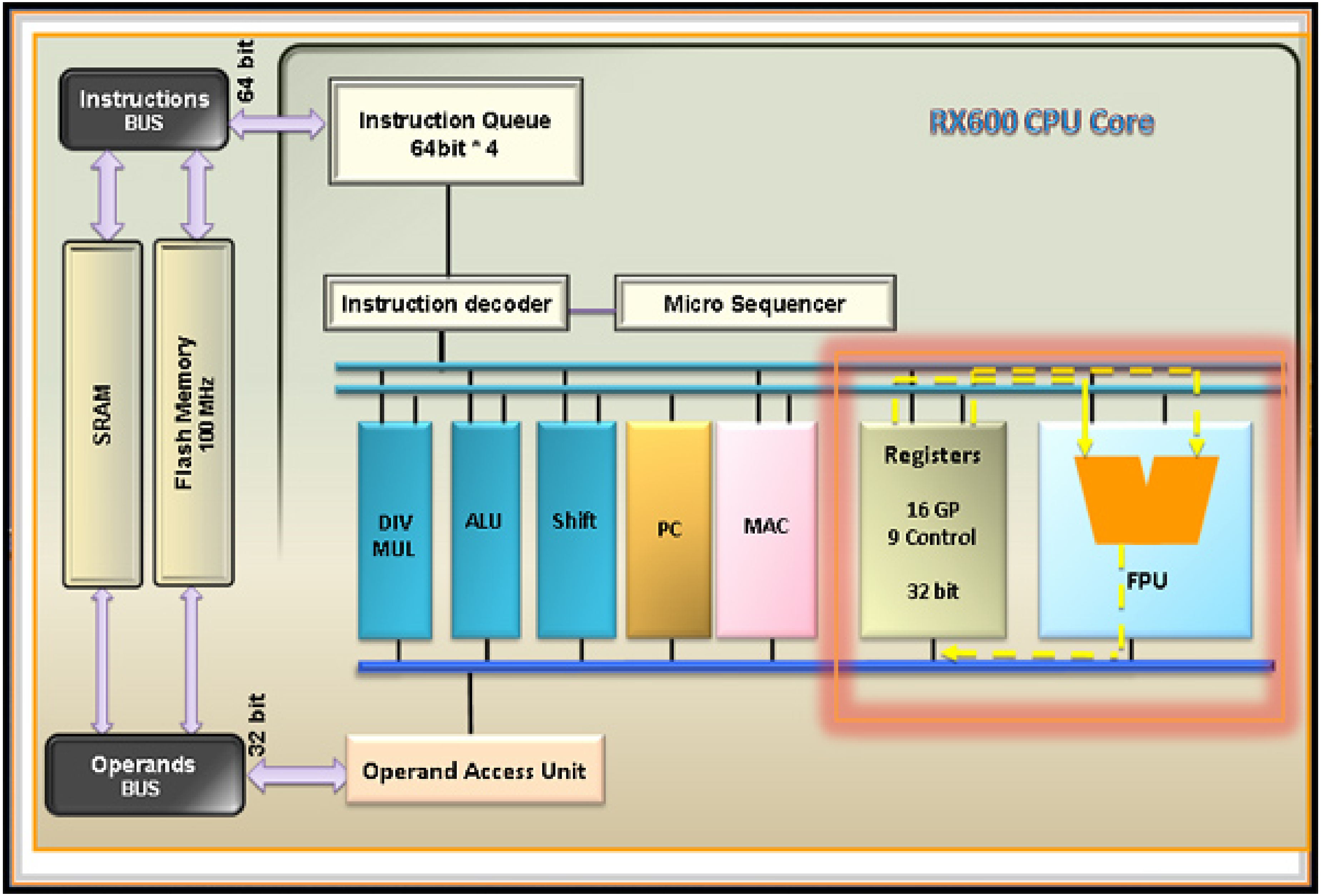
Computer Arithmetic (III)

- Integer Division
- Restoring Division
- Algorithm to Architecture


Instructions
BUS
64 bit
Instruction Queue
64bit * 4
RX600 CPU Core
SRAM
Flash Memory
100 MHz
Instruction decoder
Micro Sequencer
DIV
MUL
ALU
Shift
PC
MAC
Registers
16 GP
9 Control
32 bit
FPU
32 bit
Operands
BUS
Operand Access Unit

Division is basically repeated subtraction.
Consider the following base-10 example.
Dividing 454 by 23:

$$454 \div 23 = 19 \text{ R } 17$$

quotient

$$\begin{array}{r} 19 \\ 23 \overline{)454} \\ 230 \\ \hline 224 \\ 207 \\ \hline 17 \end{array}$$

divisor

dividend

remainder

$$\begin{array}{r} 454 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ -23 \\ \hline 17 \end{array}$$

19

Consider the following division: $13\overline{)\,321}$

How do we know what multiple of 13 to take off ?

$$
\begin{array}{r}
24 \quad \text{R } 9 \\
13\overline{)\,321} \\
\underline{26}\downarrow \\
61 \\
\underline{52} \\
9
\end{array}
$$

We make an estimate !

Digital hardware will require an algorithm to do this.

As a formula we can write:

Dividend = Quotient × Divisor + Remainder

Again, considering the division of 454 by 23:

Dividend = 454
Divisor = 23
Quotient = 19
Remainder = 17

$454 = (19 \times 23) + 17$

or in symbols: $D = Q \times V + R$

Restoring Division

$$\boxed{D = Q \times V + R}$$

$29 \div 6 = 4R5$

$29 = 4 \times 6 + 5$

As we are dealing with positive integers, $R \geq 0$

$$D \geq Q \times V$$

The problem is to find Q, which, without loss of generality, we can assume for the sake of this example to be a 3-digit number $\{q_2q_1q_0\}$ of base, B.

Hence $$Q = q_2B^2 + q_1B^1 + q_0B^0$$

The approach is to find the digits of Q starting with the most significant, q_2 since

$$q_2B^2 \leq q_2B^2 + q_1B^1 + q_0B^0$$

Remembering that $D \geq Q \times V$

then $D \geq (q_2B^2 + q_1B^1 + q_0B^0) \times V$

$$D - (q_2B^2 \times V) \geq (q_1B^1 + q_0B^0) \times V$$

We know that $(q_1B^1 + q_0B^0) \times V \geq 0$ because we are dealing with positive numbers

Therefore, the first process in division is to search for the largest number q_2, such that:

$$D - (q_2B^2 \times V) \geq 0$$

After we have found q_2, we subtract q_2B^2 from D to form a modified dividend and repeat the process to find q_1, and so on until we have found every digit of Q. Anything that is left over is the remainder.

For the example 454÷23, we know the quotient can be at most three digits for a three digit dividend;
We need to search for the largest digit of the quotient, q_2. Could q_2 = 9 or 8 or. . . ?
We can set out this search process in a table:

$q_2B^2 \times V$	$D - q_2B^2 \times V$	Is this q_2?
$9 \times 10^2 \times 23 = 20700$	-20246	No!
$8 \times 10^2 \times 23 = 18400$	-17946	No!
...	...	...
$1 \times 10^2 \times 23 = 2300$	-1846	No!
$0 \times 10^2 \times 23 = 0$	454	Yes!

So q_2 is zero. At this stage we need to modify the dividend by subtracting $q_2 \times 10^2 \times 23$ from it (which in this case leaves the dividend unchanged).

Having found q_2, we now need to find q_1 in the same manner, starting from $q_1 = 9$. . .

$q_1 B^1 \times V$	$D - q_1 B^1 \times V$	Is this q_1?
$9 \times 10^1 \times 23 = 2070$	-1616	No!
$8 \times 10^1 \times 23 = 1840$	-1386	No!
...	...	...
$2 \times 10^1 \times 23 = 460$	-6	No!
$1 \times 10^1 \times 23 = 230$	224	Yes!

So $q_1 = 1$. We modify the dividend: D = 454 − 230 = 224 and repeat the whole process for q_0. Here $9 \times 10^0 \times 23 = 207$ hence $q_0 = 9$. The remainder (which is formed in exactly the same way as modifying the dividend) is 224 - 207 = 17. Therefore, our quotient is {019} = 19, remainder 17.

$q_2 \quad q_1 \quad q_0$

$$\begin{array}{rr} & 19 \\ \cline{2-2} 23 \big) & 454 \\ & 230 \\ \cline{2-2} & 224 \\ & 207 \\ \cline{2-2} & 17 \end{array}$$

For binary numbers, q_n is either 1 or 0, so we only have to examine the case where $q_n = 1$.

Also, multiplying a number by B^n is equivalent to shifting it left by n digits. So in binary, the mechanical division rules become a series of left-shifts of the divisor and comparisons with the current dividend.

Consider the example of 454 ÷ 23 (0111000110 ÷ 010111).

n	Dividend	Divisor $\times 2^n$	Quotient
8	0111000110	01011100000000	0
7	0111000110	0101110000000	0
6	0111000110	010111000000	0
5	0111000110	01011100000	0
4	0111000110	0101110000	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

After each trial, the dividend is **restored** to its previous positive value.

The quotient is read from the right-hand column: 000010011 = 19

The remainder is what is left in the dividend column: 010001=17

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	0111000110	0101110000000	0
6	0111000110	010111000000	0
5	0111000110	01011100000	0
4	0111000110	0101110000	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	0111000110	010111000000	0
5	0111000110	01011100000	0
4	0111000110	0101110000	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	0111000110	01011100000	0
4	0111000110	0101110000	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	454 → 0111000110	01011100000 ← 736	0
4	0111000110	0101110000	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	454 → 0111000110	01011100000 ← 736	0
4	454 → 0111000110	0101110000 ← 368	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

Process could begin at step 4 as both numbers have their MSB aligned.

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	454 → 0111000110	01011100000 ← 736	0
4	454 - 368 0111000110	0101110000 ← 368	1
3	0111000110 - 0101110000 = 01010110	010111000	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	454 → 0111000110	01011100000 ← 736	0
4	454 → 0111000110	0101110000 ← 368	1
3	0111000110 - 0101110000 = 01010110 ← 86	010111000 ← 184	0
2	01010110	01011100	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	454 → 0111000110	01011100000 ← 736	0
4	454 → 0111000110	0101110000 ← 368	1
3	0111000110 - 0101110000 = 01010110 ← 86	010111000 ← 184	0
2	86 → 01010110	01011100 ← 92	0
1	01010110	0101110	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

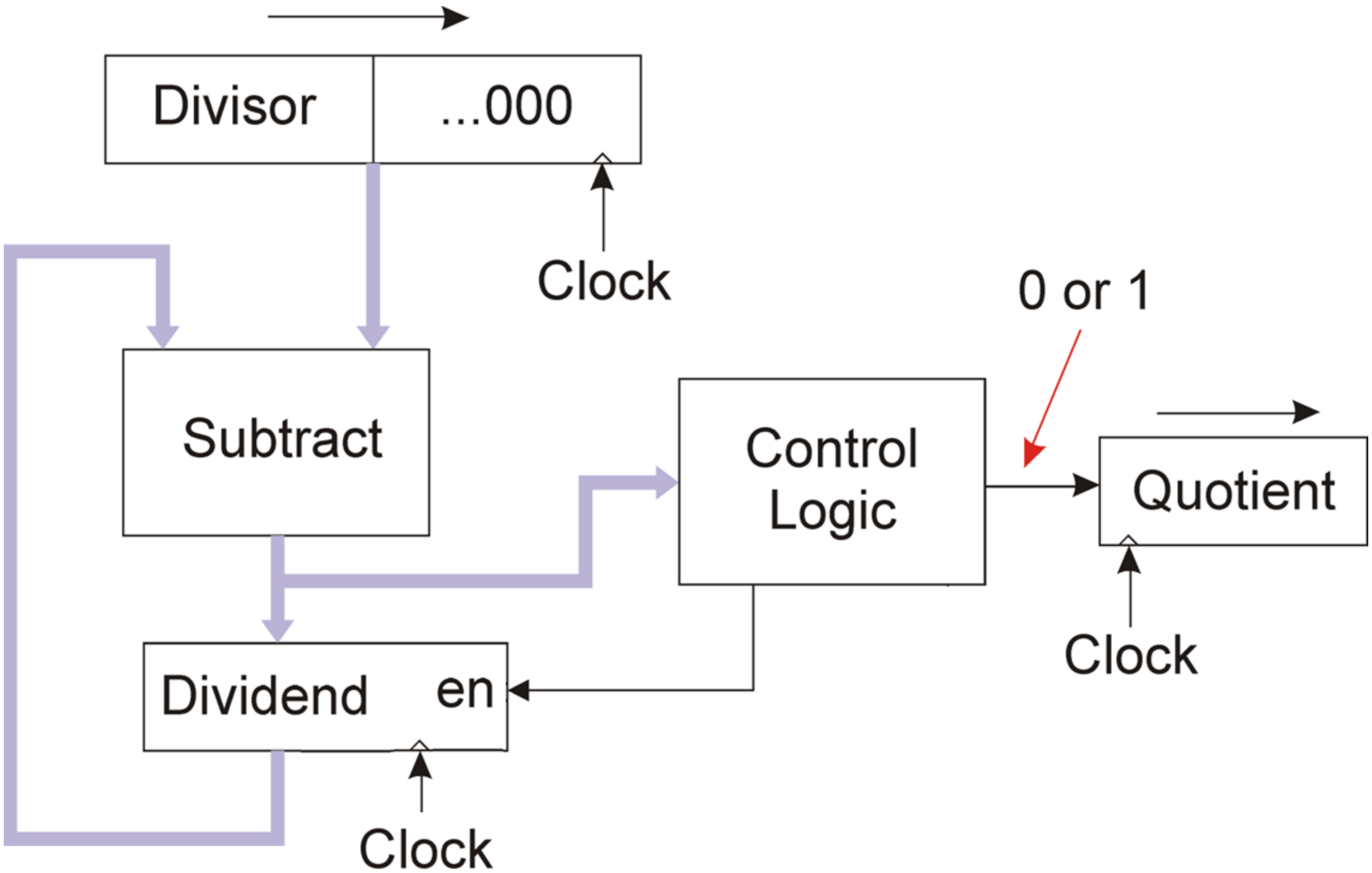
n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	010111000000000 ← 5888	0
7	454 → 0111000110	01011100000000 ← 2944	0
6	454 → 0111000110	0101110000000 ← 1472	0
5	454 → 0111000110	010111000000 ← 736	0
4	454 → 0111000110	01011100000 ← 368	1
3	0111000110 - 0101110000 = 01010110 ← 86	010111000 ← 184	0
2	86 → 01010110	01011100 ← 92	0
1	86 → 01010110	0101110 ← 46	1
0	01010110 - 0101110 = 0101000	010111	1
	0101000 - 010111 = 010001		

n	Dividend	Divisor $\times 2^n$	Quotient
8	454 $\rightarrow$ 0111000110	01011100000000 $\leftarrow$ 5888	0
7	454 $\rightarrow$ 0111000110	0101110000000 $\leftarrow$ 2944	0
6	454 $\rightarrow$ 0111000110	010111000000 $\leftarrow$ 1472	0
5	454 $\rightarrow$ 0111000110	01011100000 $\leftarrow$ 736	0
4	454 $\rightarrow$ 0111000110	0101110000 $\leftarrow$ 368	1
3	0111000110 - 0101110000 = 01010110 $\leftarrow$ 86	010111000 $\leftarrow$ 184	0
2	86 $\rightarrow$ 01010110	01011100 $\leftarrow$ 92	0
1	86 - 46 01010110	0101110 $\leftarrow$ 46	1
0	01010110 - 0101110 = 0101000 $\leftarrow$ 40	010111	1
	0101000 - 010111 = 010001		

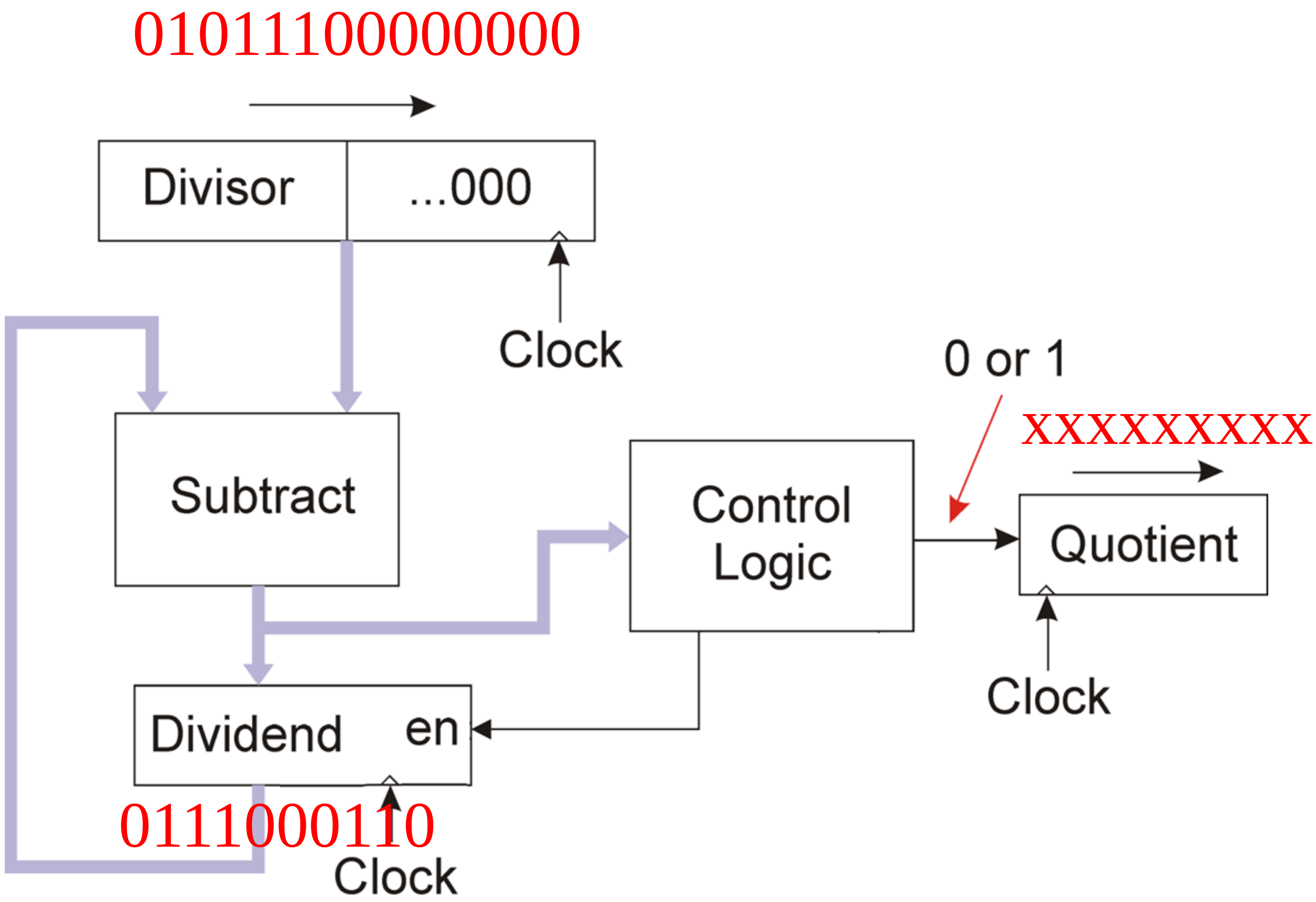
n	Dividend	Divisor $\times 2^n$	Quotient
8	454 → 0111000110	01011100000000 ← 5888	0
7	454 → 0111000110	0101110000000 ← 2944	0
6	454 → 0111000110	010111000000 ← 1472	0
5	454 → 0111000110	01011100000 ← 736	0
4	454 → 0111000110	0101110000 ← 368	1
3	0111000110 - 0101110000 = 01010110 ← 86	010111000 ← 184	0
2	86 → 01010110	01011100 ← 92	0
1	86 → 01010110	0101110 ← 46	1
0	01010110 - 0101110 = 0101000 ← 40	010111 ← 23	1
	0101000 - 010111 = 010001 ← 17		

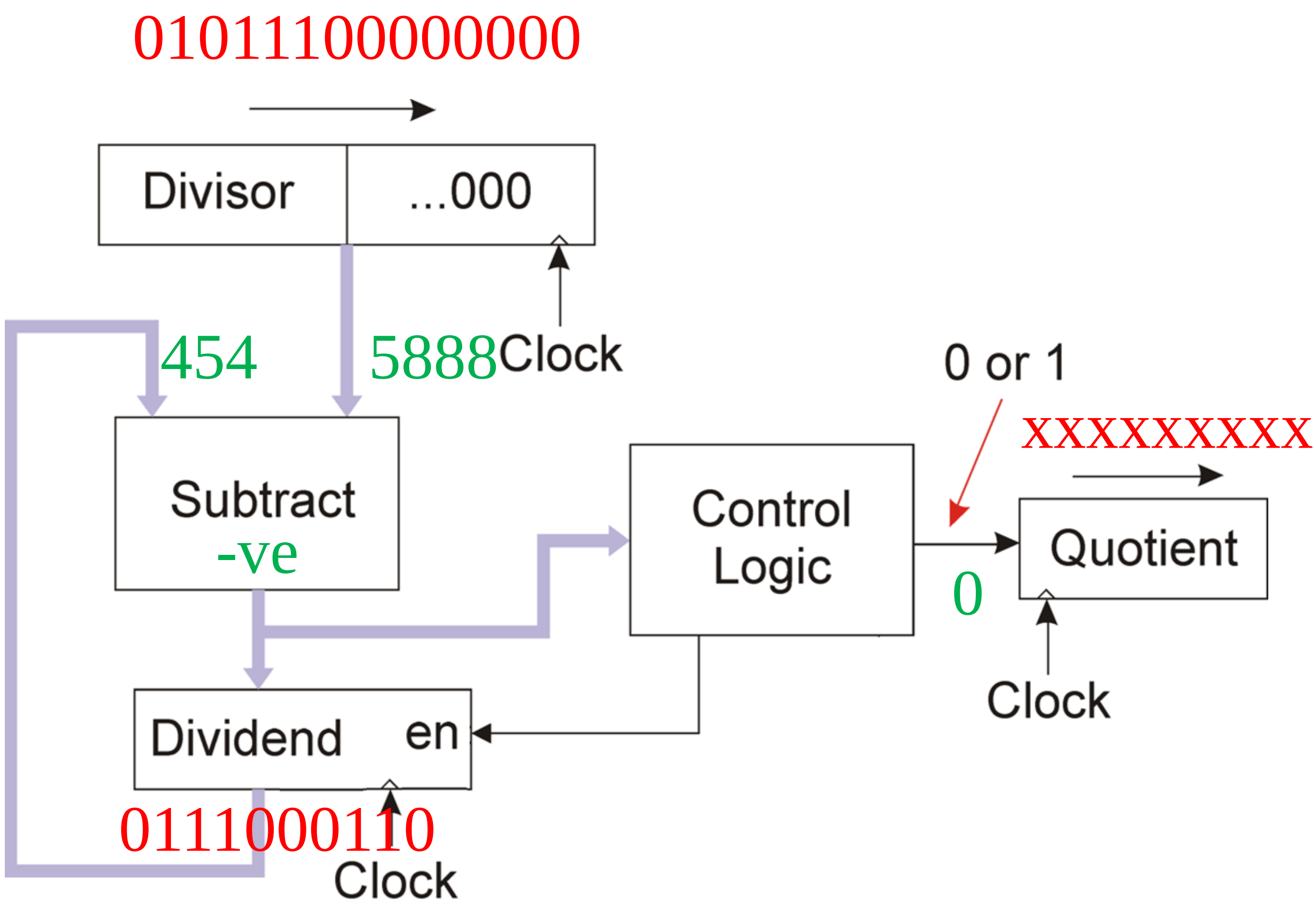
40 – 23 = 17

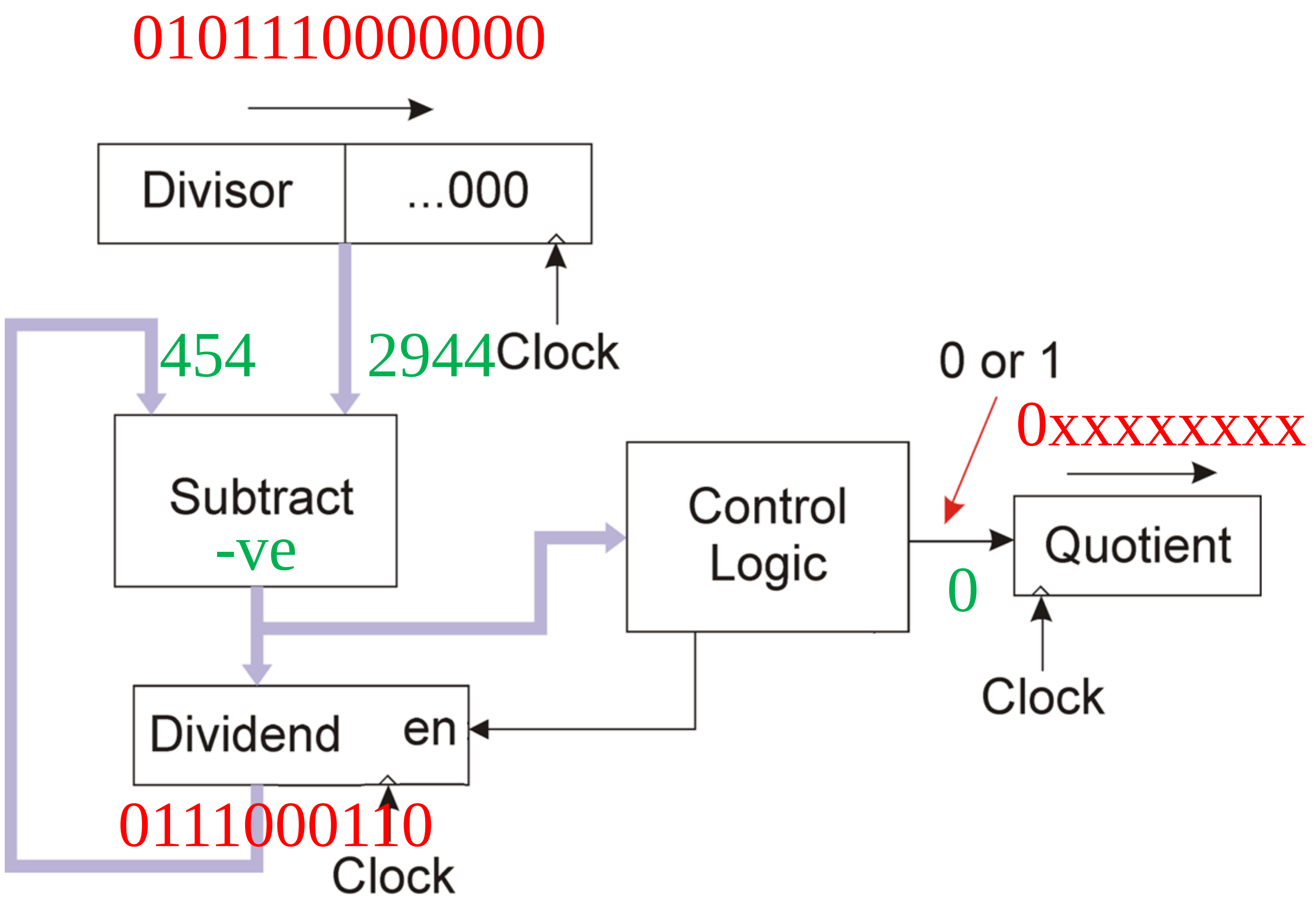
Possible Hardware Solution

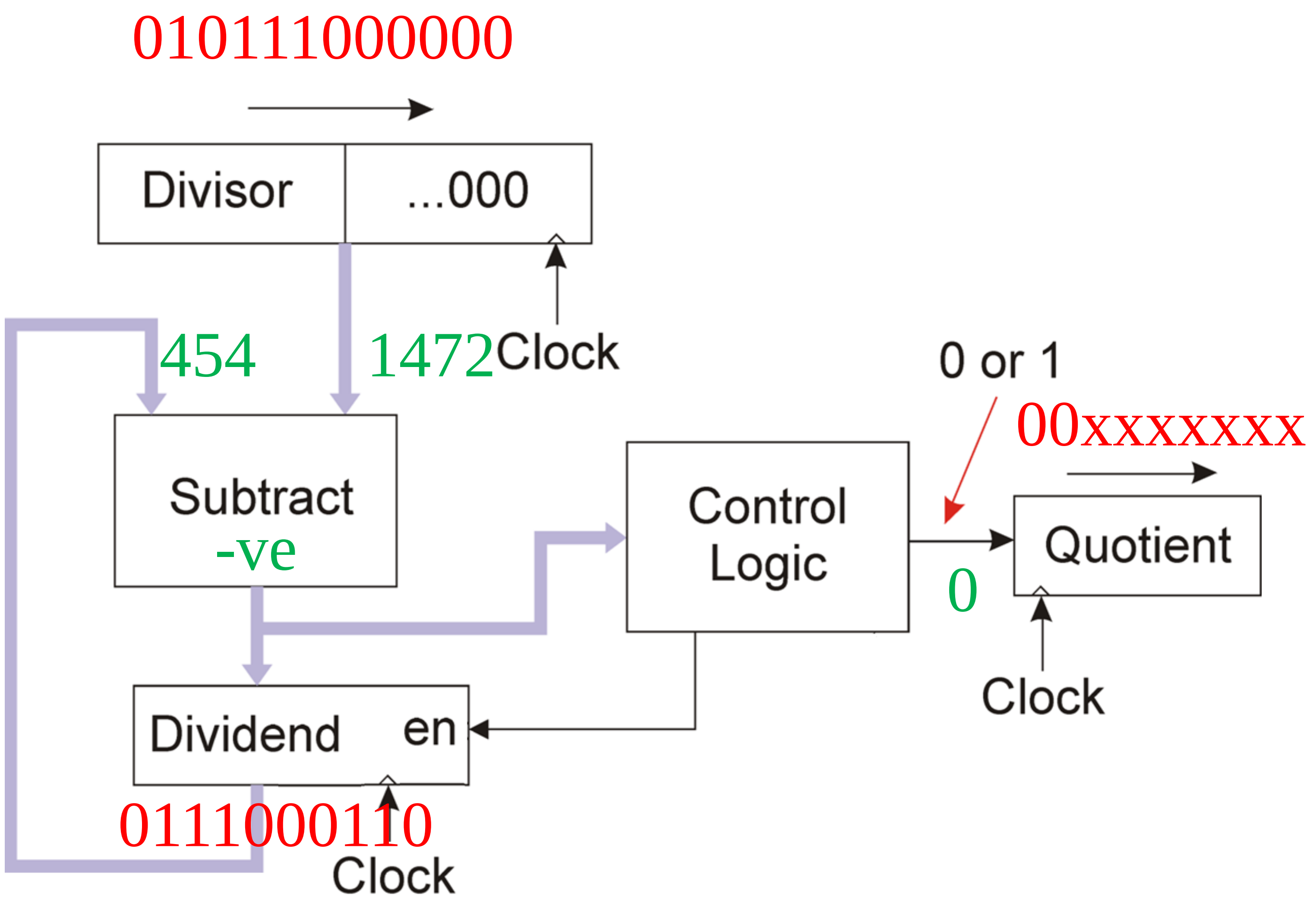


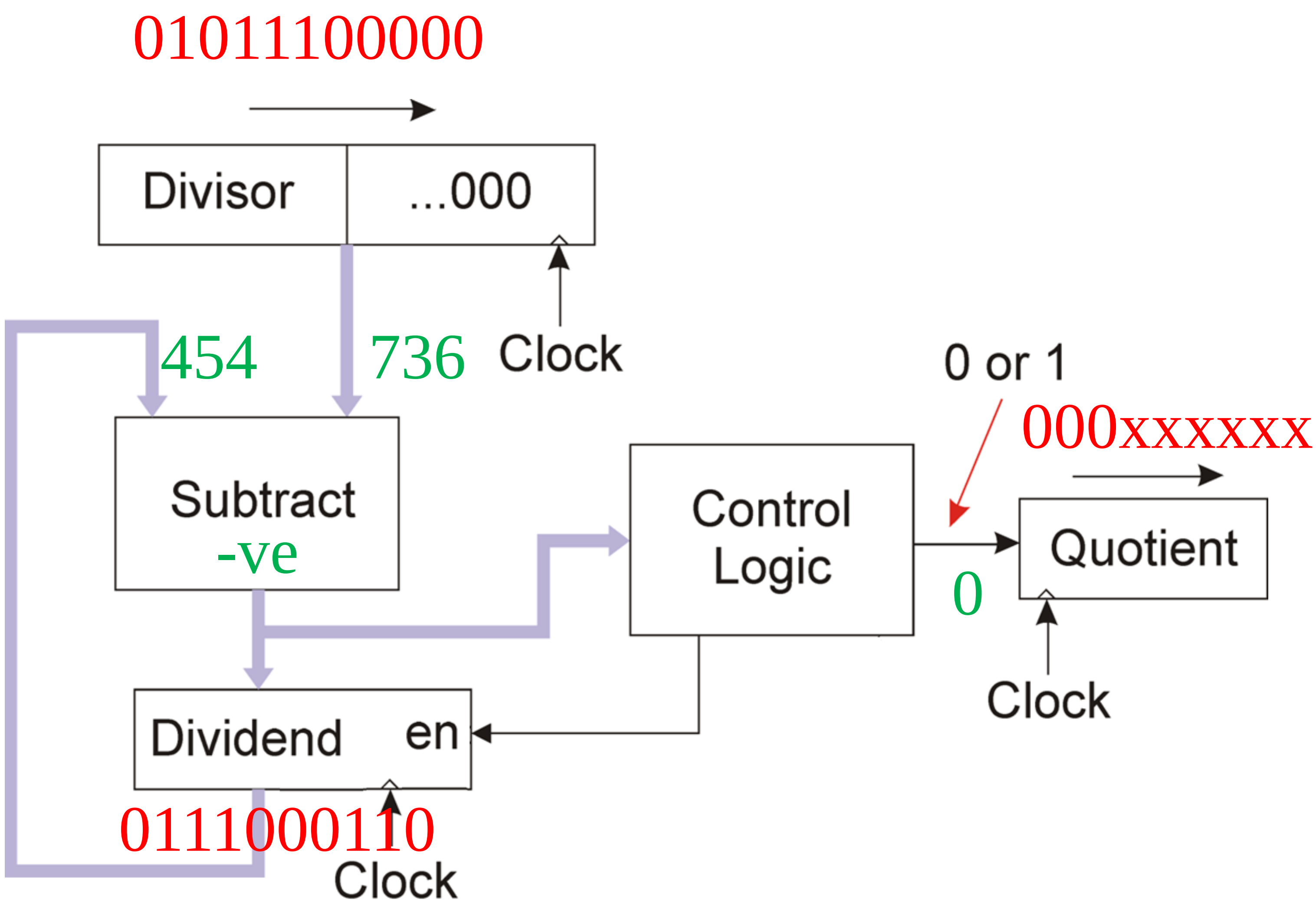
- Right shift register for extended divisor
- Combinatorial subtractor (2s complement)
- Dividend register with enable. (only enable when there is a +ve result to be clocked in, this is how the *restoring* functionality is obtained)
- Control logic – inverter on the sign bit of the subtraction result
- Quotient register (contents initially unknown

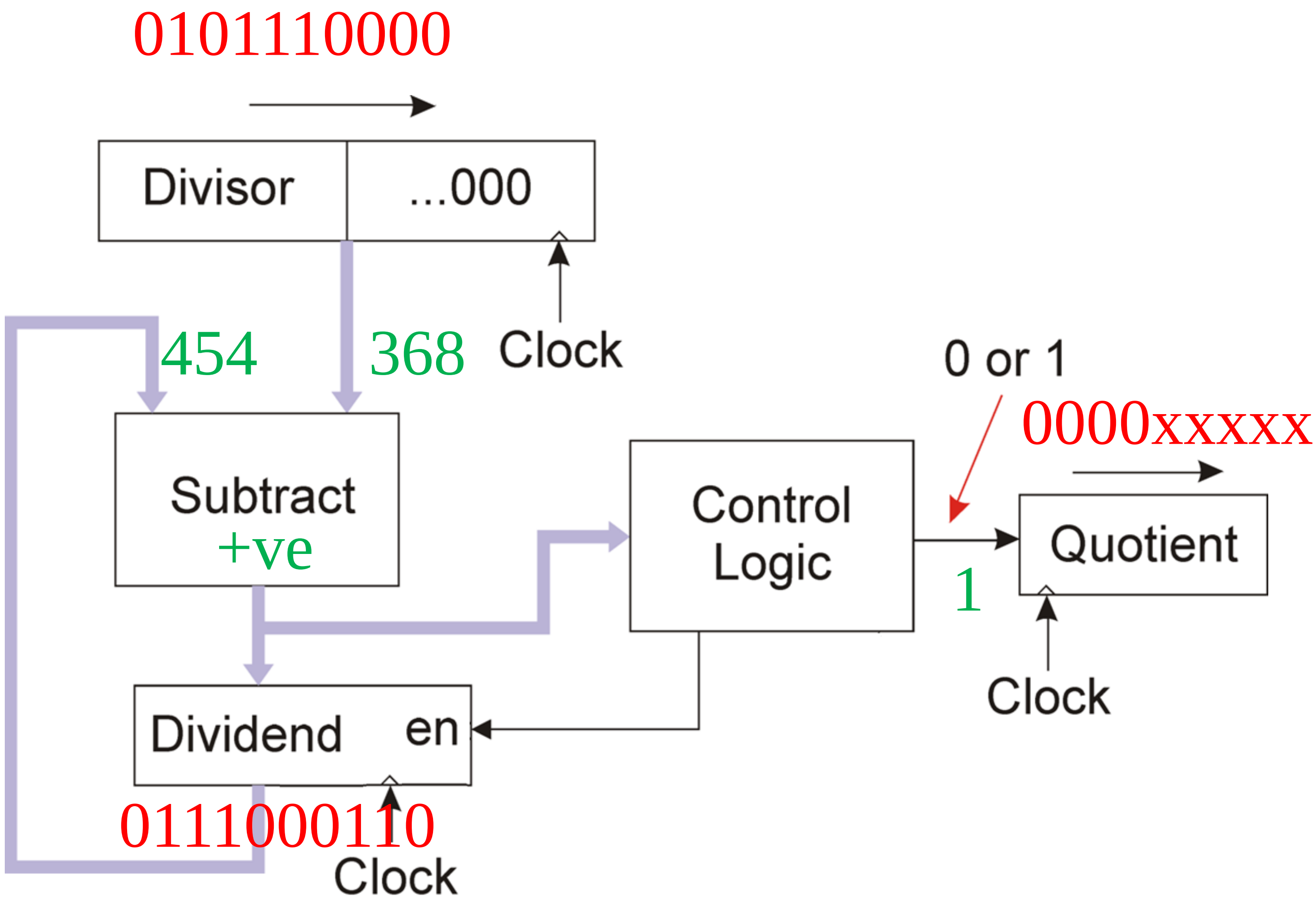

01011100000000
Divisor
...000
Clock
Subtract
Control
Logic
0 or 1
XXXXXXXXX
Quotient
Clock
Dividend
en
0111000110
Clock

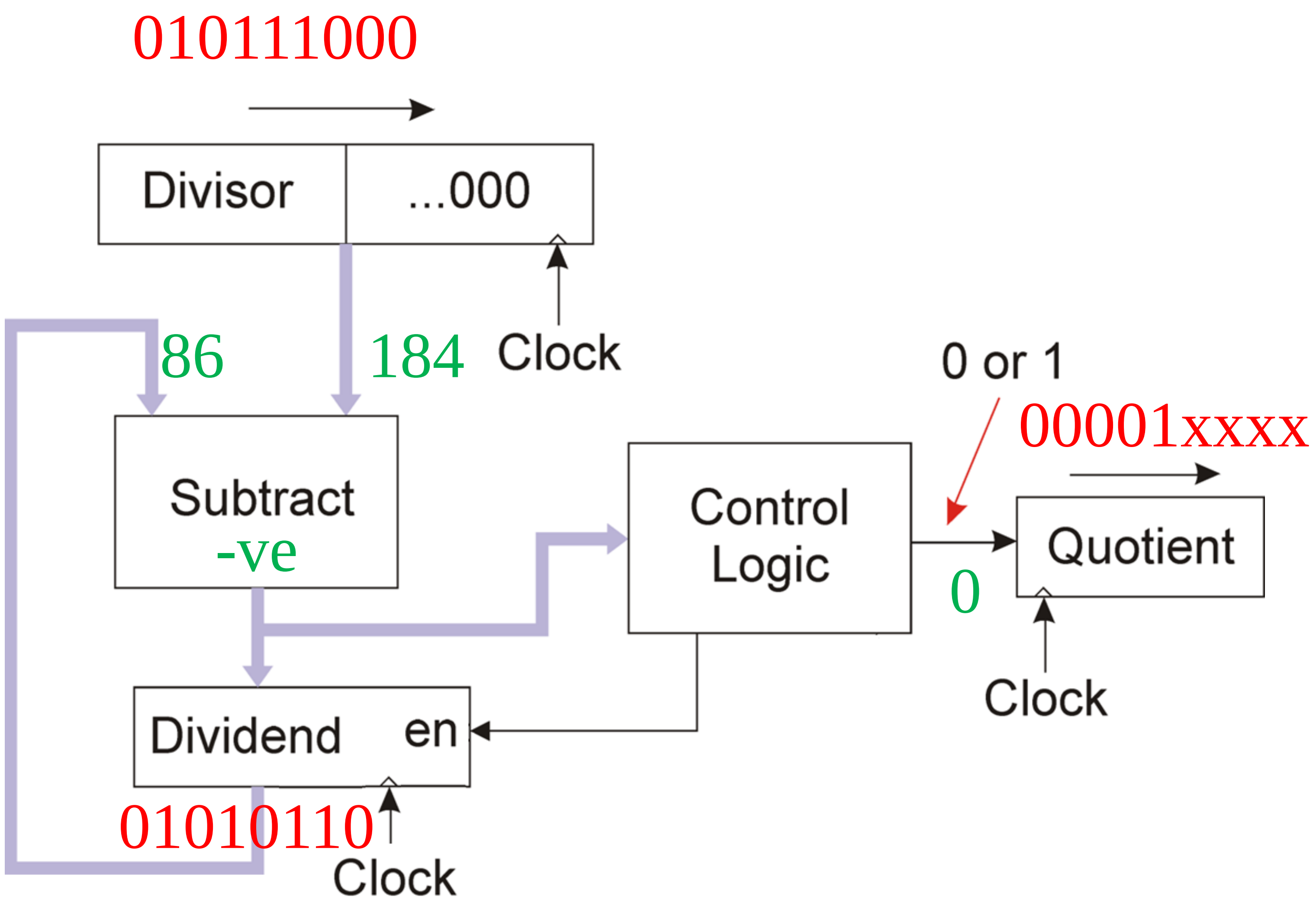

0101110000000
Divisor
...000
454
5888
Clock
Subtract
-ve
Control
Logic
0 or 1
xxxxxxxxx
Quotient
0
Clock
Dividend
en
0111000110
Clock

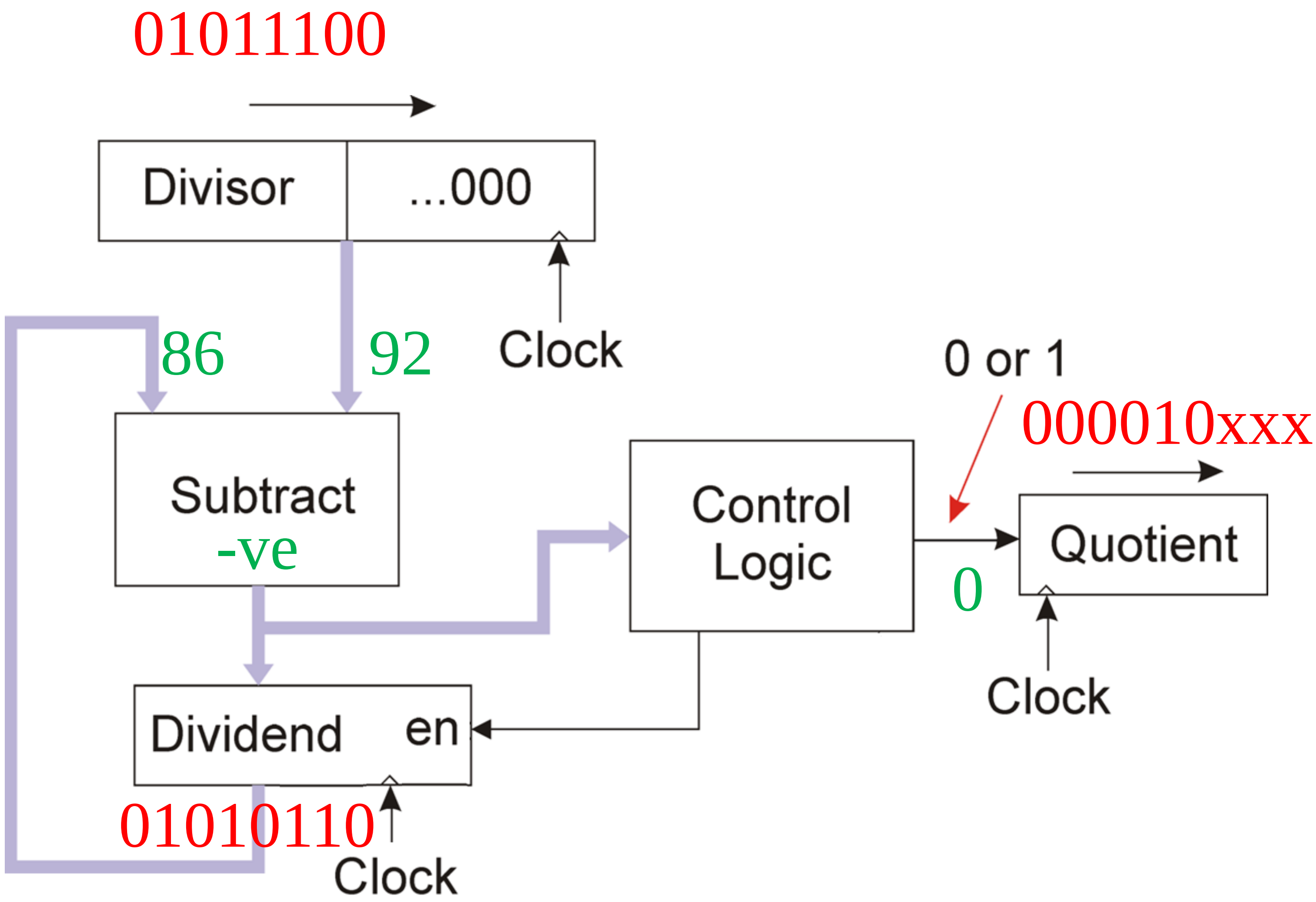

010111000000
Divisor
...000
454
2944
Clock
Subtract
-ve
Control
Logic
0 or 1
0xxxxxxxx
Quotient
0
Clock
Dividend
en
0111000110
Clock

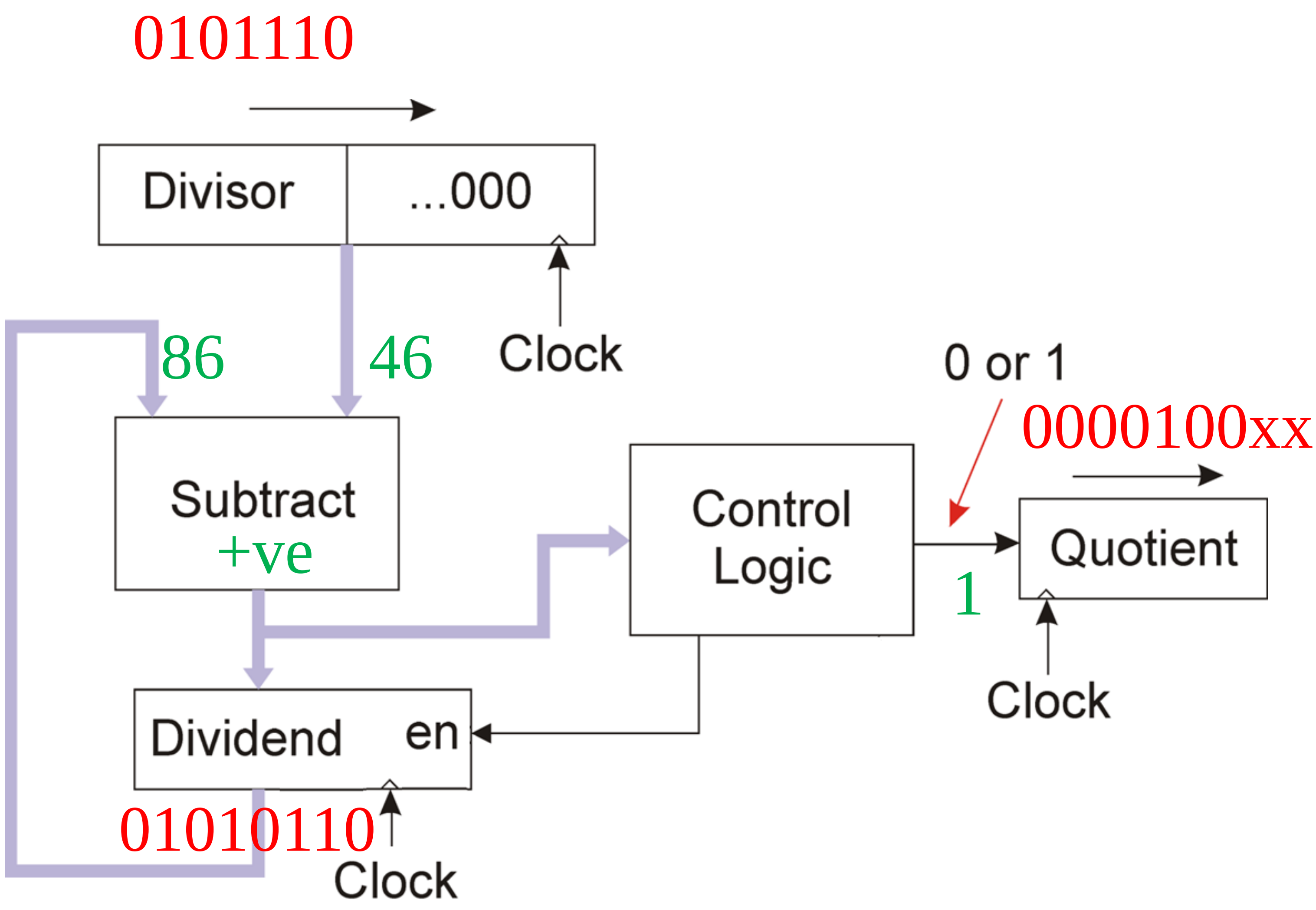

010111000000
Divisor
...000
454
1472
Clock
Subtract
-ve
Control
Logic
0 or 1
00xxxxxxx
Quotient
0
Clock
Dividend
en
0111000110
Clock

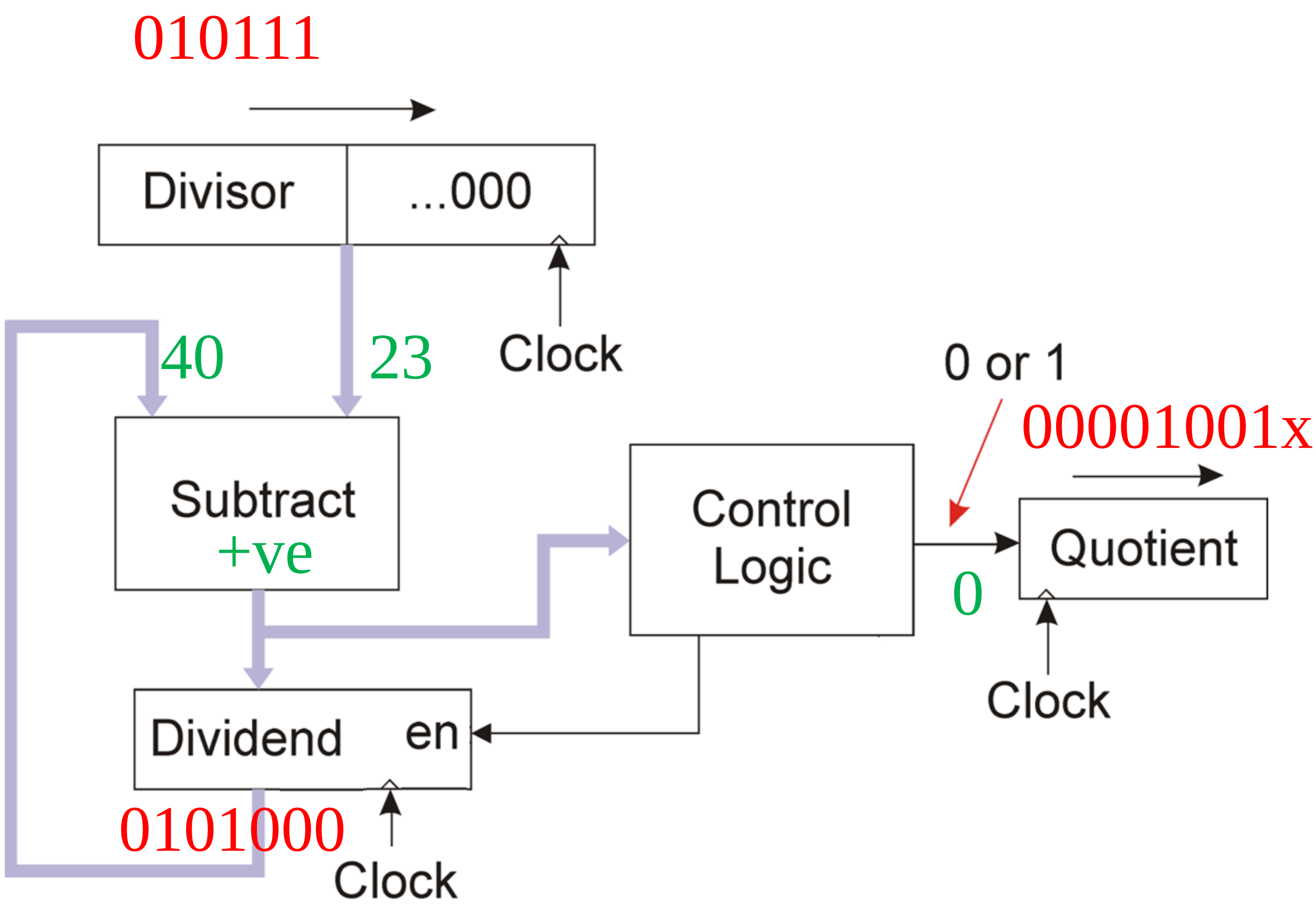

01011100000
Divisor
...000
Clock
454
736
Subtract
-ve
0 or 1
000xxxxxx
Control
Logic
Quotient
0
Clock
Dividend
en
0111000110
Clock

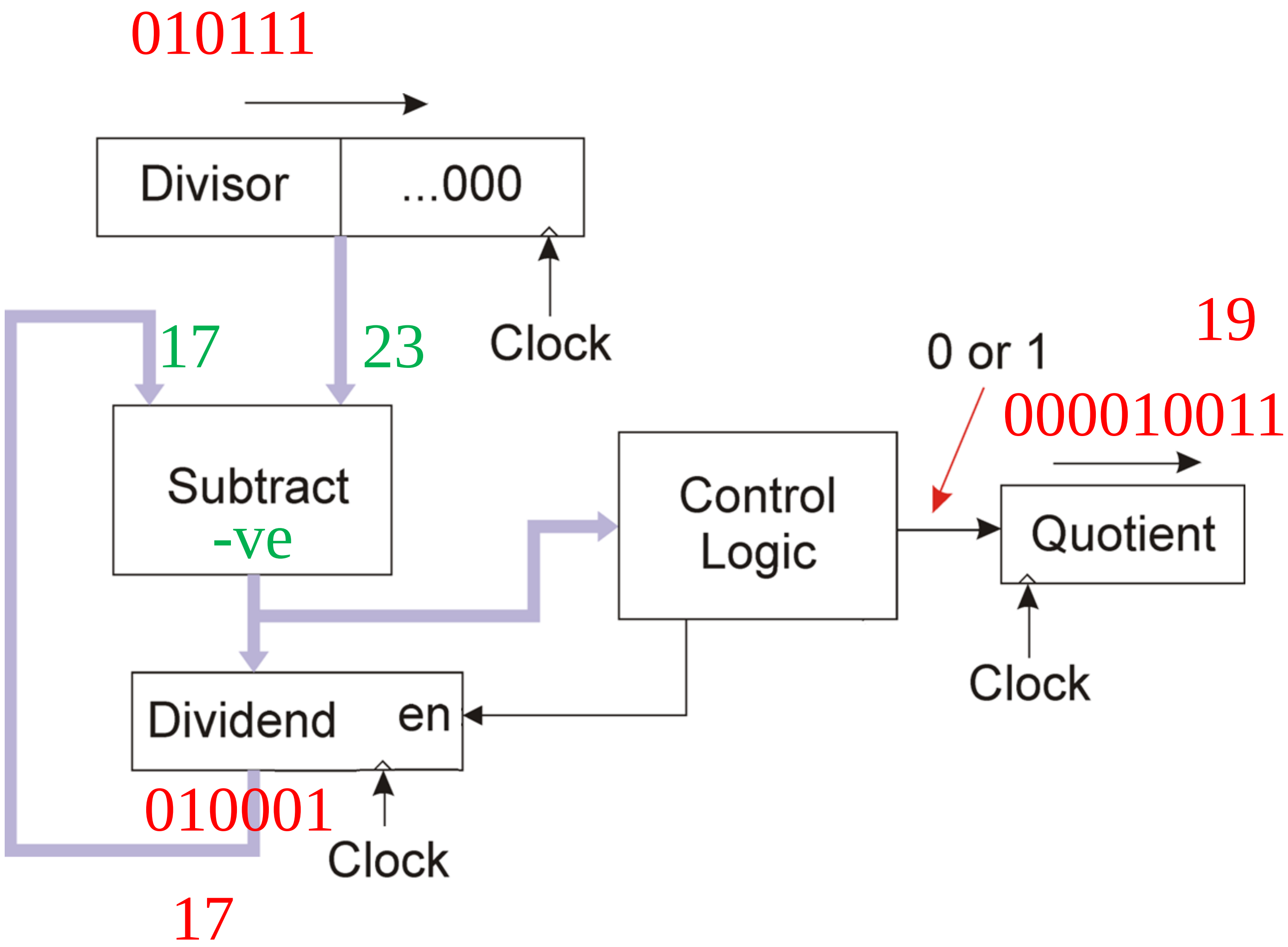

0101110000
Divisor
...000
454
368
Clock
0 or 1
0000xxxxx
Subtract
+ve
Control
Logic
Quotient
1
Clock
Dividend
en
0111000110
Clock


010111000
Divisor
...000
Clock
86
184
Clock
0 or 1
00001xxxx
Subtract
-ve
Control
Logic
Quotient
0
Clock
Dividend
en
01010110
Clock


01011100
Divisor
...000
86
92
Clock
0 or 1
000010xxx
Subtract
-ve
Control
Logic
Quotient
0
Clock
Dividend
en
01010110
Clock


0101110
Divisor
...000
86
46
Clock
0 or 1
0000100xx
Subtract
+ve
Control
Logic
Quotient
1
Clock
Dividend
en
01010110
Clock


010111
Divisor
...000
Clock
40
23
Subtract
+ve
0 or 1
00001001x
Control
Logic
Quotient
0
Clock
Dividend
en
0101000
Clock


010111
Divisor
...000
Clock
17
23
Subtract
-ve
Control
Logic
0 or 1
19
000010011
Quotient
Clock
Dividend
en
010001
Clock
17