Booth Multiplication

Example:

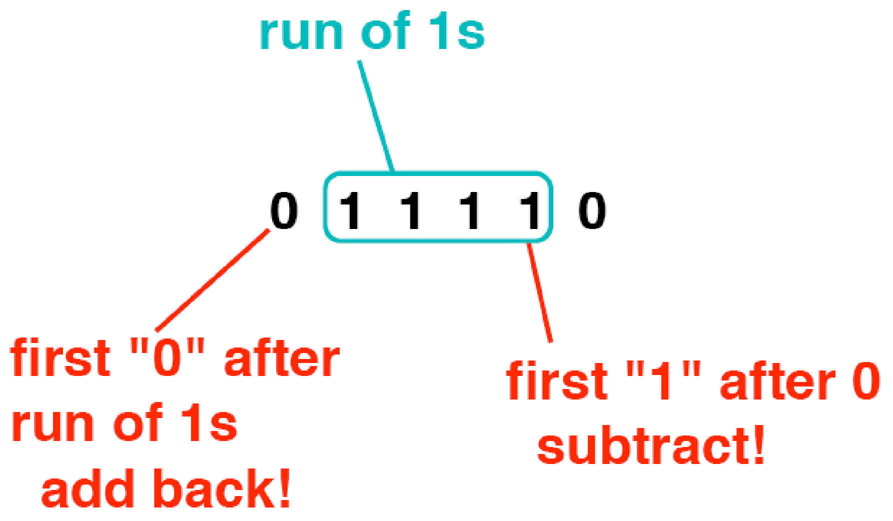
$$\begin{array}{lcccccccc}
\text{multiplicand} & & & & & 1 & 0 & 1 & 0 \\
\text{multiplier} & & & & \times & 0 & 1 & 1 & 0 \\
\cline{5-9}
 & & & & & 0 & 0 & 0 & 0 \\
 & & & & 1 & 0 & 1 & 0 & \\
 & & & 1 & 0 & 1 & 0 & & \\
 & + & 0 & 0 & 0 & 0 & & & \\
\cline{2-9}
 & & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
\cline{2-9}
\end{array}$$

Instead we could subtract early and add later...

$$6x = 2x + 4x = -2x + 8x$$

$$11110000 = 10000XXXX - 0001XXXX$$

Booth Multiplication



run of 1s

0 1 1 1 1 0

first "0" after run of 1s add back!

first "1" after 0 subtract!

Current	Right	Explanation
1	0	beginning of run of 1s
0	1	end of run of 1s
1	1	middle of run of 1s
0	0	middle of run of 0s

Originally for speed: shifts faster than adds

Booth Multiplication

Depending on current and previous bits, do one of the following:

- 00: middle of a run of 0s ⇒ no operation
- 01: end of a run of 1s ⇒ add multiplicand to left half of product
- 10: start of a run of 1s ⇒ subtract multiplicand from left half of product
- 11: middle of a run of 1s ⇒ no operation

As before, shift product register right by 1 bit per step.

Integer Division

```
                       0101      quotient
                     ________
divisor         0010 | 1011      dividend

                   0010
                 - 0010
                     ________
                       0011
                     0010
                   - 0010
                     ________
                       0001      remainder
```

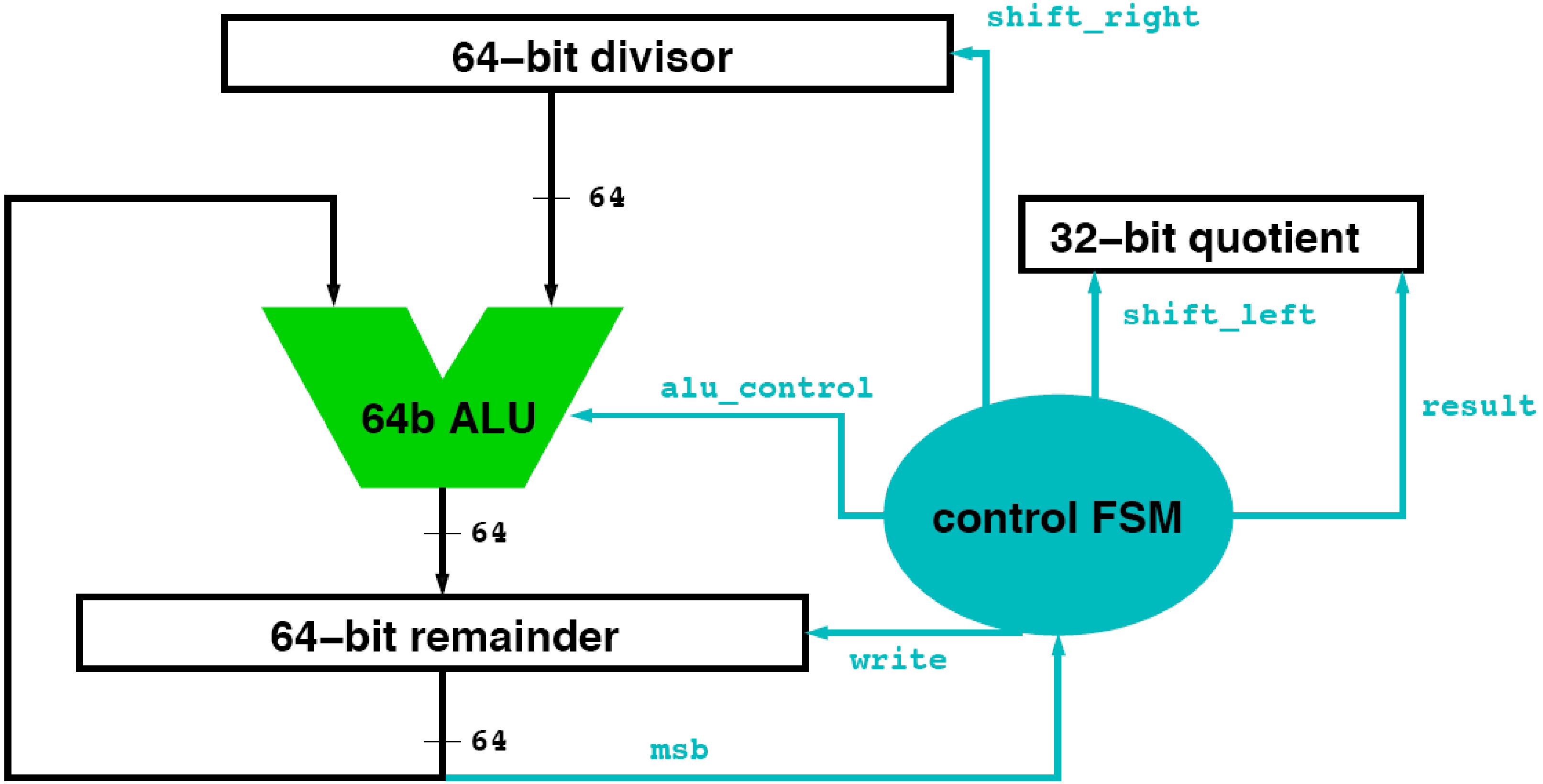
Red: **steps where subtracting would result in a negative number, i.e. quotient bit is zero.**

Integer Division

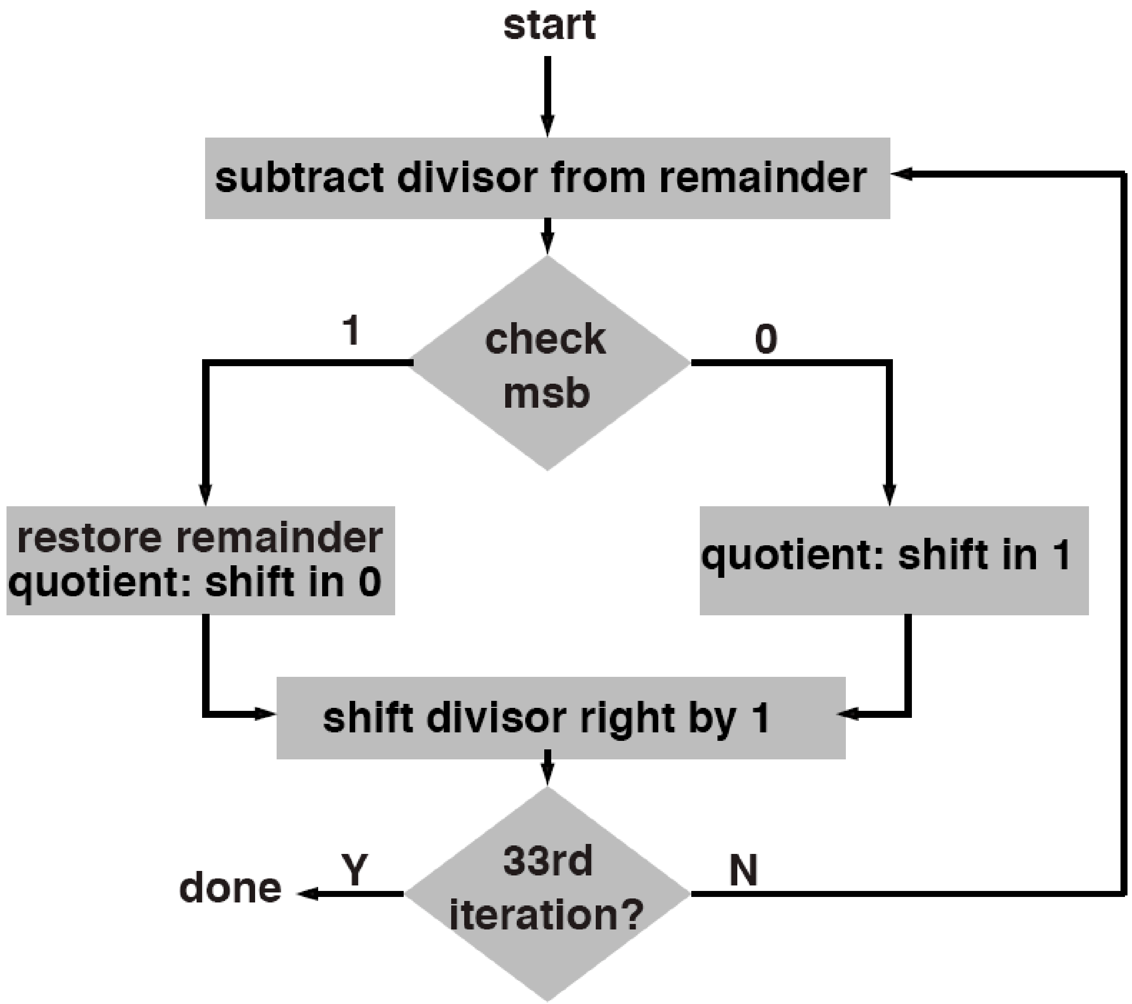
```
                        0101        quotient
divisor      0010  |    1011        dividend
                    00010000
                  - 00001000
                 -----------
                    00000011
                    00000100
                  - 00000010
                 -----------
                    00000001        remainder
```

Pad out the dividend and divisor to 8 bits.

Integer Division



Integer Division



Integer Division

Observations:

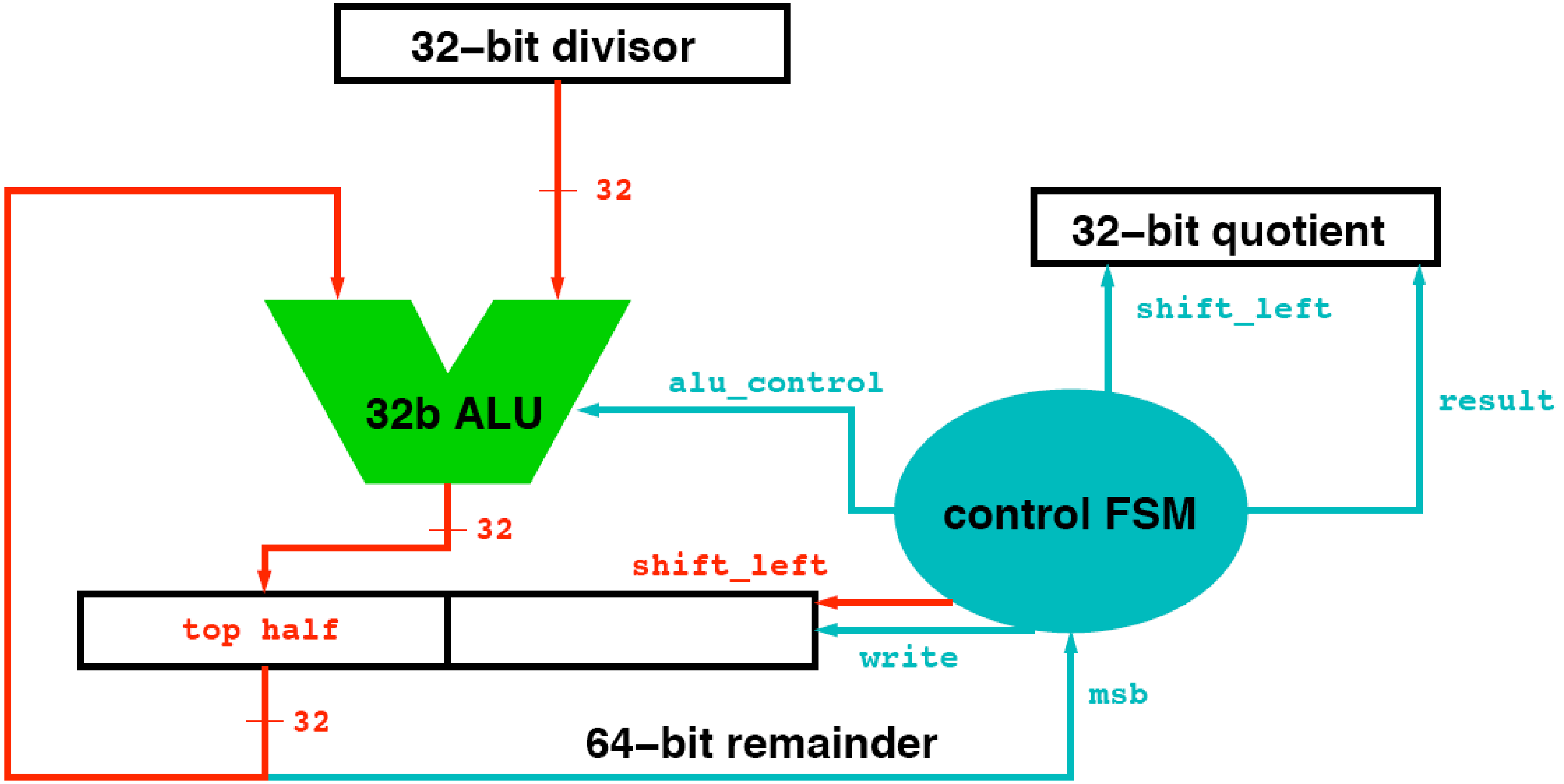
- Half the bits in the divisor are zero

  ⇒ 64-bit ALU wasted
- Instead of shifting divisor right, we can shift remainder left
- When does the first iteration shift in a 1 into the quotient?

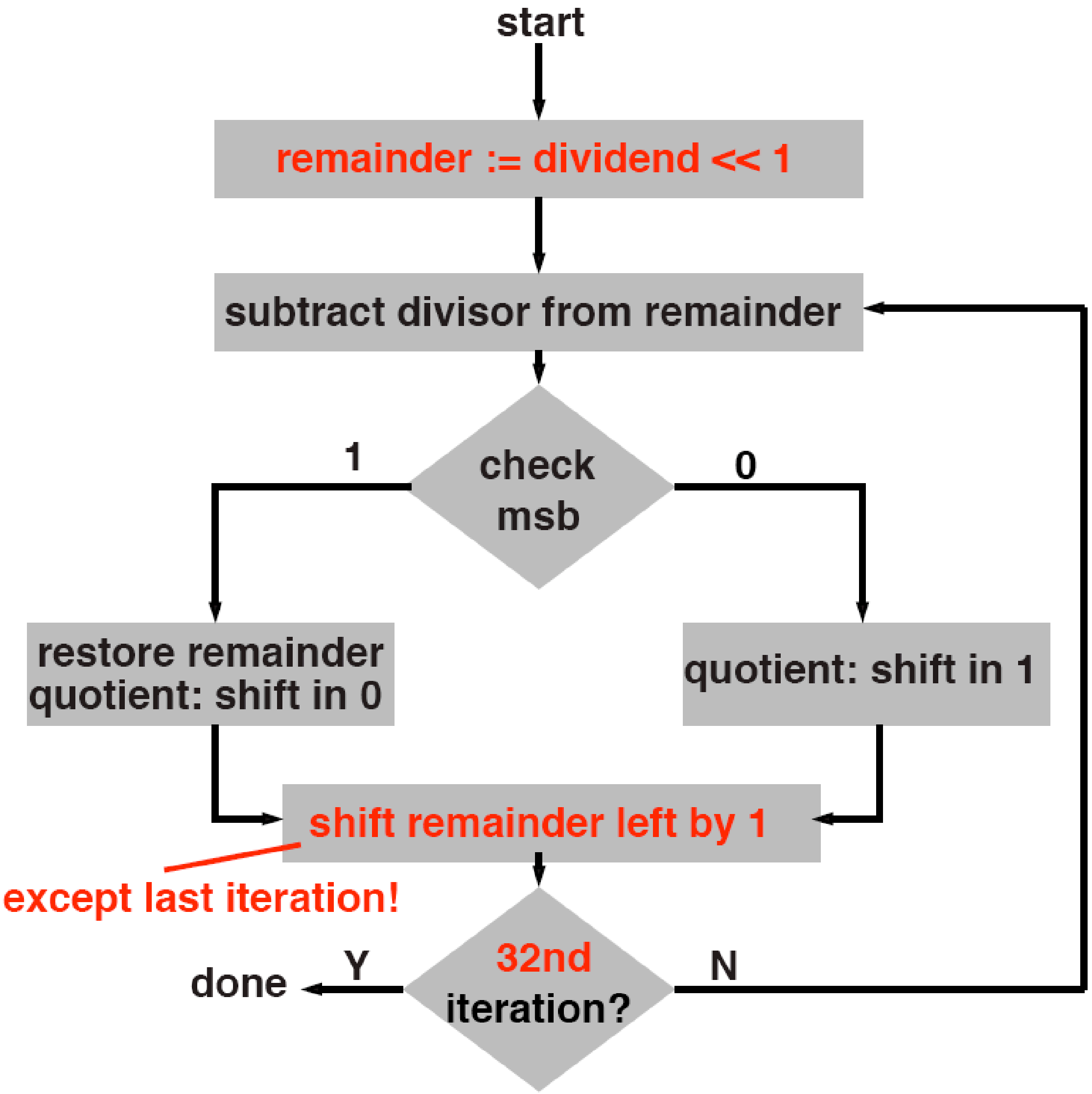
  ⇒ save 1 iteration

What is the initial value of the divisor?

Integer Division



New Control

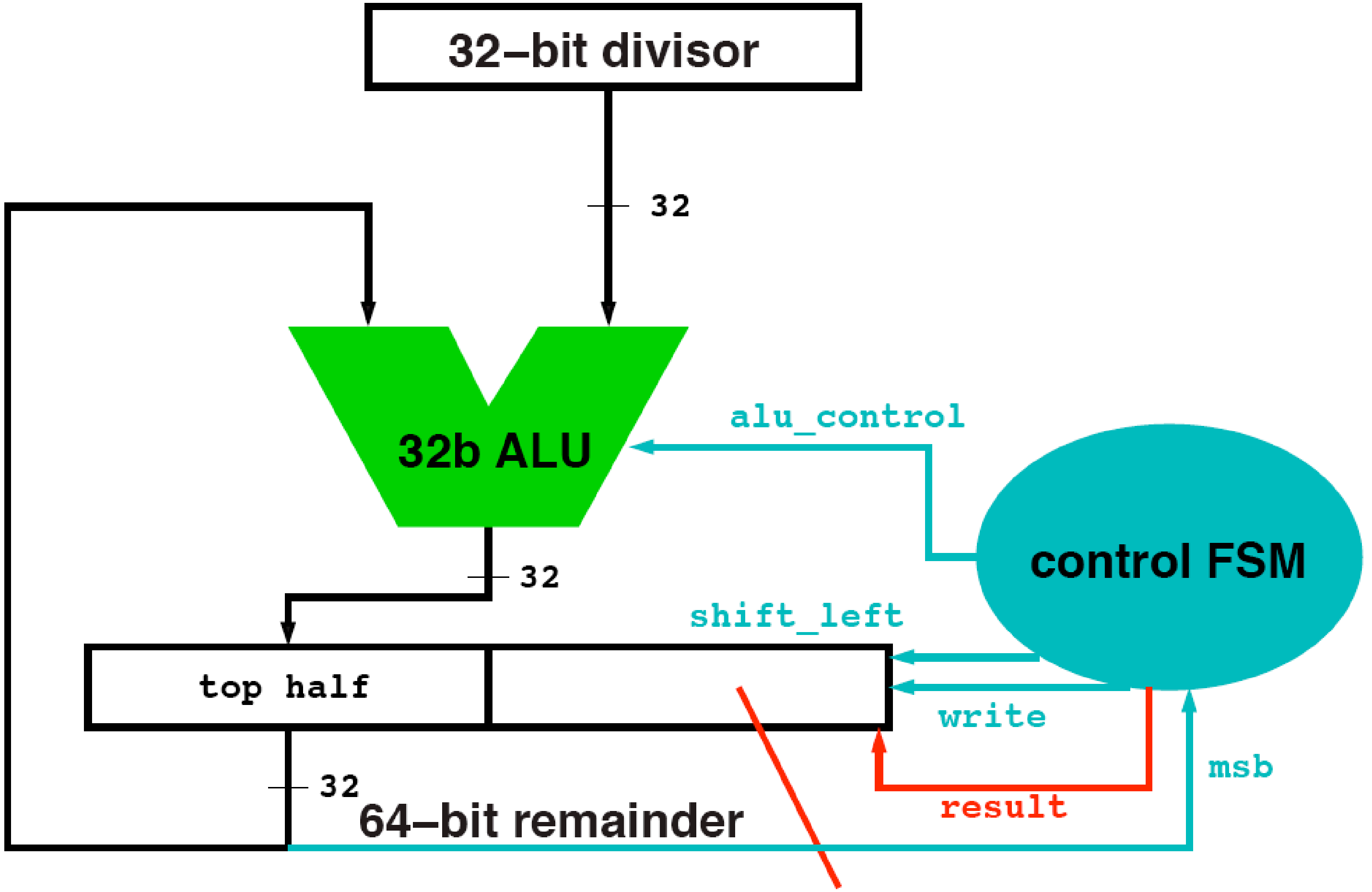


Remainder loses one bit per iteration;

Quotient gains one bit per iteration.

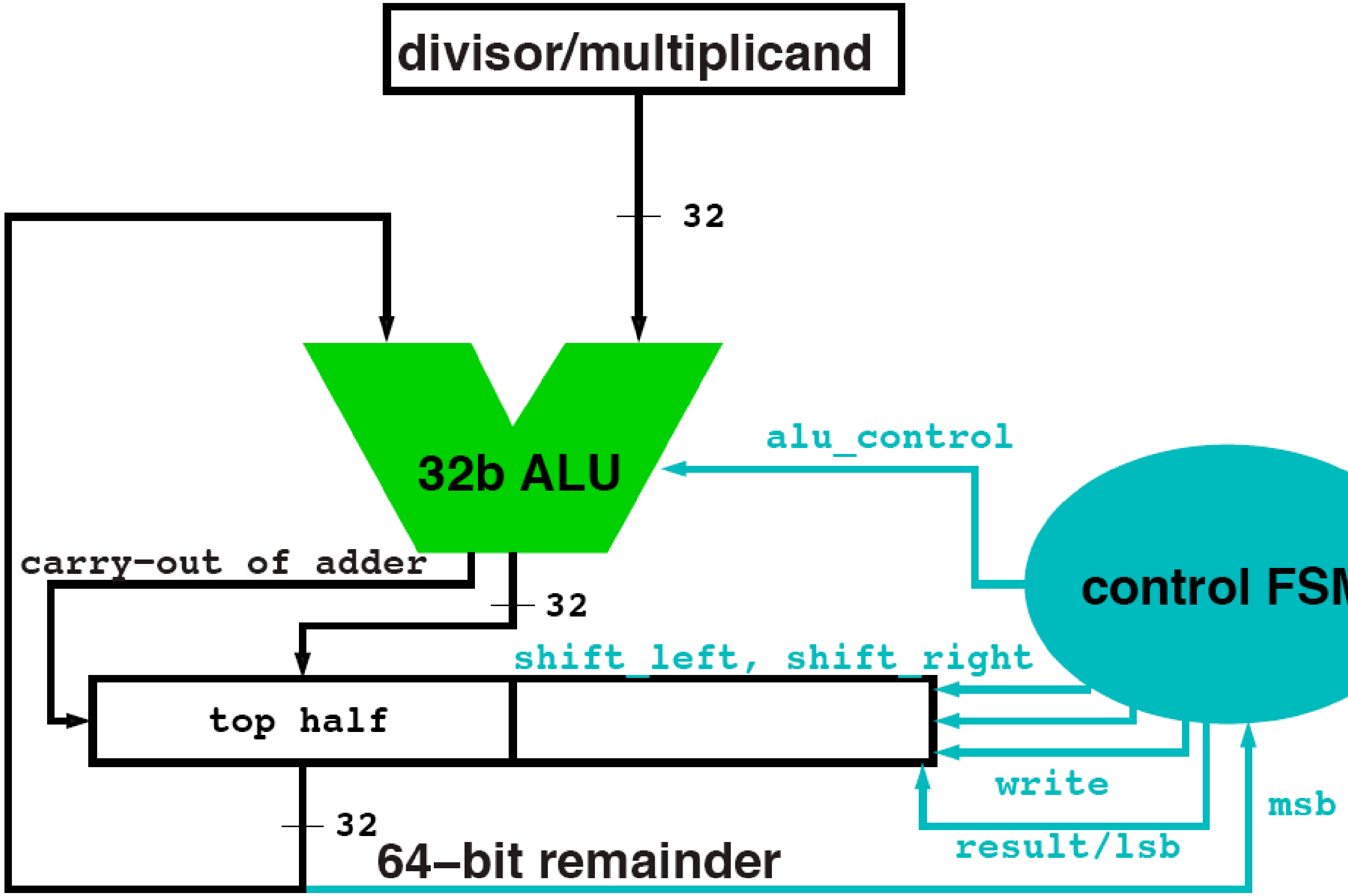
=> share registers!

Final Divider Hardware



quotient bits go here

Mult/Div



It's the same hardware...