CS/ECE 552: Mul/Div/FP

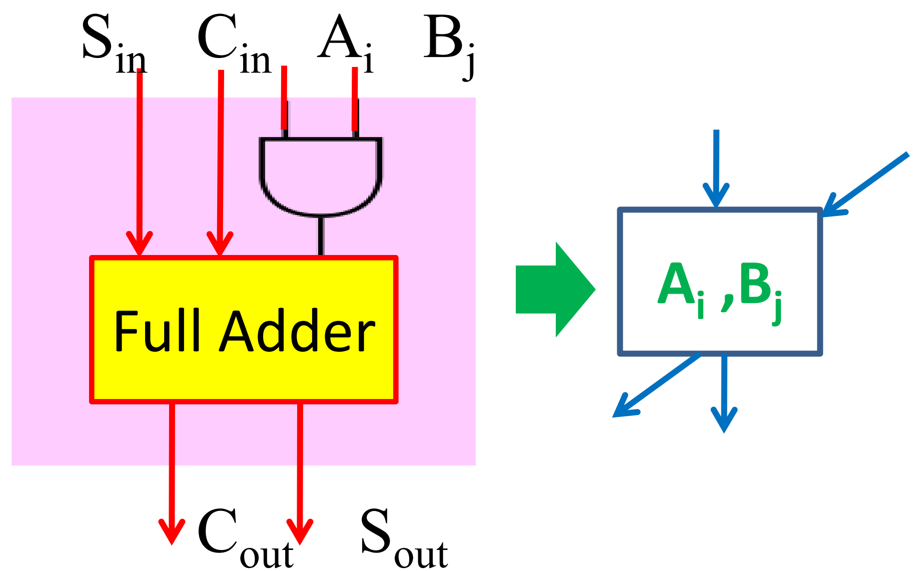
Prof. Matthew D. Sinclair

Lecture notes based in part on slides created by Mikko Lipasti, Mark Hill, David Wood, Guri Sohi, John Shen, Josh San Miguel, and Jim Smith

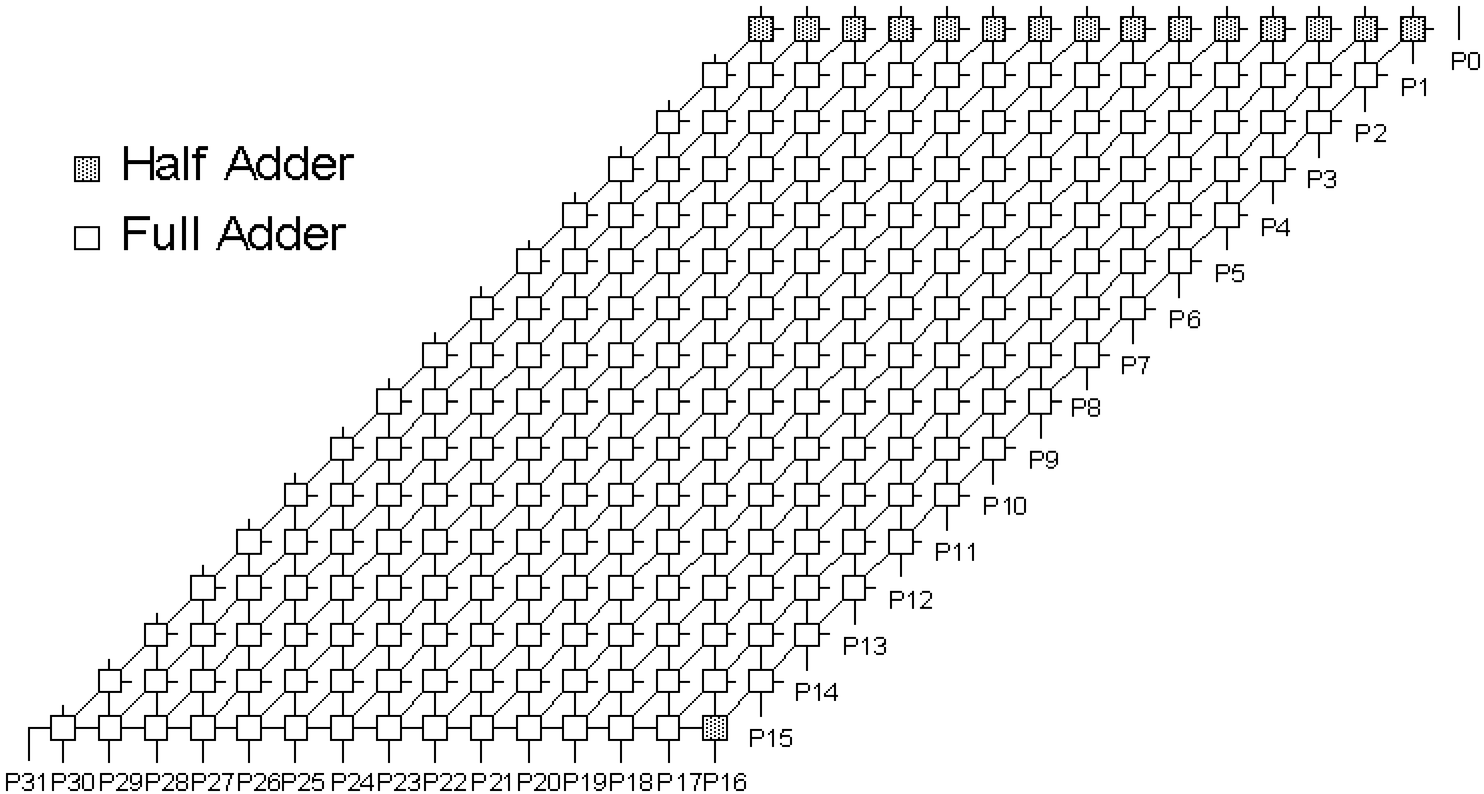
Array Multiplier

				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	1	0	0	1	0	0	0

- Adding all partial products simultaneously using an array of basic cells



16-bit Array Multiplier

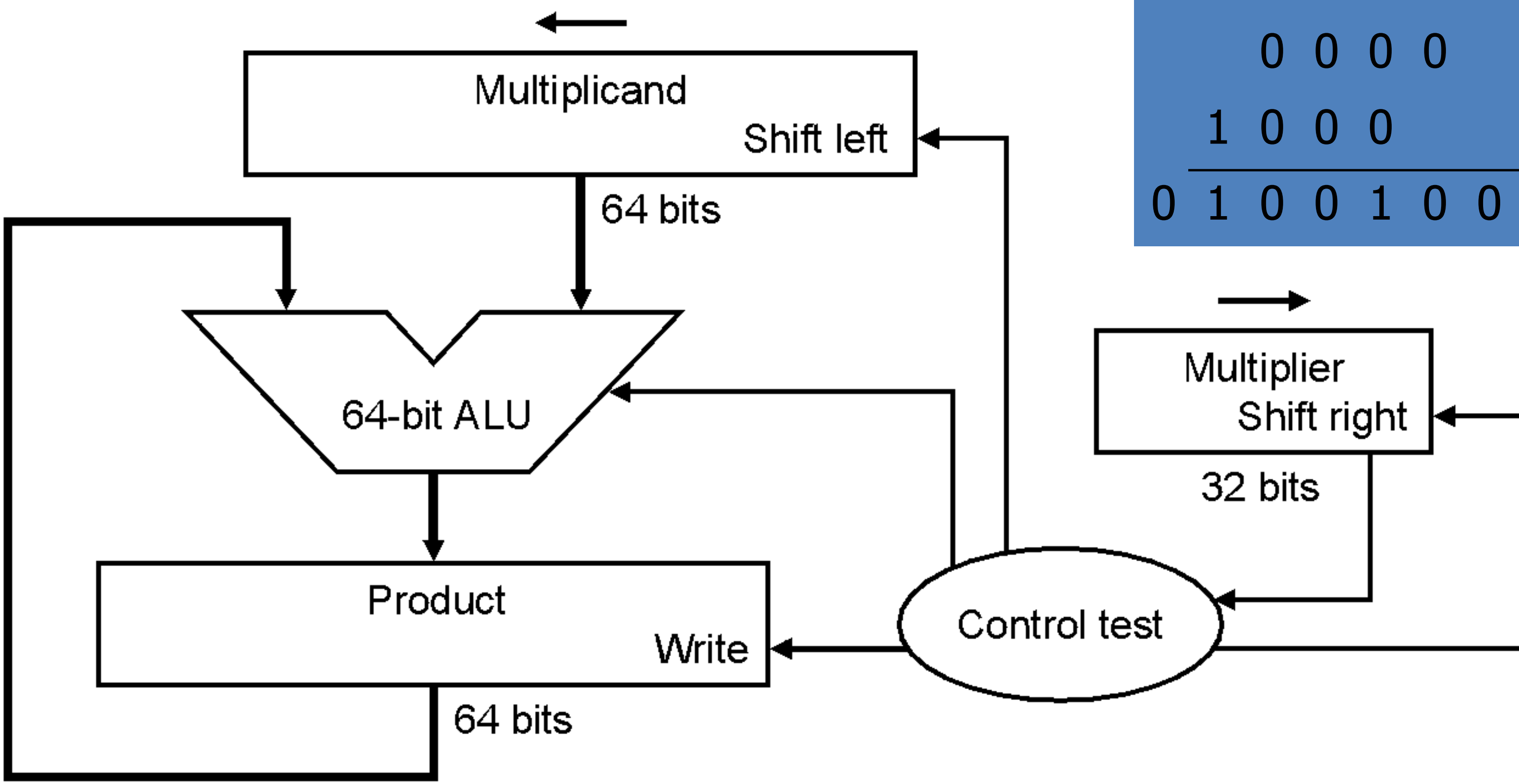


+ Conceptually straightforward
- Fairly expensive hardware
- Integer multiplies relatively rare

[Source: J. Hayes, Univ. of Michigan]

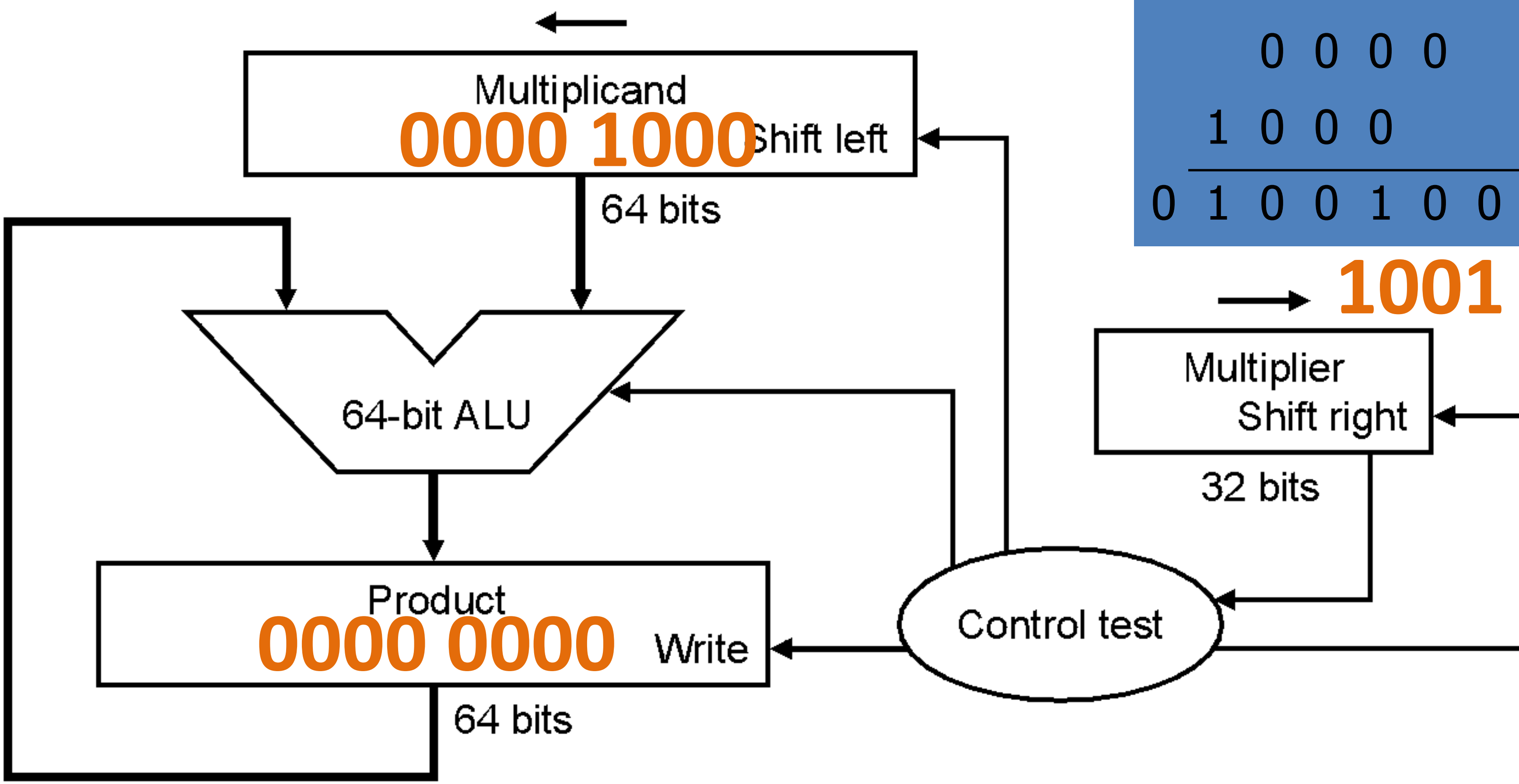
Multicycle Multiplier

```
        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 1 0 0 1 0 0 0
```

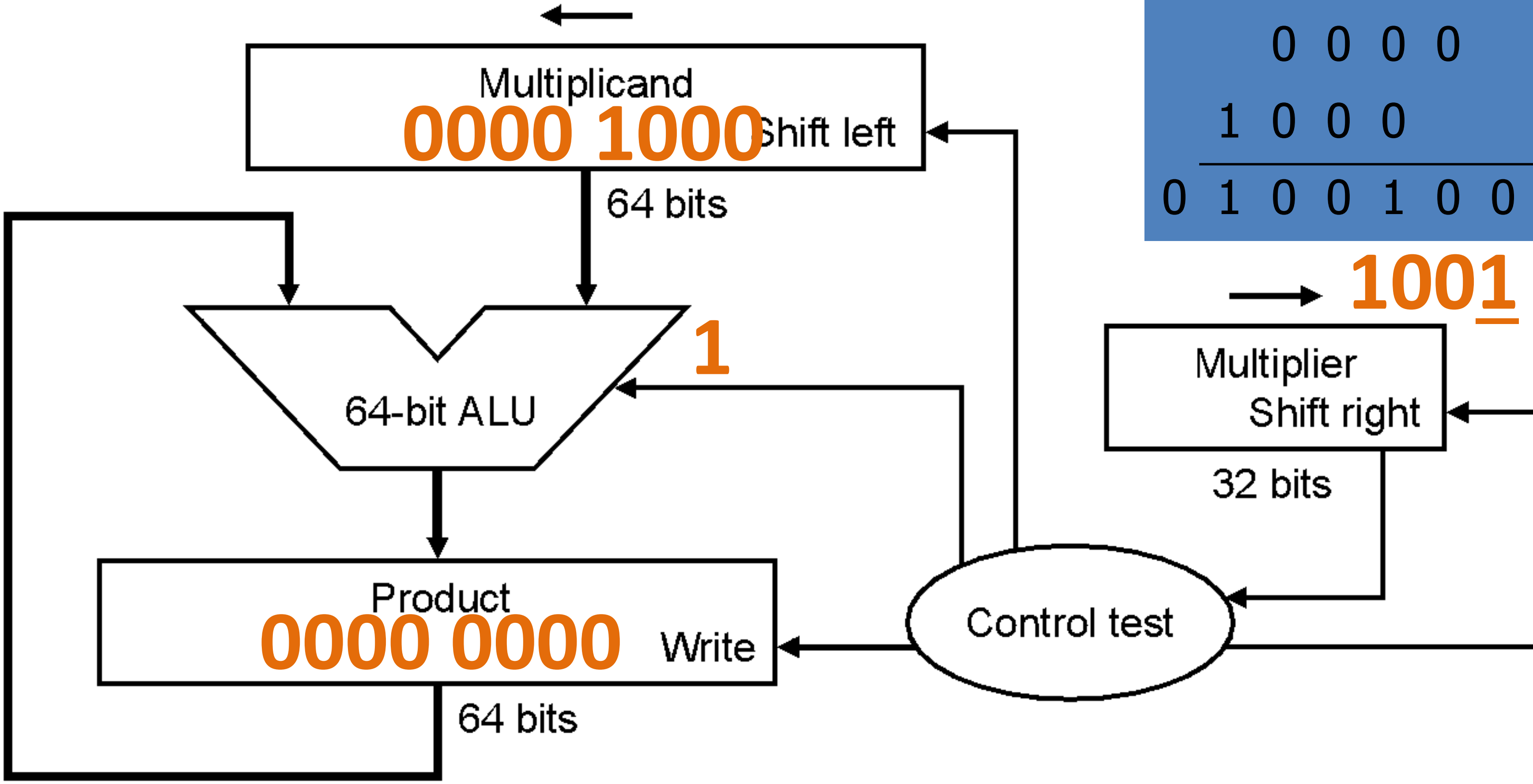


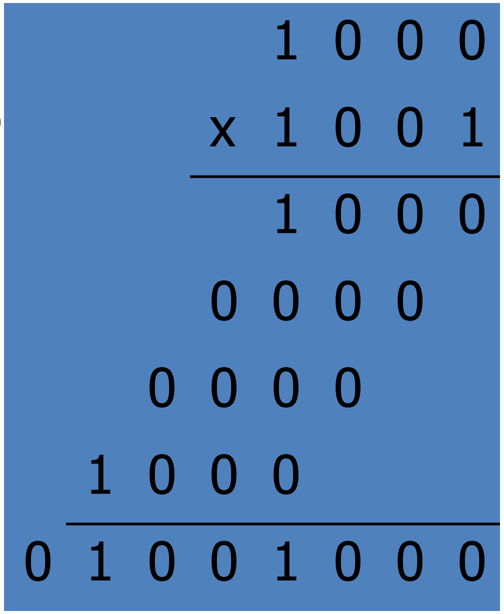
Multicycle Multiplier

```
      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 1 0 0 1 0 0 0
```



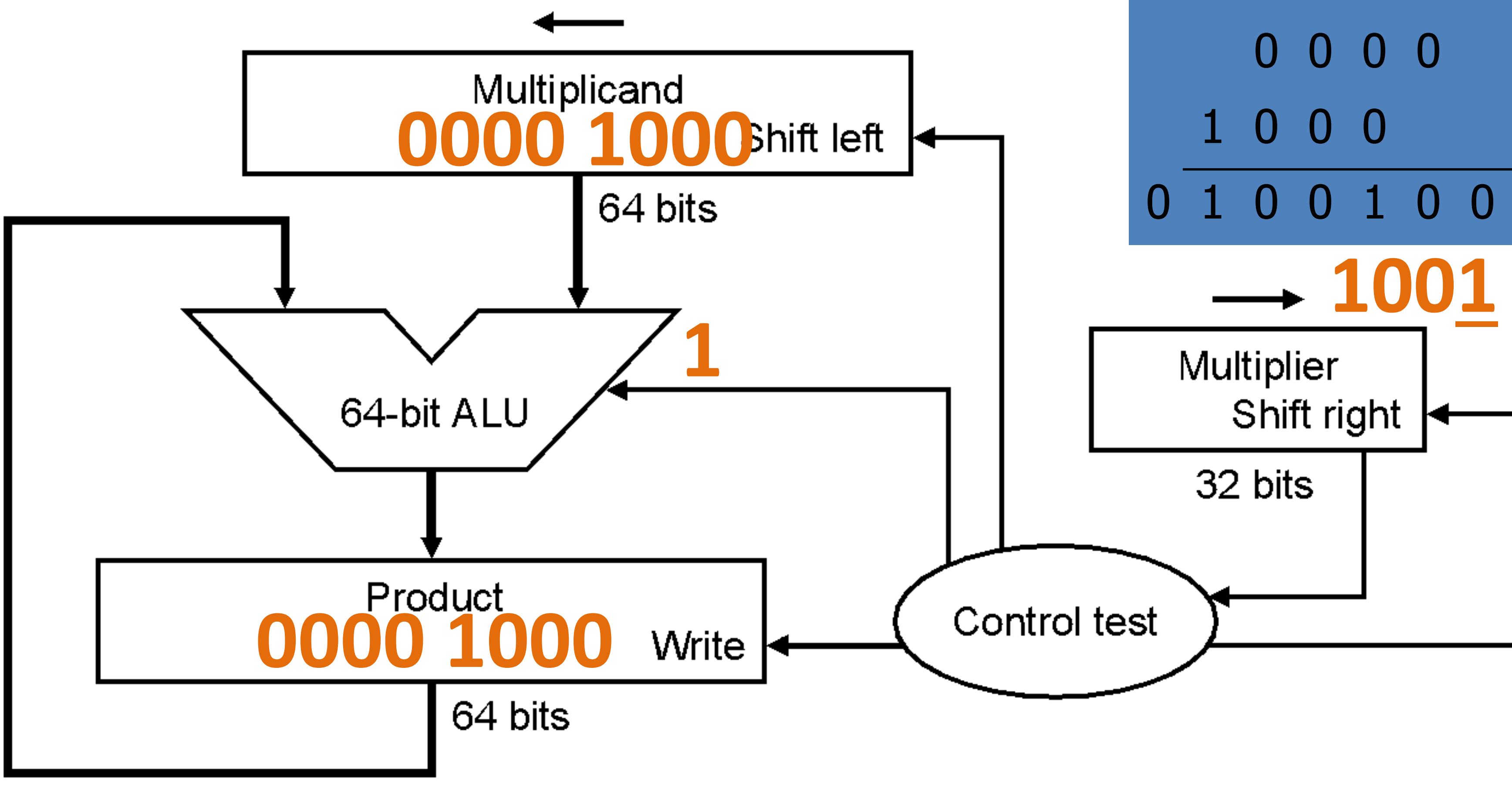
Multicycle Multiplier





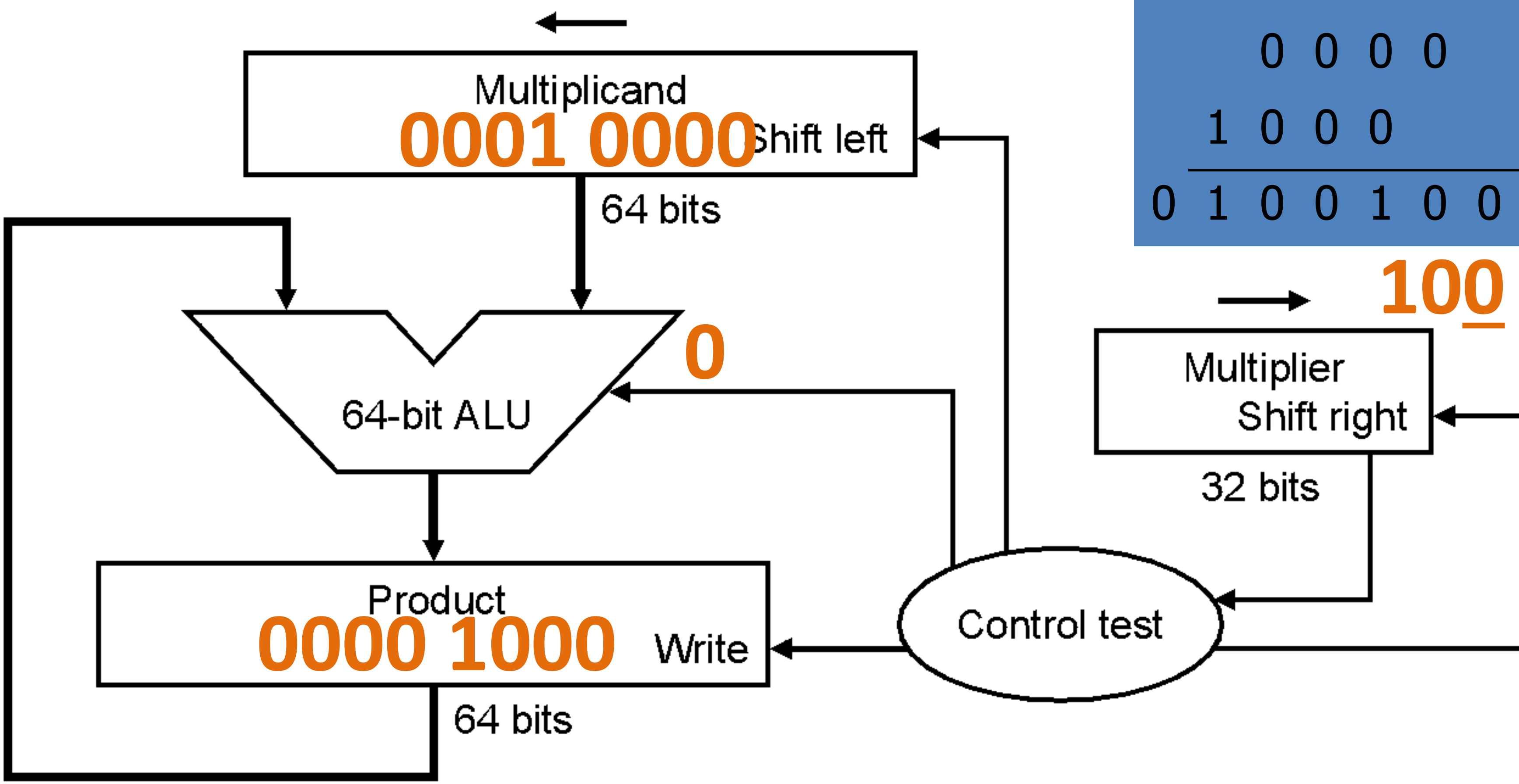
Multicycle Multiplier

```
      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 1 0 0 1 0 0 0
```



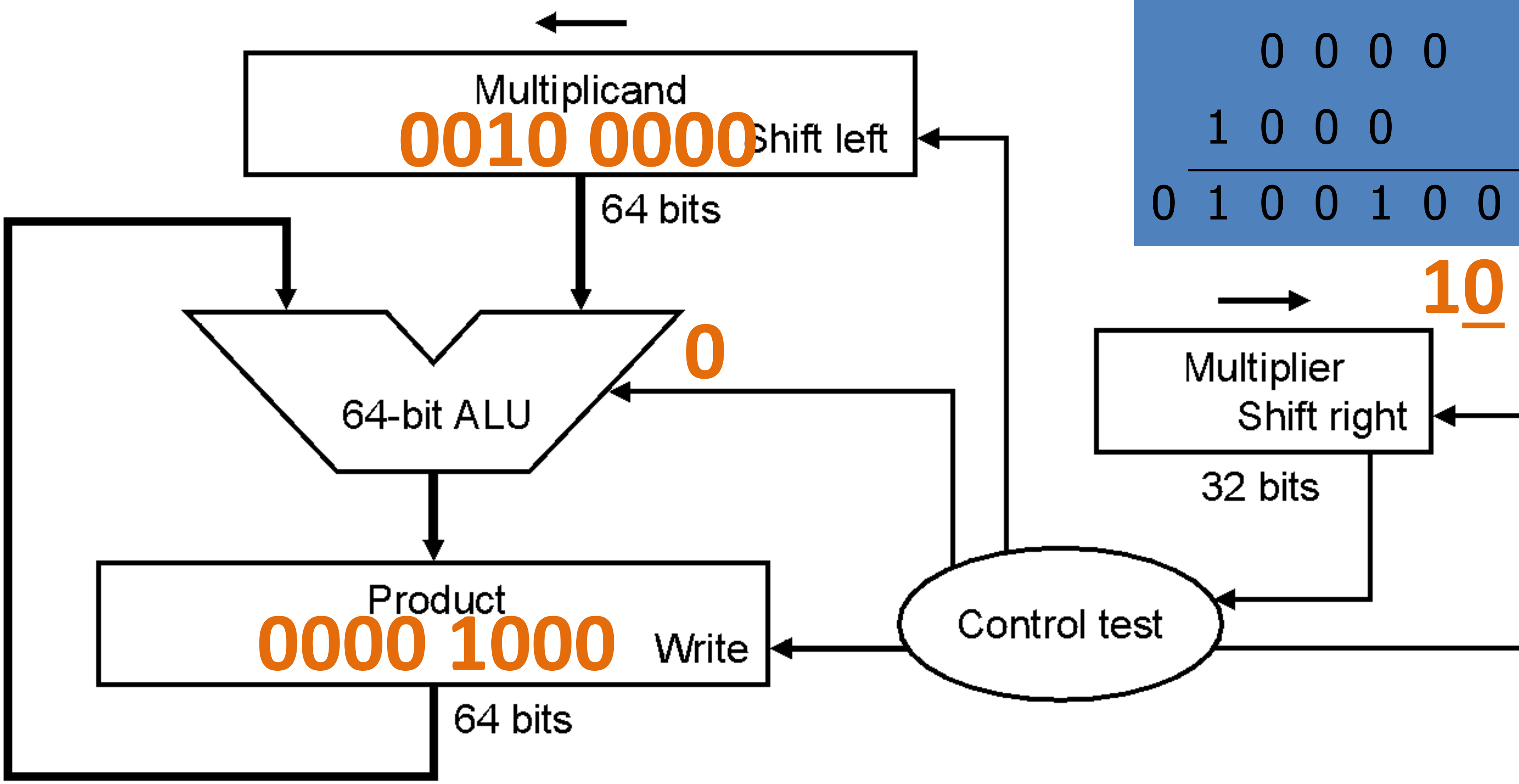
Multicycle Multiplier

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



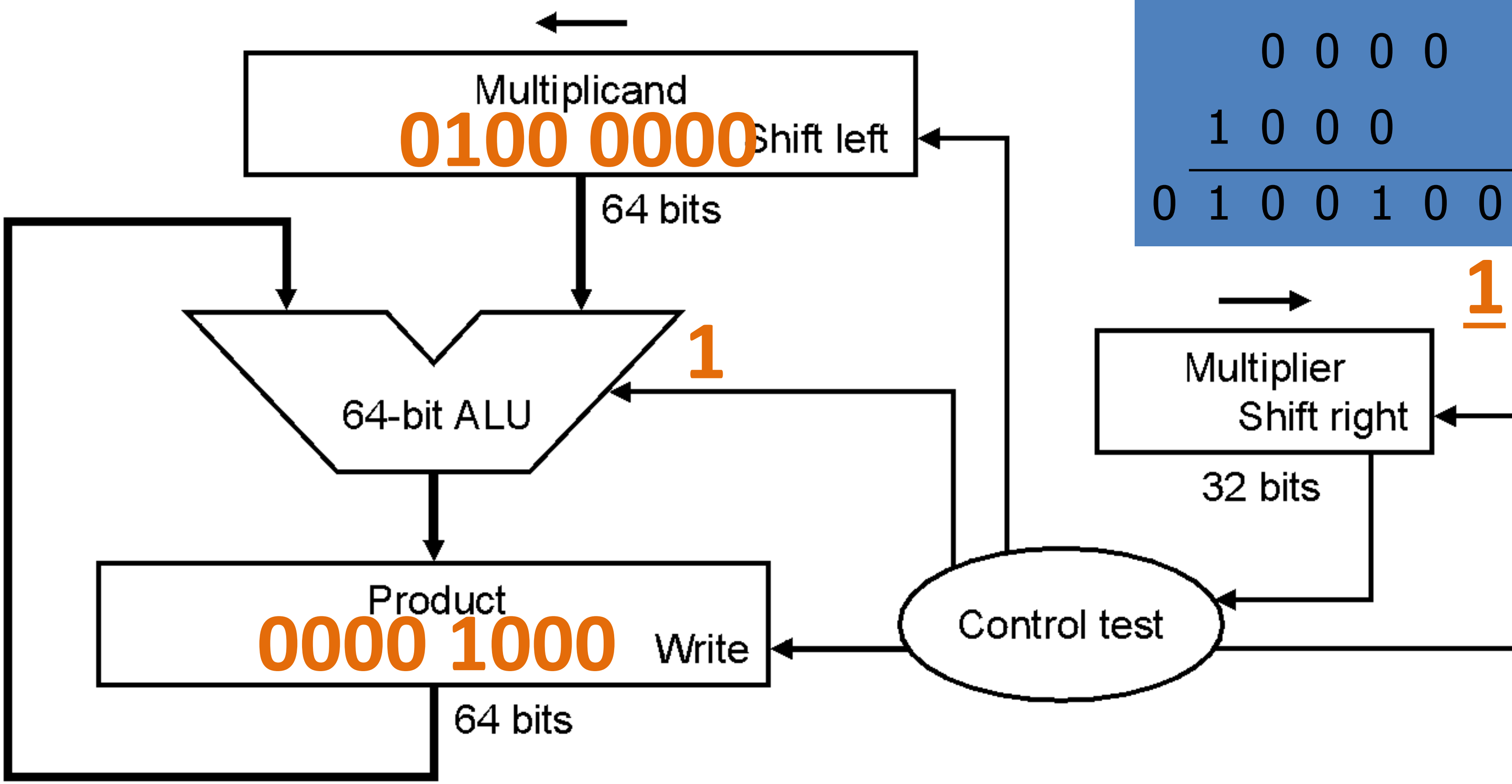
Multicycle Multiplier

					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	1	0	0	1	0	0	0



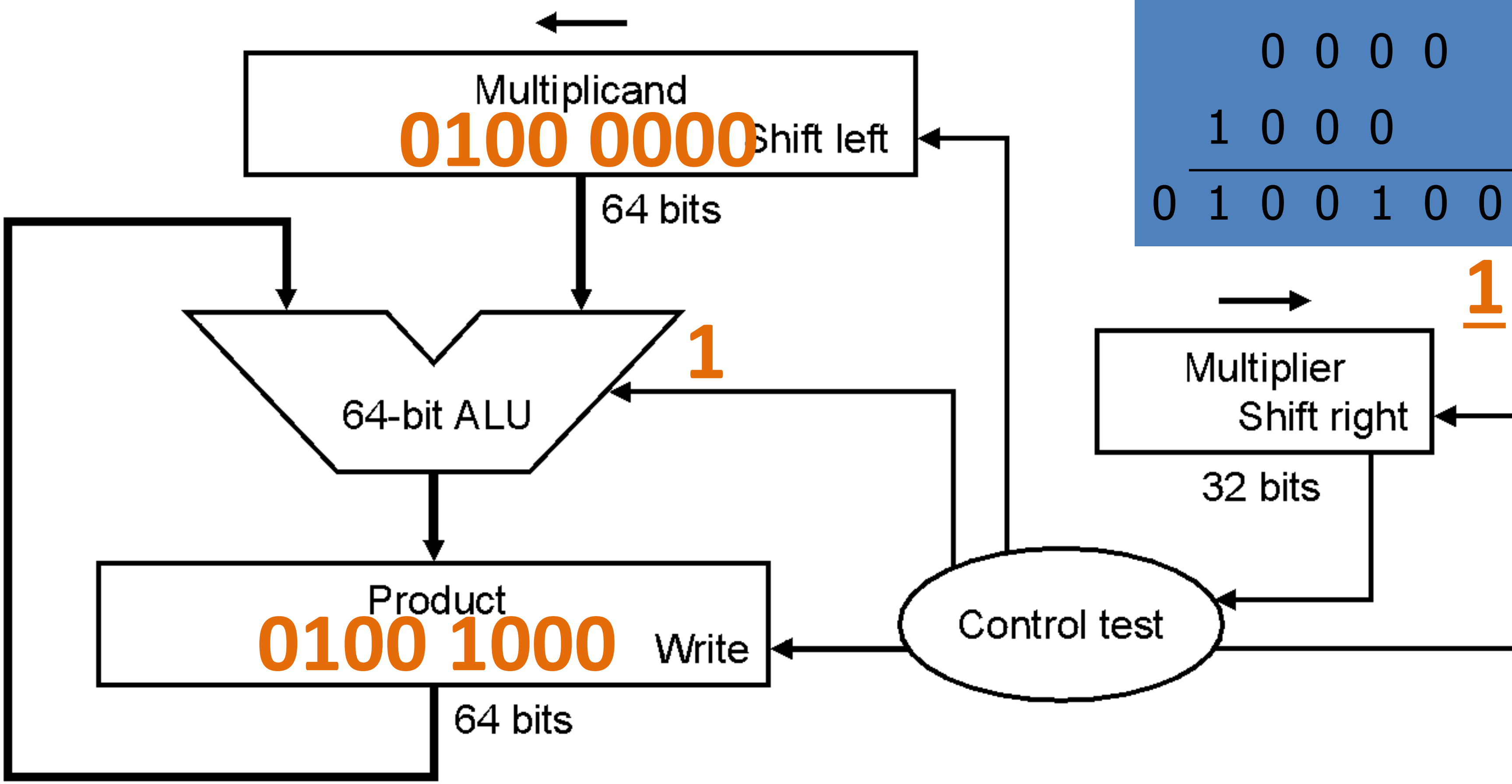
Multicycle Multiplier

```
         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 1 0 0 1 0 0 0
```



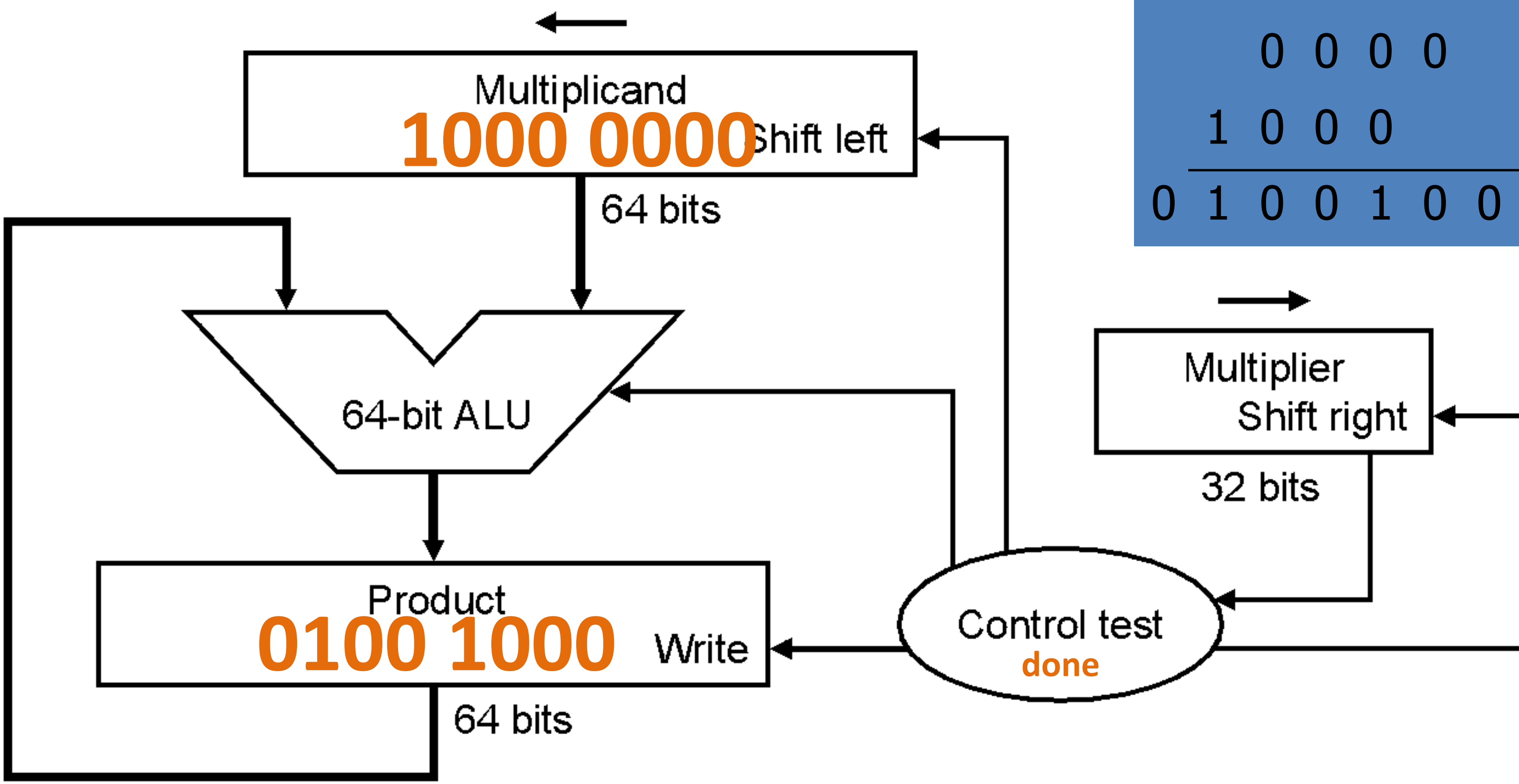
Multicycle Multiplier

```
      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 1 0 0 1 0 0 0
```

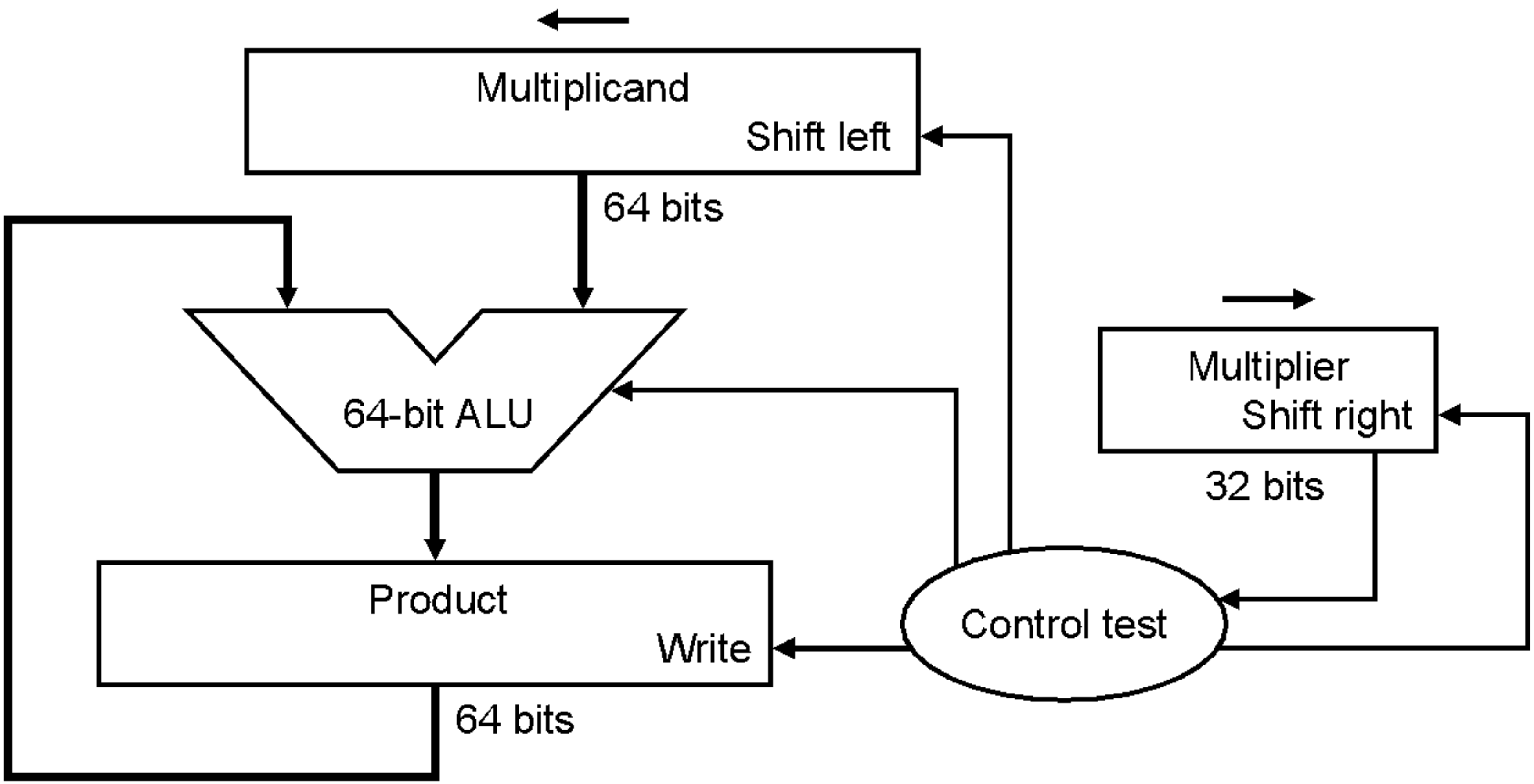


Multicycle Multiplier

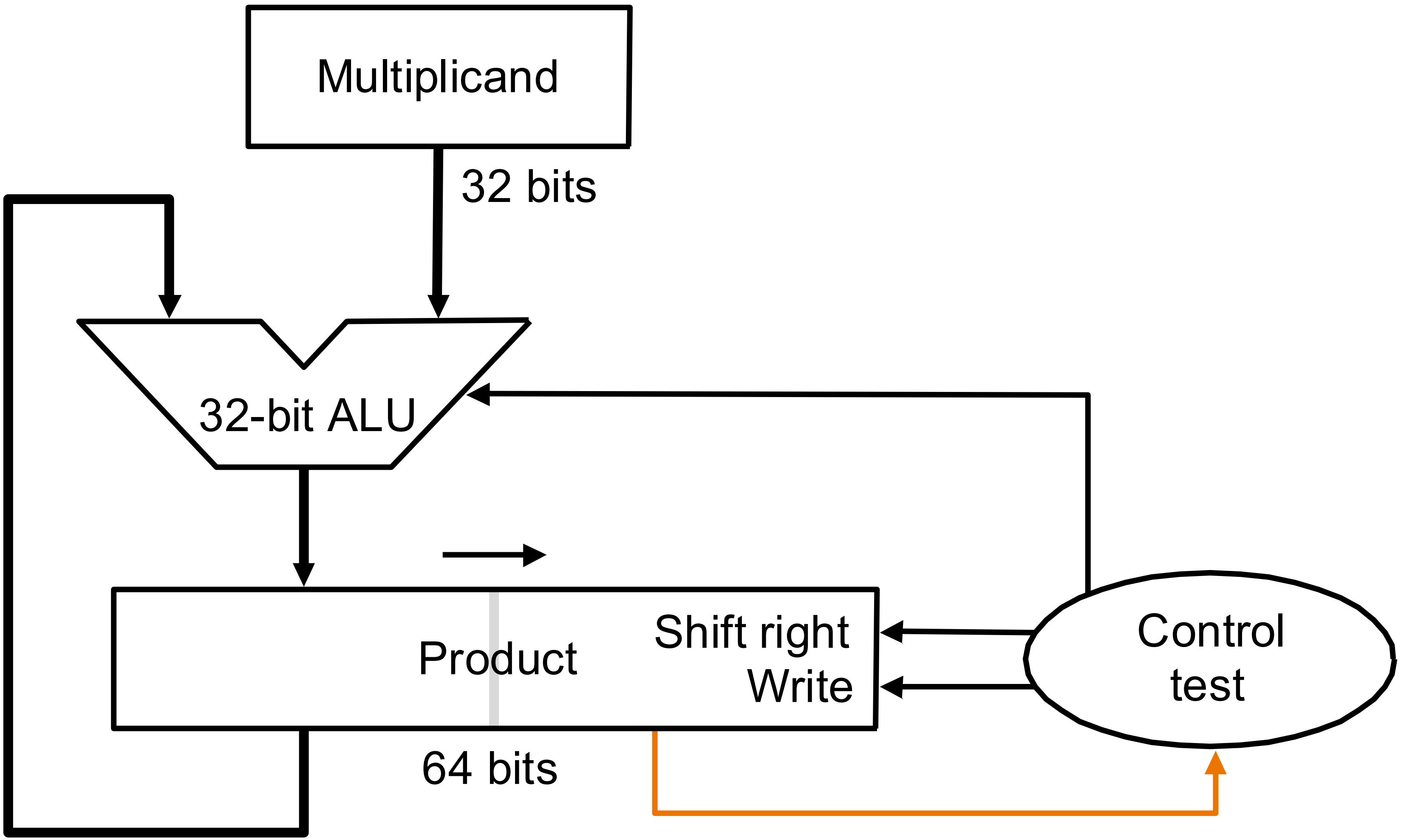
```
      1 0 0 0
    x 1 0 0 1
    ---------
      1 0 0 0
    0 0 0 0
  0 0 0 0
1 0 0 0
---------------
0 1 0 0 1 0 0 0
```



Multicycle Multiplier

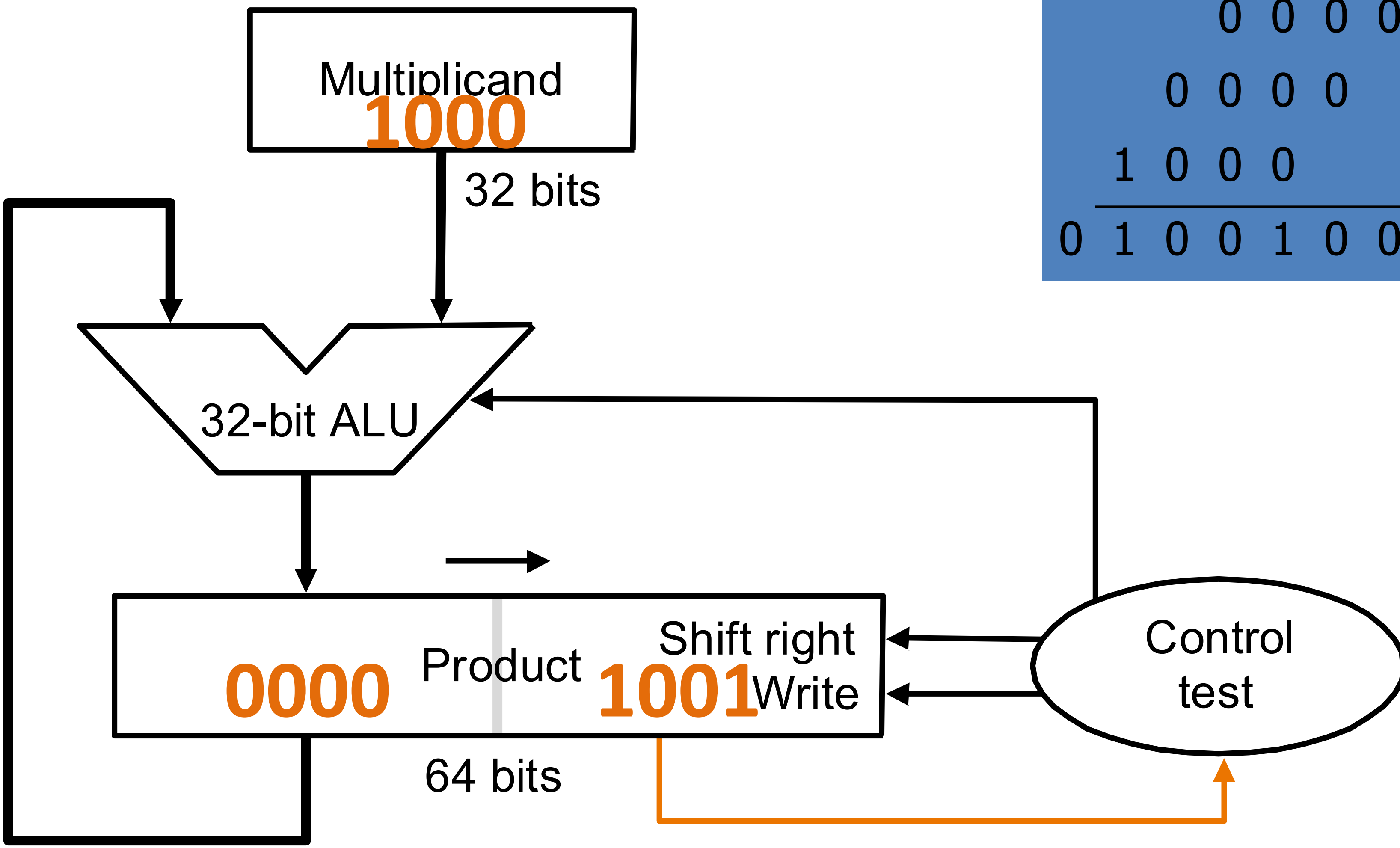


Multicycle Multiplier

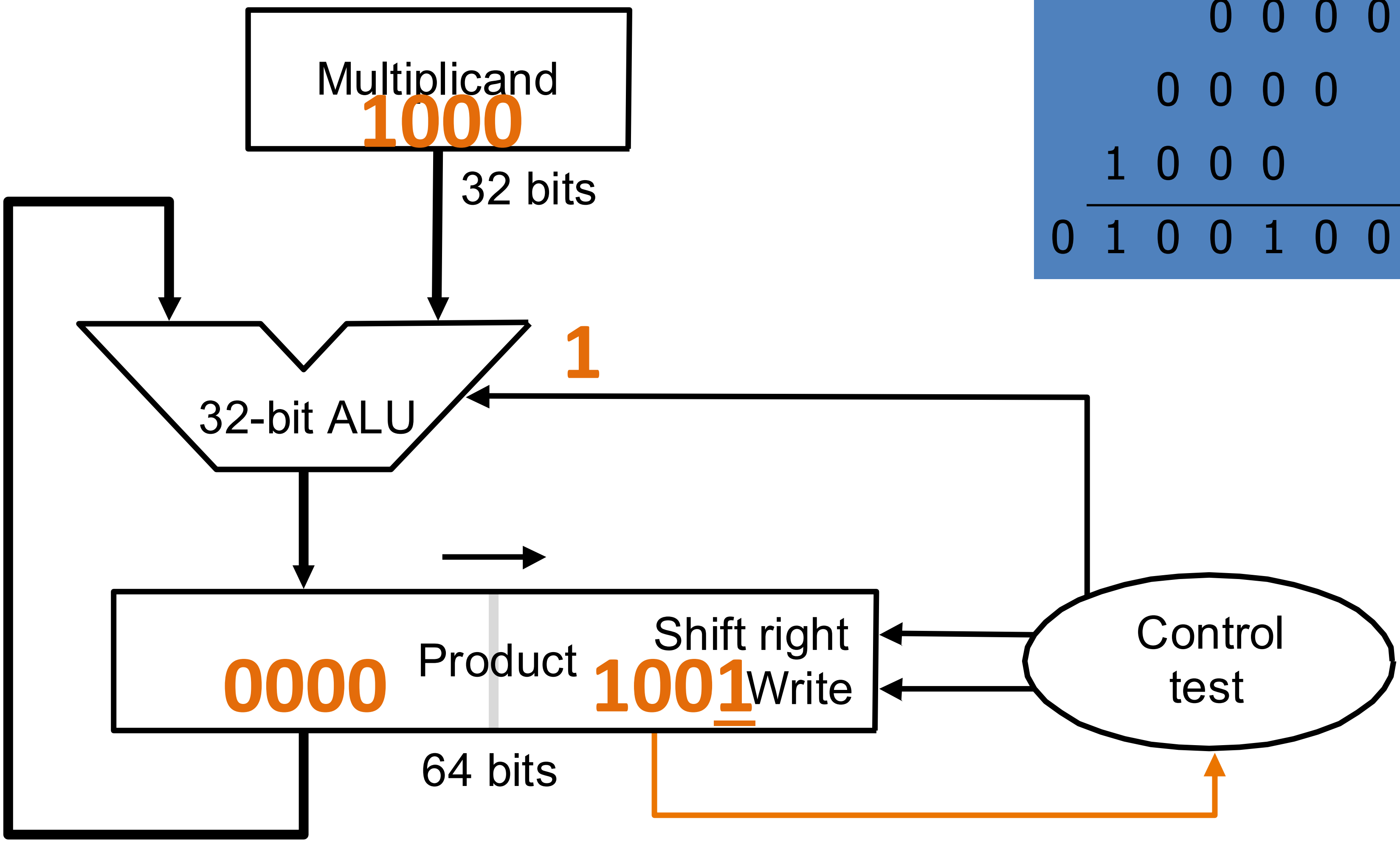


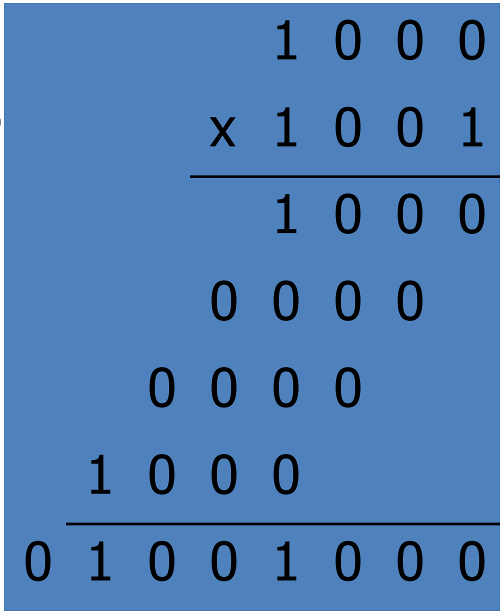
Multicycle Multiplier

					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	1	0	0	1	0	0	0



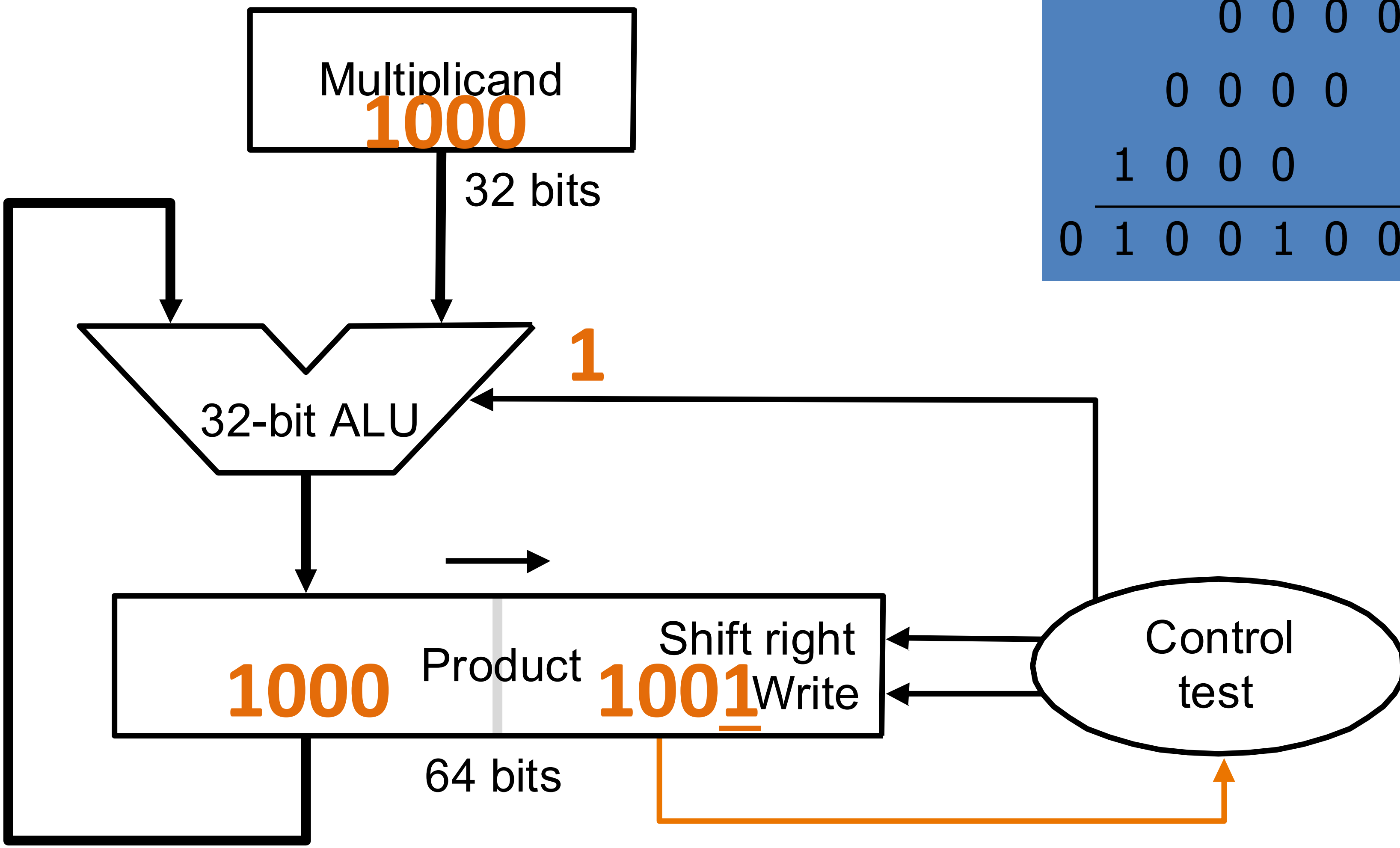
Multicycle Multiplier





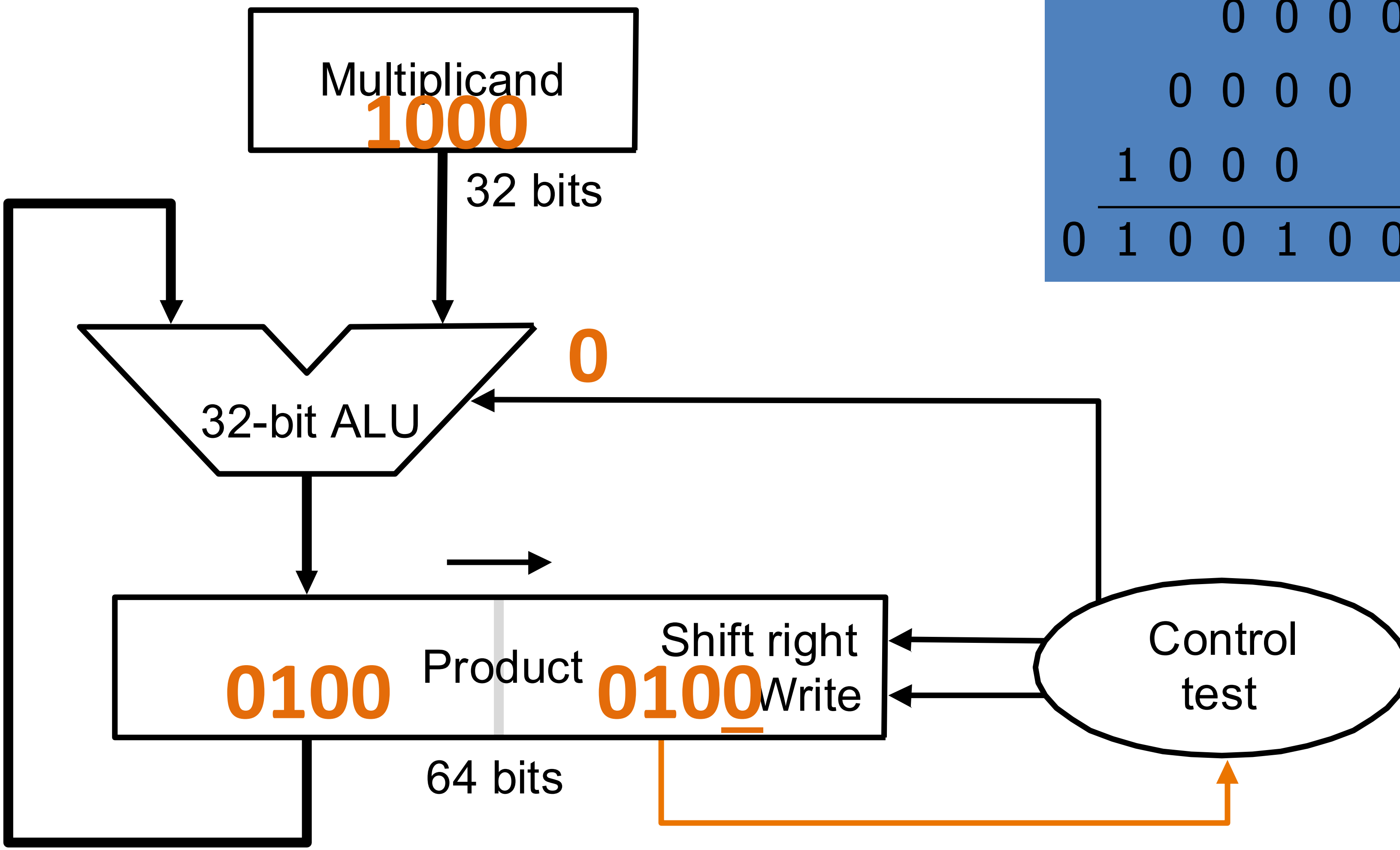
Multicycle Multiplier

```
        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 1 0 0 1 0 0 0
```



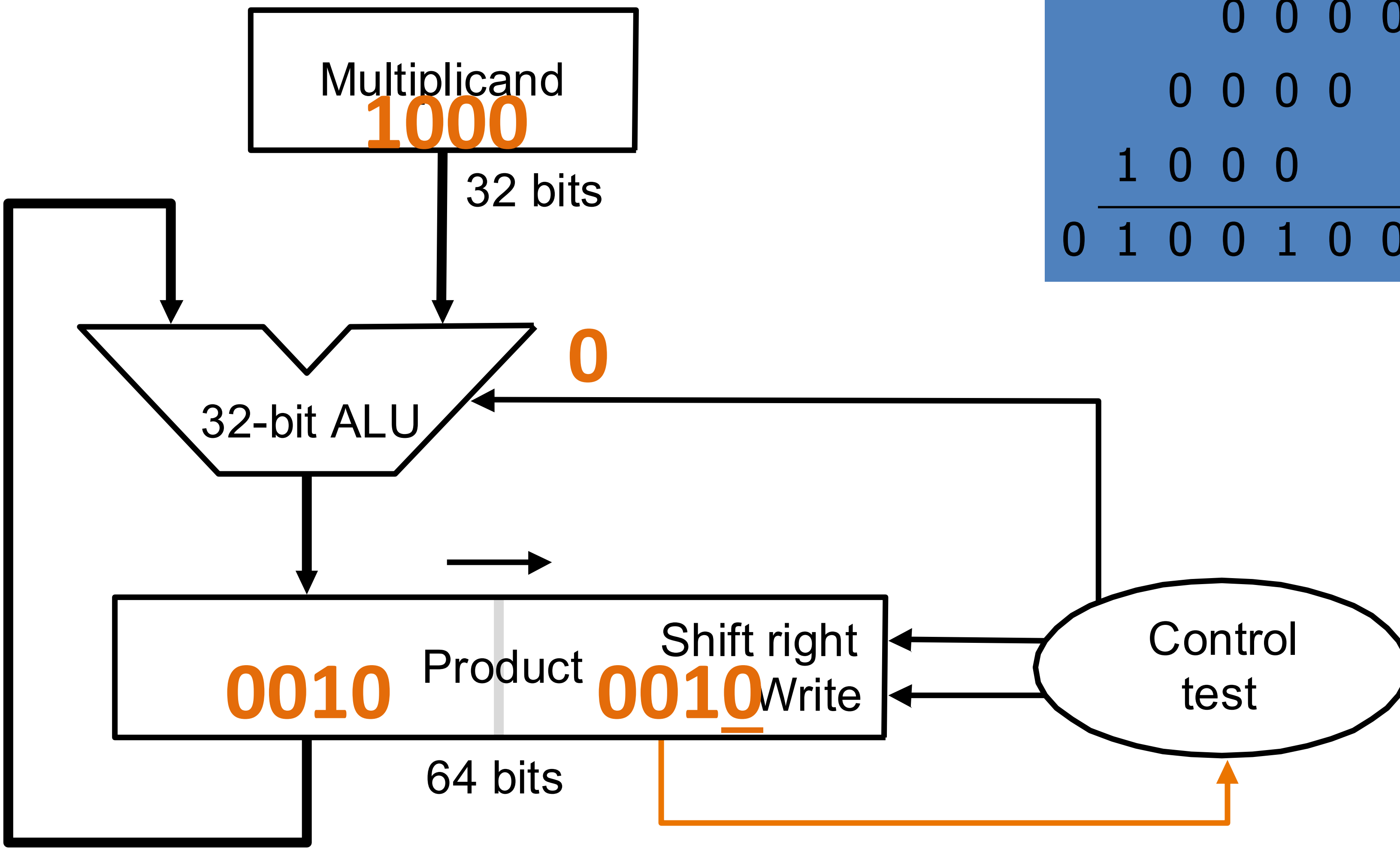
Multicycle Multiplier

```
      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 1 0 0 1 0 0 0
```



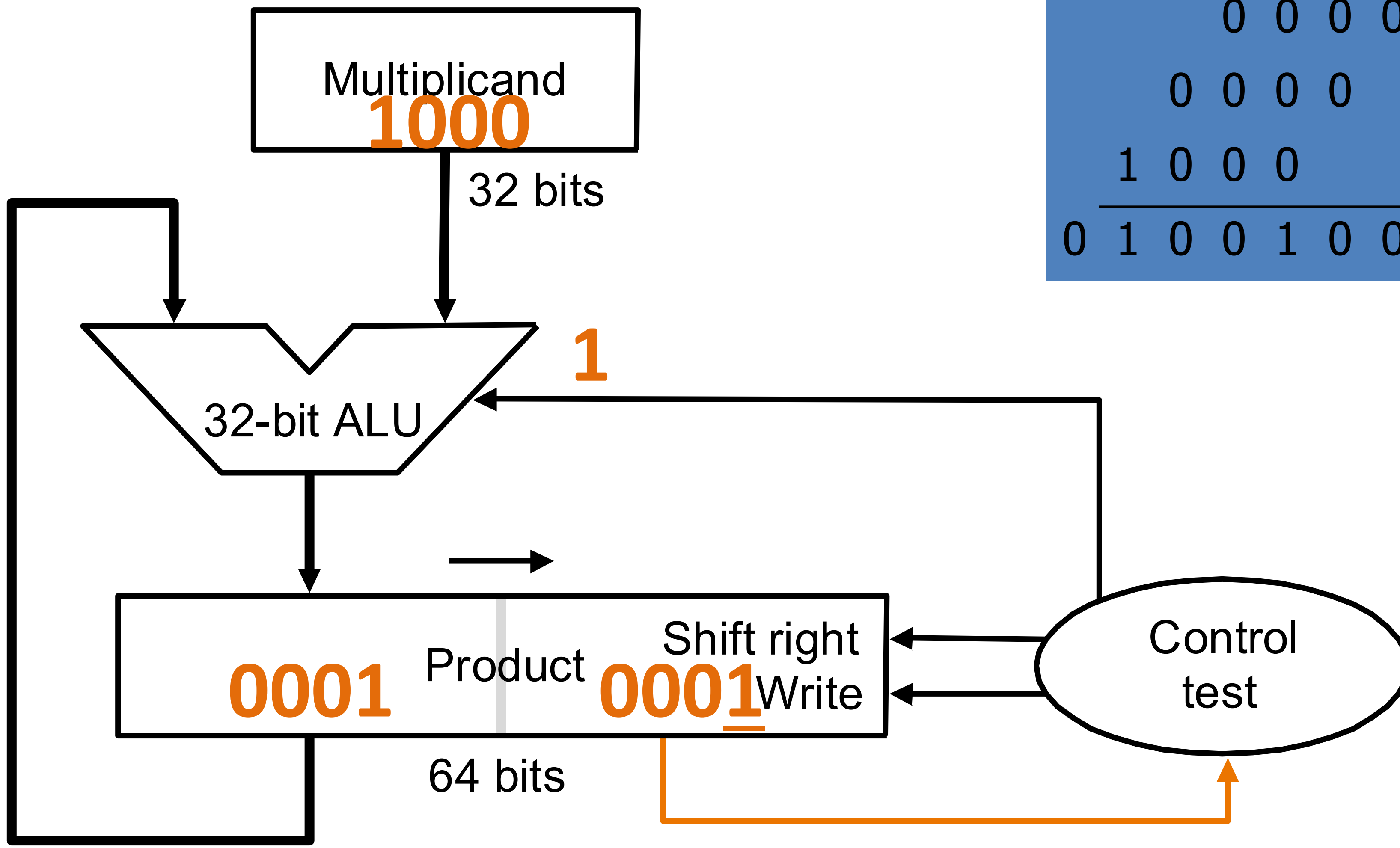
Multicycle Multiplier

```
        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 1 0 0 1 0 0 0
```



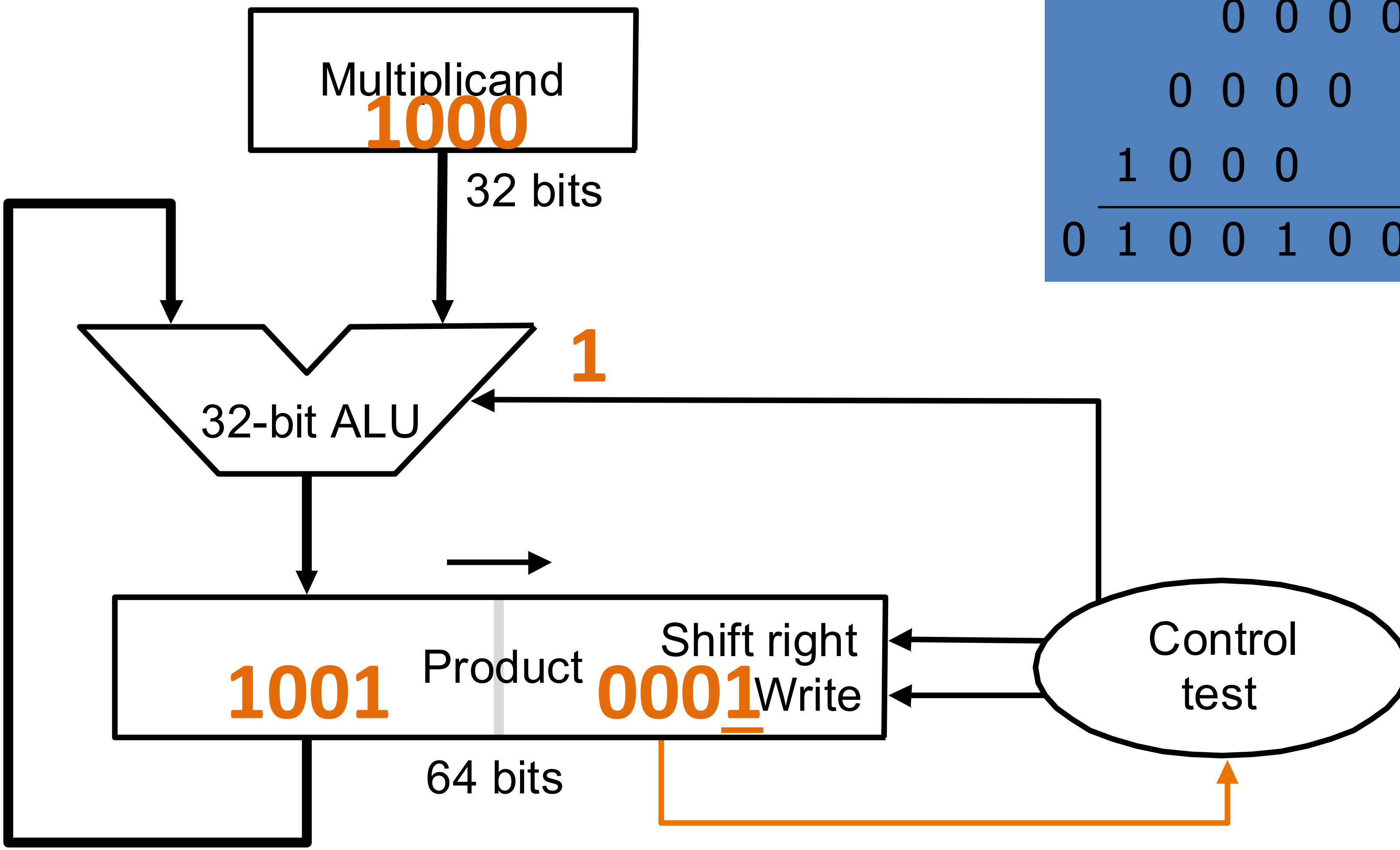
Multicycle Multiplier

```
      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 1 0 0 1 0 0 0
```



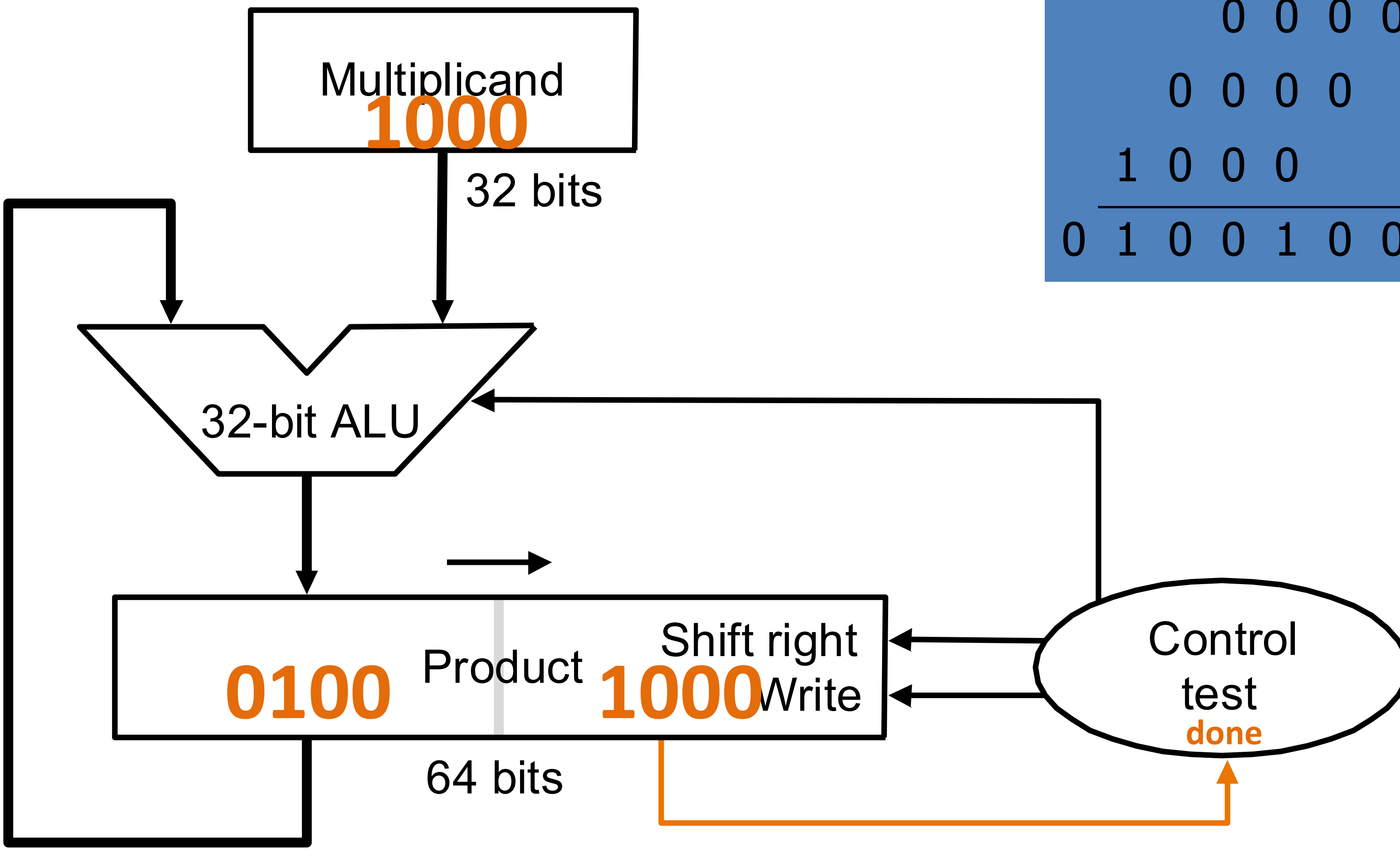
Multicycle Multiplier

```
        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 1 0 0 1 0 0 0
```



Multicycle Multiplier

```
        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 1 0 0 1 0 0 0
```



Booth's Encoding

From right (least significant) to left (most significant):

Current bit	Previous bit	Explanation	Example	Operation
1	0	Begins run of '1'	0000111000	Subtract (-)
1	1	Middle of run of '1'	0000111000	Nothing (0)
0	1	End of a run of '1'	0000111000	Add (+)
0	0	Middle of a run of '0'	0000111000	Nothing (0)

Booth's encoding: **000+000-000**

Booth's Multiplier

- Positive multiplier:

  -6 x 6 = -36

  1010 x 0110

  - Multiplier 0110 in Booth's encoding is +0-0

	Final Sum:	

Booth's Multiplier

- Positive multiplier:

  -6 x 6 = -36

  1010 x 0110

  - Multiplier 0110 in Booth's encoding is +0-0

1111 1010	x 0	0000 0000
	Final Sum:	

Booth’s Multiplier

- Positive multiplier:

  -6 x 6 = -36

  1010 x 0110

  - Multiplier 0110 in Booth’s encoding is +0-0

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
	Final Sum:	

Booth's Multiplier

- Positive multiplier:

  -6 x 6 = -36

  1010 x 0110

  - Multiplier 0110 in Booth's encoding is +0-0

1111 1010	x 0	0000 0000
1111 0100	x −1	0000 1100
1110 1000	x 0	0000 0000
	Final Sum:	

Booth's Multiplier

- Positive multiplier:

  -6 x 6 = -36

  1010 x 0110

  - Multiplier 0110 in Booth's encoding is +0-0

1111 1010	x 0	0000 0000
1111 0100	x −1	0000 1100
1110 1000	x 0	0000 0000
1101 0000	x +1	+ 1101 0000
	Final Sum:	

Booth’s Multiplier

- Positive multiplier:

  -6 x 6 = -36

  1010 x 0110

  - Multiplier 0110 in Booth’s encoding is +0-0

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
1110 1000	x 0	0000 0000
1101 0000	x +1	+ 1101 0000
	Final Sum:	1101 1100 (-36)

Booth's Multiplier

- Negative multiplier:

  -6 x -2 = 12

  1010 x 1110

  - Multiplier 1110 in Booth's encoding is 00-0

	Final Sum:	

Booth's Multiplier

- Negative multiplier:

  -6 x -2 = 12

  1010 x 1110

  - Multiplier 1110 in Booth's encoding is 00-0

1111 1010	x 0	0000 0000
	Final Sum:	

Booth's Multiplier

- Negative multiplier:

  -6 x -2 = 12

  1010 x 1110

  - Multiplier 1110 in Booth's encoding is 00-0

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
	Final Sum:	

Booth's Multiplier

- Negative multiplier:

  -6 x -2 = 12

  1010 x 1110

  - Multiplier 1110 in Booth's encoding is 00-0

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
1110 1000	x 0	0000 0000
	Final Sum:	

Booth's Multiplier

- Negative multiplier:

  -6 x -2 = 12

  1010 x 1110

  - Multiplier 1110 in Booth's encoding is 00-0

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
1110 1000	x 0	0000 0000
1101 0000	x 0	+ 0000 0000
	Final Sum:	

Booth's Multiplier

- Negative multiplier:

  -6 x -2 = 12

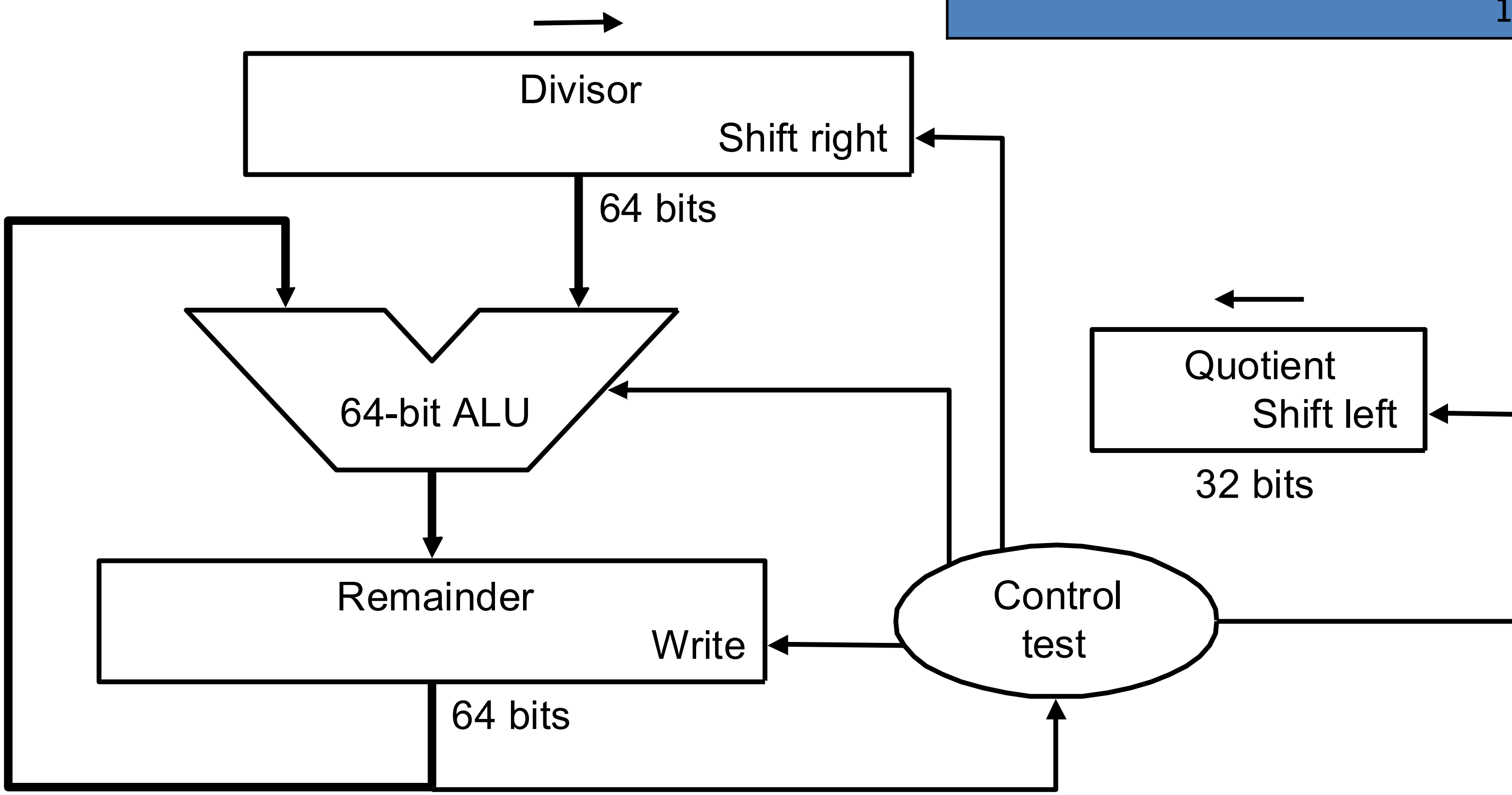
  1010 x 1110

  - Multiplier 1110 in Booth's encoding is 00-0

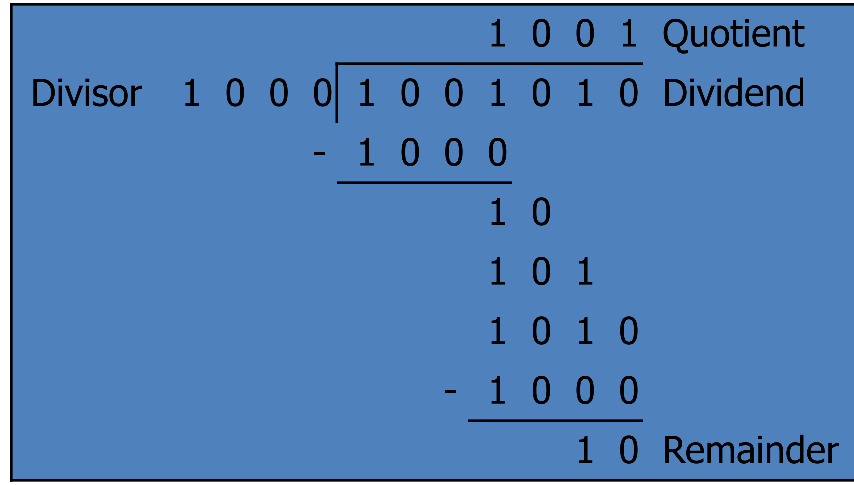
1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
1110 1000	x 0	0000 0000
1101 0000	x 0	+ 0000 0000
	Final Sum:	0000 1100 (12)

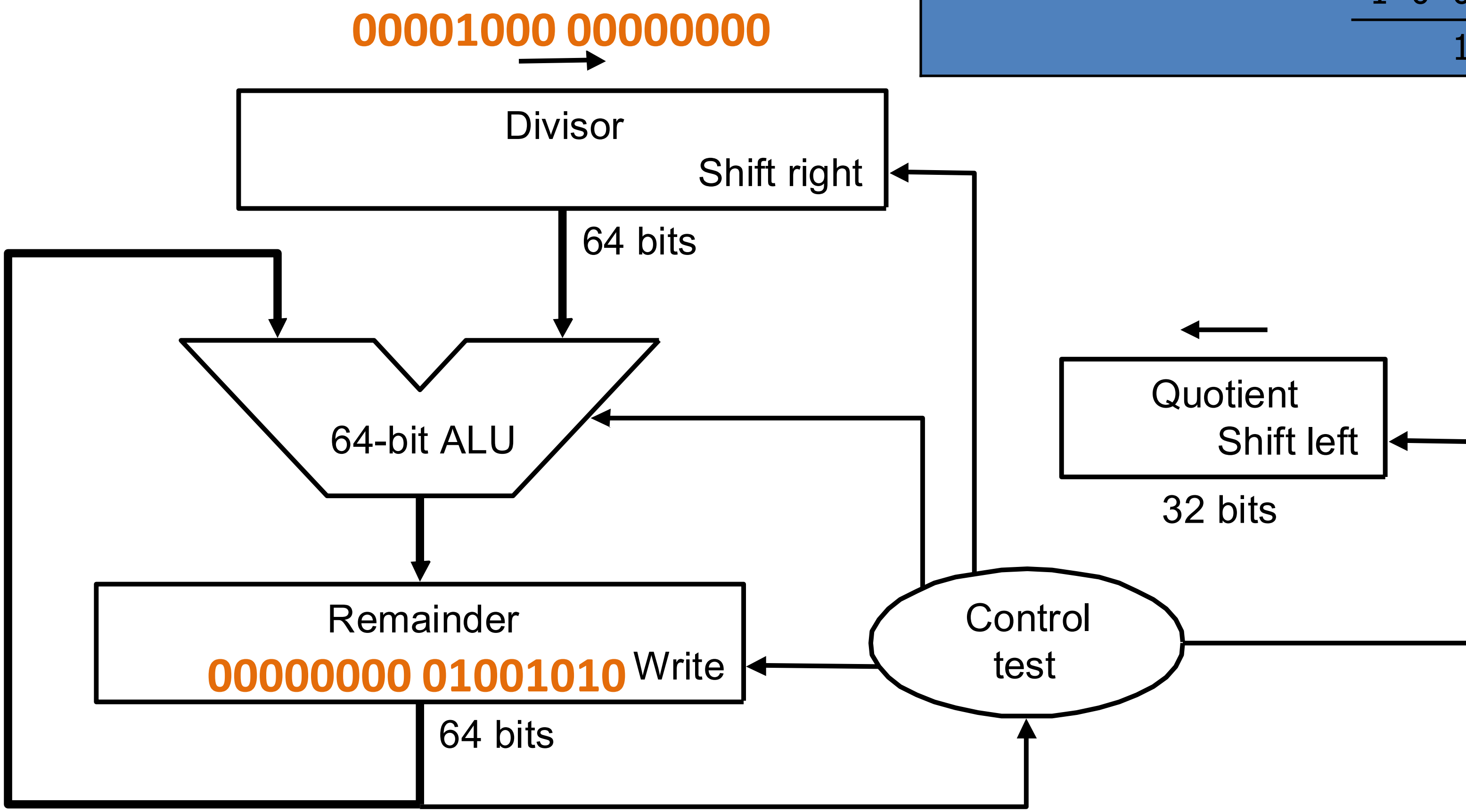
Restoring Division

```
                          1 0 0 1   Quotient
Divisor  1 0 0 0 ) 1 0 0 1 0 1 0   Dividend
                 - 1 0 0 0
                   -------
                         1 0
                         1 0 1
                         1 0 1 0
                       - 1 0 0 0
                         -------
                             1 0   Remainder
```

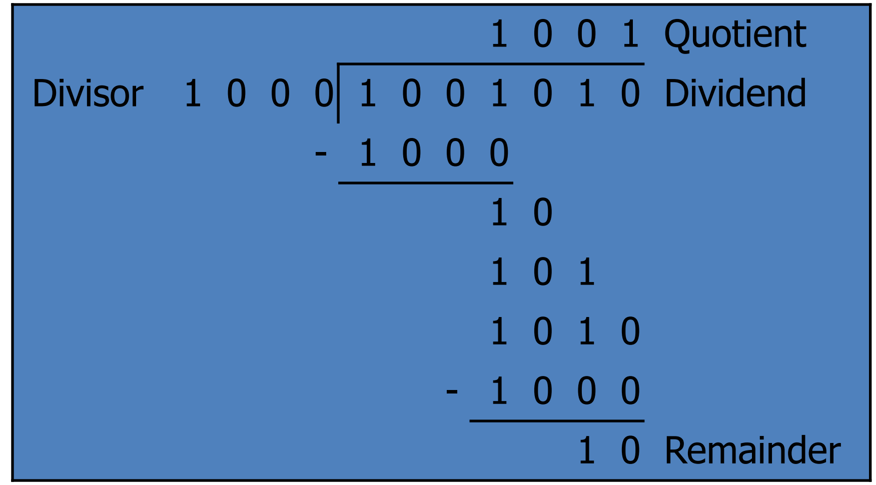


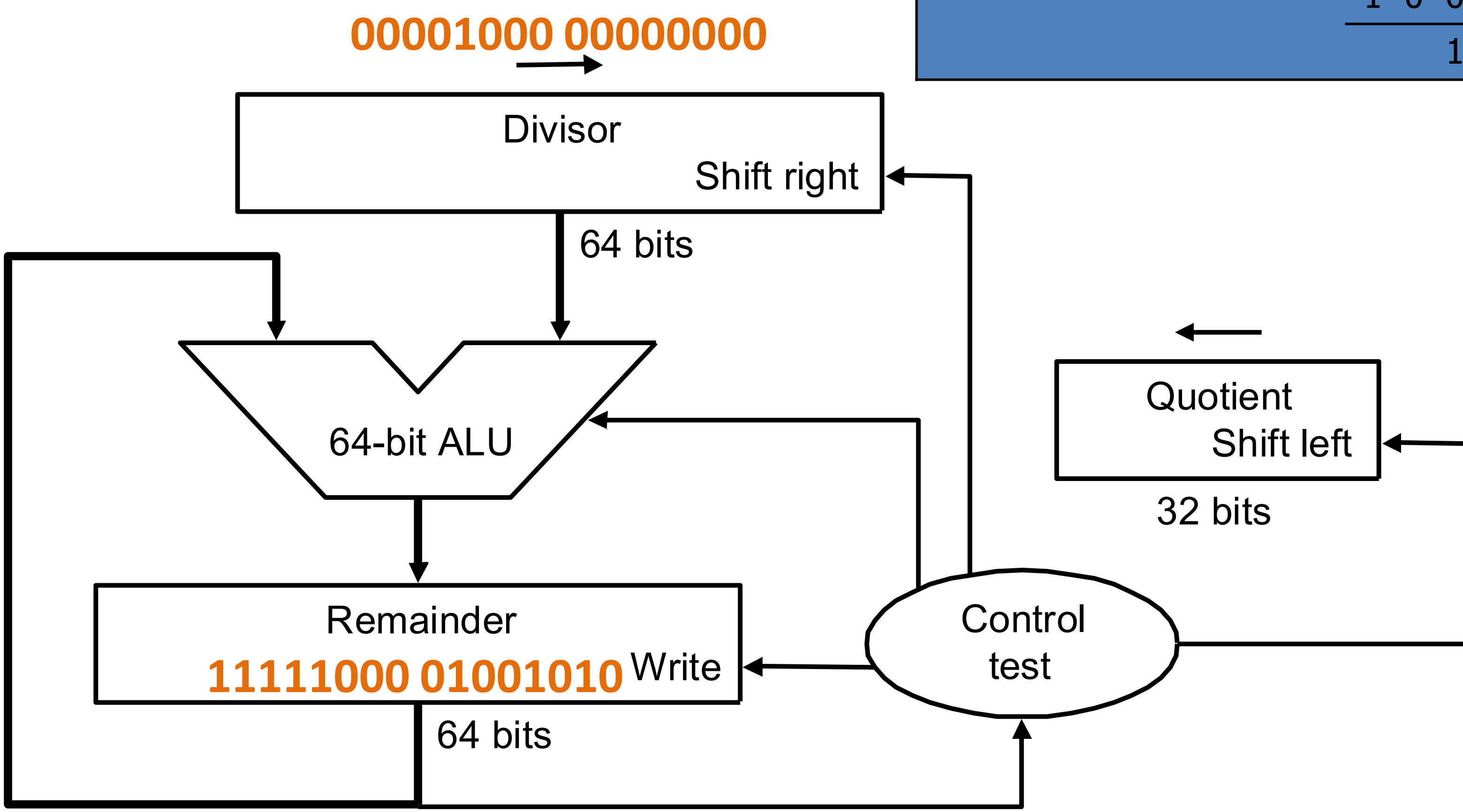
Restoring Division





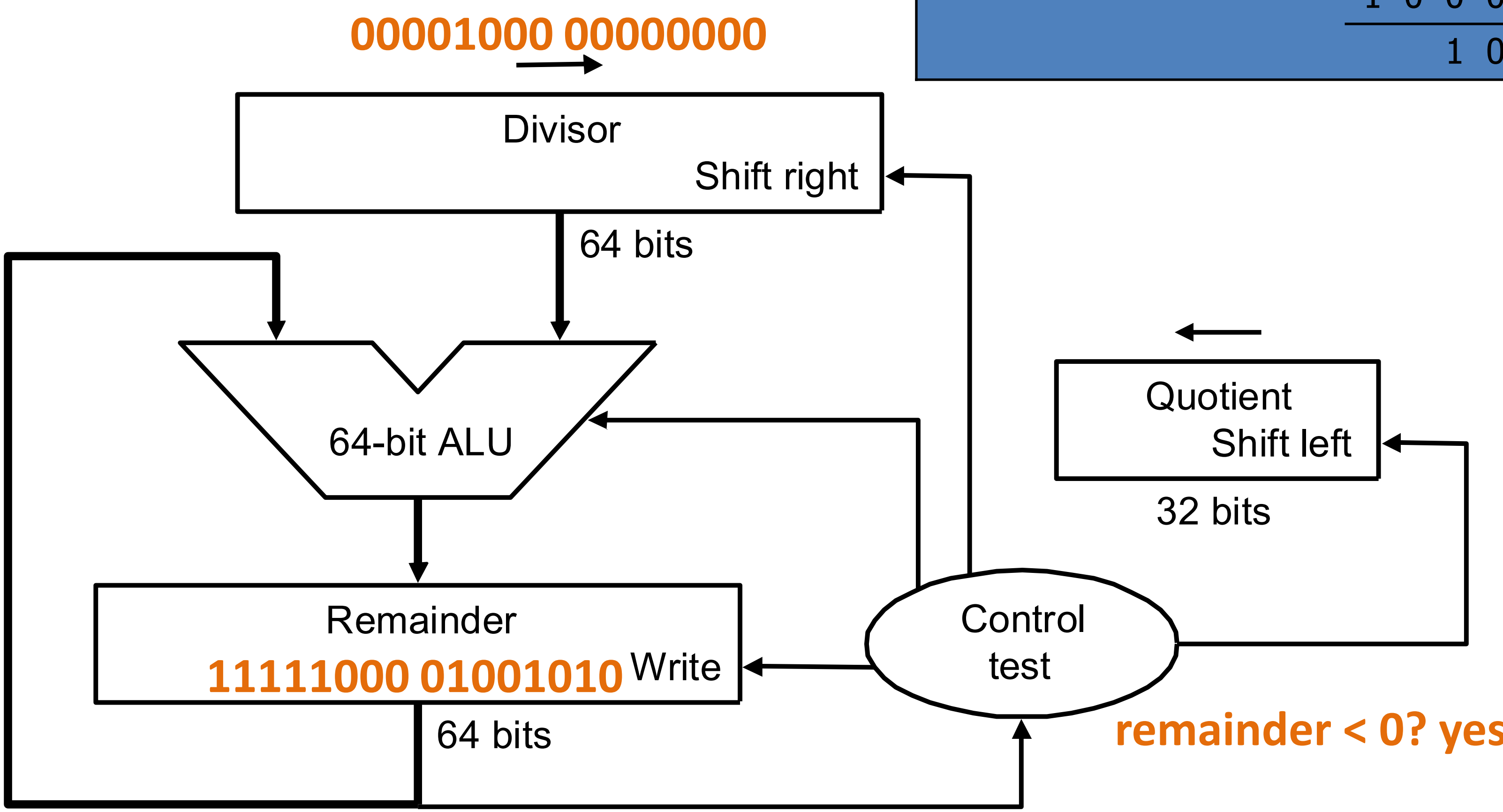
Restoring Division



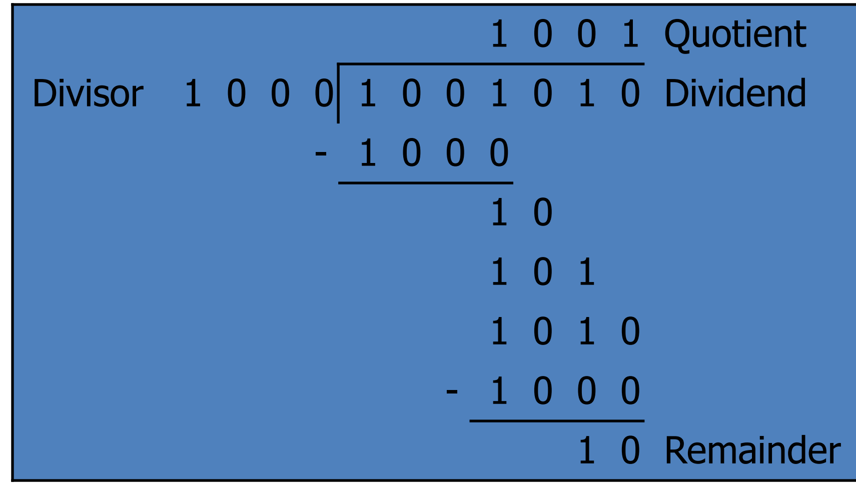


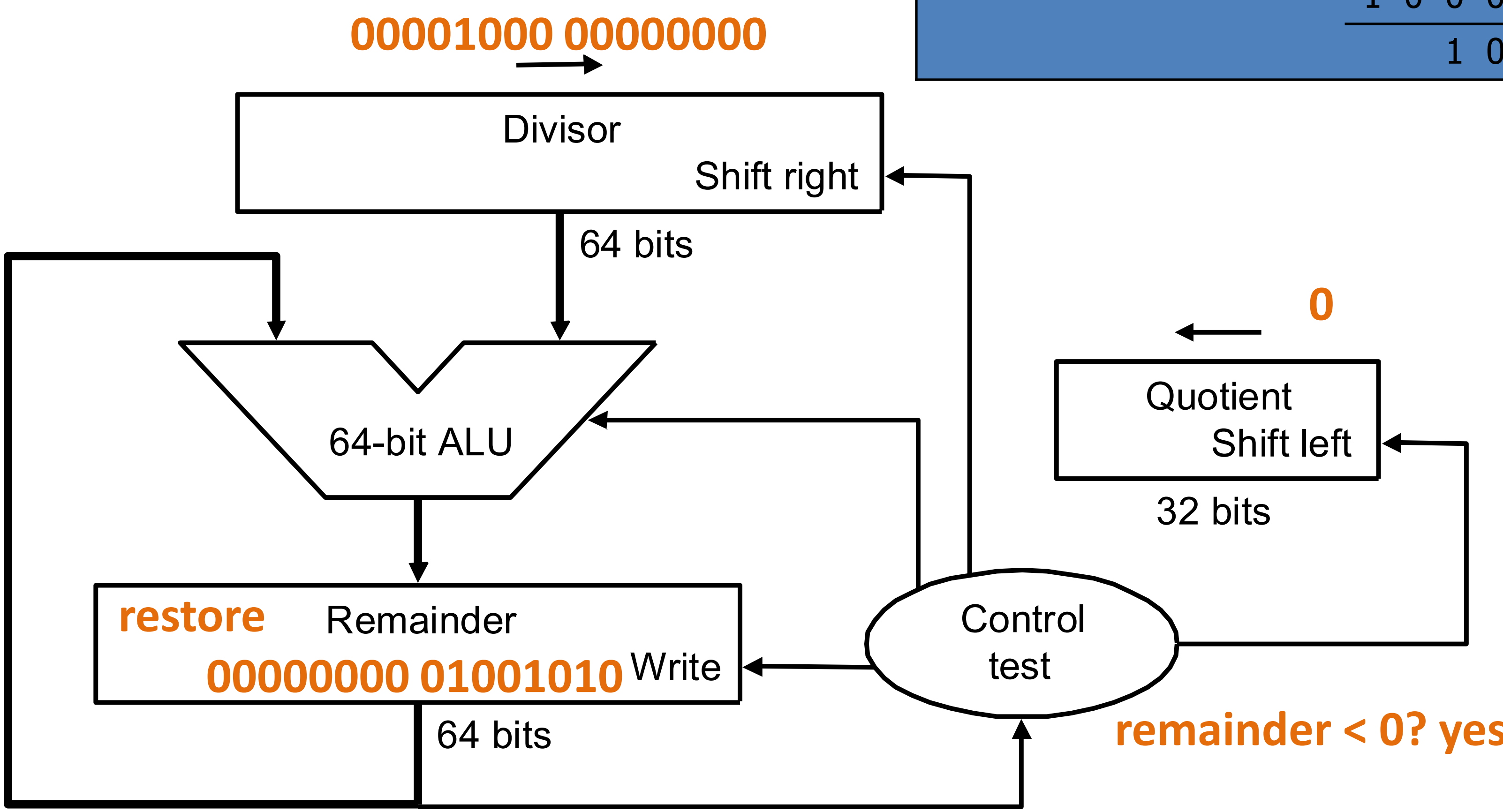
Restoring Division

```
                    1 0 0 1        Quotient
Divisor  1 0 0 0 | 1 0 0 1 0 1 0   Dividend
                 - 1 0 0 0
                   ---------
                         1 0
                         1 0 1
                         1 0 1 0
                       - 1 0 0 0
                         -------
                             1 0   Remainder
```

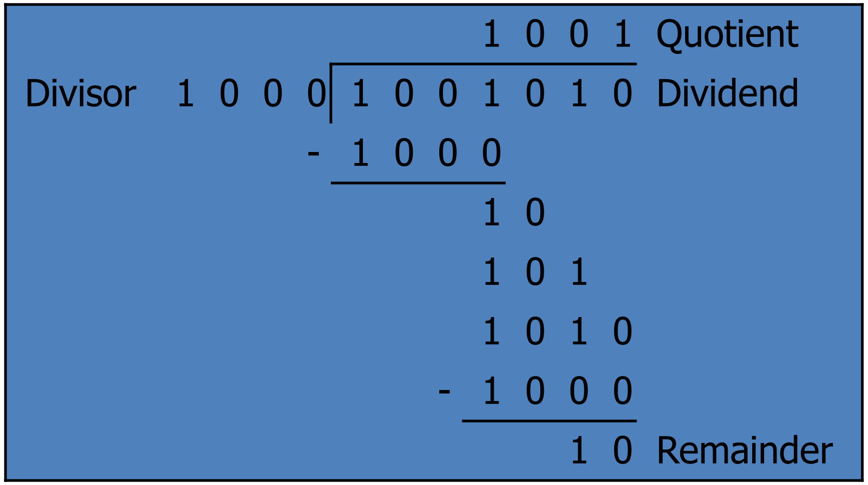


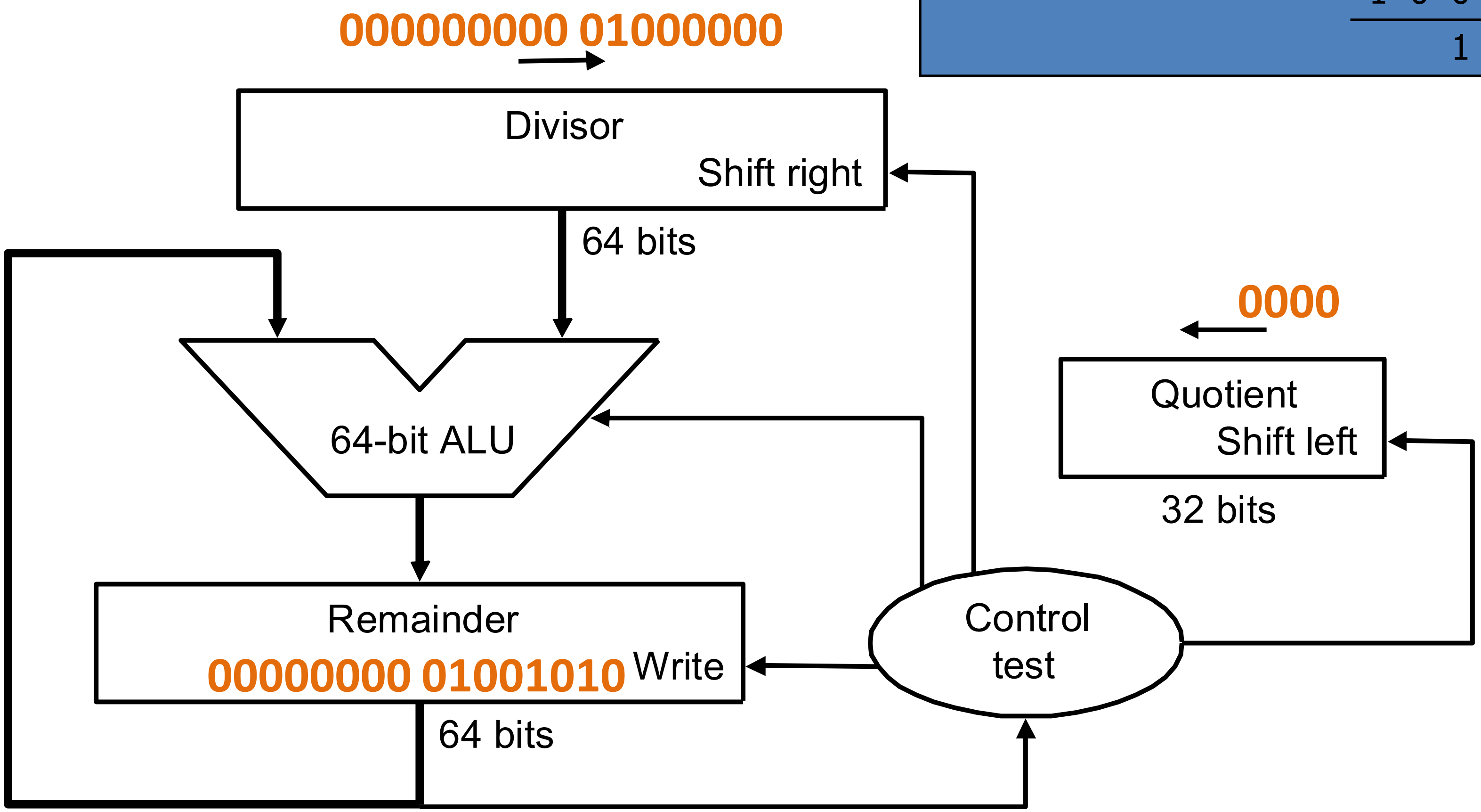
Restoring Division



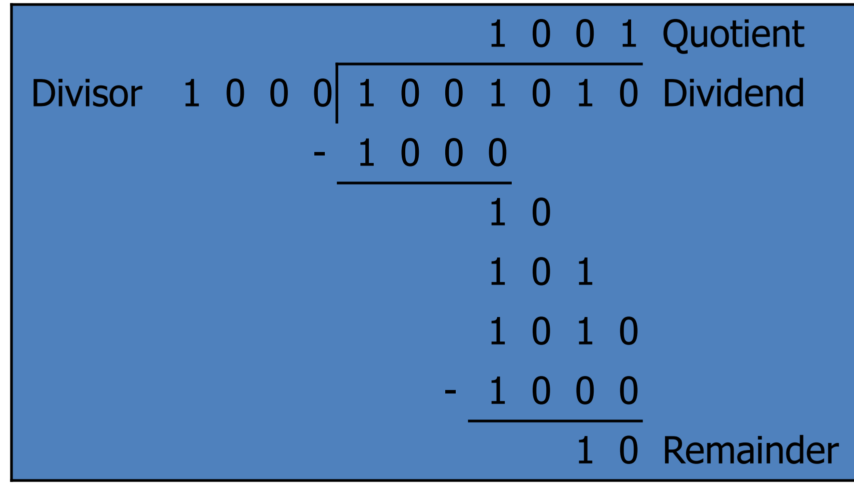


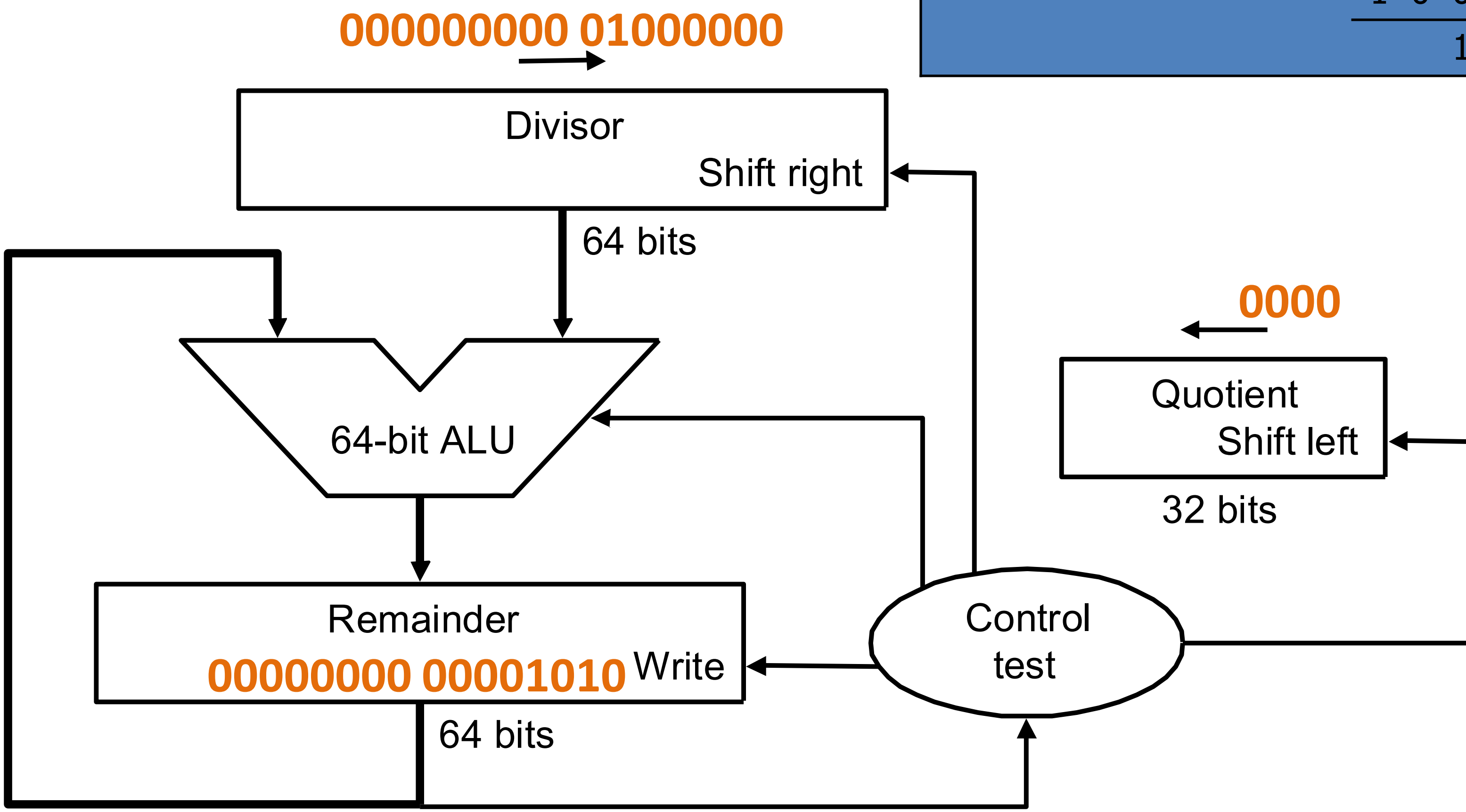
Restoring Division



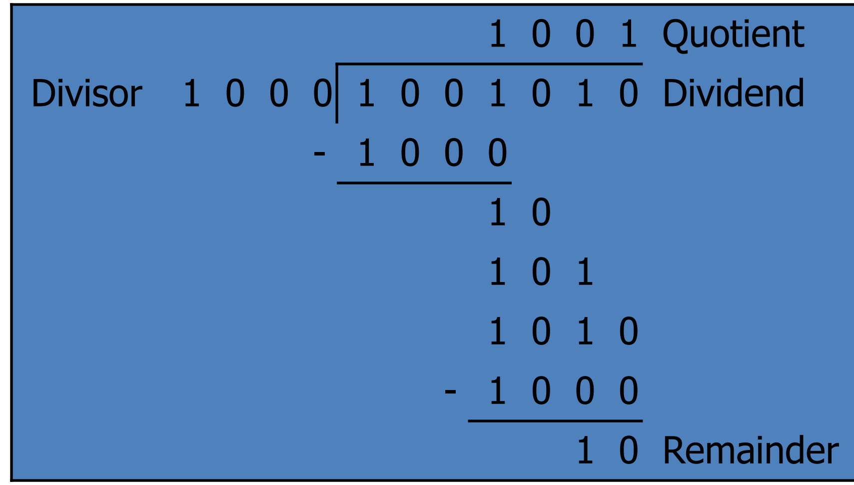


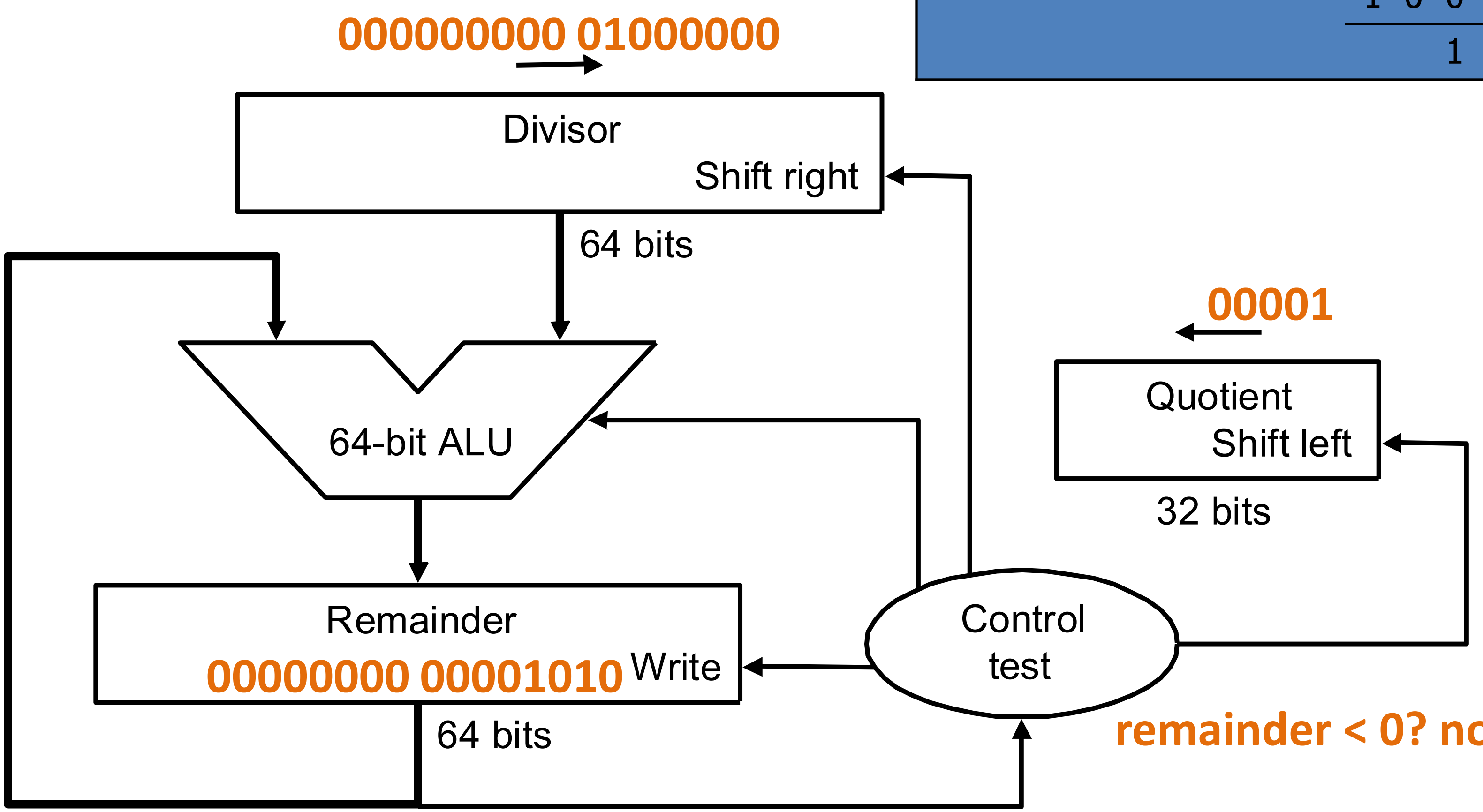
Restoring Division



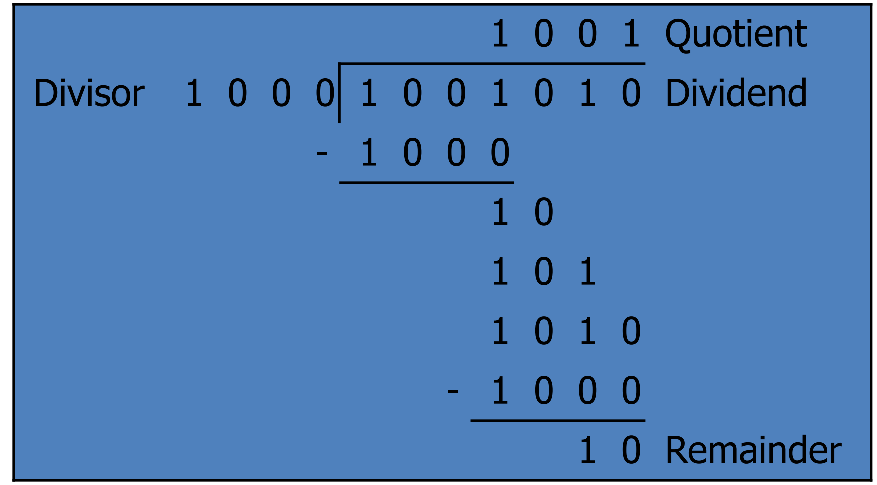


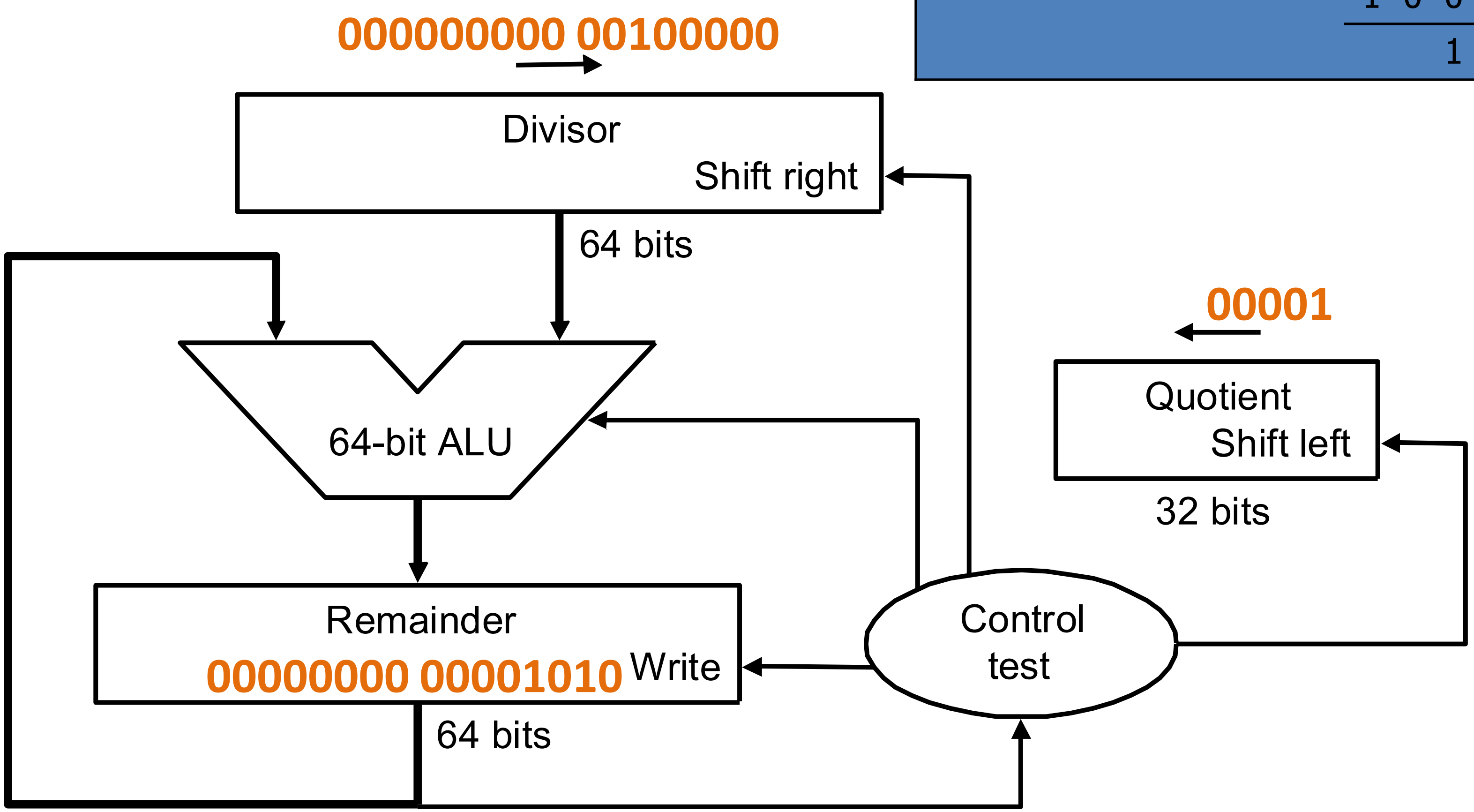
Restoring Division



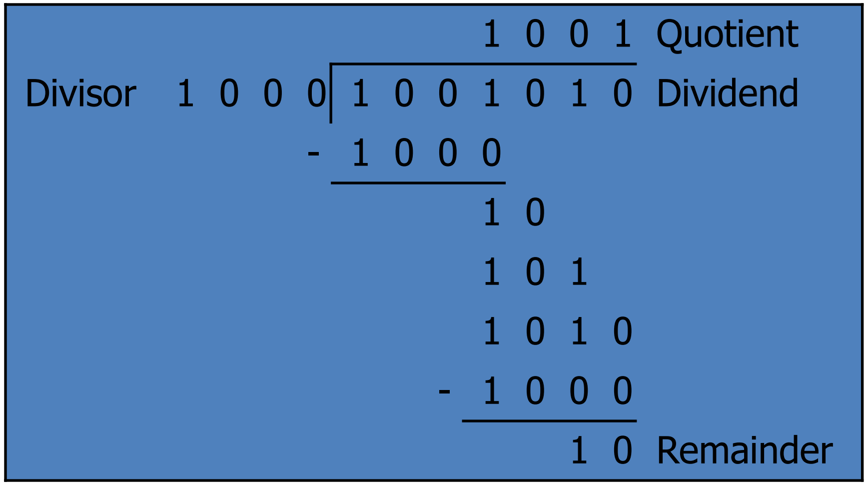


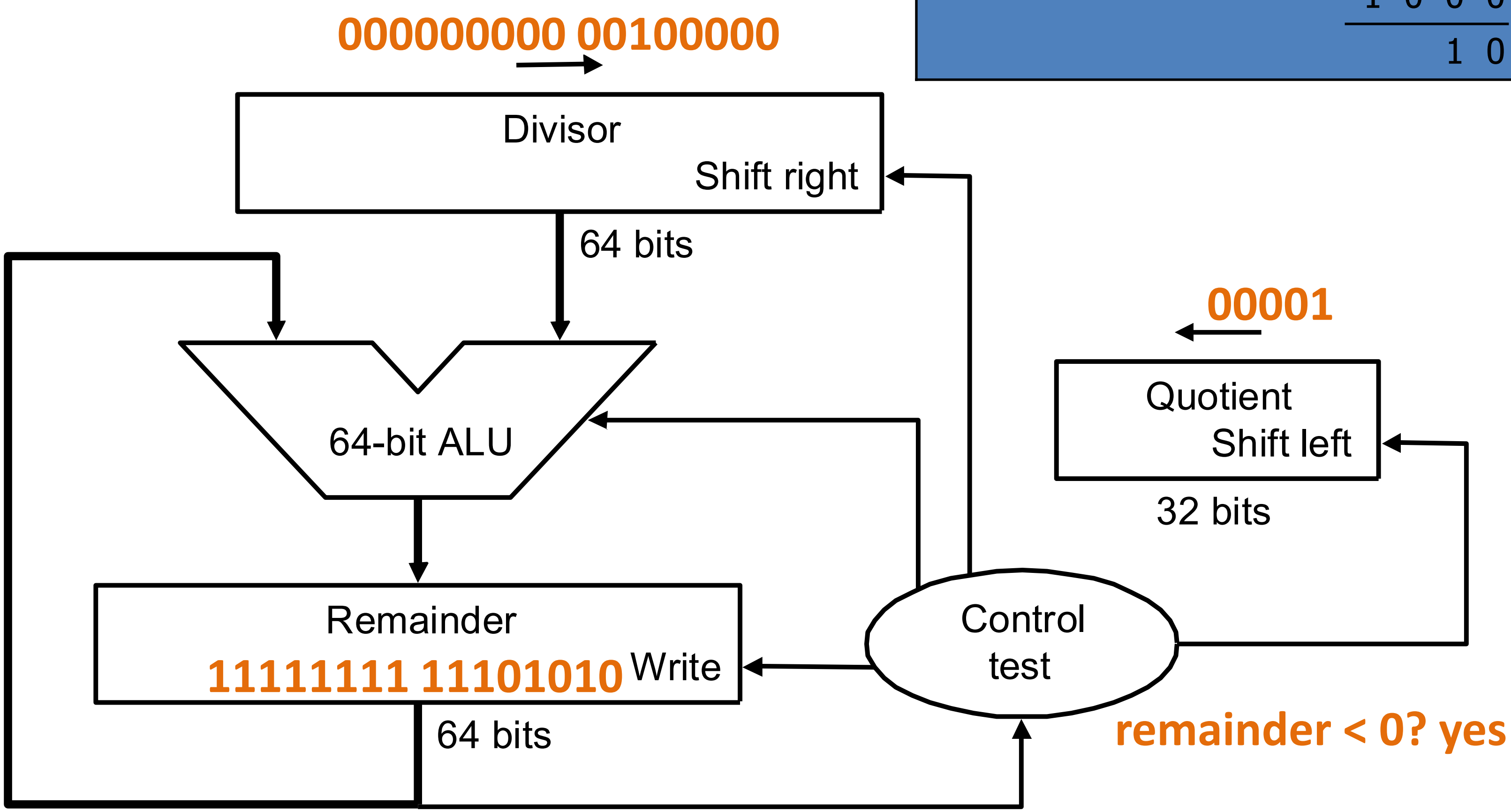
Restoring Division



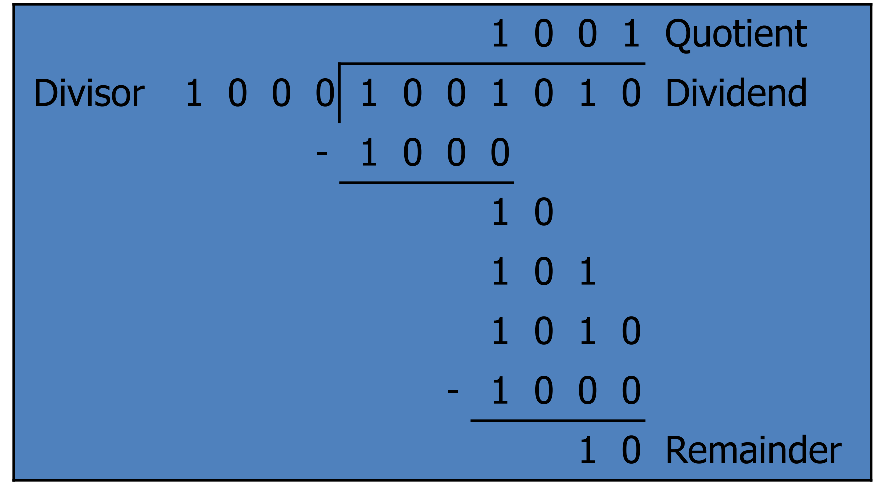


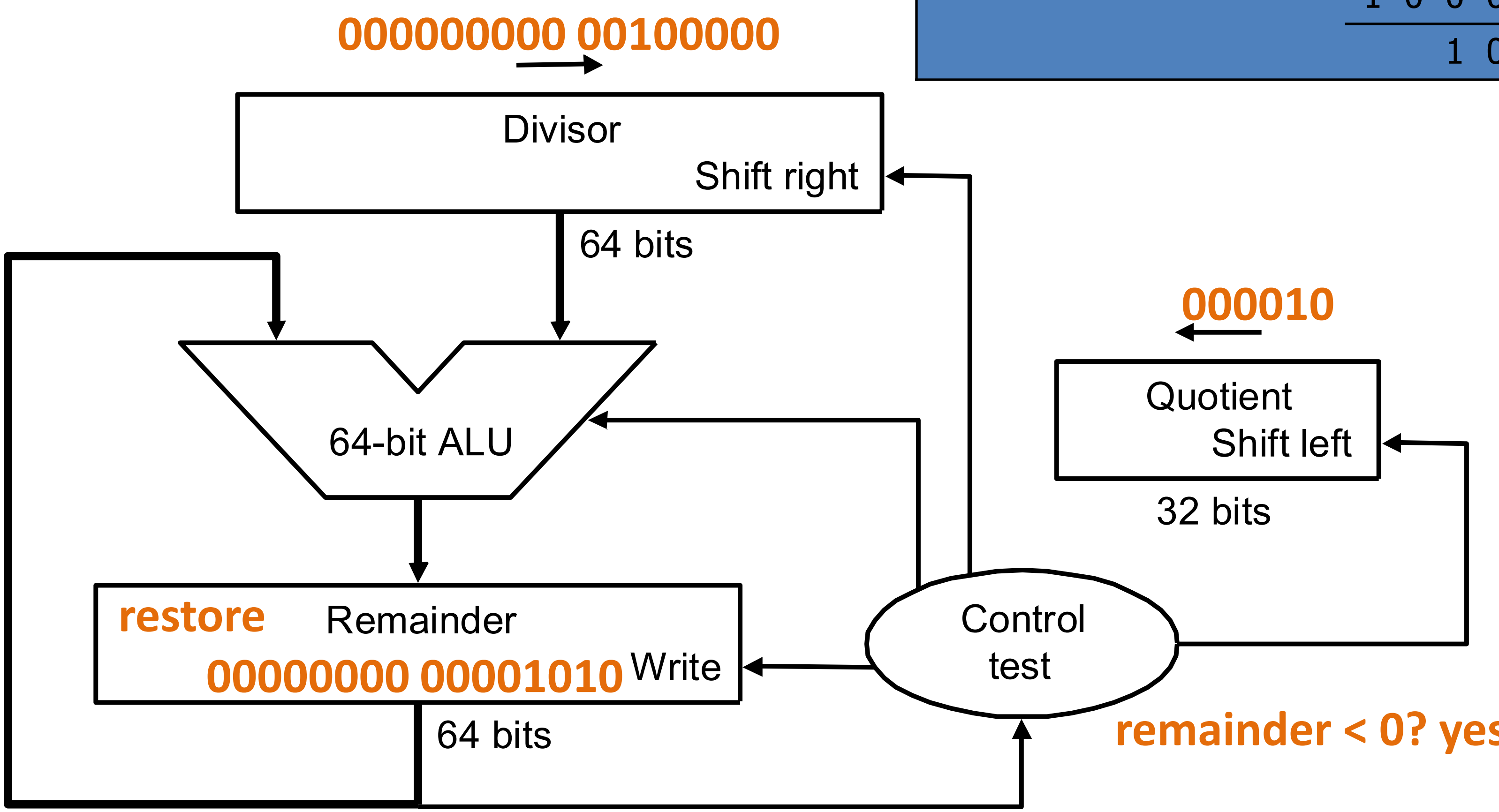
Restoring Division



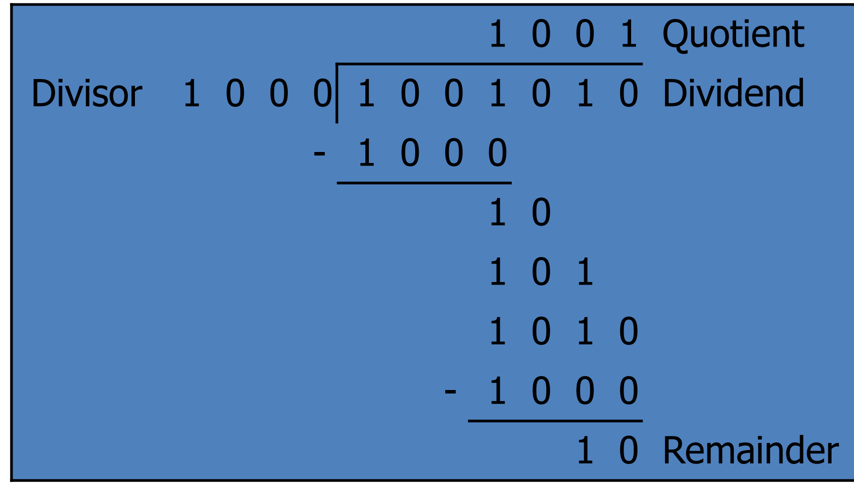


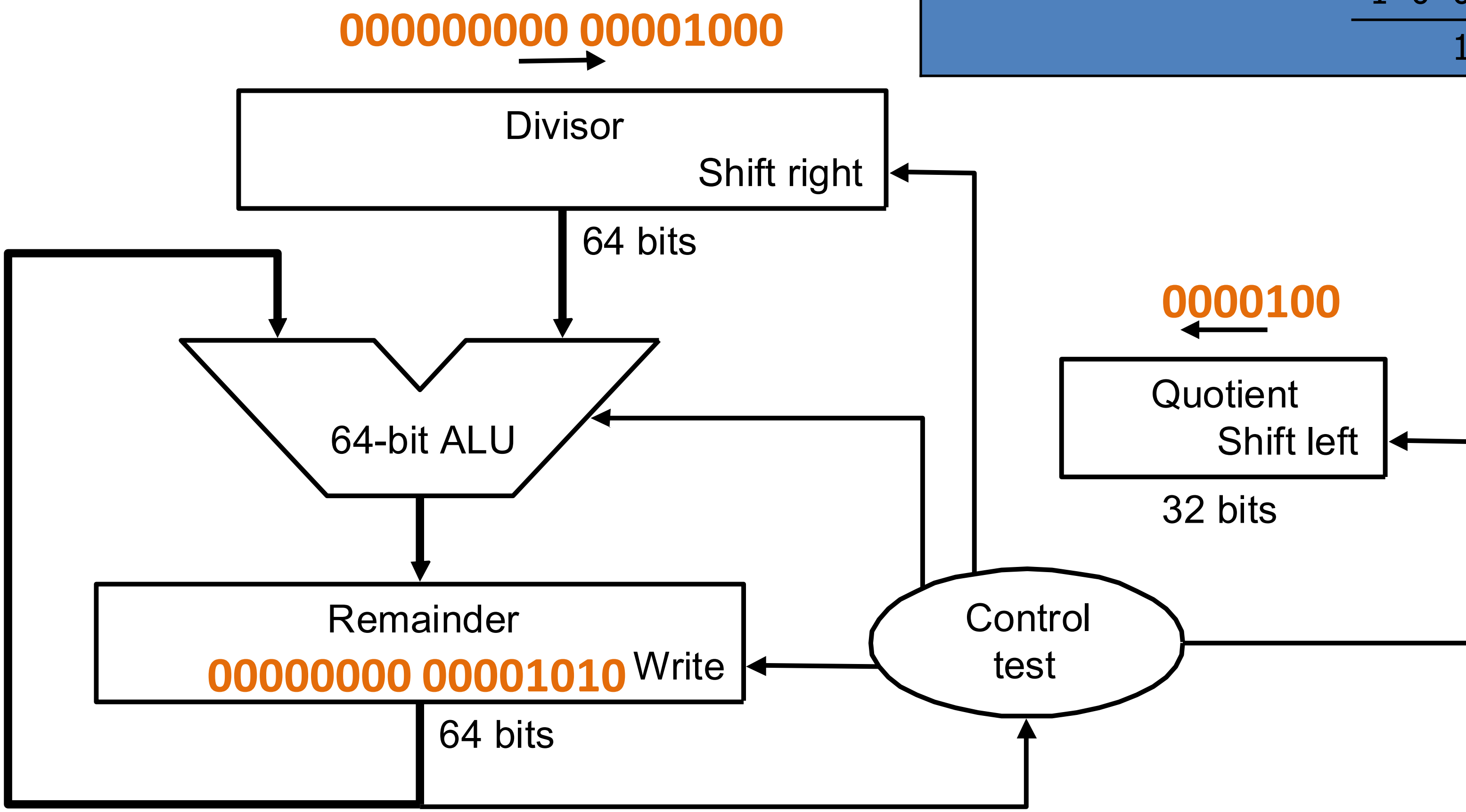
Restoring Division



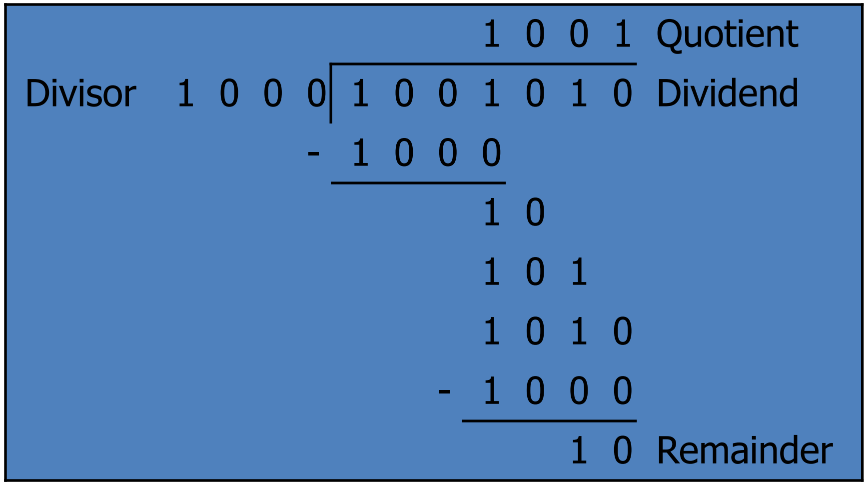


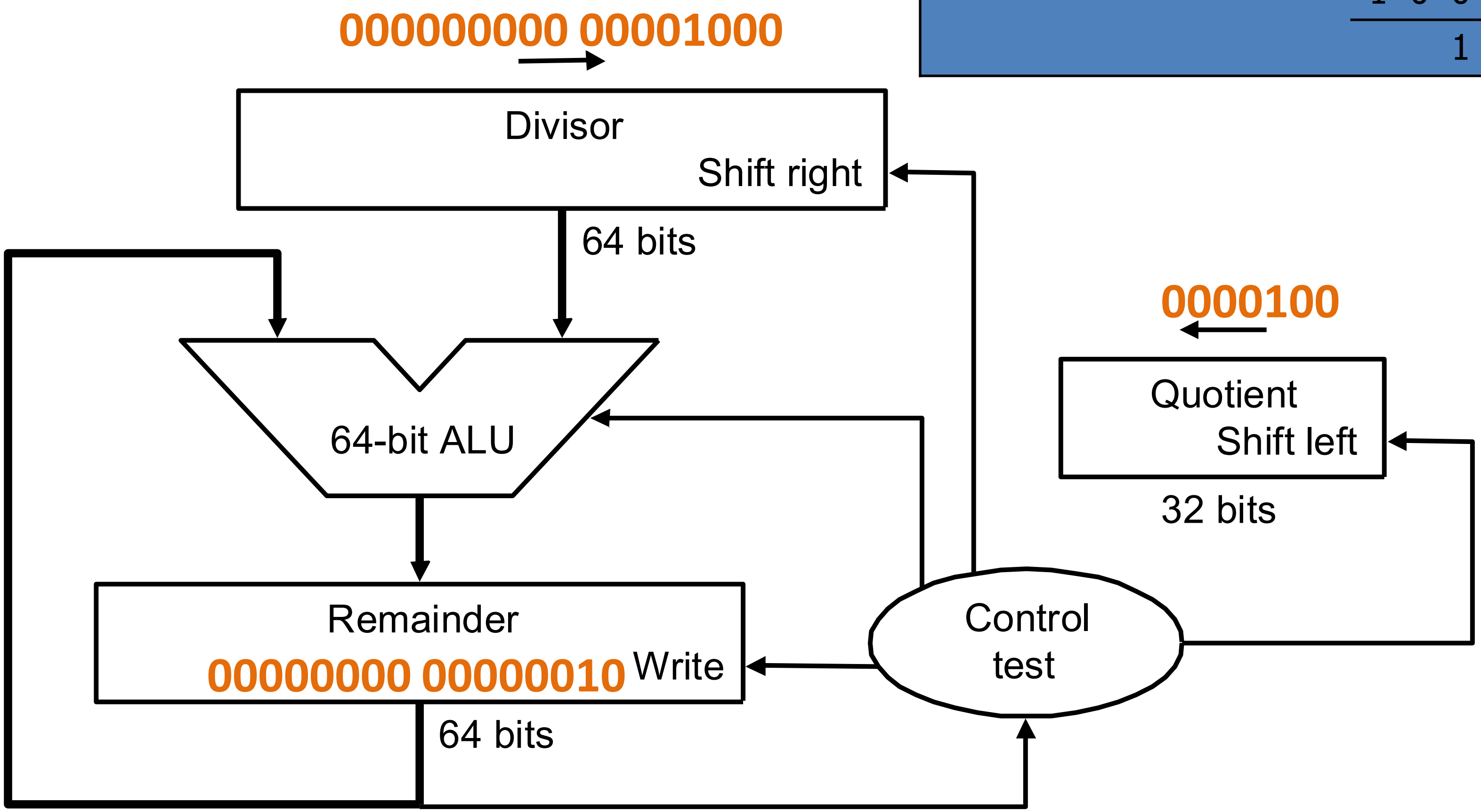
Restoring Division





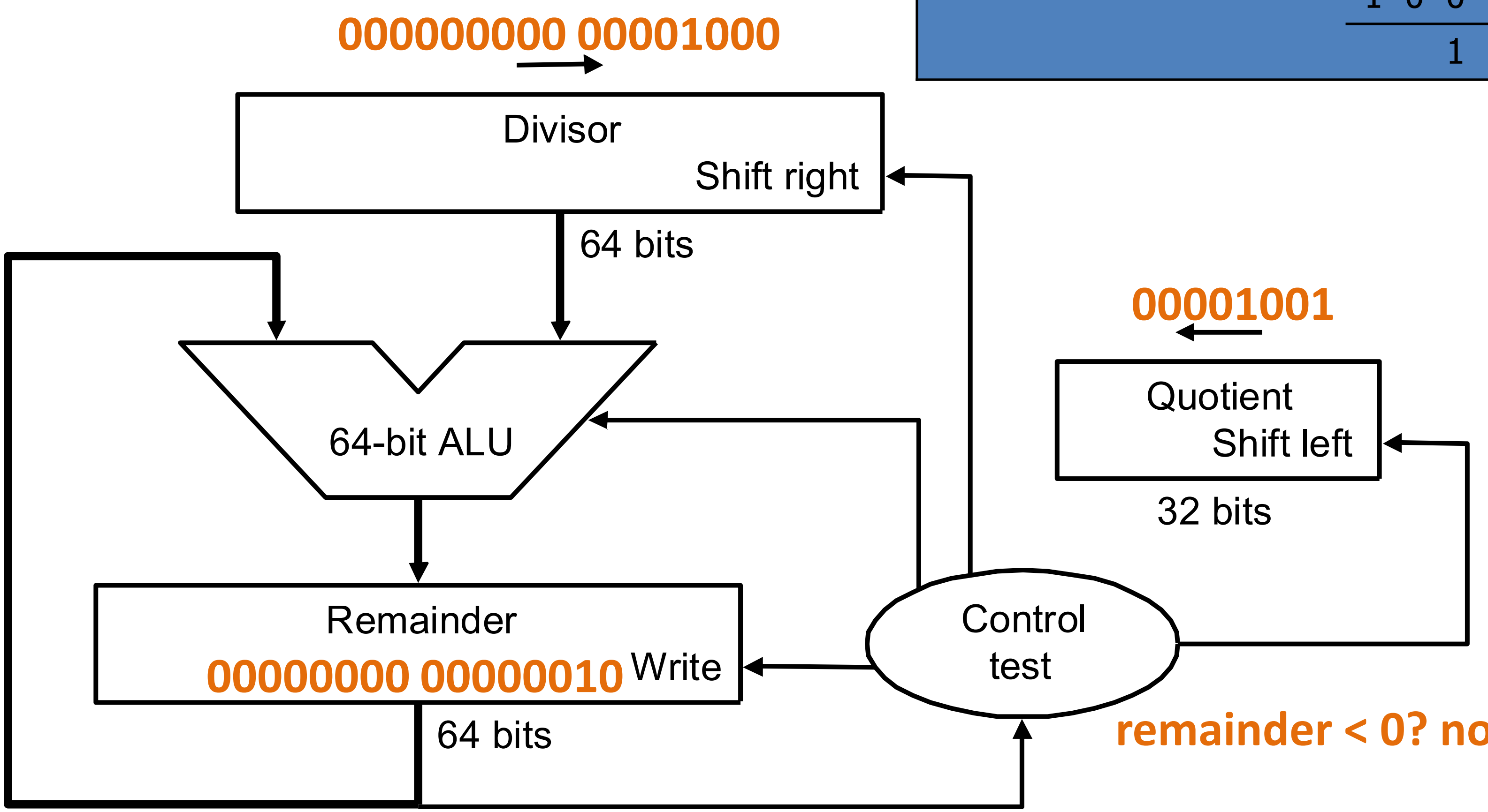
Restoring Division



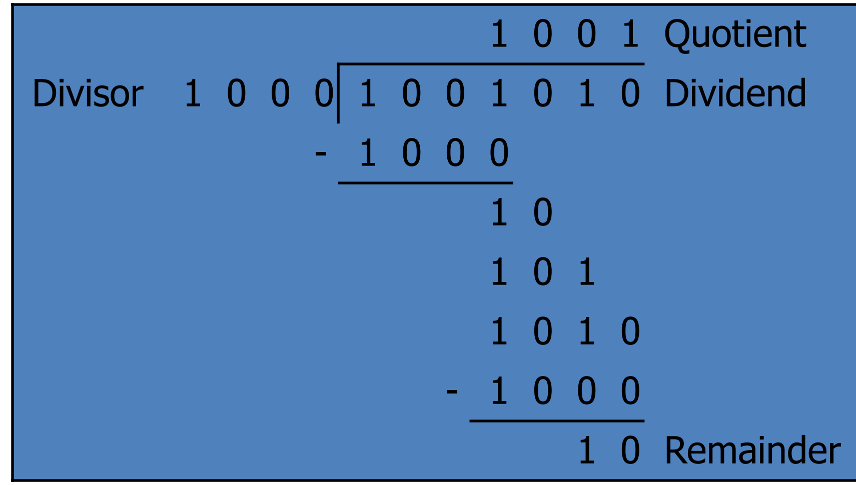


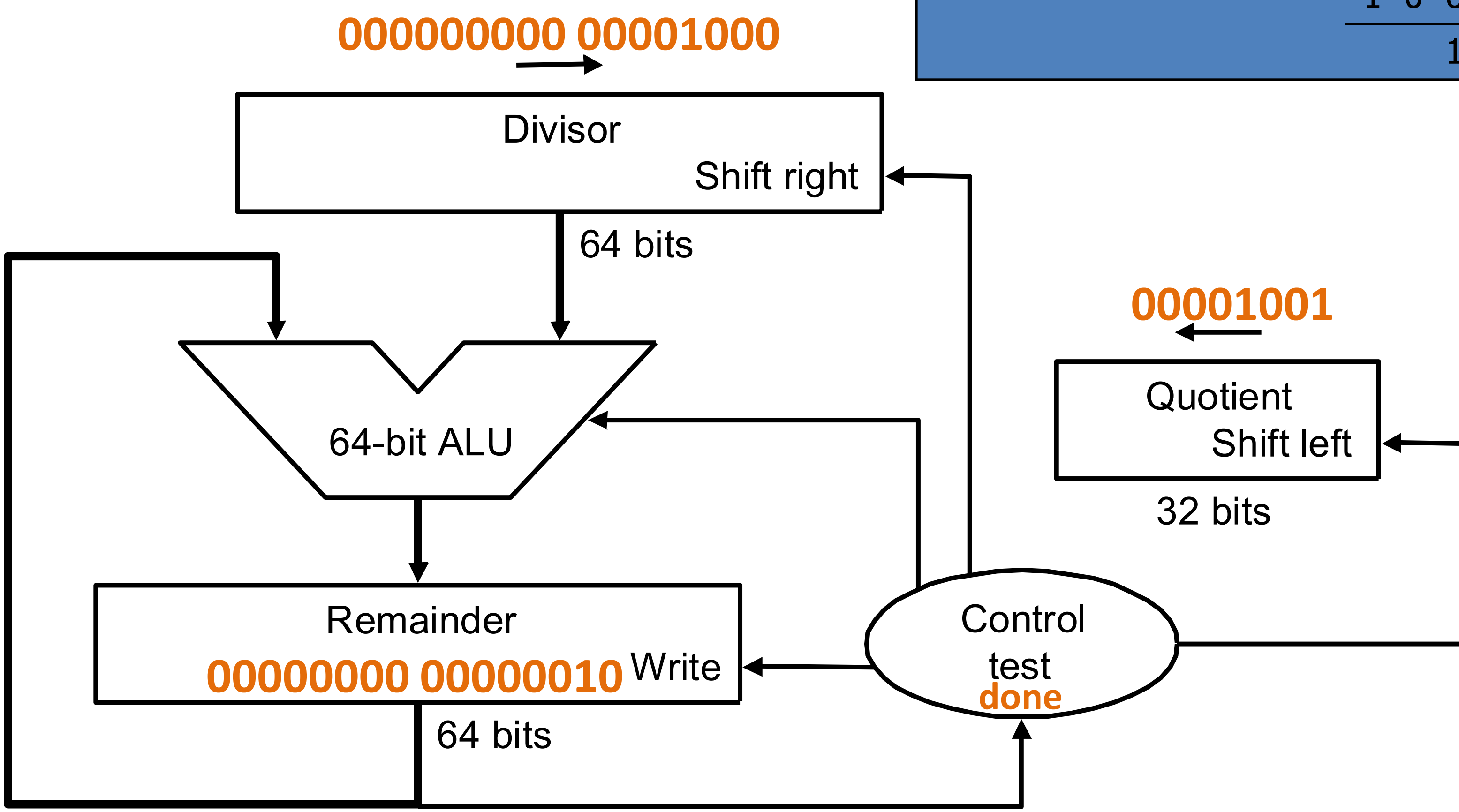
Restoring Division

```
                    1 0 0 1    Quotient
Divisor   1 0 0 0 | 1 0 0 1 0 1 0   Dividend
                  - 1 0 0 0
                        1 0
                        1 0 1
                        1 0 1 0
                      - 1 0 0 0
                            1 0   Remainder
```



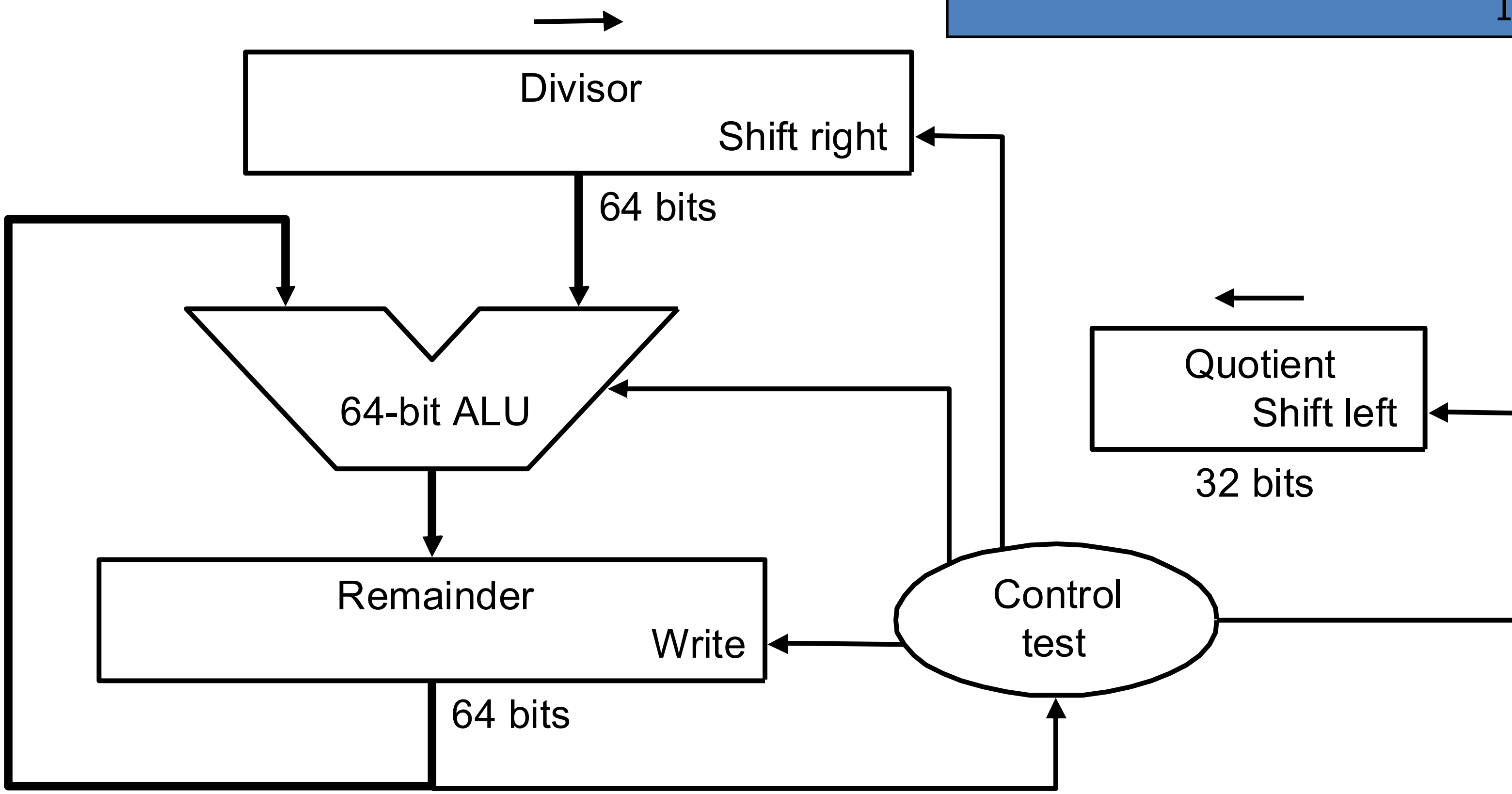
Restoring Division



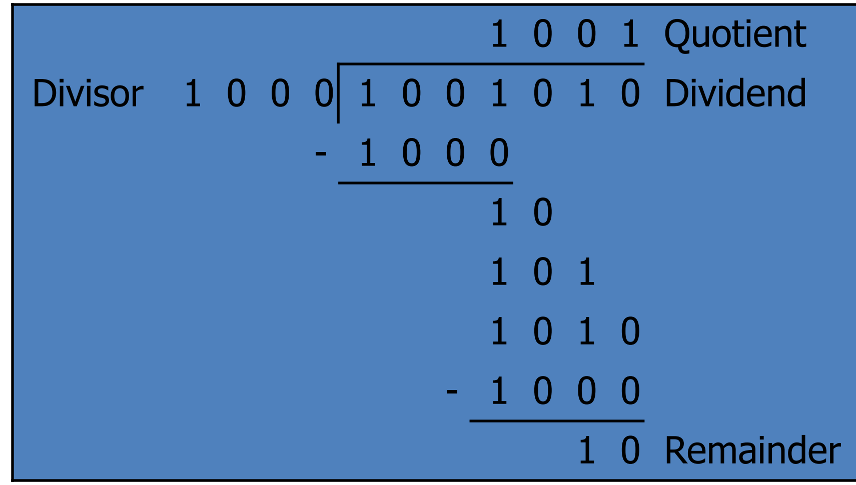


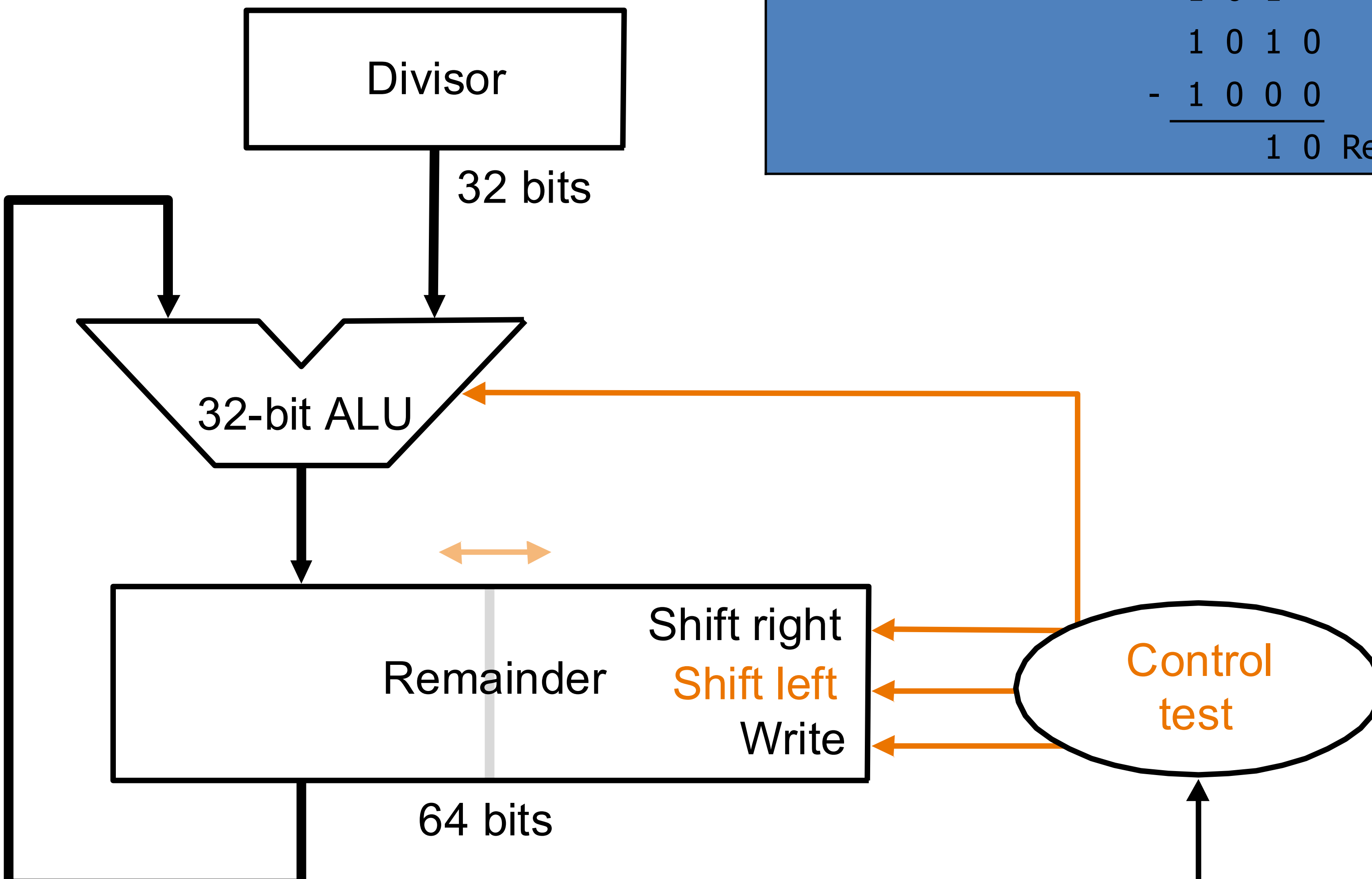
Restoring Division

```
                 1 0 0 1      Quotient
Divisor 1 0 0 0 | 1 0 0 1 0 1 0  Dividend
              - 1 0 0 0
                      1 0
                      1 0 1
                      1 0 1 0
                    - 1 0 0 0
                          1 0  Remainder
```

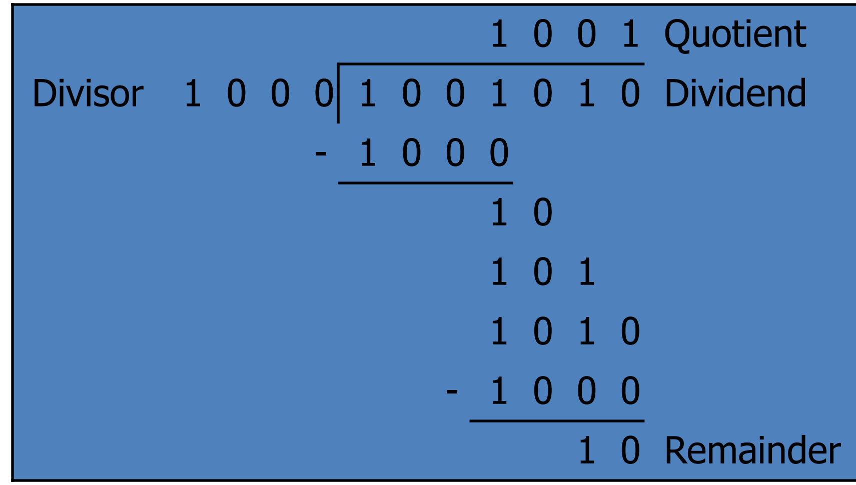


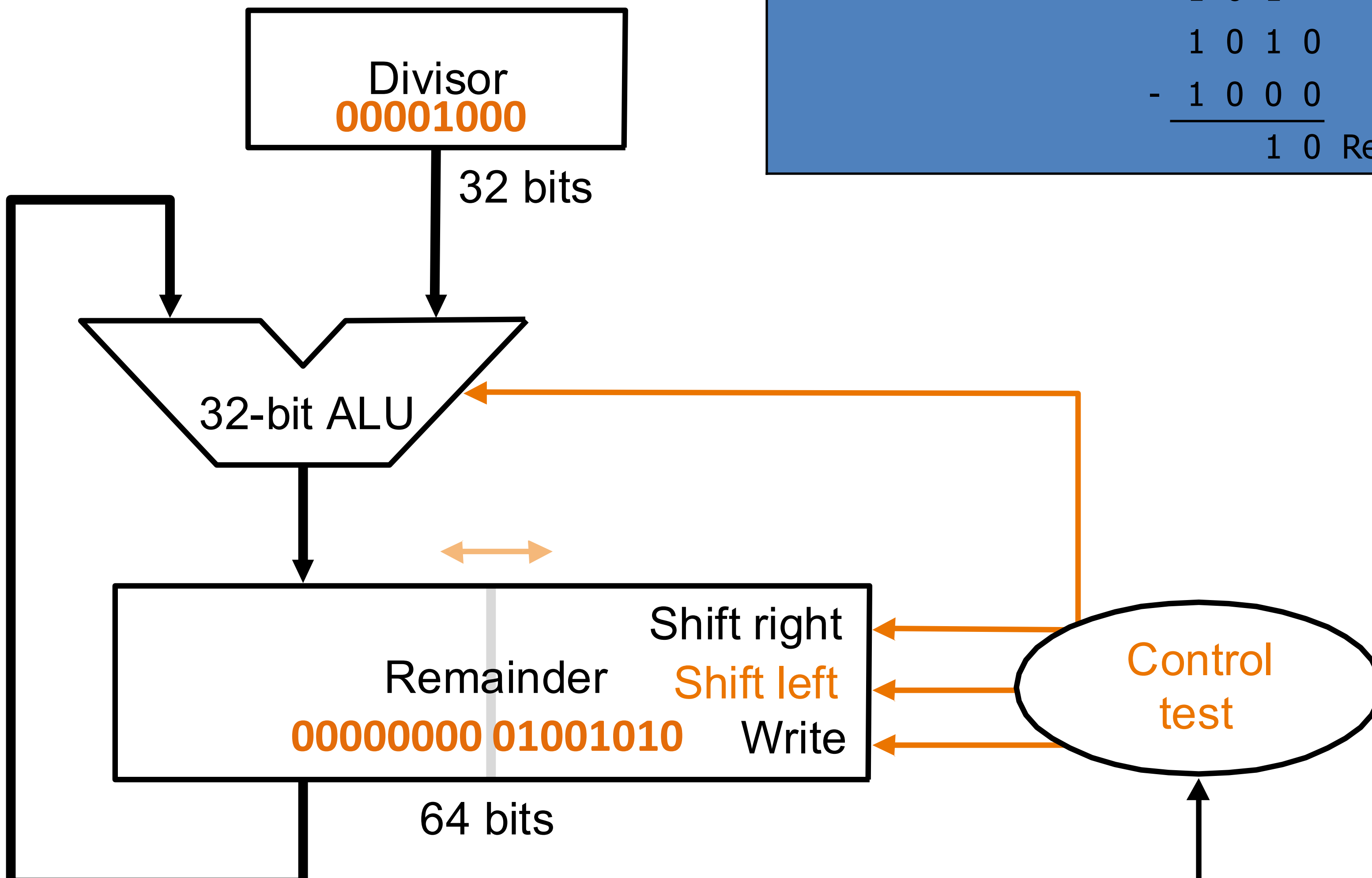
Restoring Division



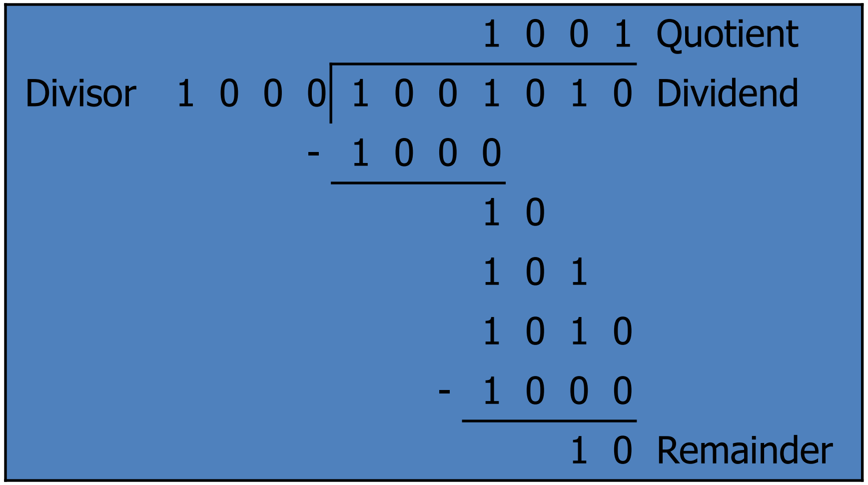


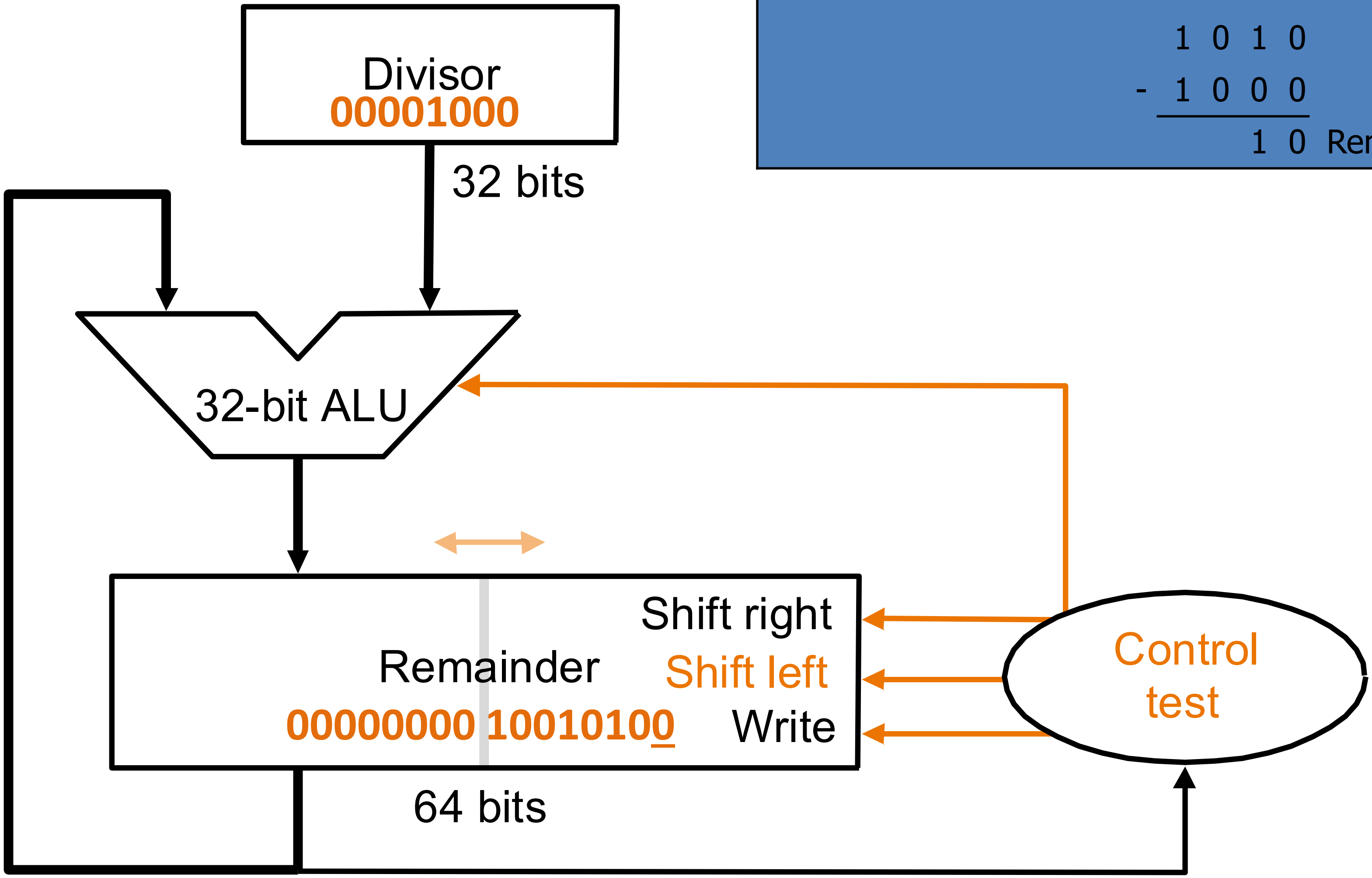
Restoring Division





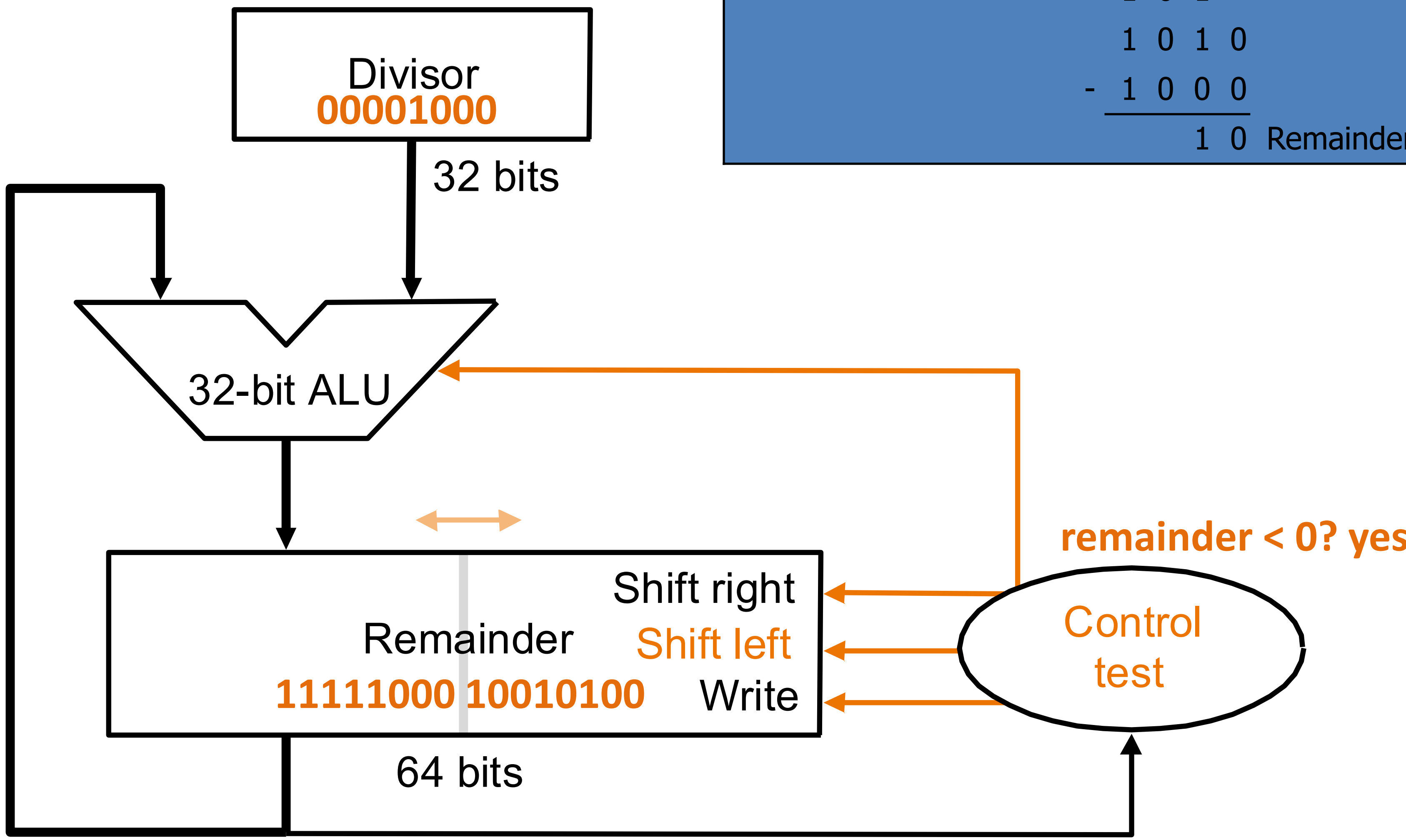
Restoring Division



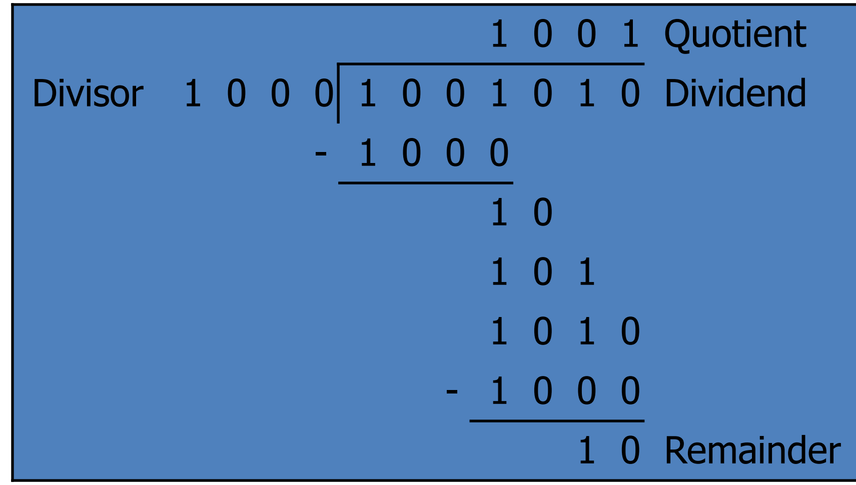


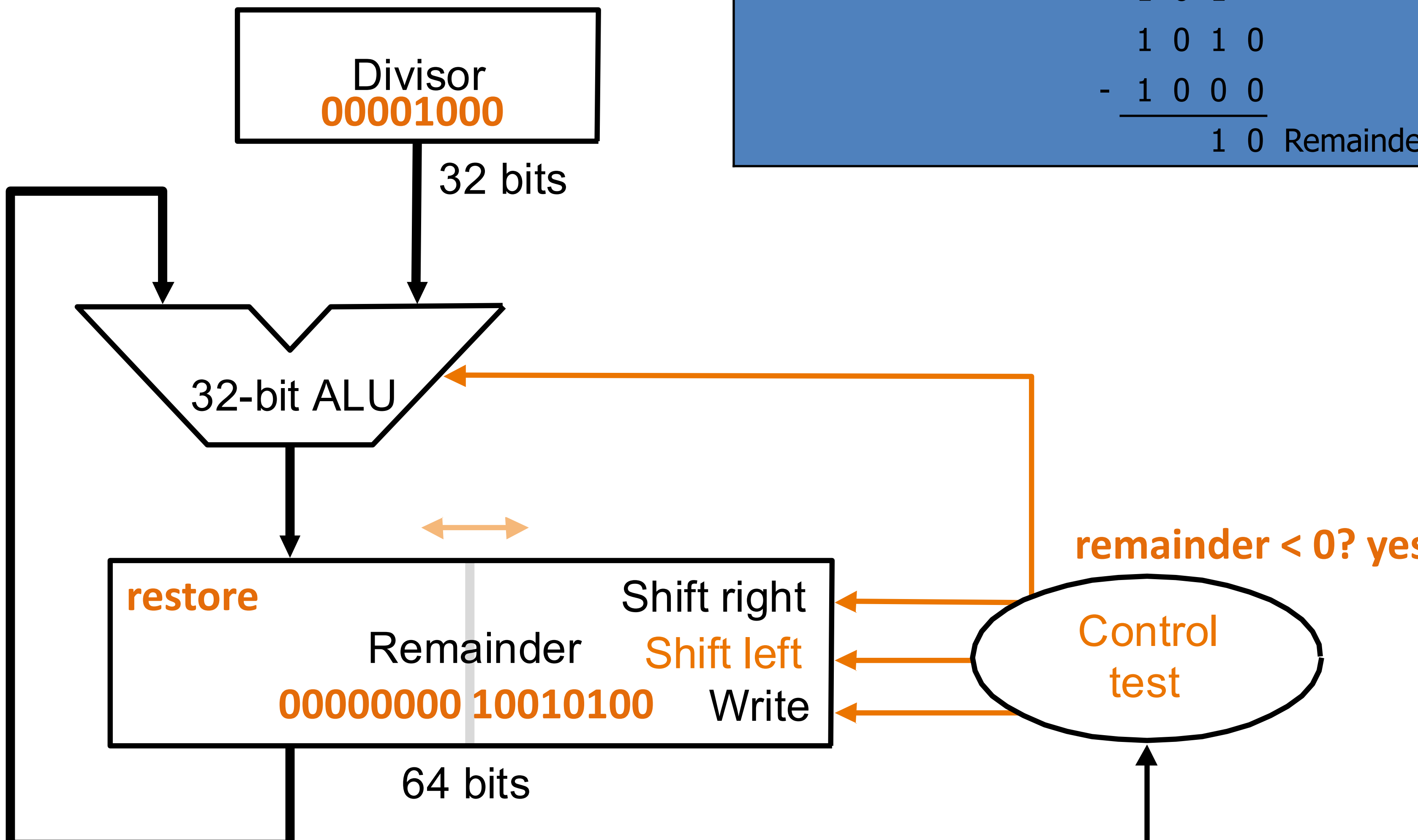
Restoring Division

```
           1 0 0 1        Quotient
Divisor 1 0 0 0 | 1 0 0 1 0 1 0   Dividend
              - 1 0 0 0
                      1 0
                      1 0 1
                      1 0 1 0
                    - 1 0 0 0
                          1 0     Remainder
```

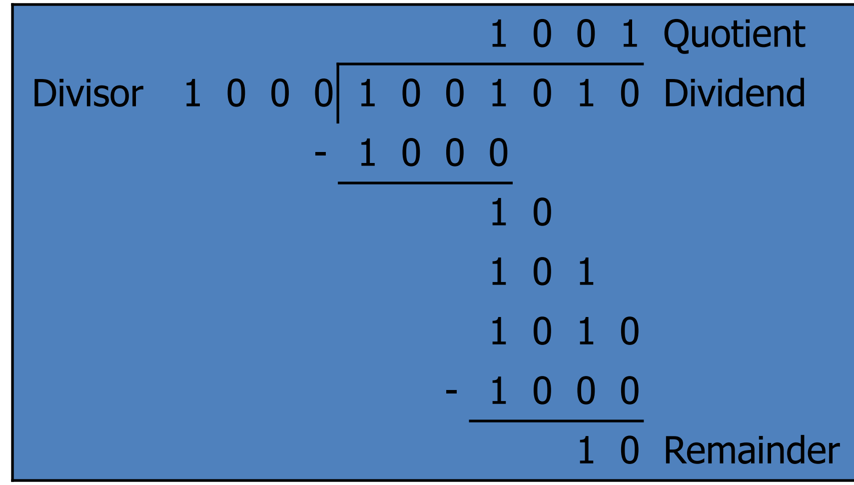


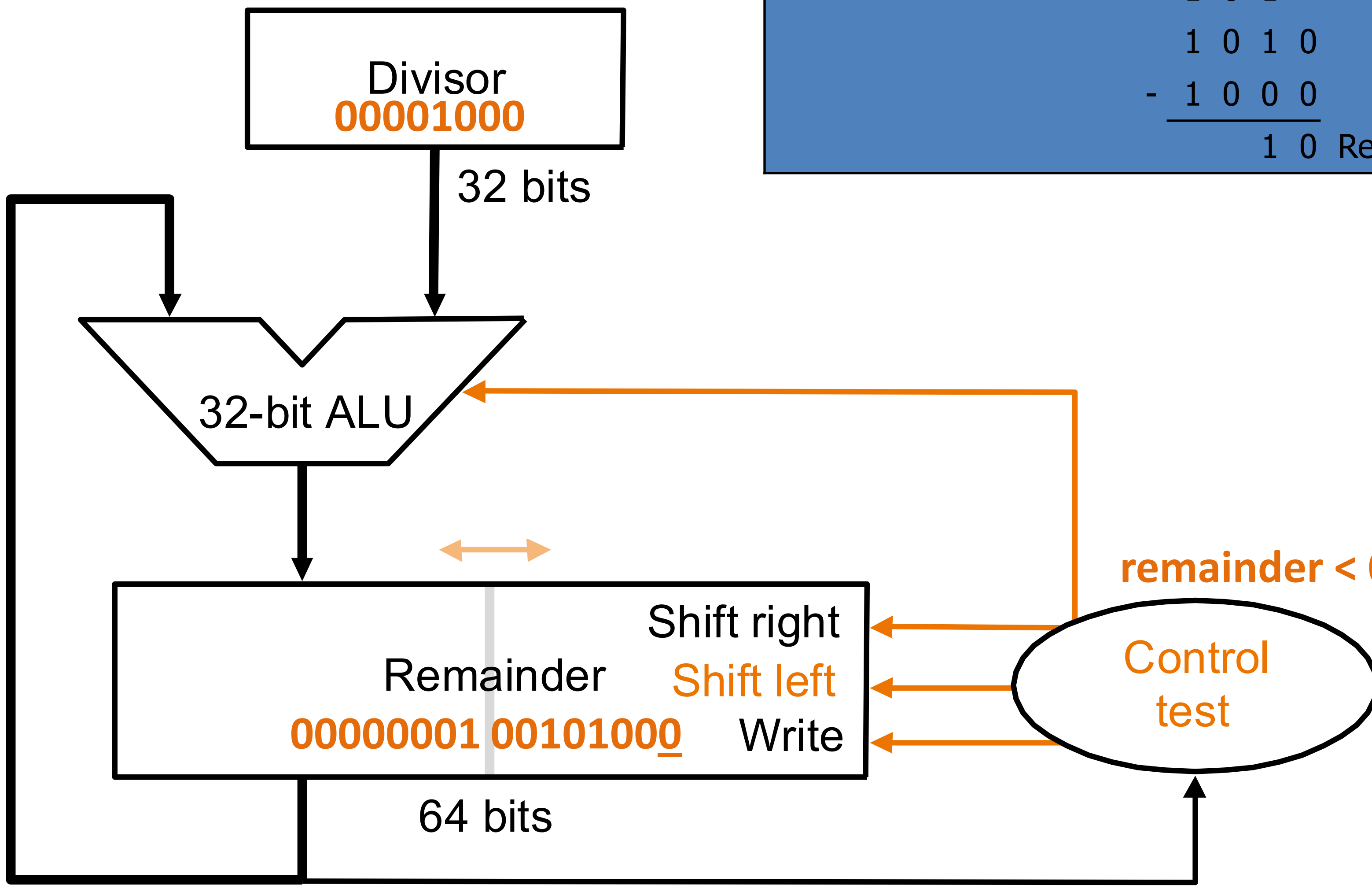
Restoring Division



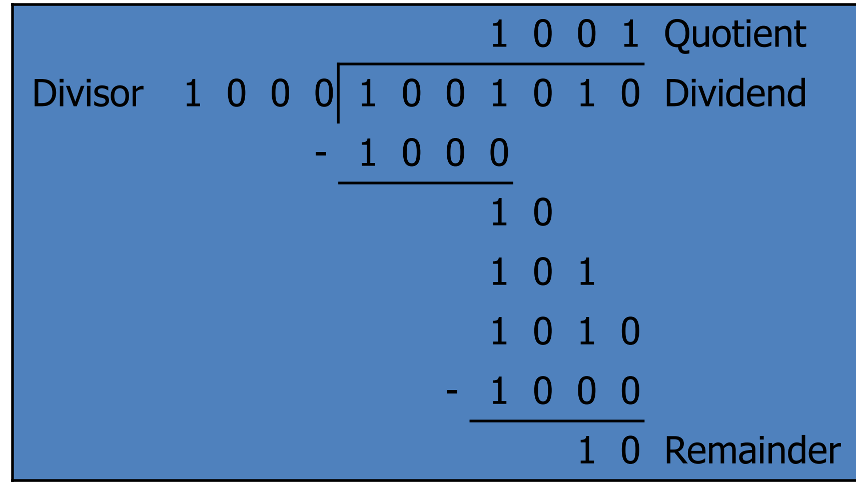


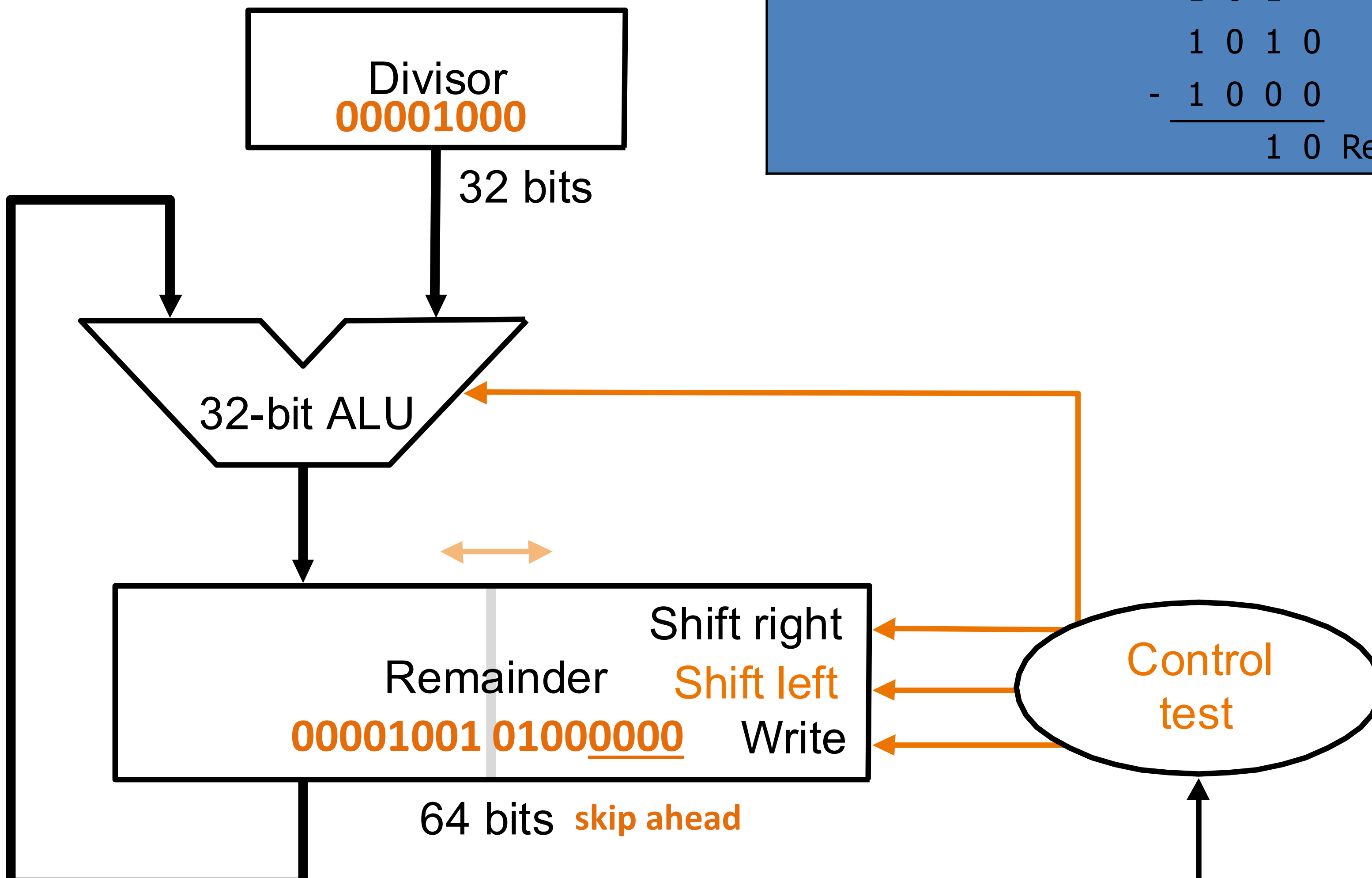
Restoring Division





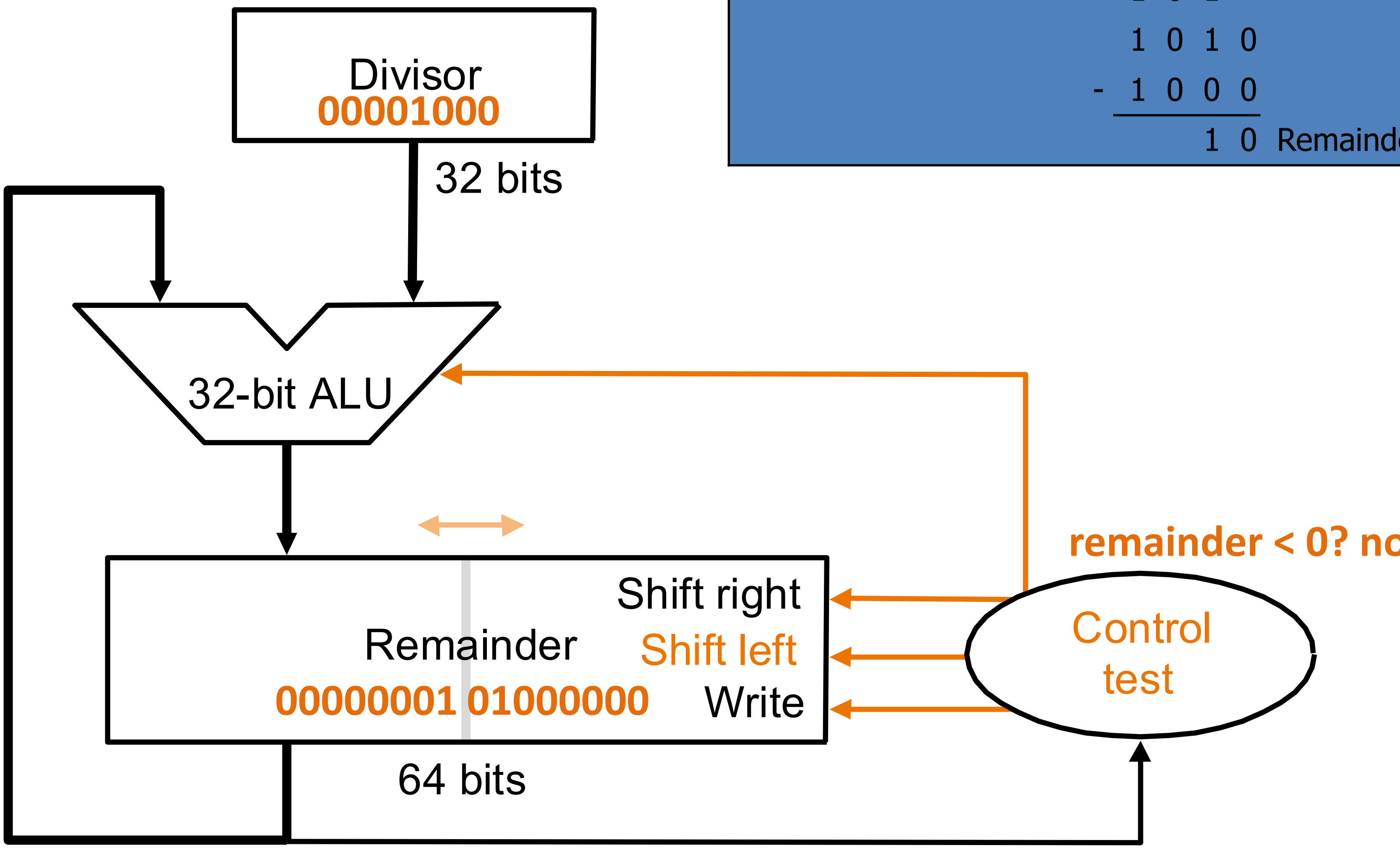
Restoring Division



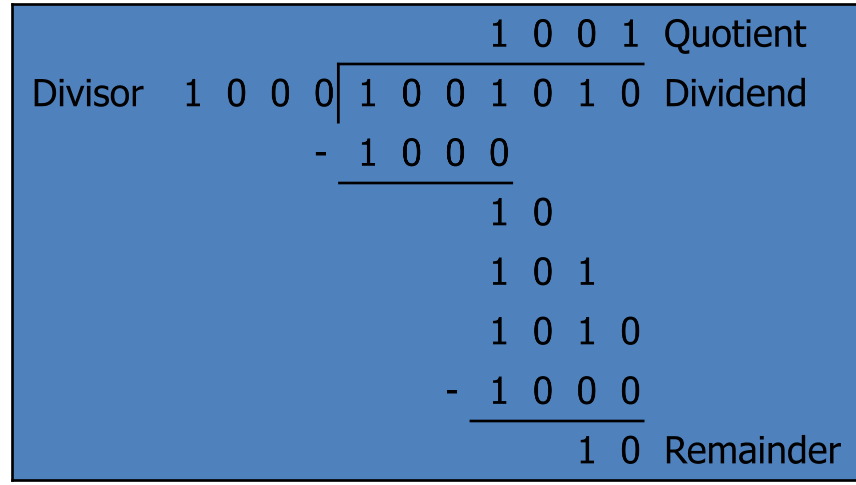


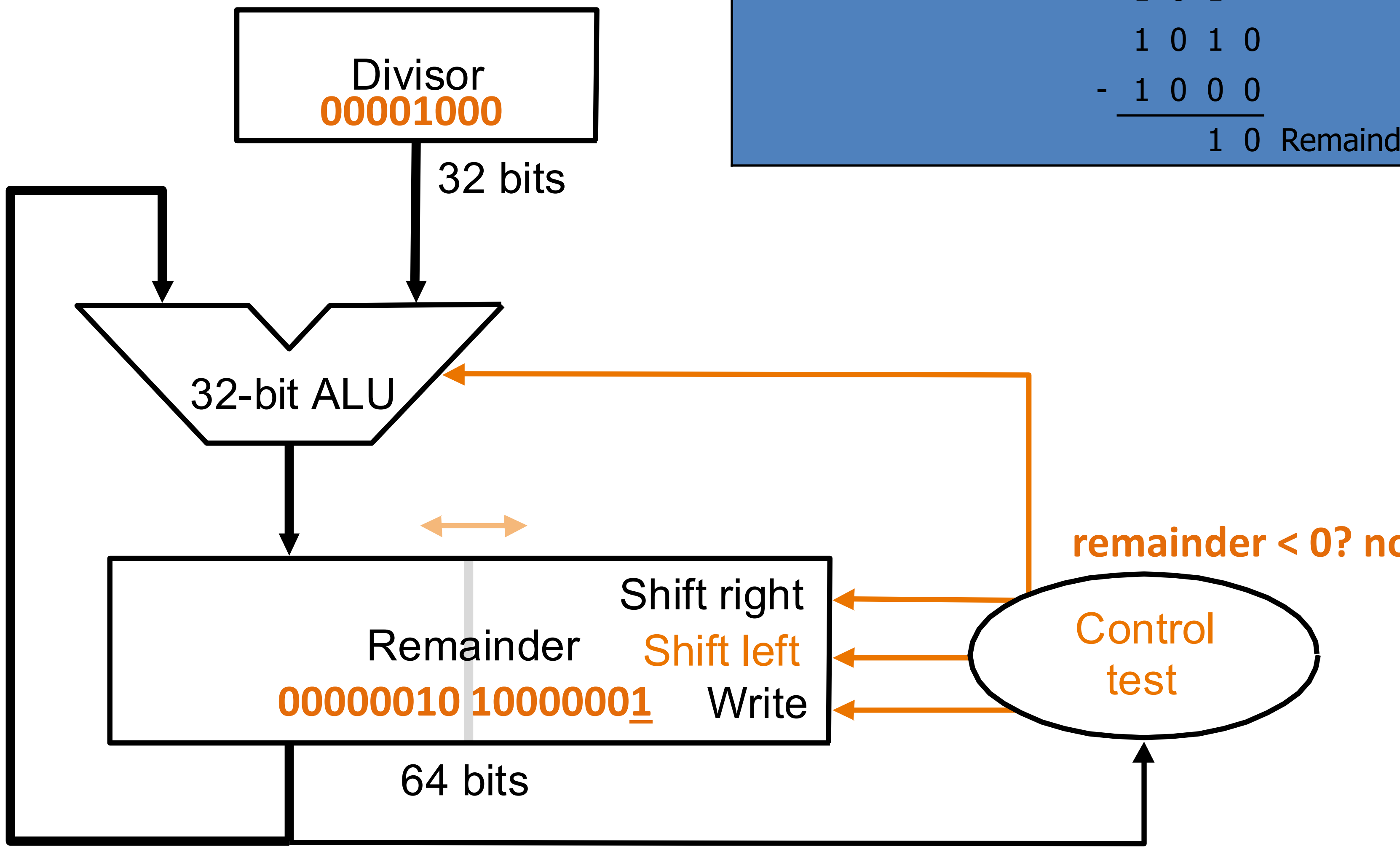
Restoring Division

```
                         1  0  0  1   Quotient
Divisor    1  0  0  0 | 1  0  0  1  0  1  0   Dividend
                    - 1  0  0  0
                               1  0
                               1  0  1
                               1  0  1  0
                            -  1  0  0  0
                                     1  0   Remainder
```

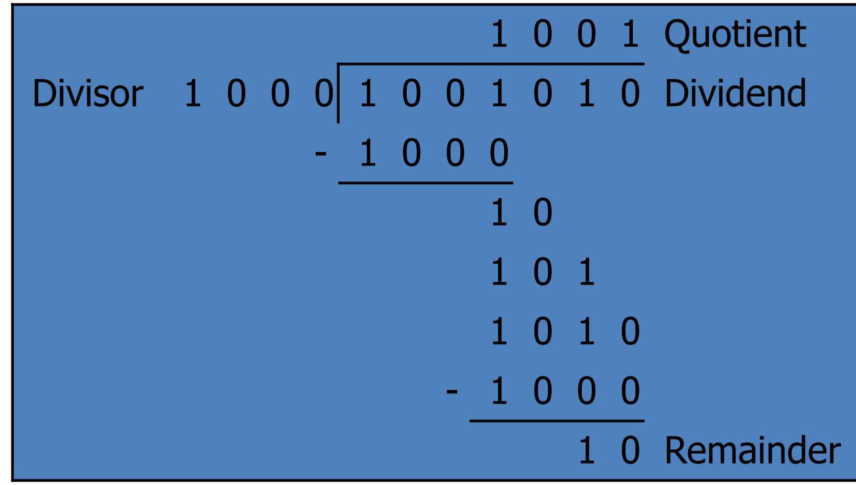


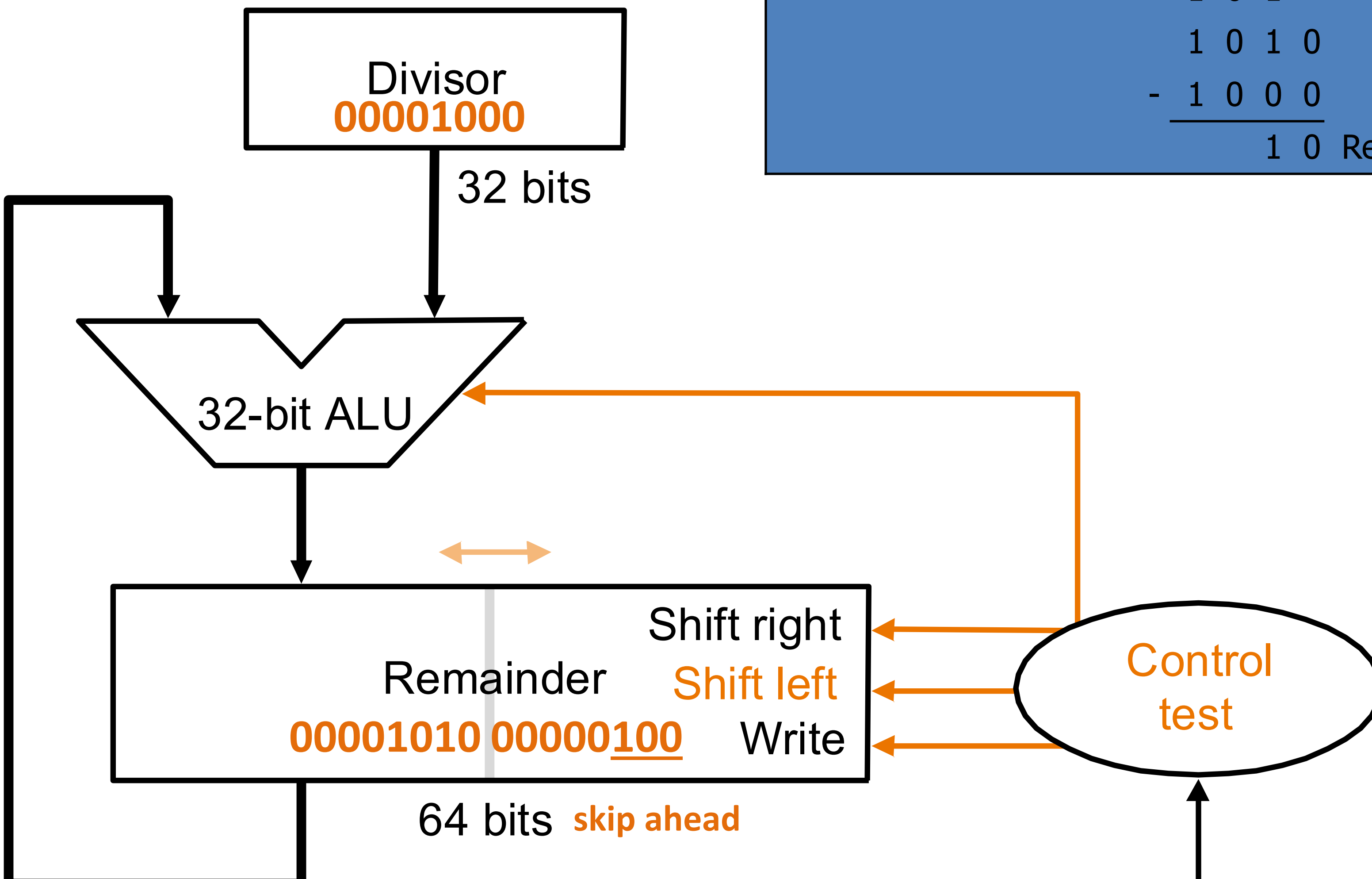
Restoring Division





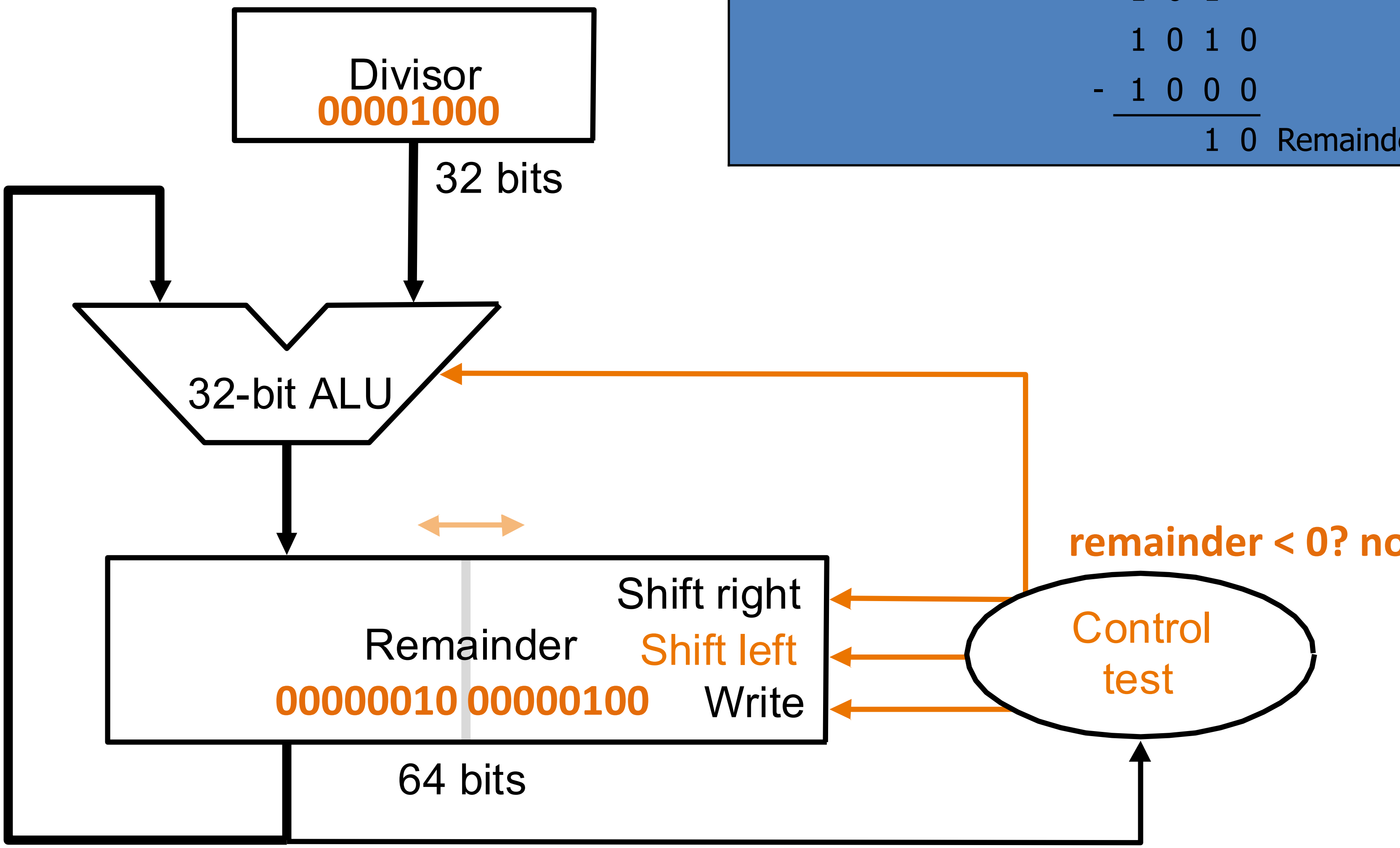
Restoring Division



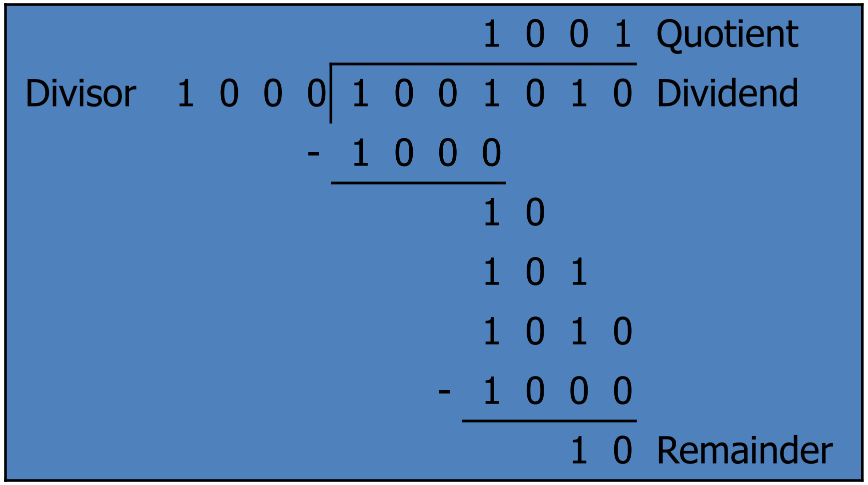


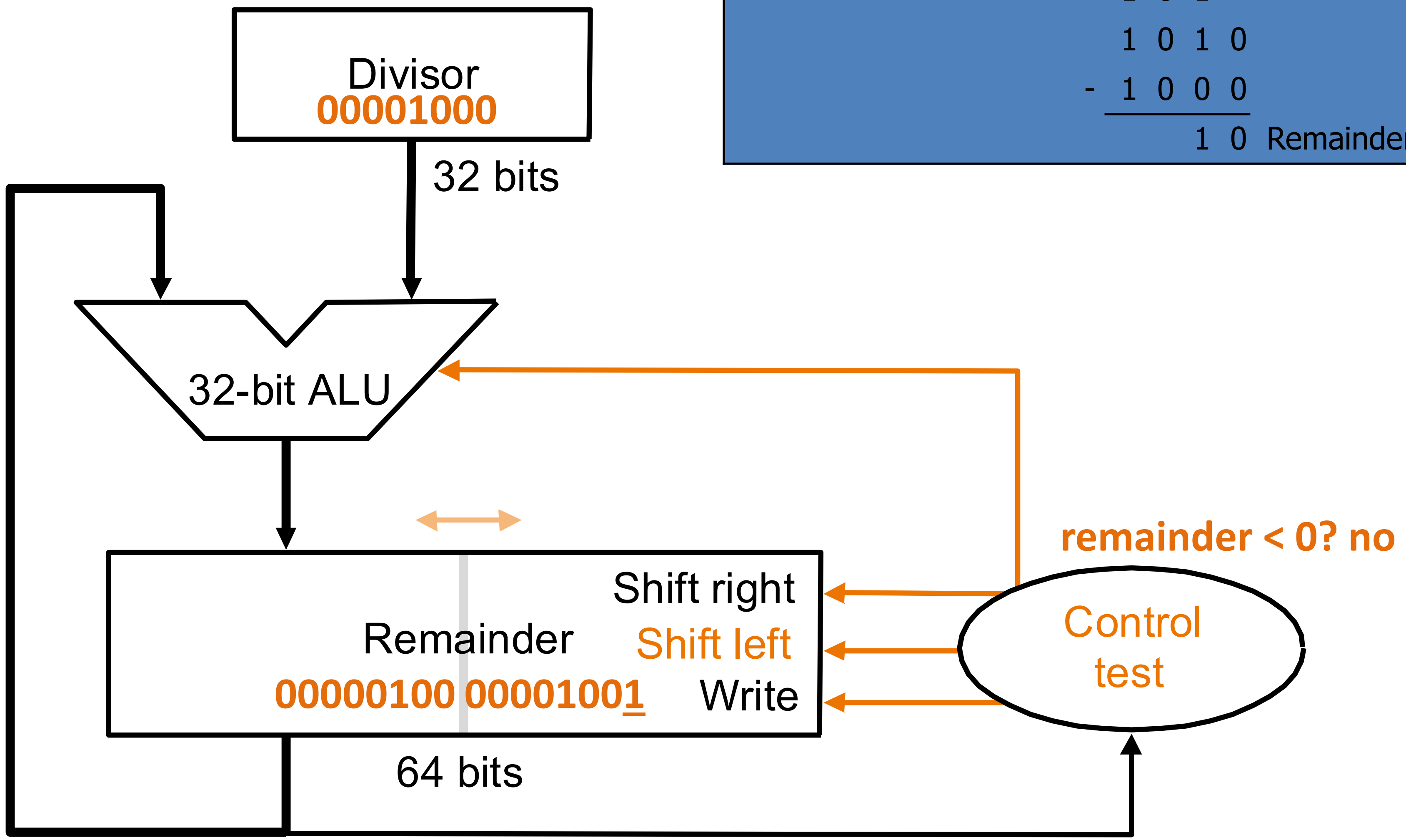
Restoring Division

```
                 1 0 0 1     Quotient
Divisor 1 0 0 0 |1 0 0 1 0 1 0 Dividend
               - 1 0 0 0
                       1 0
                       1 0 1
                       1 0 1 0
                     - 1 0 0 0
                           1 0 Remainder
```

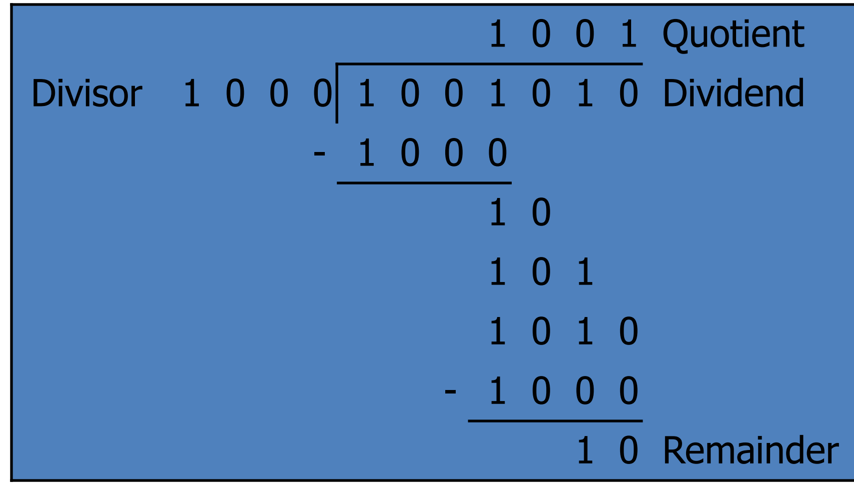


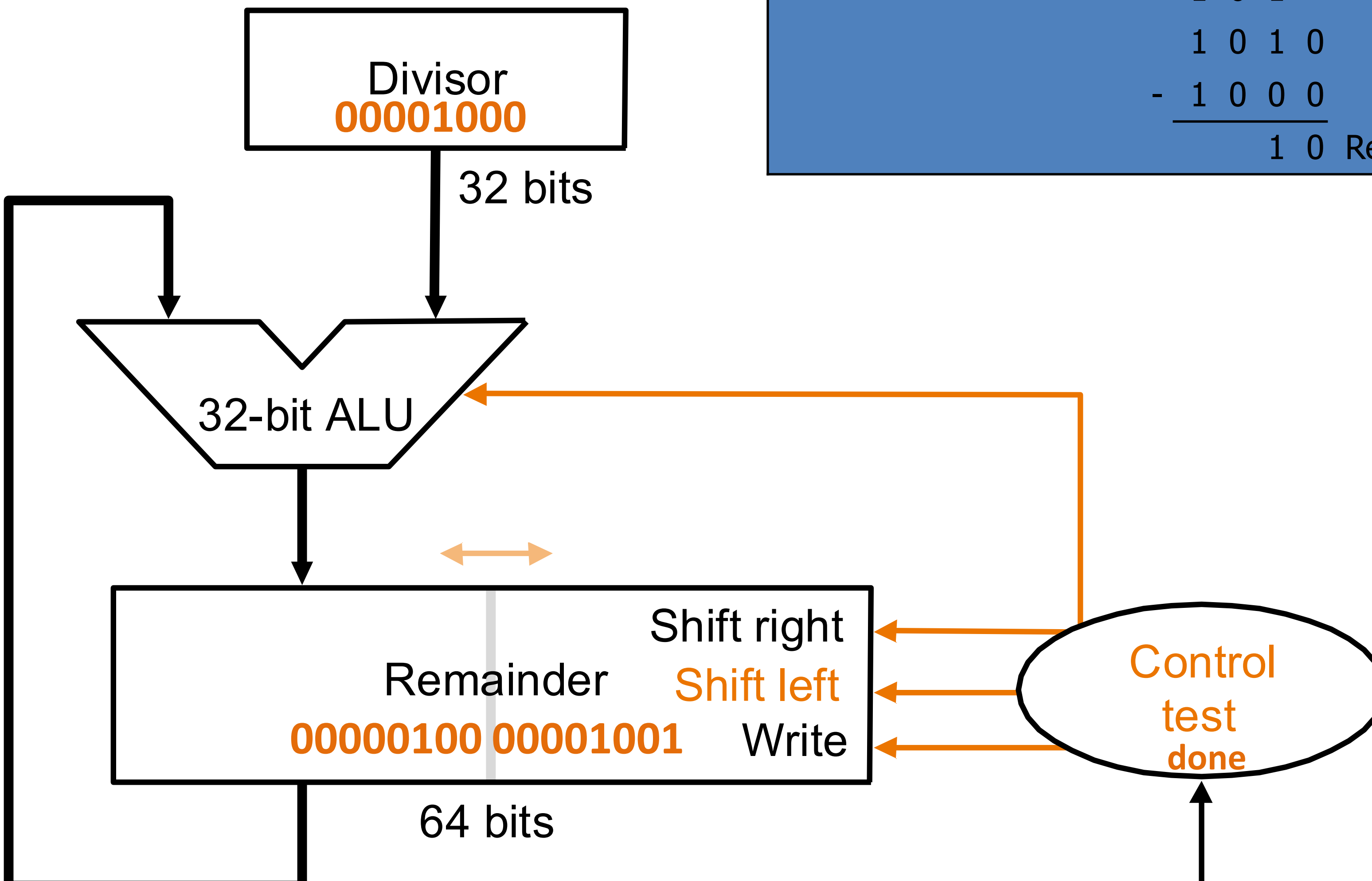
Restoring Division





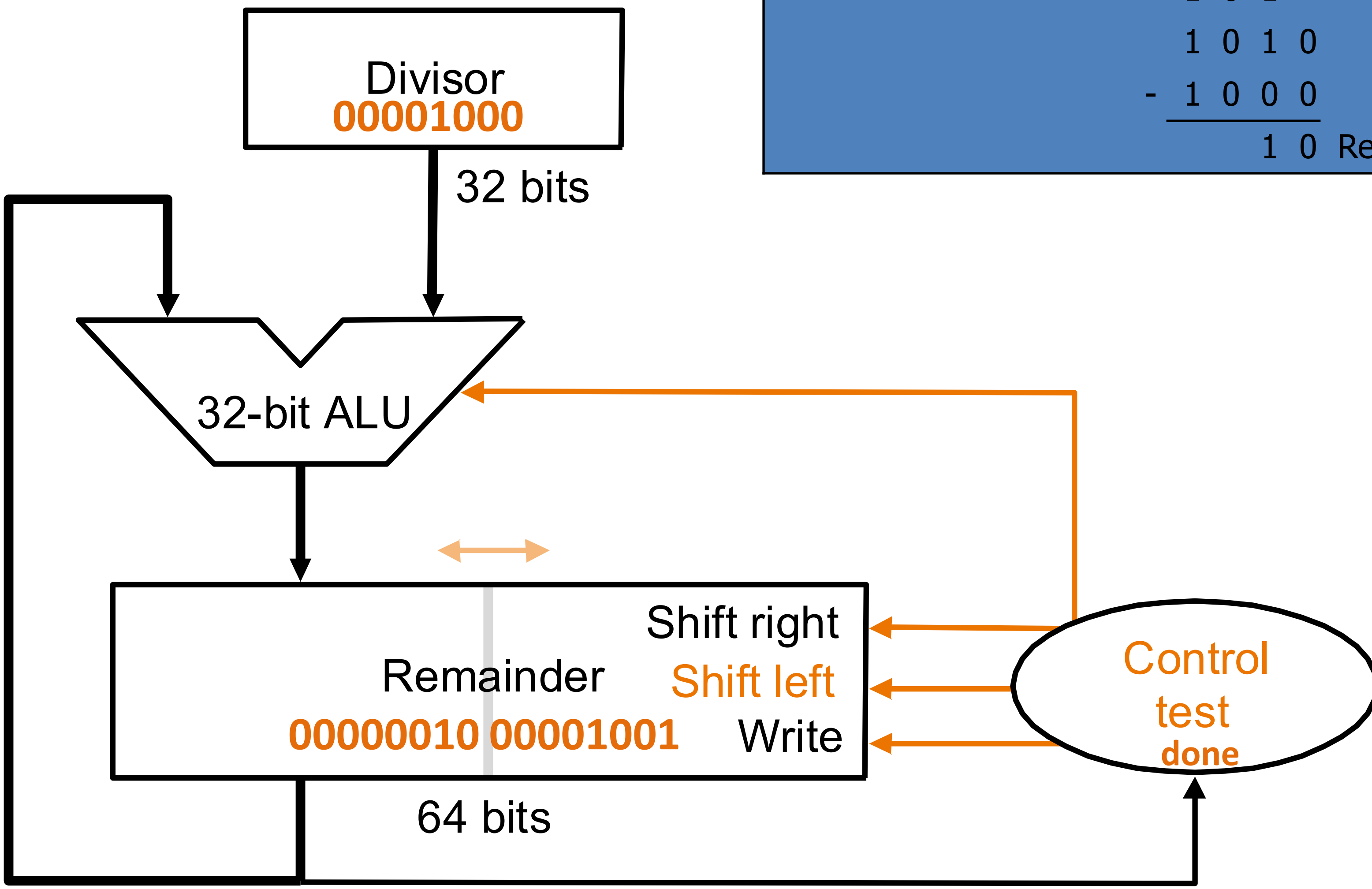
Restoring Division



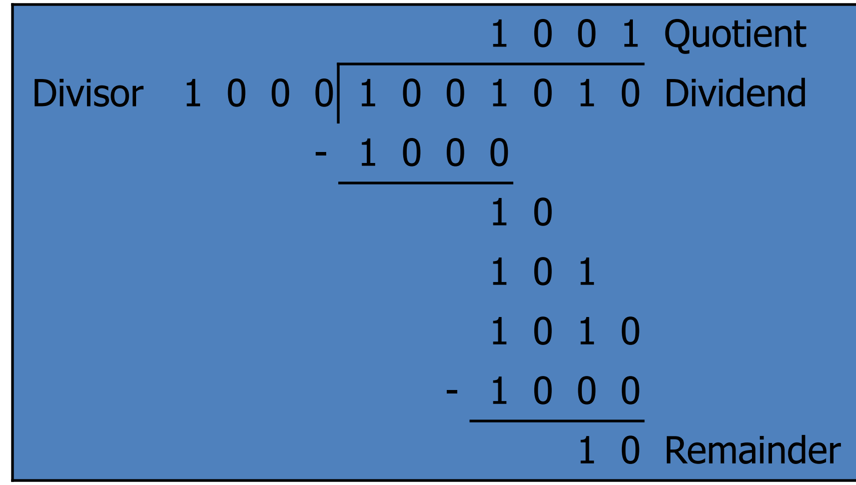


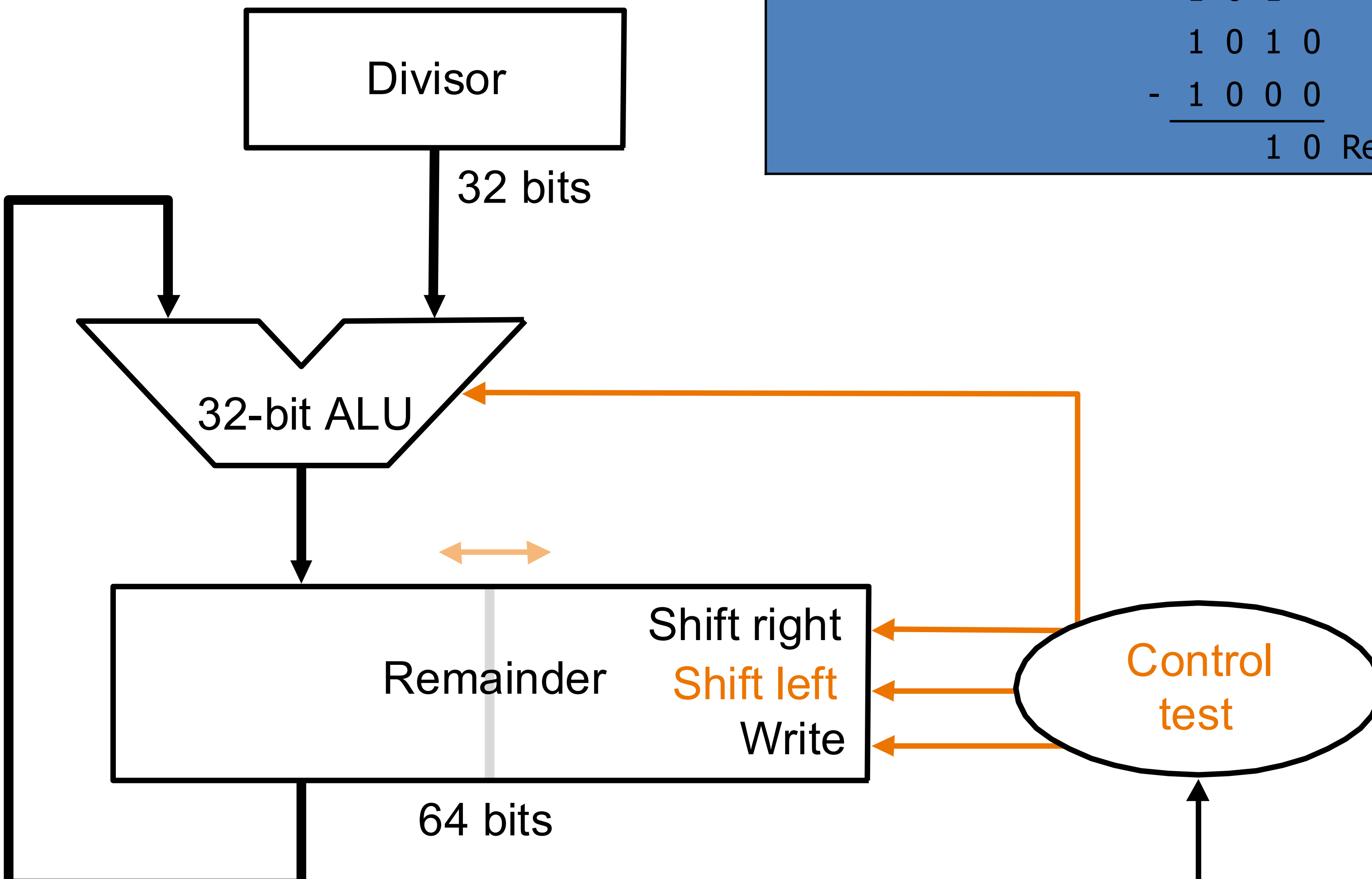
Restoring Division

```
                    1 0 0 1      Quotient
Divisor  1 0 0 0 | 1 0 0 1 0 1 0 Dividend
                 - 1 0 0 0
                   -------
                         1 0
                         1 0 1
                         1 0 1 0
                       - 1 0 0 0
                         -------
                             1 0 Remainder
```



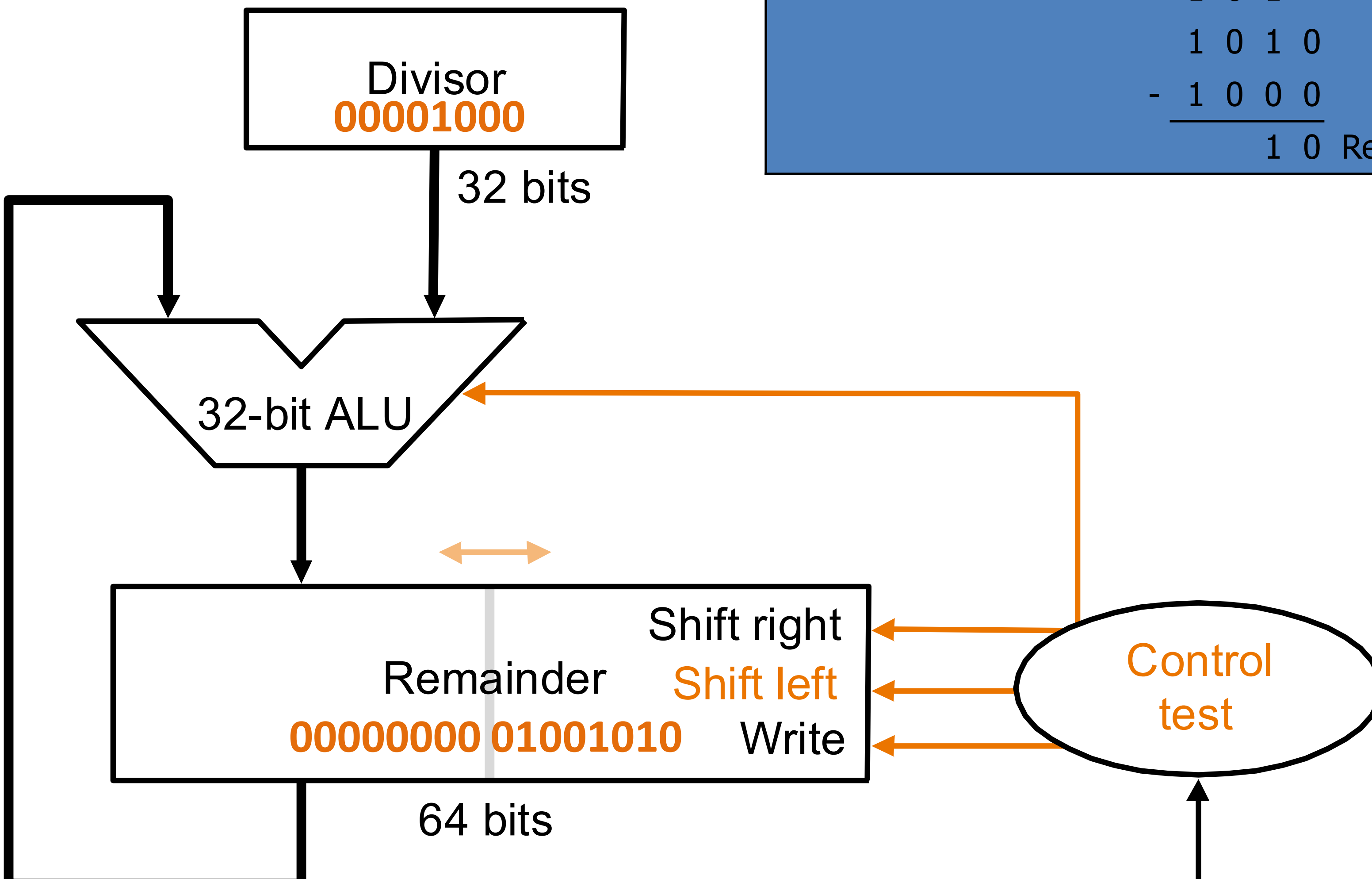
Non-Restoring Division



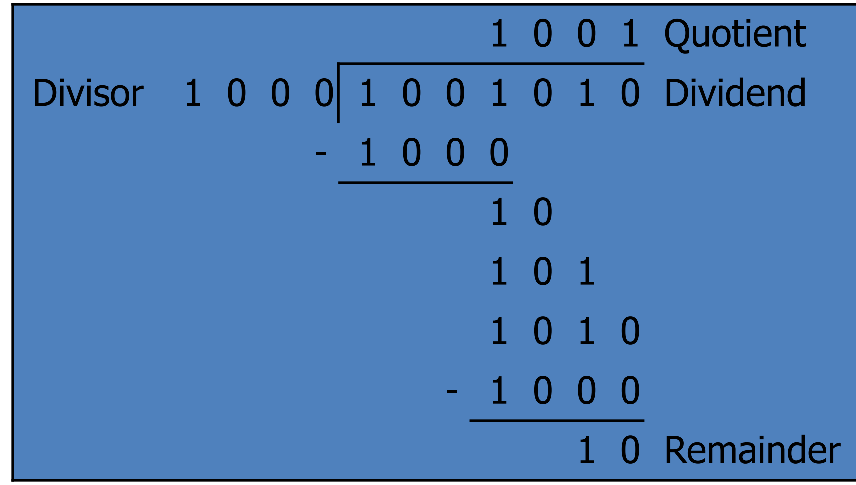


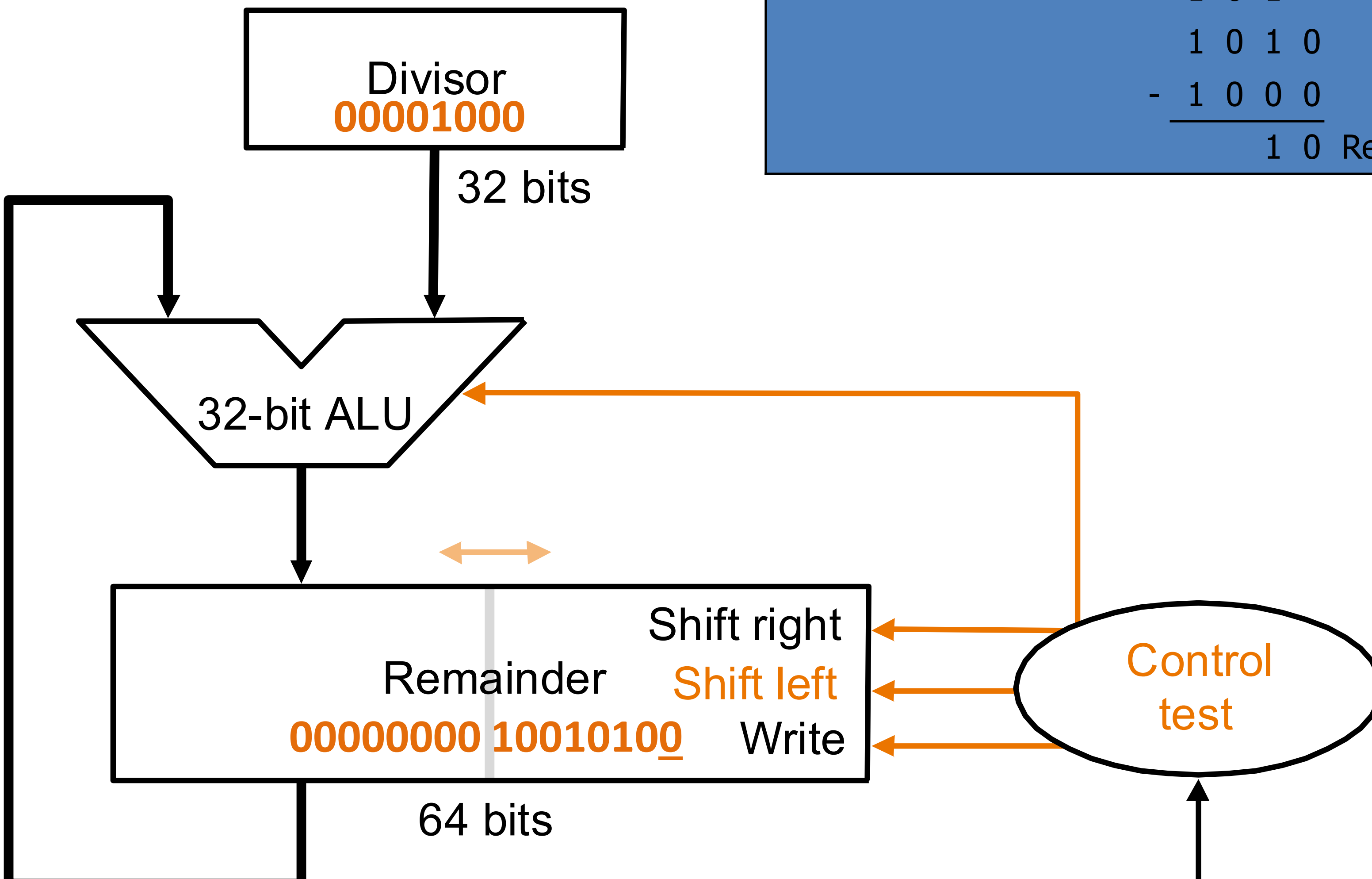
Non-Restoring Division

1 0 0 1 Quotient
Divisor 1 0 0 0 | 1 0 0 1 0 1 0 Dividend
- 1 0 0 0
1 0
1 0 1
1 0 1 0
- 1 0 0 0
1 0 Remainder

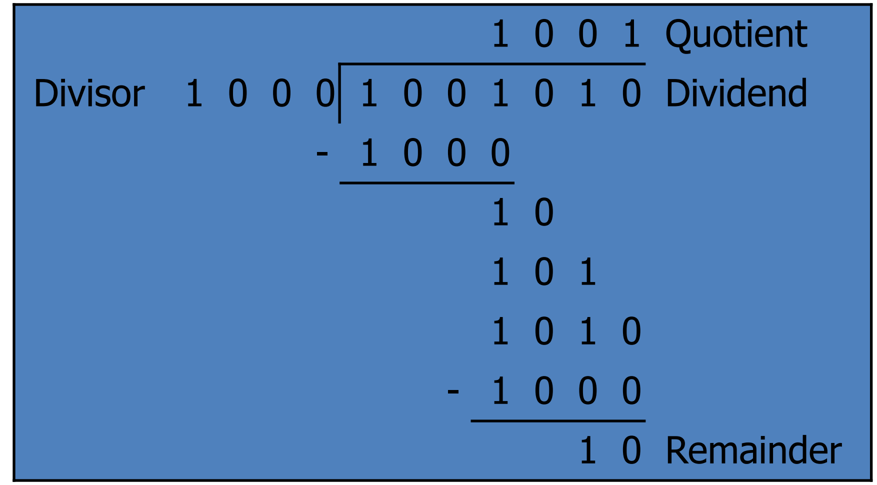


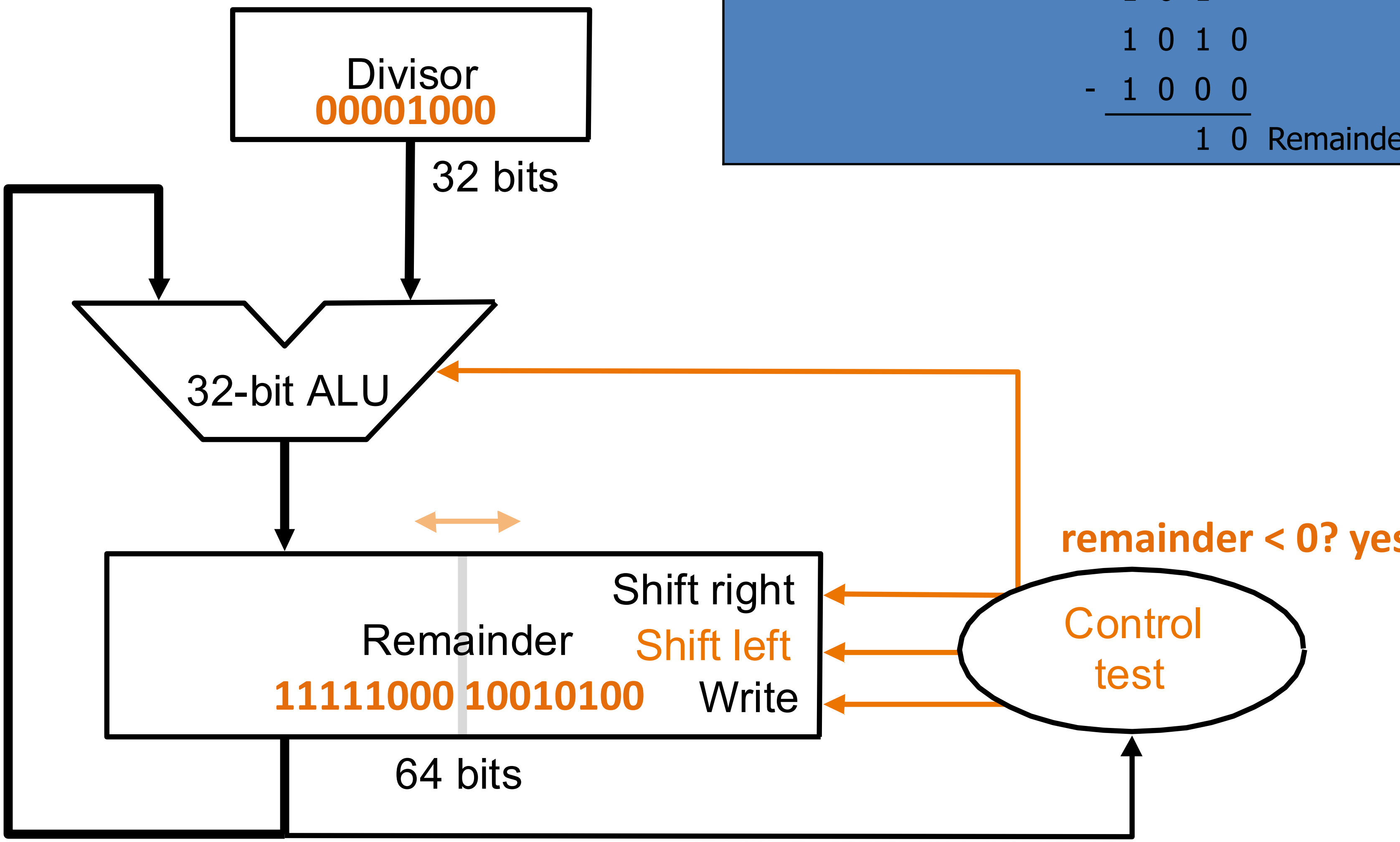
Non-Restoring Division



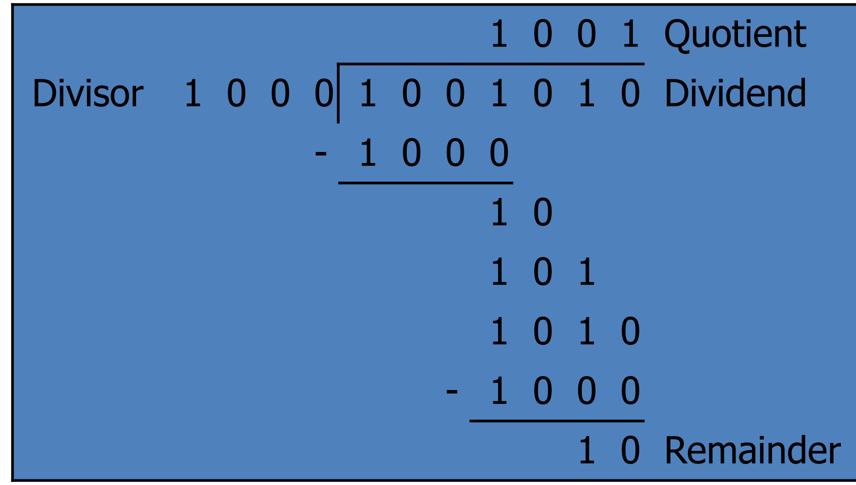


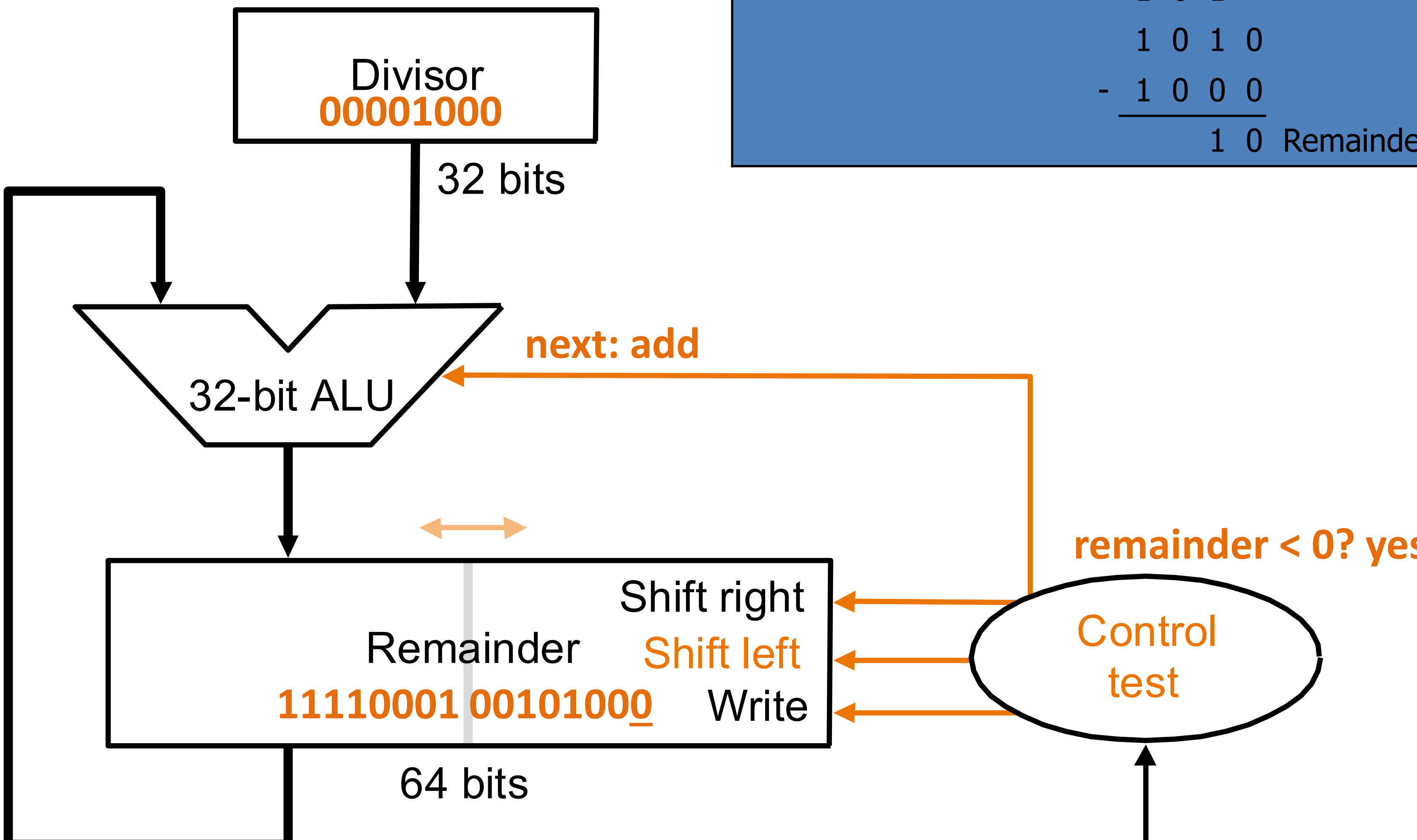
Non-Restoring Division



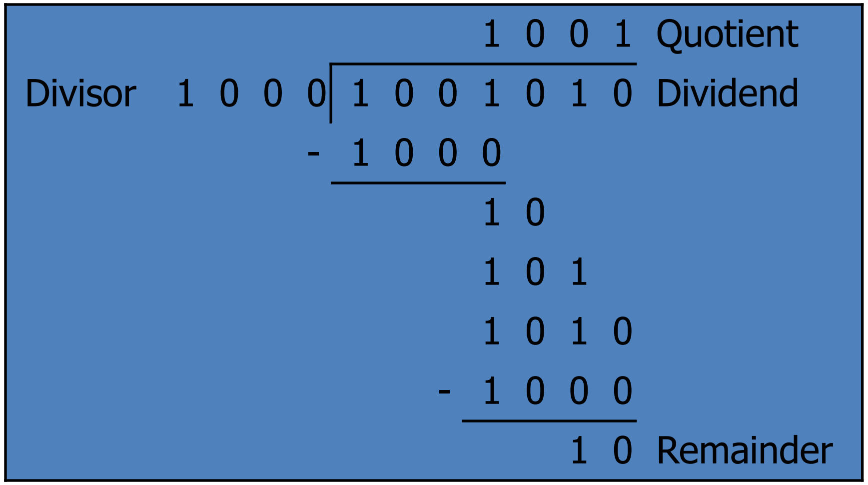


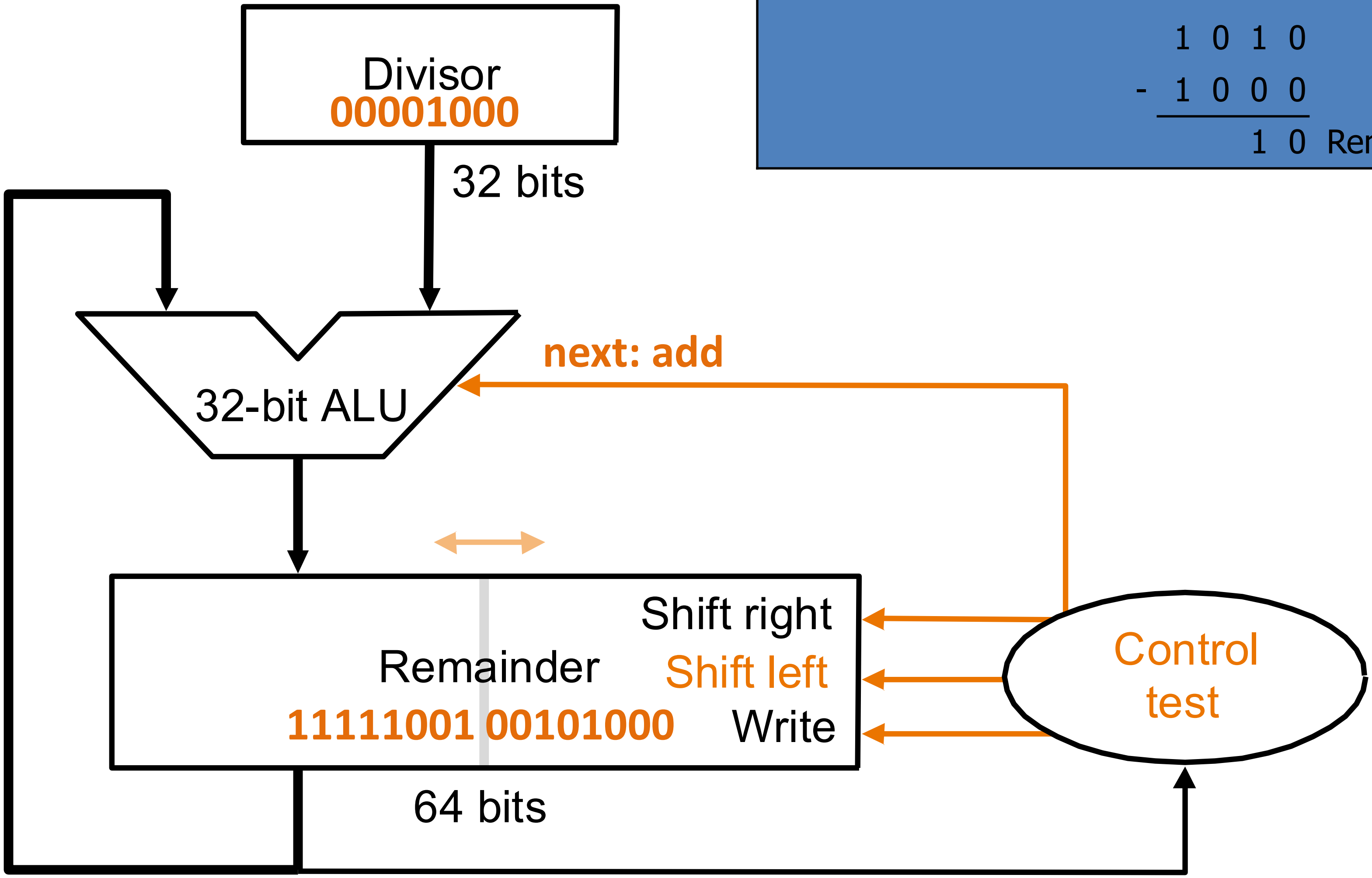
Non-Restoring Division





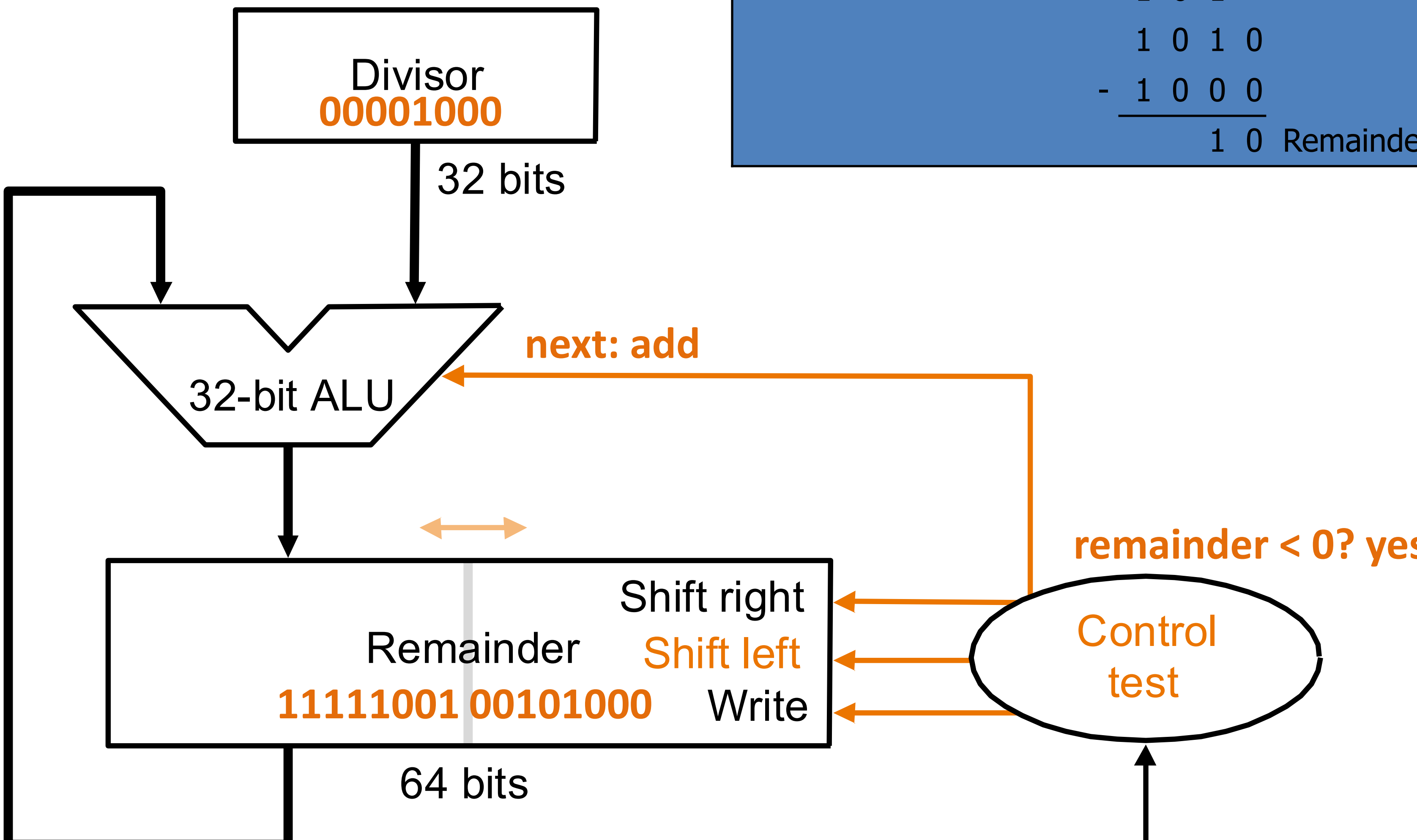
Non-Restoring Division



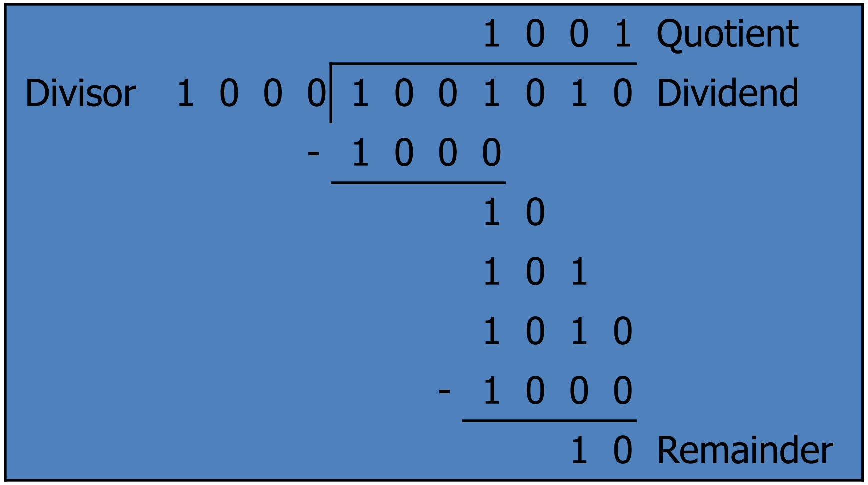


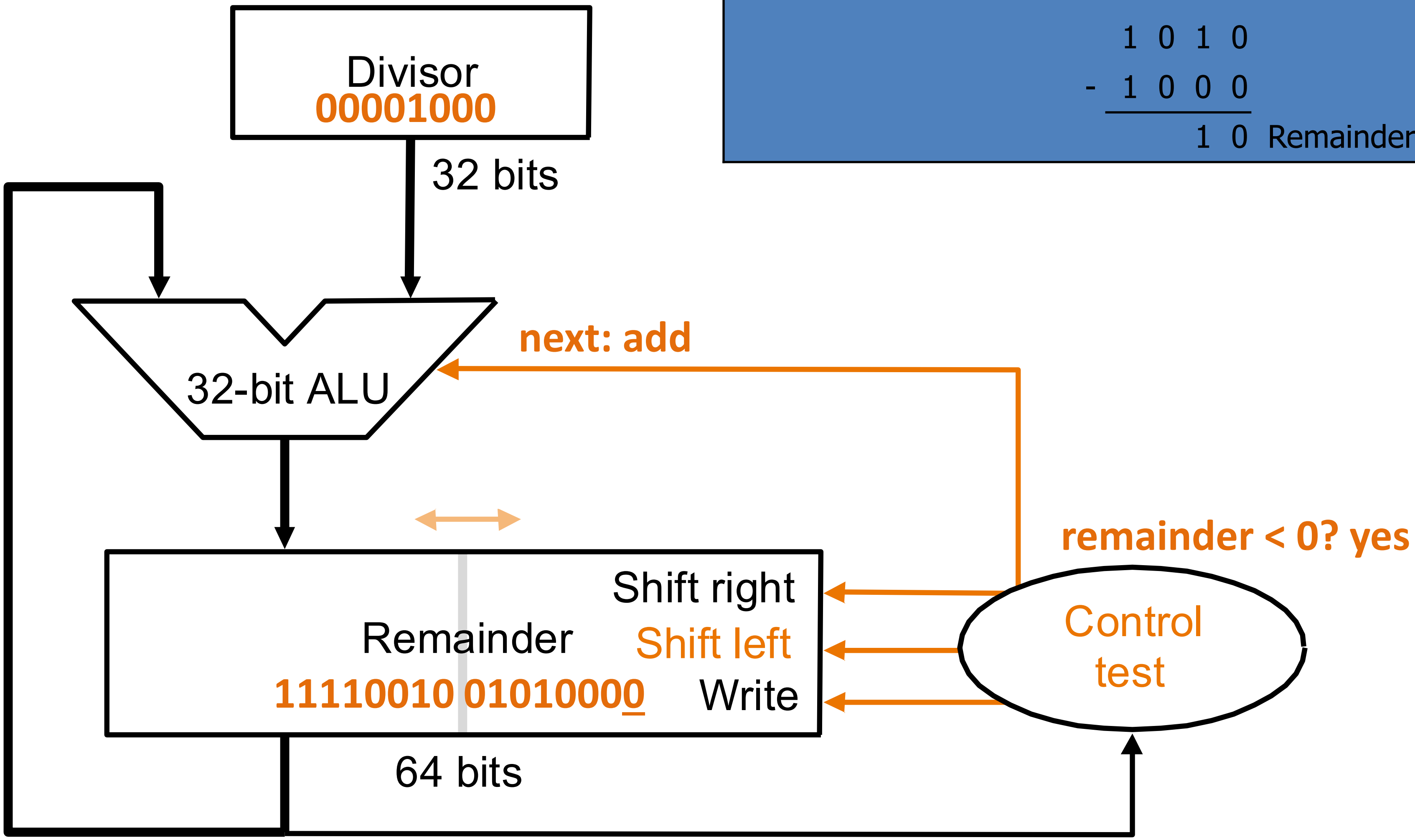
Non-Restoring Division

```
                      1 0 0 1   Quotient
Divisor  1 0 0 0 | 1 0 0 1 0 1 0   Dividend
               - 1 0 0 0
                     1 0
                     1 0 1
                     1 0 1 0
                   - 1 0 0 0
                       1 0   Remainder
```



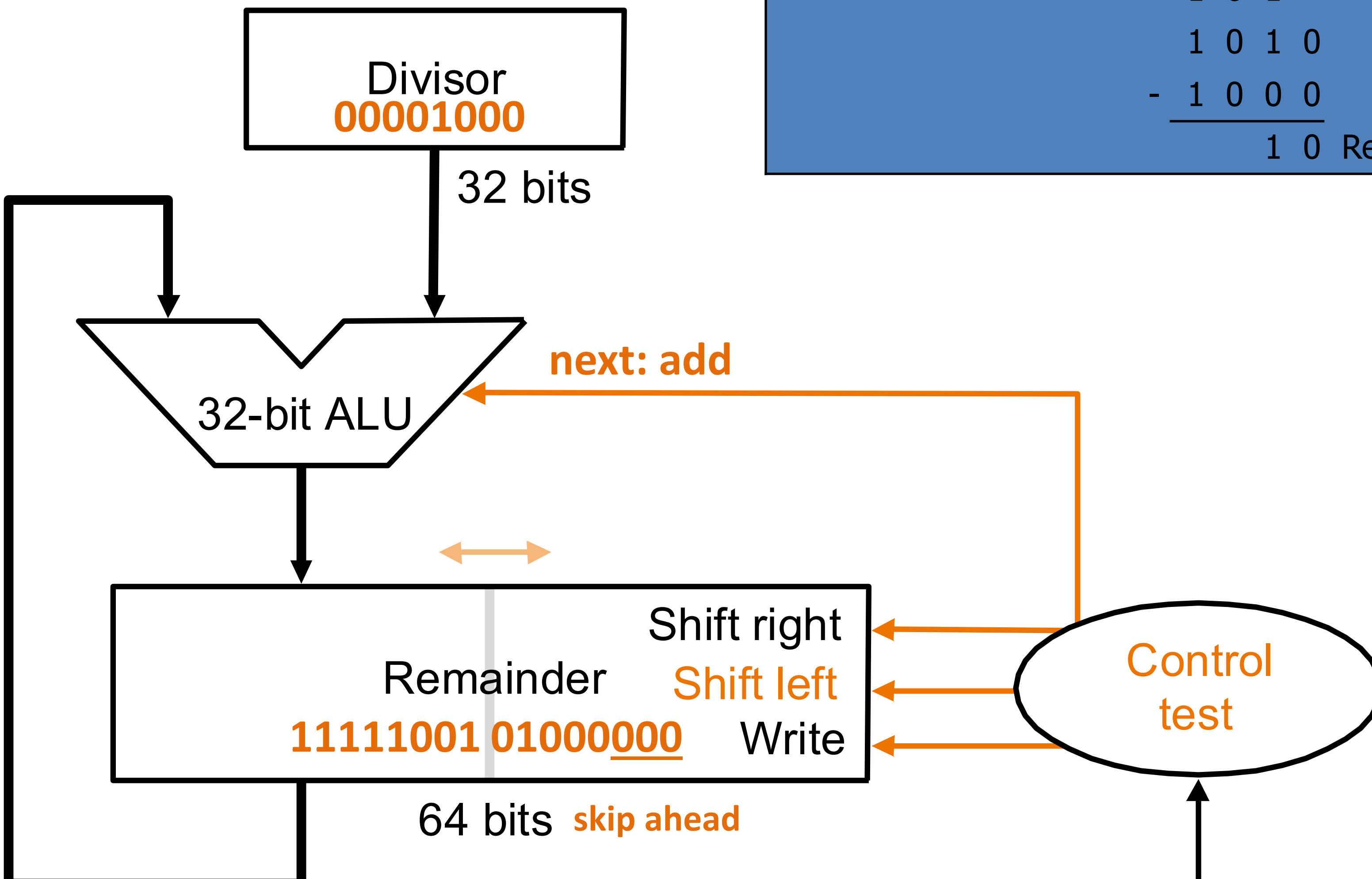
Non-Restoring Division



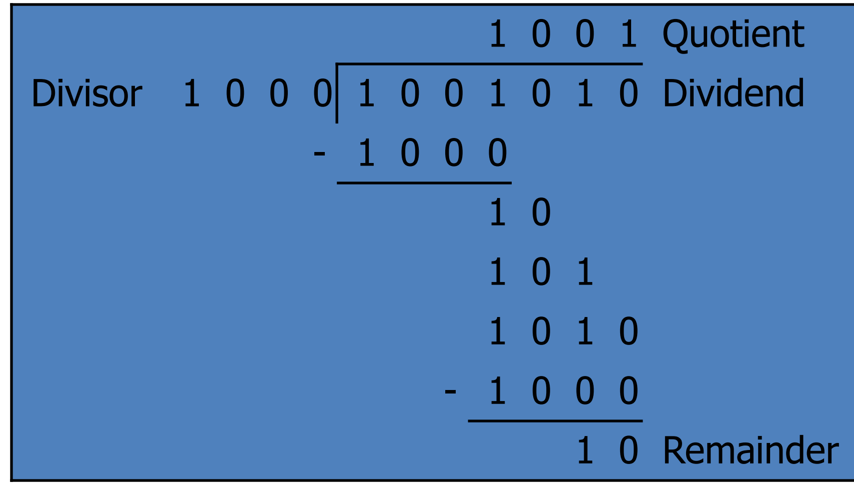


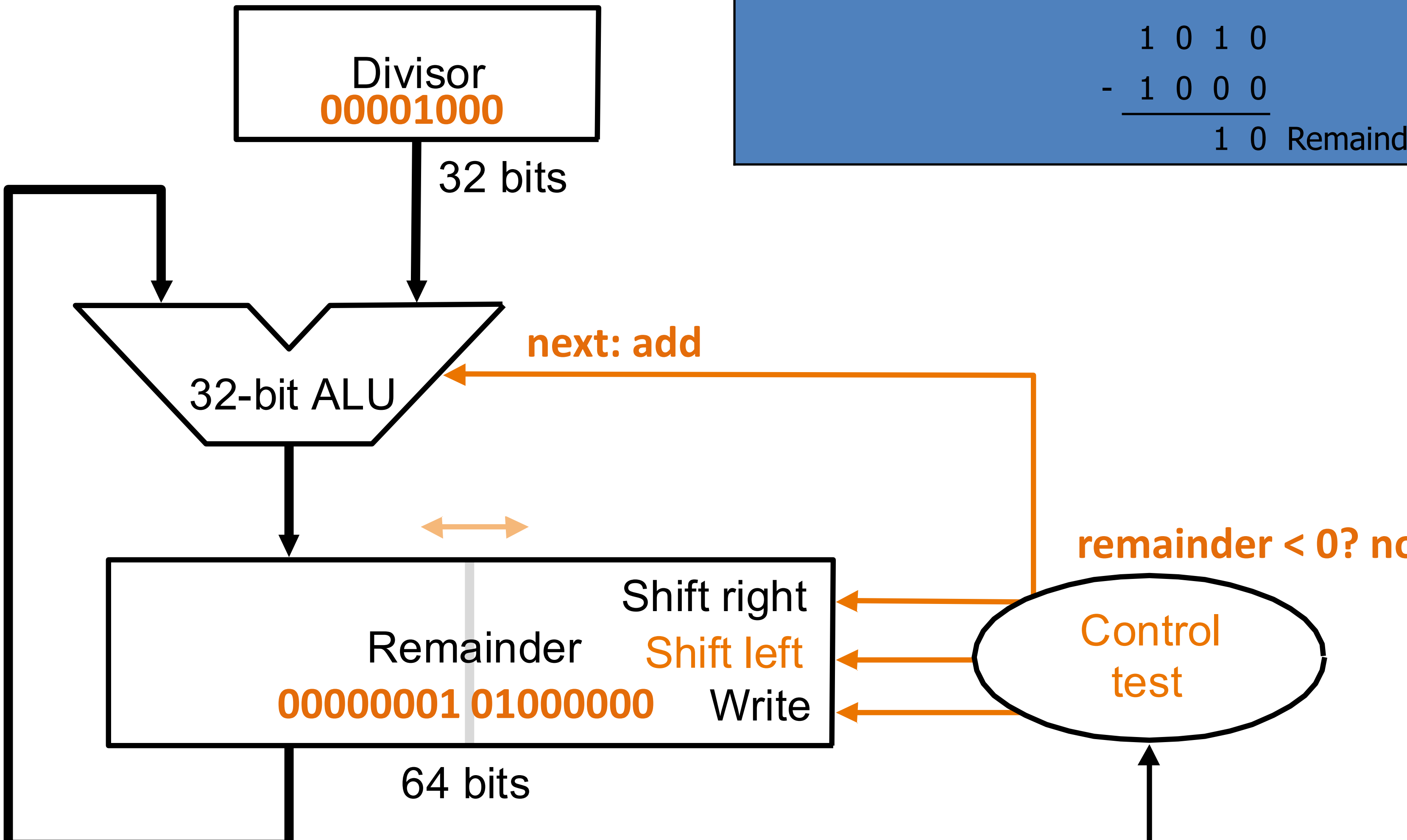
Non-Restoring Division

```
                      1 0 0 1        Quotient
Divisor  1 0 0 0 | 1 0 0 1 0 1 0    Dividend
                 - 1 0 0 0
                         1 0
                         1 0 1
                         1 0 1 0
                       - 1 0 0 0
                             1 0    Remainder
```

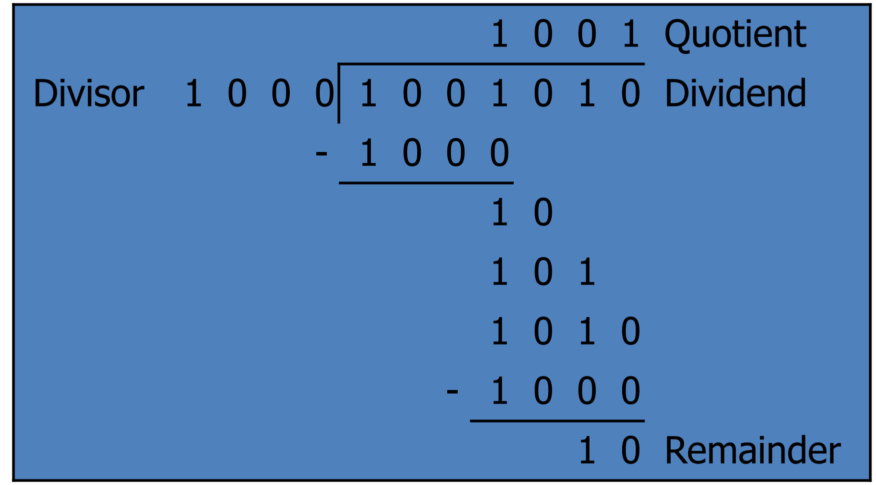


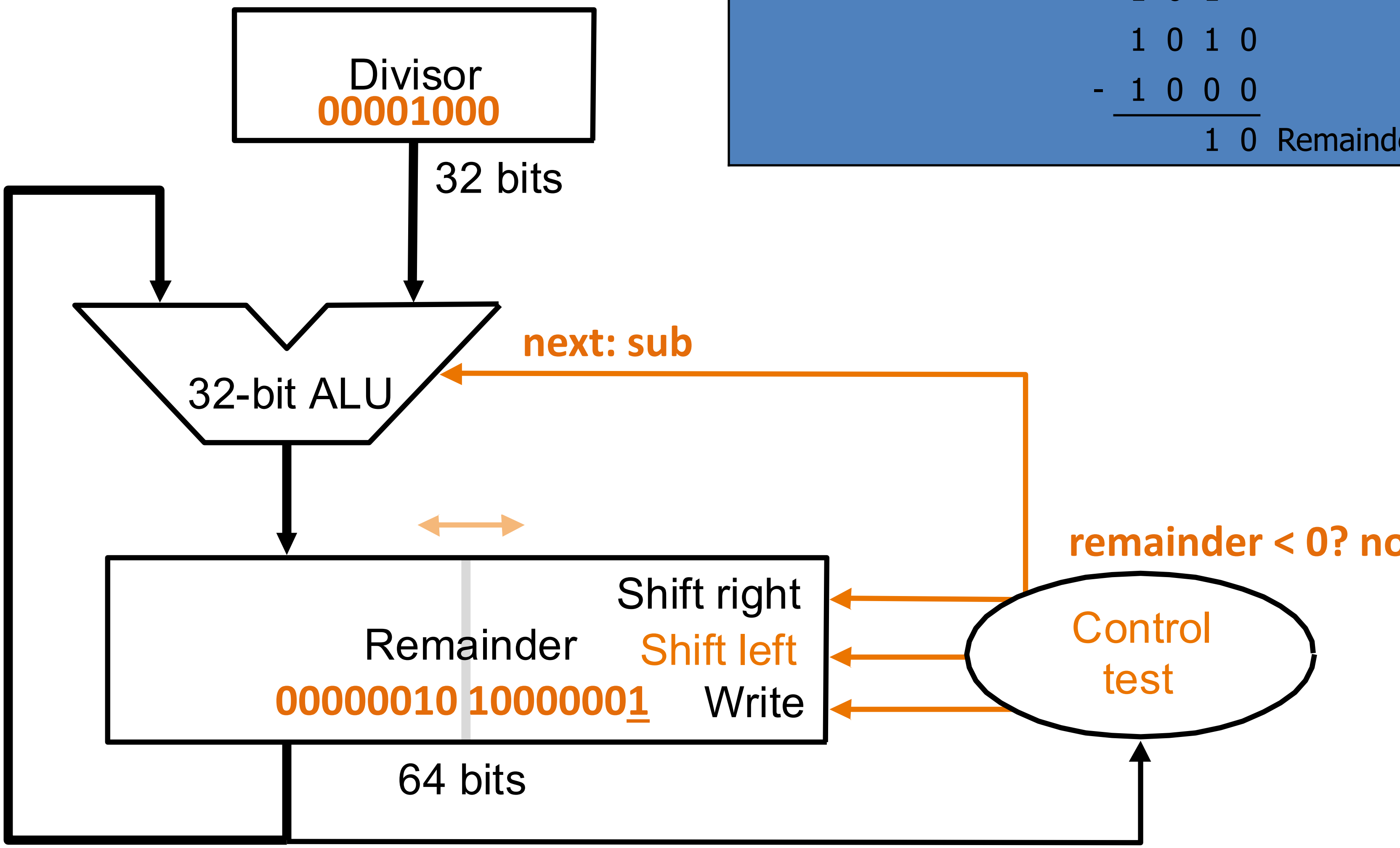
Non-Restoring Division



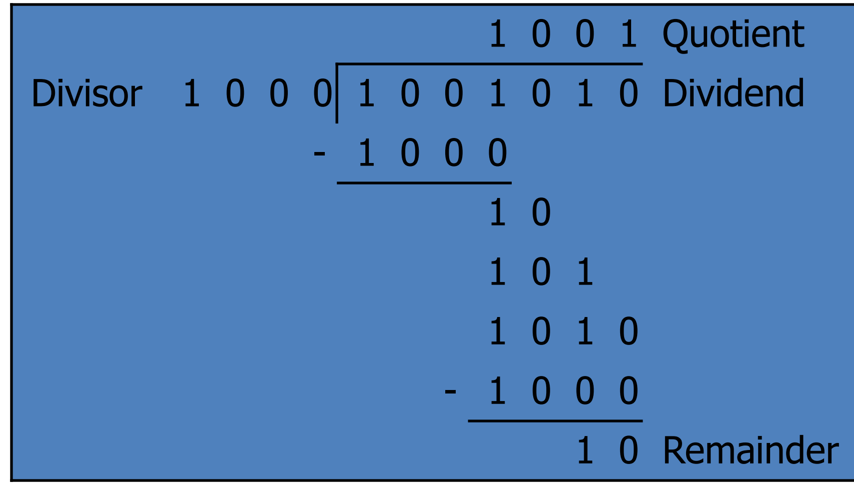


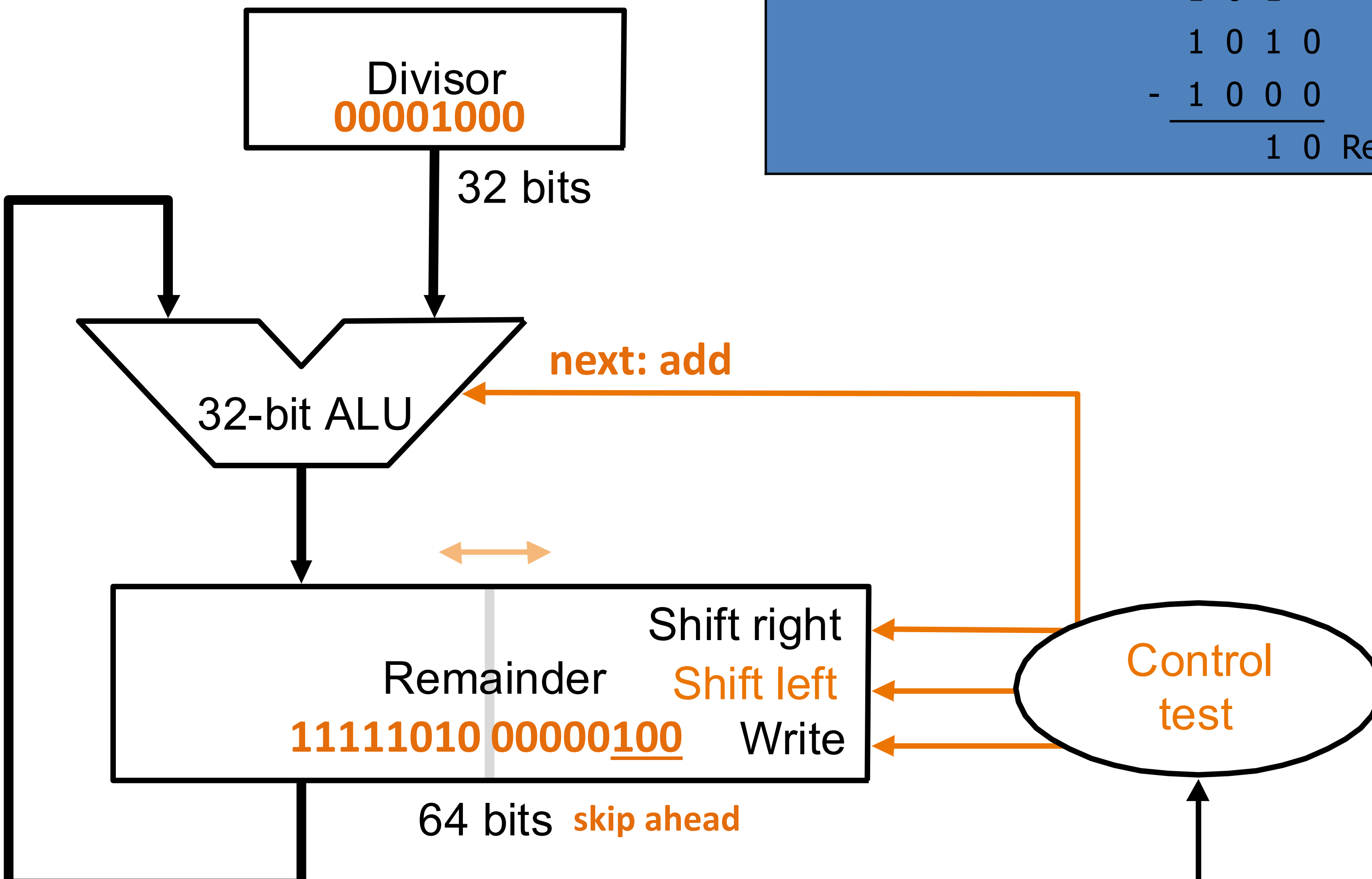
Non-Restoring Division





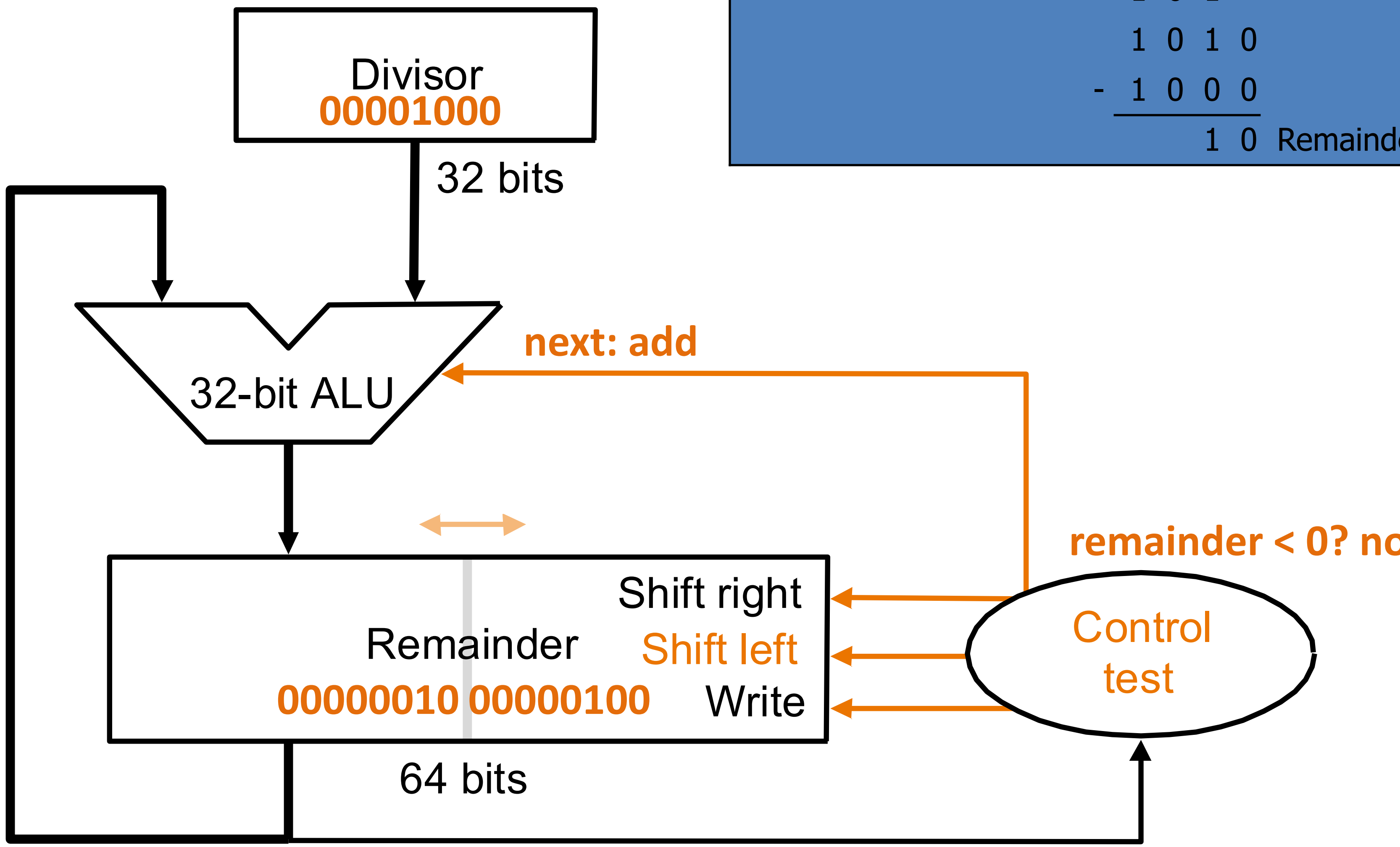
Non-Restoring Division



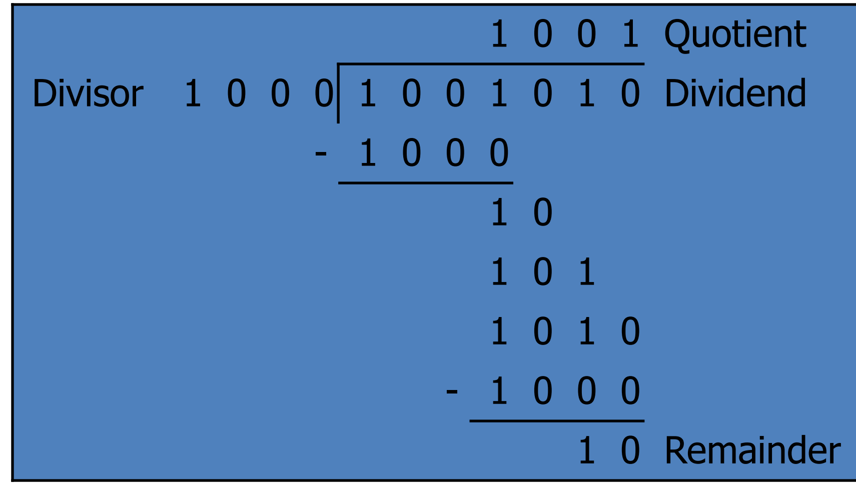


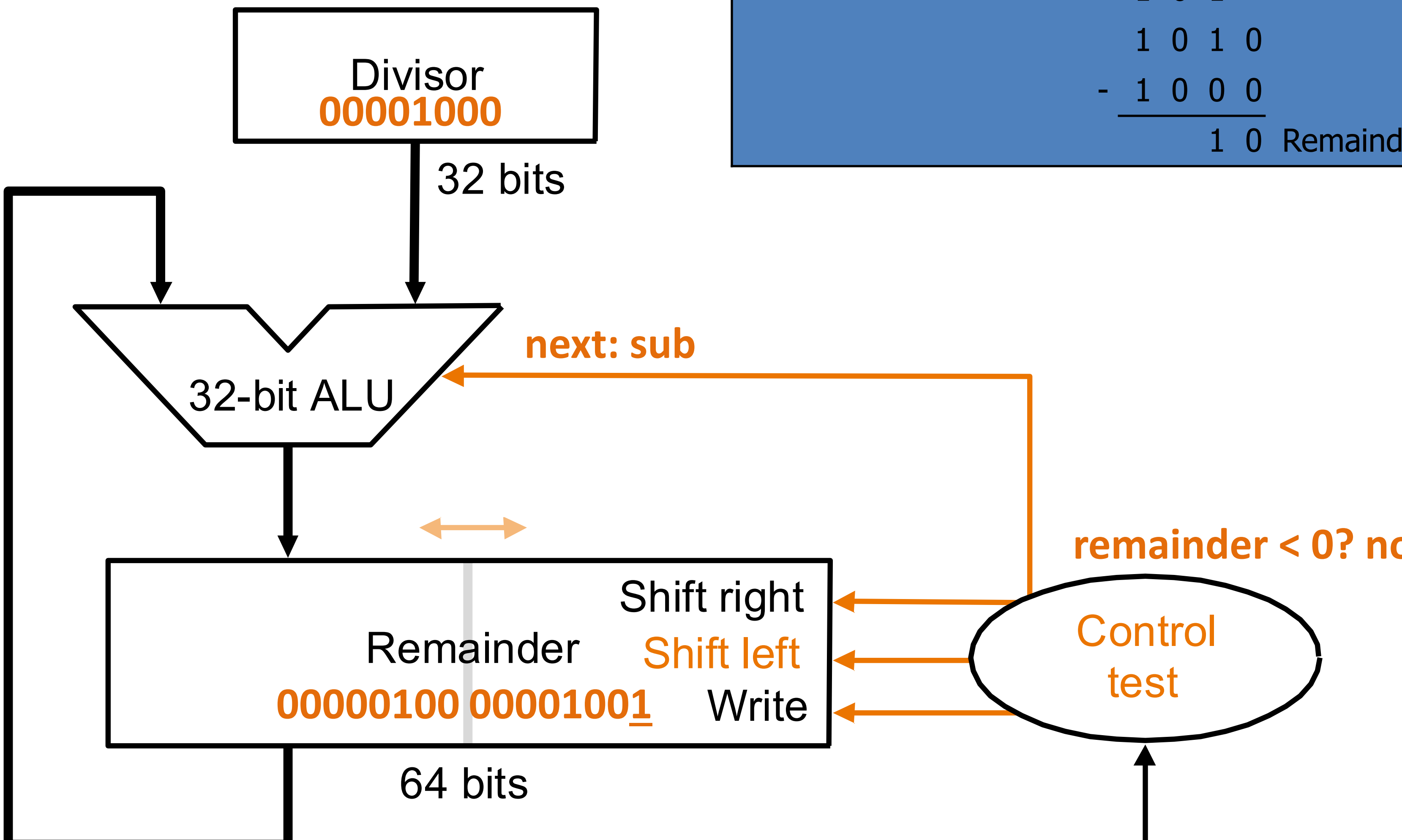
Non-Restoring Division

```
                    1 0 0 1     Quotient
Divisor   1 0 0 0 | 1 0 0 1 0 1 0   Dividend
                - 1 0 0 0
                  -------
                        1 0
                        1 0 1
                        1 0 1 0
                      - 1 0 0 0
                        -------
                            1 0   Remainder
```



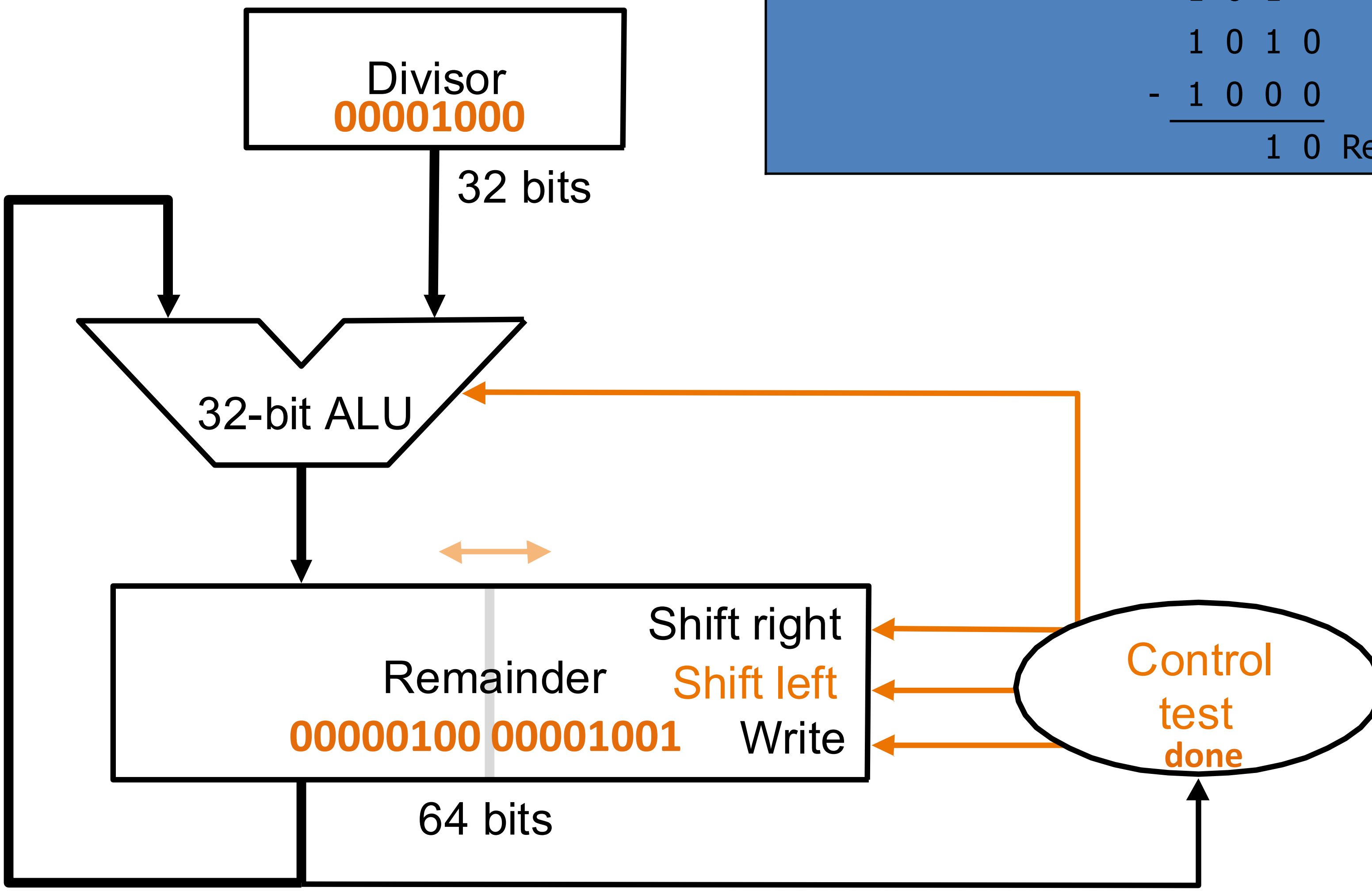
Non-Restoring Division



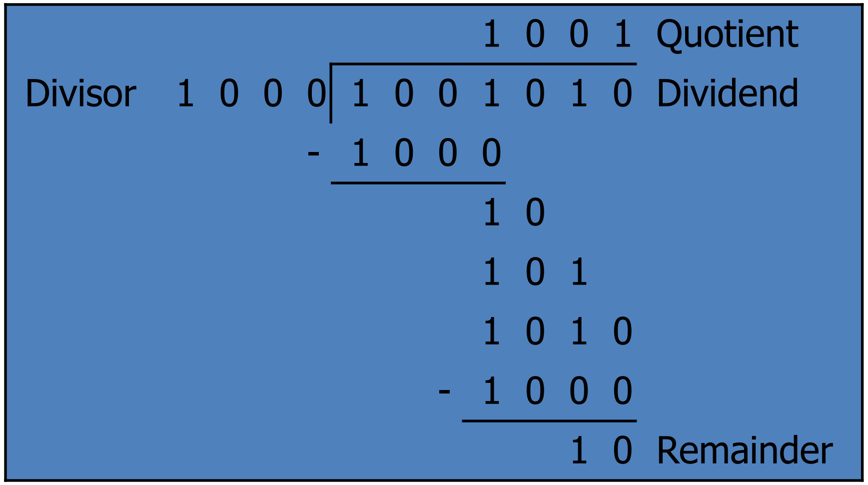


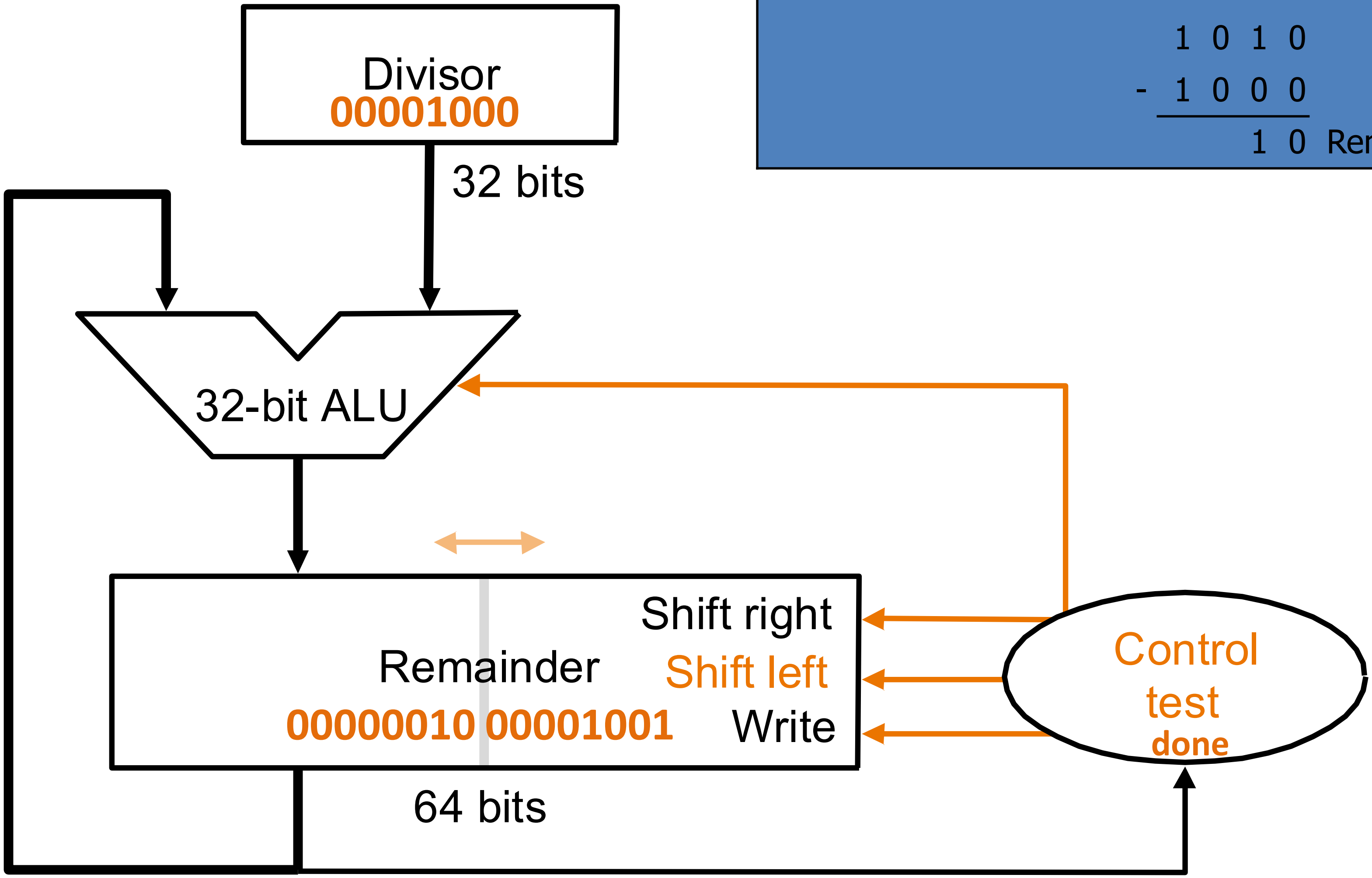
Non-Restoring Division

```
                  1 0 0 1   Quotient
Divisor  1 0 0 0 |1 0 0 1 0 1 0   Dividend
                - 1 0 0 0
                  ---------
                        1 0
                        1 0 1
                        1 0 1 0
                      - 1 0 0 0
                        -------
                            1 0   Remainder
```



Non-Restoring Division





Floating Point

- Why floating-point representation?
  - In many cases, we want to be able to represent a large **range of values** without needing much **precision**
- e.g., represent [3 … 1572864] in fixed point:

  [11 … 110000000000000000000]
- But really that's just:

  [1.1×2^{1} … 1.1×2^{20}]

Floating Point

- Want to represent larger range of numbers
  - Fixed point (unsigned integer): $[0 \ldots (2^n - 1)]$
  - Fixed point (signed integer): $[-2^{n-1} \ldots (2^{n-1} - 1)]$
- Like scientific notation:

  1.337×10^{-5}
- Floating point:

  1.00101×2^3

Floating Point

- Represent 9.25 (decimal) in floating point?

  1001.01

Floating Point

- Represent 9.25 (decimal) in floating point?
  1001.01×2^0

Floating Point

- Represent 9.25 (decimal) in floating point?

  1001.01×2^0

  100.101×2^1

  10.0101×2^2

  $\mathbf{1.00101 \times 2^3}$

Floating Point

- Represent 9.25 (decimal) in floating point?

  1001.01×2^0

  100.101×2^1

  10.0101×2^2

  $\mathbf{1.00101 \times 2^3}$

normalized
(implicit leading 1)

e.g., 1-bit sign, 4-bit exponent, 5-bit significand/mantissa/fraction:

value = $(-1)^S \times F \times 2^E$

$9.25 = 1.00101 \times 2^3$

S = sign	E = exp	F = significand
0	0011	00101

Floating Point

- Still use a fixed number of bits
  - Sign bit S, exponent E, significand F
  - Value: $(-1)^S \times F \times 2^E$
- IEEE 754 standard

S	E	F

	Size	Exponent	Significand	Range
Single precision	32b	8b	23b	$2 \times 10^{+/-38}$
Double precision	64b	11b	52b	$2 \times 10^{+/-308}$

Floating Point - Exponent

- Exponent specified in *biased* or *excess* notation
- Why?
  - To simplify sorting, we want:

    ↑unsigned exponent => ↑value

Floating Point - Biased Exponent

Exponent Value	**2’s Complement**	**E (with bias 127)**
-127	1000 0001	0000 0000
-126	1000 0010	0000 0001
…	…	…
+127	0111 1111	1111 1110

- Value: $(-1)^S \times F \times 2^{(E\text{-bias})}$
  - IEEE 754 single precision: bias = 127
  - IEEE 754 double precision: bias = 1023

Floating Point - Biased Exponent

Exponent Value	2's Complement	E (with bias 127)
-127	1000 0001	0000 0000
-126	1000 0010	0000 0001
…	…	…
+127	0111 1111	1111 1110

e.g., 1-bit sign, 4-bit exponent, 5-bit significand:

value = $(-1)^S \times F \times 2^E$

$9.25 = 1.00101 \times 2^3$

S = sign	E = exp	F = significand
0	0011	00101

Floating Point - Biased Exponent

Exponent Value	2's Complement	E (with bias 127)
-127	1000 0001	0000 0000
-126	1000 0010	0000 0001
…	…	…
+127	0111 1111	1111 1110

e.g., 1-bit sign, 4-bit exponent (with bias 7), 5-bit significand:

value = $(-1)^S \times F \times 2^{(E-7)}$

$9.25 = 1.00101 \times 2^3$

S = sign	E = exp	F = significand
0	1010	00101