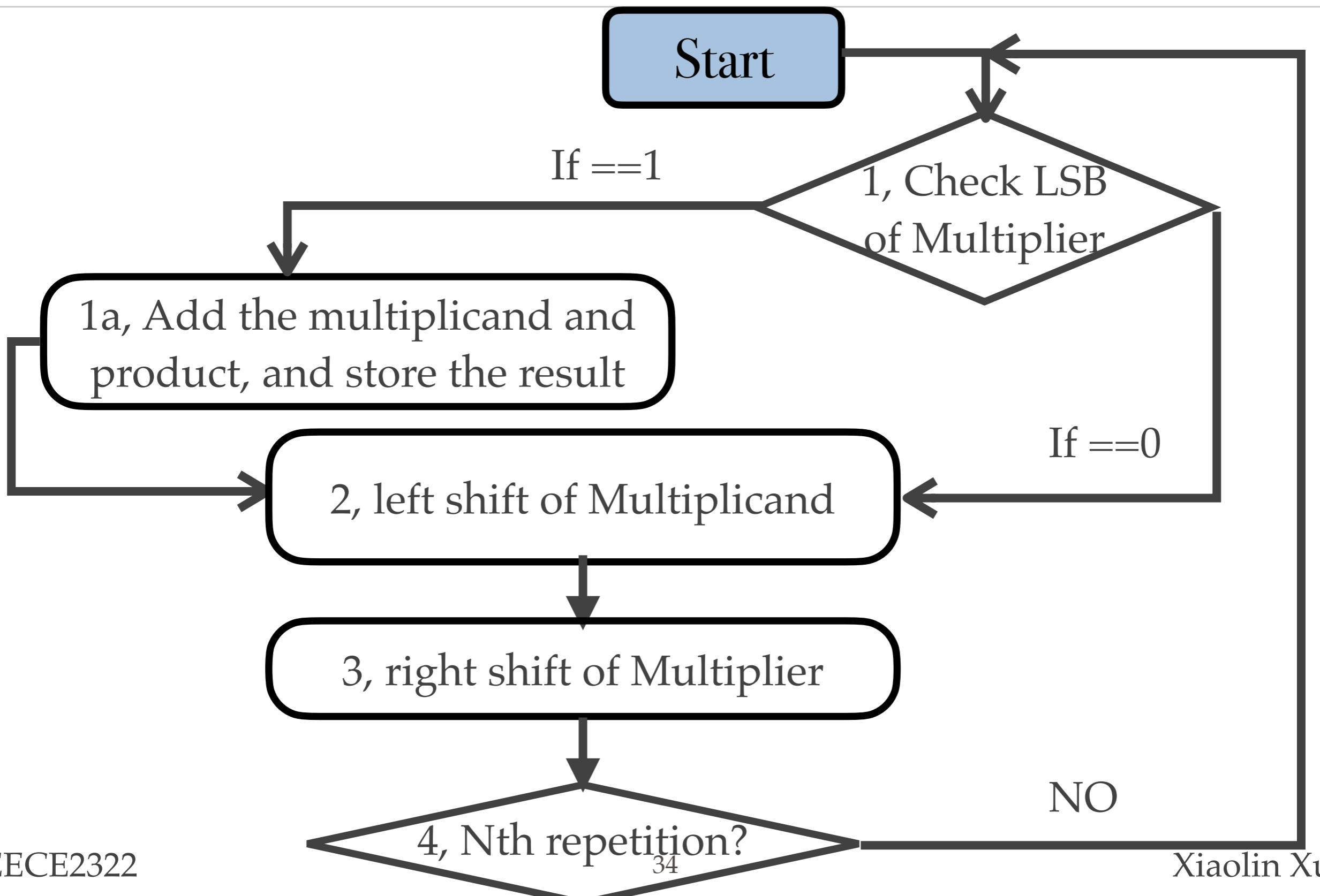
What about N-bit Multiplication?



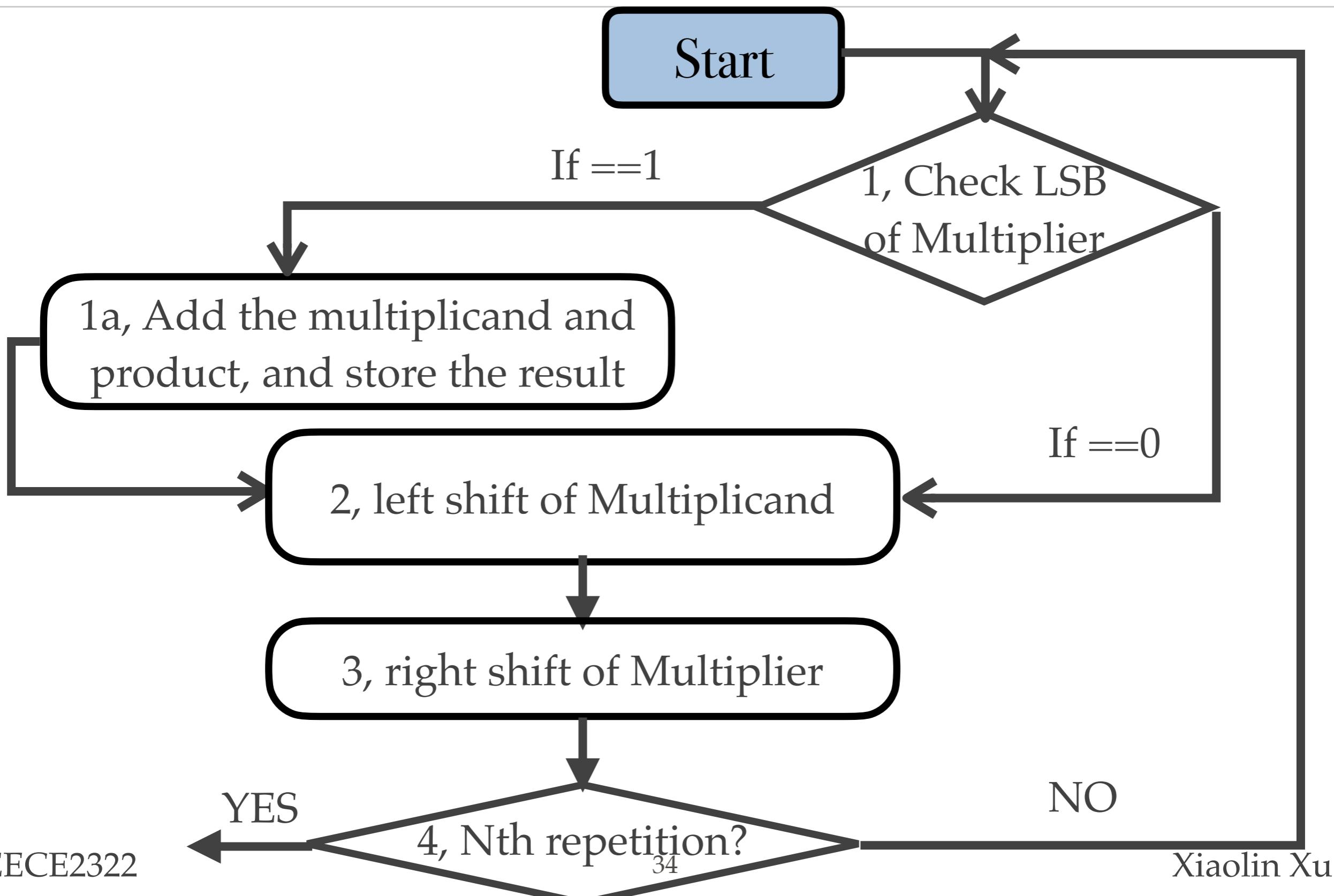
What about N-bit Multiplication?



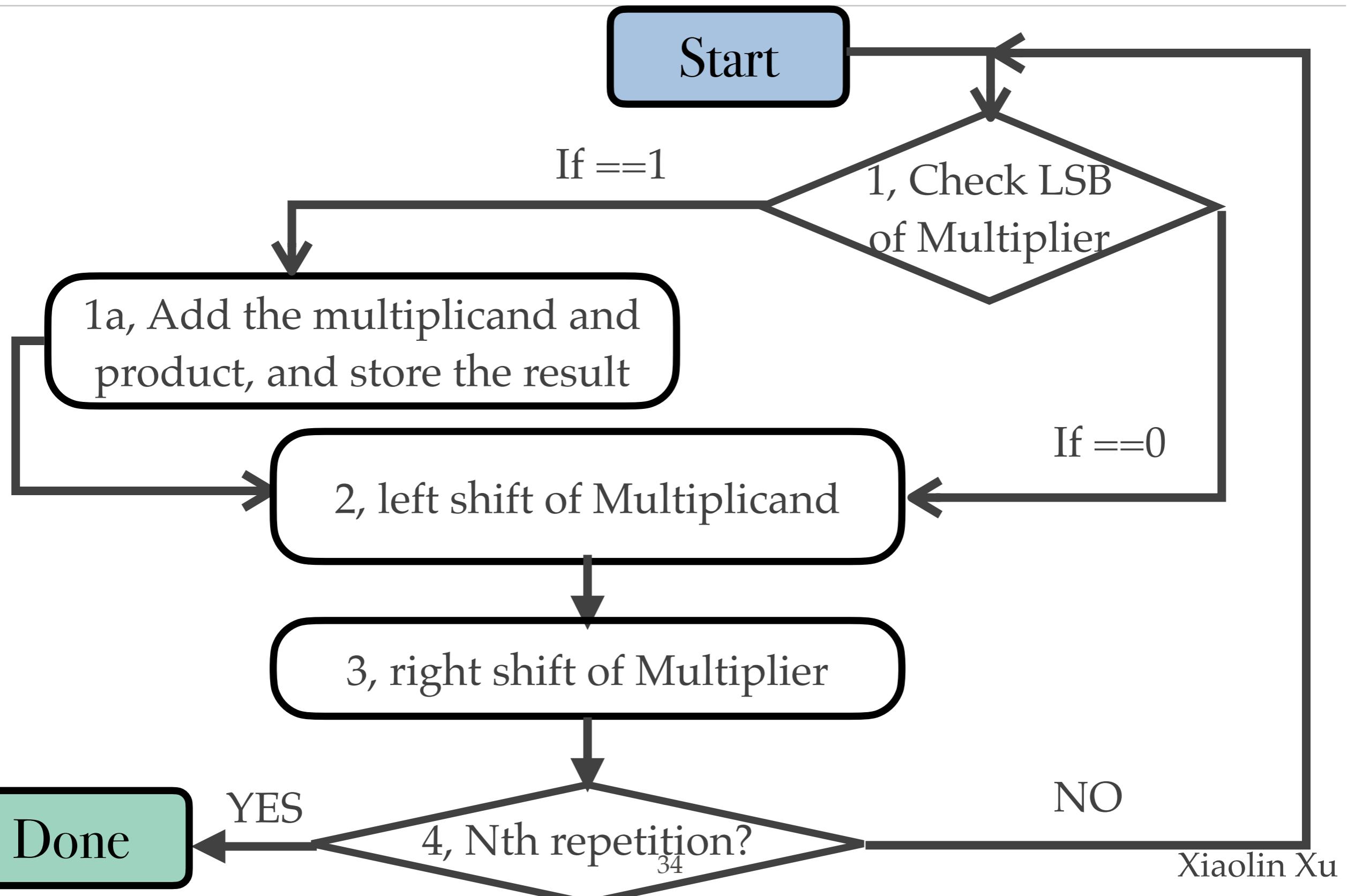
What about N-bit Multiplication?



What about N-bit Multiplication?



What about N-bit Multiplication?



EECE 2322: Fundamentals of Digital Design and Computer Organization

Lecture 5_2: MIPS ISA

Xiaolin Xu

Department of ECE
Northeastern University