Integer Division

Division of Unsigned Integers

- Two operands: Divisor and Dividend

Dividend (D): 274

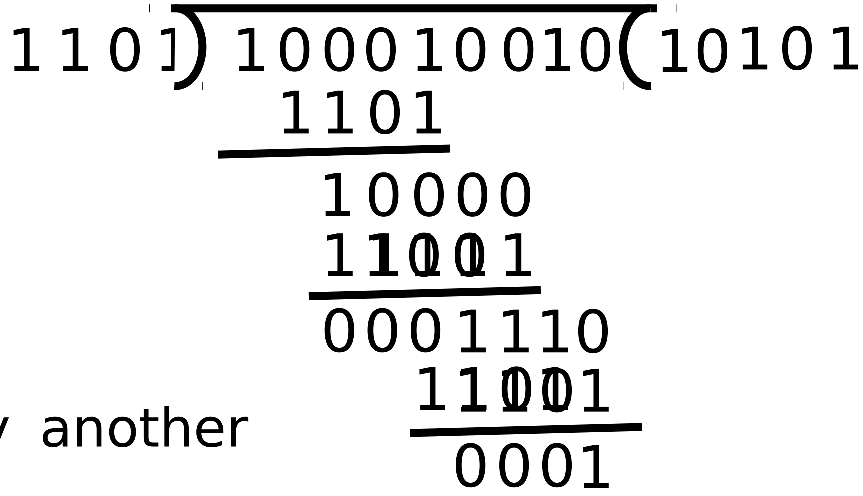
Divisor (M): 13

```
13 ) 274 ( 21  ← Quotient (Q)
     26
     ----
      14
      13
     ----
       1  ← Remainder (R)
```

- D = Q x M + R
- Long-hand-division (paper-and-pencil method)

Binary Division

- In binary division the only possibilities for quotient bits are 0 and 1
- Divisor is positioned appropriately with respect to the dividend and perform subtraction
- If the remainder is 0 or positive, a quotient bit of 1 is determined
- The remainder is extended by another bit of the dividend



- The devisor is repositioned and another subtraction is performed
- If the remainder is negative, a quotient bit of 0 is determined
- The dividend is restored by adding back the divisor

n-bit Unsigned Division – Restoring Division

- For any n-bit division, we will have $n/2$-bit divisor and n-bit dividend
- It produces n-bit quotient and n-bit remainder
- Restoring division method uses 3 registers
  - Register M: $m_n m_{n\text{-}1} \ldots m_1 m_0$. $(n+1)$-bit length
    - It holds $n/2$-bit positive divisor. The MSB holds 0
  - Register Q: $q_{n\text{-}1} \ldots q_1 q_0$
    - It holds n-bit positive dividend at the start of the operation
    - After the division is complete, it will contain n-bit quotient
  - Register A: $a_n a_{n\text{-}1} \ldots a_1 a_0$. $(n+1)$-bit length
    - Set to 0 at the beginning of the operation
    - After the division is complete, it will contain n-bit remainder
  - The extra bit in MSB of A and M accommodates the sign bit during subtraction
  - Subtraction is done using 2's complement arithmetic

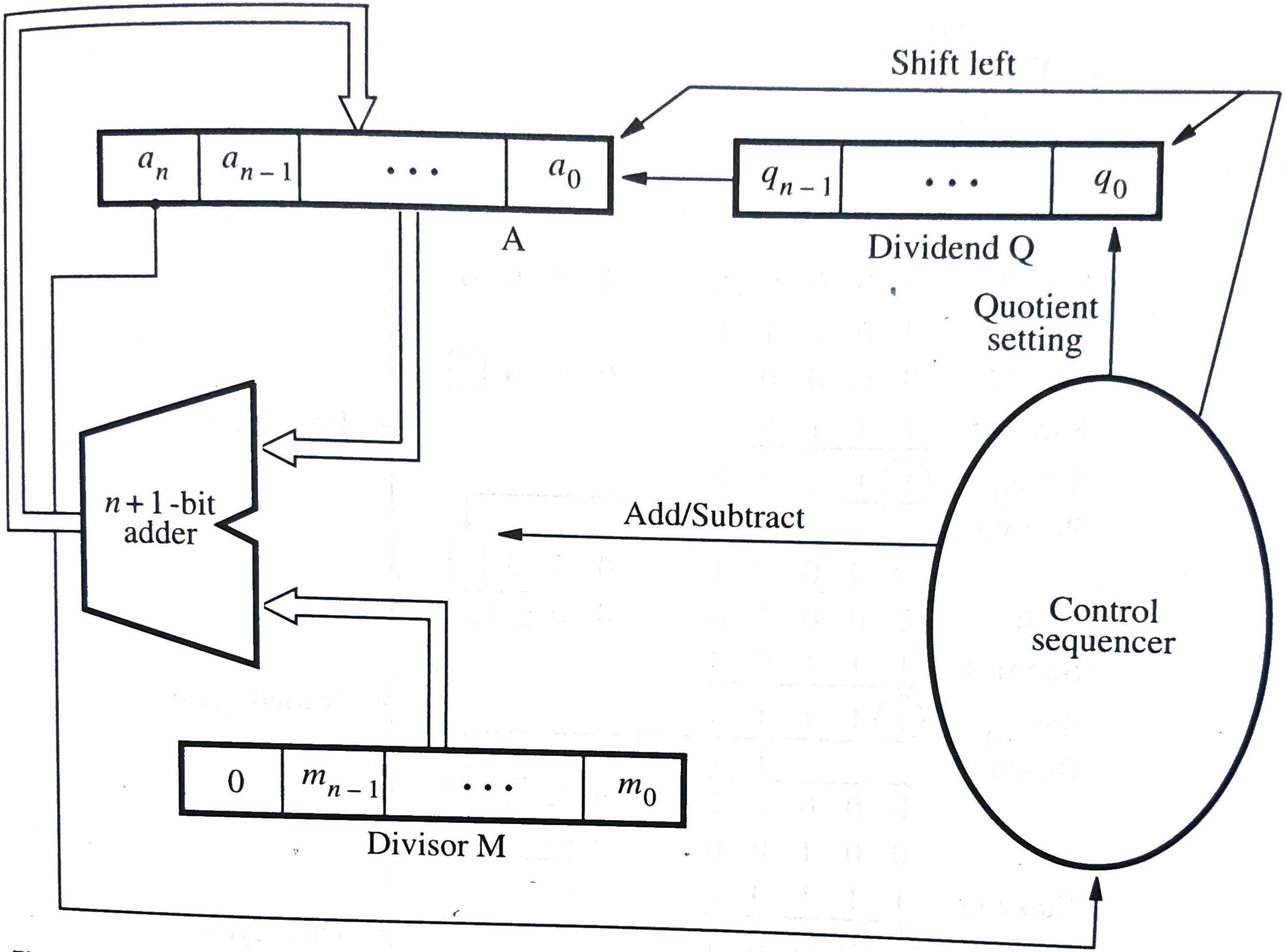


Figure 6.21 Circuit arrangement for binary division.

Restoring Division

- Algorithm:

Do the following steps n times

1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1
    1. Set q_0 to 0
    2. Add M back to A (i.e. restore A)
4. Otherwise, set q_0 to 1

- 4-bit division:
- 8/3
  - Quotient: 2
  - Remainder: 2

```
11 ) 1000 ( 10
     11
     --
       10
```

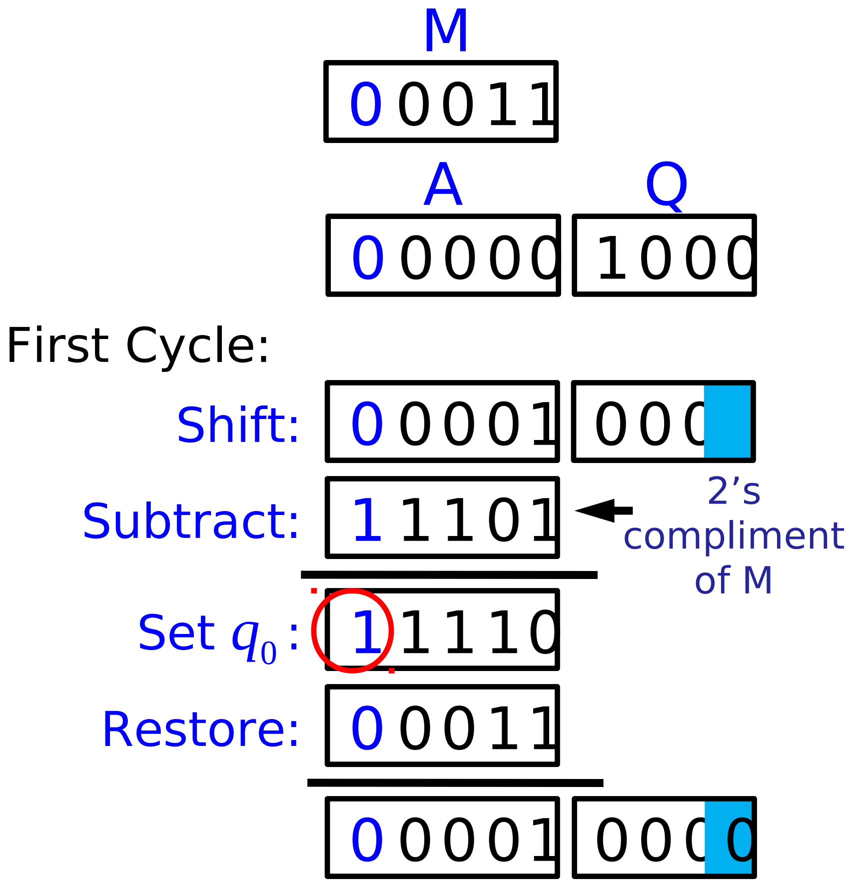
Restoring Division

- Algorithm:

Do the following steps n times

1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1
   1. Set q_0 to 0
   2. Add M back to A (i.e. restore A)
4. Otherwise, set q_0 to 1

- 4-bit division:



Restoring Division

- Algorithm:

Do the following steps n times

1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1
   1. Set q_0 to 0
   2. Add M back to A (i.e. restore A)
4. Otherwise, set q_0 to 1

- 4-bit division:

M

0 0 0 1 1

A Q

0 0 0 0 1 0 0 0 0

Second Cycle:

Shift: 0 0 0 1 0 0 0 0

Subtract: 1 1 1 0 1

Set q_0: 1 1 1 1 1

Restore: 0 0 0 1 1

0 0 0 1 0 0 0 0 0

Restoring Division

- Algorithm:

Do the following steps n times

1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1
   1. Set q_0 to 0
   2. Add M back to A (i.e. restore A)
4. Otherwise, set q_0 to 1

- 4-bit division:

M

0 0 0 1 1

A | Q

0 0 0 1 0 | 0 0 0 0

Third Cycle:

Shift: 0 0 1 0 0 | 0 0 0

Subtract: 1 1 1 0 1

Set q_0: 0 0 0 0 1

0 0 0 0 1 | 0 0 0 1

Restoring Division

- Algorithm:

Do the following steps n times

1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1
   1. Set q_0 to 0
   2. Add M back to A (i.e. restore A)
4. Otherwise, set q_0 to 1

- 4-bit division:

M

0 0 0 1 1

A Q

0 0 0 0 1 | 0 0 0 1

Fourth Cycle:

Shift: 0 0 0 1 0 | 0 0 1

Subtract: 1 1 1 0 1

Set q_0: 1 1 1 1 1

Restore: 0 0 0 1 1

0 0 0 1 0 | 0 0 1 0

Remainder **Quotient**

Restoring Division

- The restoring division can be improved by avoiding the need for restoring A after unsuccessful subtraction
- Subtraction is said to be unsuccessful if the result is negative

Non-Restoring Division

- Algorithm:

Step-1:

Do the following steps n times

1. Shift A and Q left one binary position
2. If the sign of A is 0

   then, subtract M from A

   else, add M to A
3. If the sign of resulting A is 0

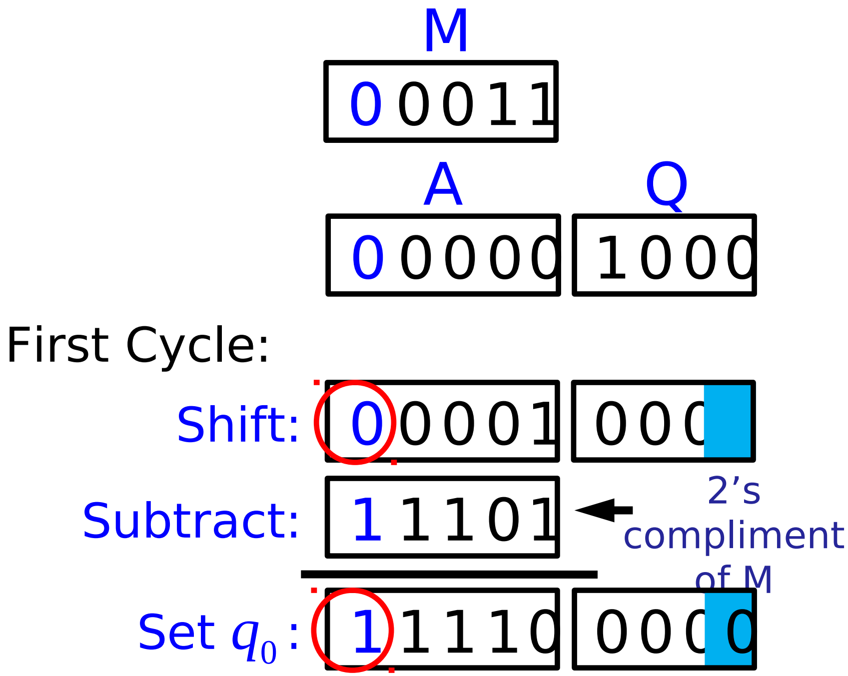
   then set q_0 to 1

   Otherwise, set q_0 to 0

Step-2:

If the sign of A is 1, add M to A

- 4-bit division:



Non-Restoring Division

- Algorithm:

Step-1:

Do the following steps n times

1. Shift A and Q left one binary position
2. If the sign of A is 0

   then, subtract M from A

   else, add M to A
3. If the sign of resulting A is 0

   then set q_0 to 1

   Otherwise, set q_0 to 0

Step-2:

If the sign of A is 1, add M to A

- 4-bit division:

	M	Q
	0 0 0 1 1	
	A	Q
	1 1 1 1 0	0 0 0 0

Second Cycle:

	A	Q
Shift:	1 1 1 0 0	0 0 0
Add:	0 0 0 1 1	
Set q_0:	1 1 1 1 1	0 0 0 0

Non-Restoring Division

- Algorithm:

Step-1:

Do the following steps n times

1. Shift A and Q left one binary position
2. If the sign of A is 0
   then, subtract M from A
   else, add M to A
3. If the sign of resulting A is 0
   then set q_0 to 1
   Otherwise, set q_0 to 0

Step-2:

If the sign of A is 1, add M to A

- 4-bit division:

M: 00011

A: 11111 Q: 0000

Third Cycle:

Shift: 11110 000

Add: 00011

Set q_0: 00001 0001

Non-Restoring Division

- Algorithm:

Step-1:

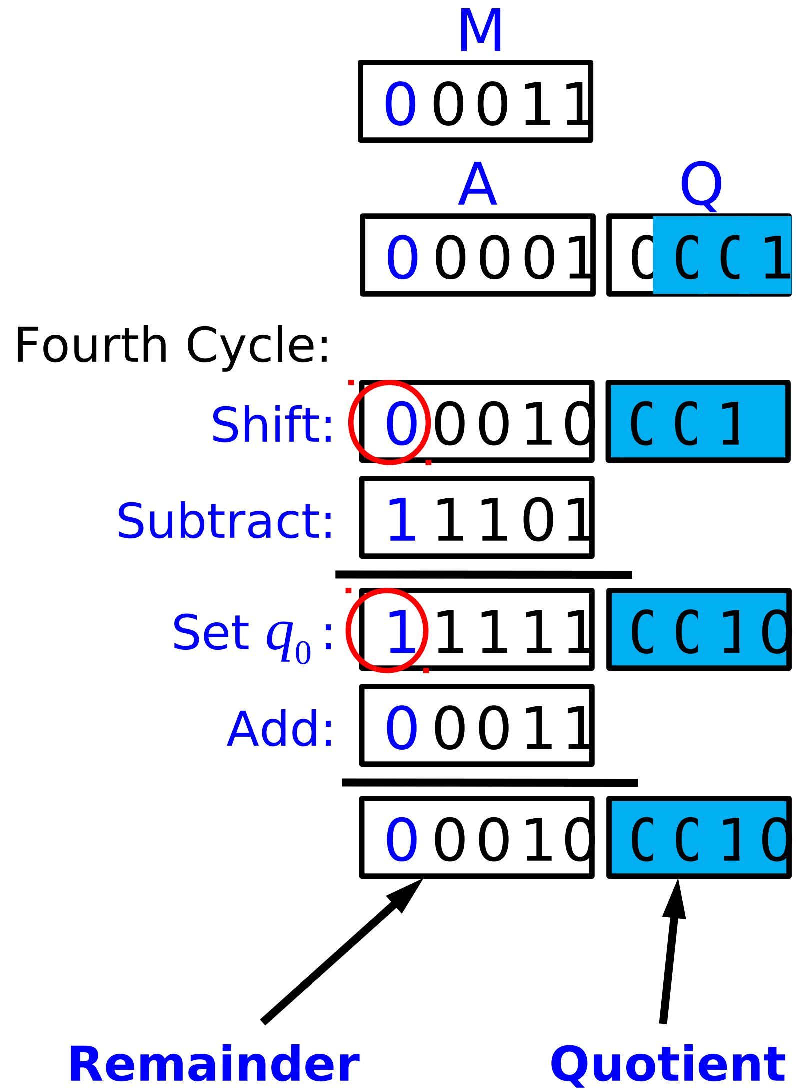
Do the following steps n times

1. Shift A and Q left one binary position
2. If the sign of A is 0
   then, subtract M from A
   else, add M to A
3. If the sign of resulting A is 0
   then set q_0 to 1
   Otherwise, set q_0 to 0

Step-2:

If the sign of A is 1, add M to A

- 4-bit division:

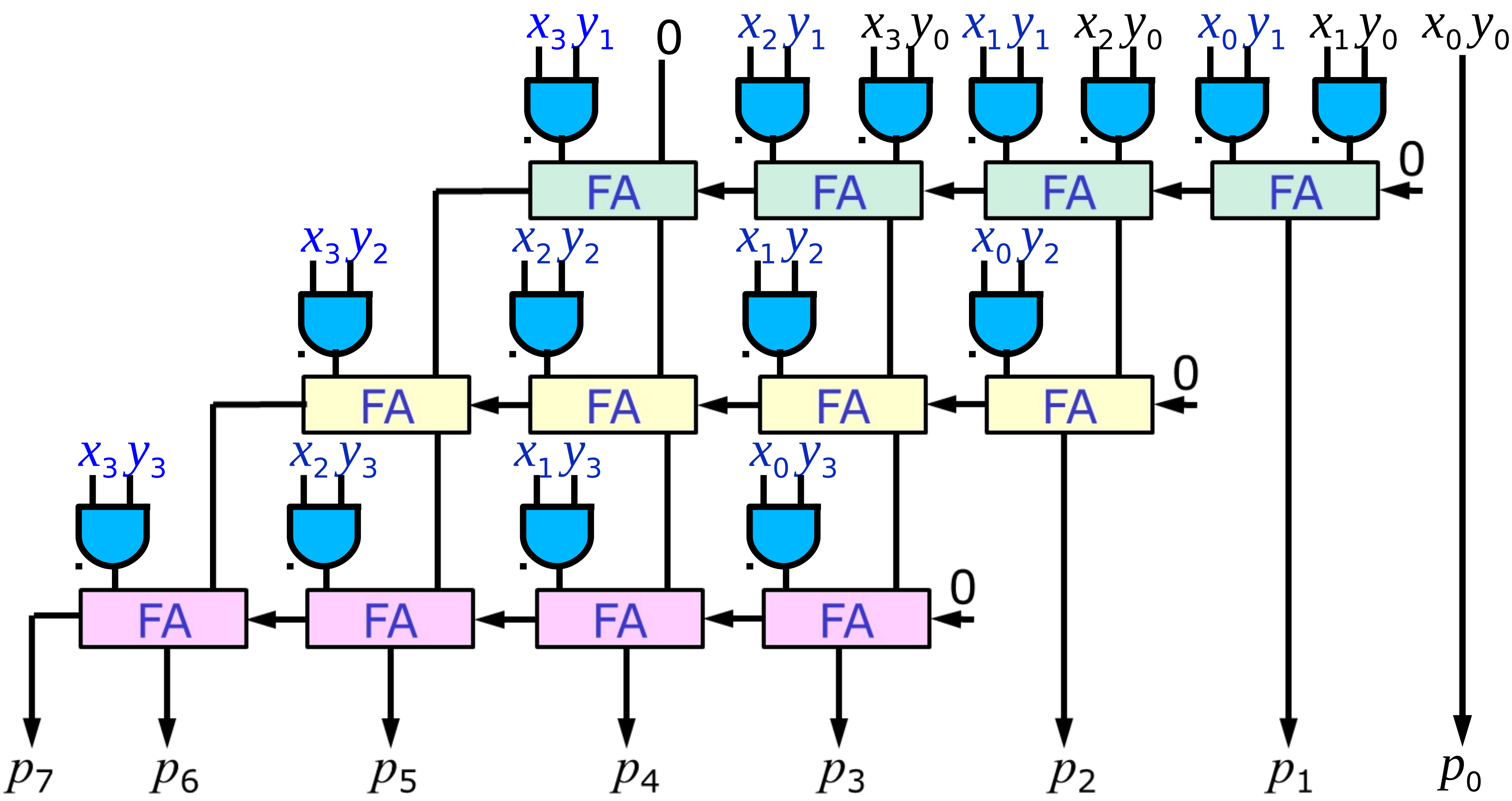


Signed Integer Division

- There is no simple algorithms for directly performing division on signed operands
- In signed division, the negative operands can be pre-processed to transform them into positive values
- After using restoring or non-restoring division method, the results are transformed to the correct signed values

Combinational Array Division Circuit

Combinational Array Multiplier Circuit



- Uses ripple carry adder

Combinational Array (CA) Division Circuit

- Analogous to the array multiplier
- Array divider can be realized by implementing the behaviour of each division step on a row of basic cells
- Basic cell in array divider depends on the specific division algorithm (restored or non-restored) to be implemented
- **Non-Restoring Algorithm**:
  - A row of cells accepts the intermediate remainder and divisor as input
  - Addition or Subtraction should be implemented i.e. 1-bit full adder/subtractor to be implemented based on the sign

Non-Restoring Division

- Algorithm:

Step-1:

Do the following steps n times

1. Shift A and Q left one binary position
2. If the sign of A is 0

   then, subtract M from A

   else, add M to A
3. If the sign of resulting A is 0

   then set q_0 to 1

   Otherwise, set q_0 to 0

Step-2:

If the sign of A is 1, add M to A

- 4-bit division:

M: 0 0 0 1 1

A: 0 0 0 0 0 Q: 1 0 0 0

First Cycle:

Shift: 0 0 0 0 1 | 0 0 0 _

Subtract: 1 1 1 0 1 ← 2's compliment of M

Set q_0: 1 1 1 1 0 | 0 0 0 0

Basic Cell of a CA Division – Non-restoring Algorithm

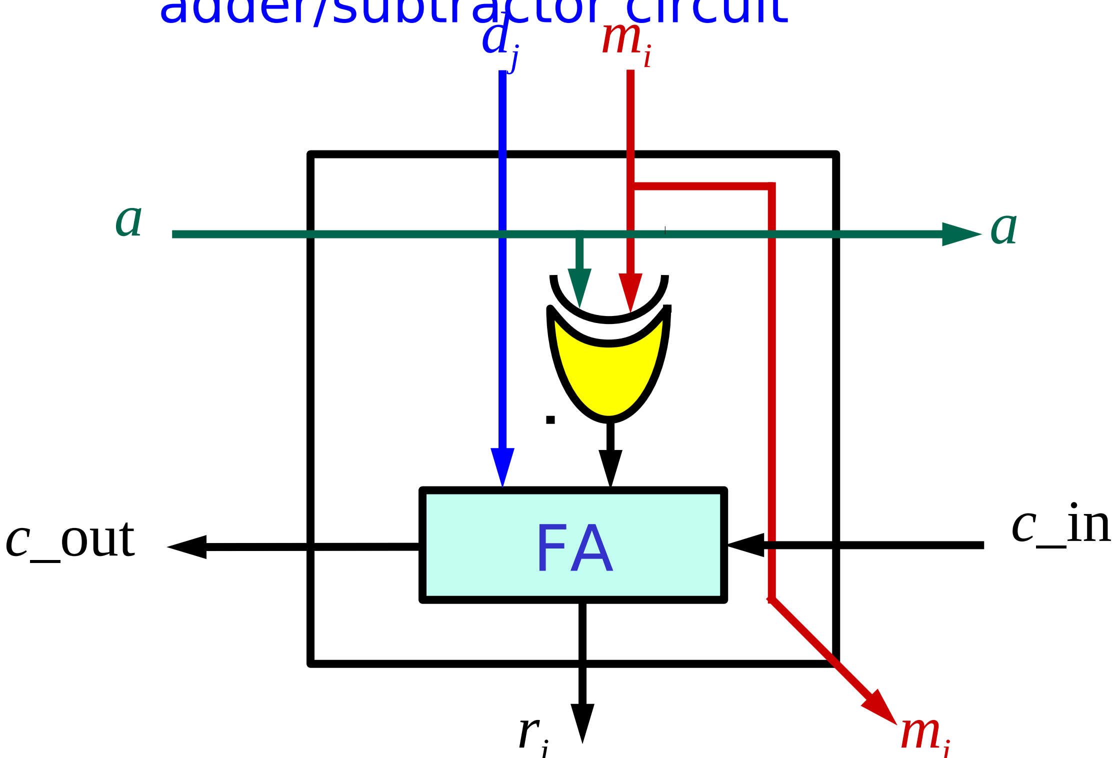
- Dividend is $2n$-bit length

  $D = d_{2n\text{-}1}\, d_{2n\text{-}2} \ldots d_1\, d_0$

- Divisor is n-bit length

  $M = m_{n\text{-}1}\, m_{n\text{-}2} \ldots m_1\, m_0$

- Basic cell is a full adder (FA) that acts as an adder/subtractor circuit



c_in : Carry in

c_out : Carry out

r_j : Remainder bit

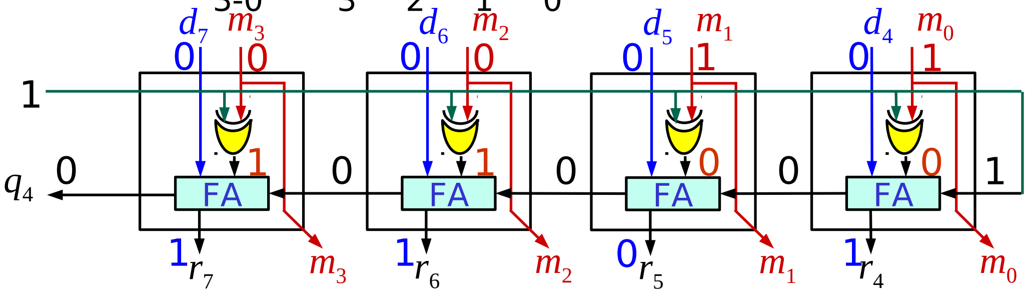
when,

a=1, m_i is 1's complimented

a =0, m_i is unchanged

4-bit CA Division: Non-restore Algorithm

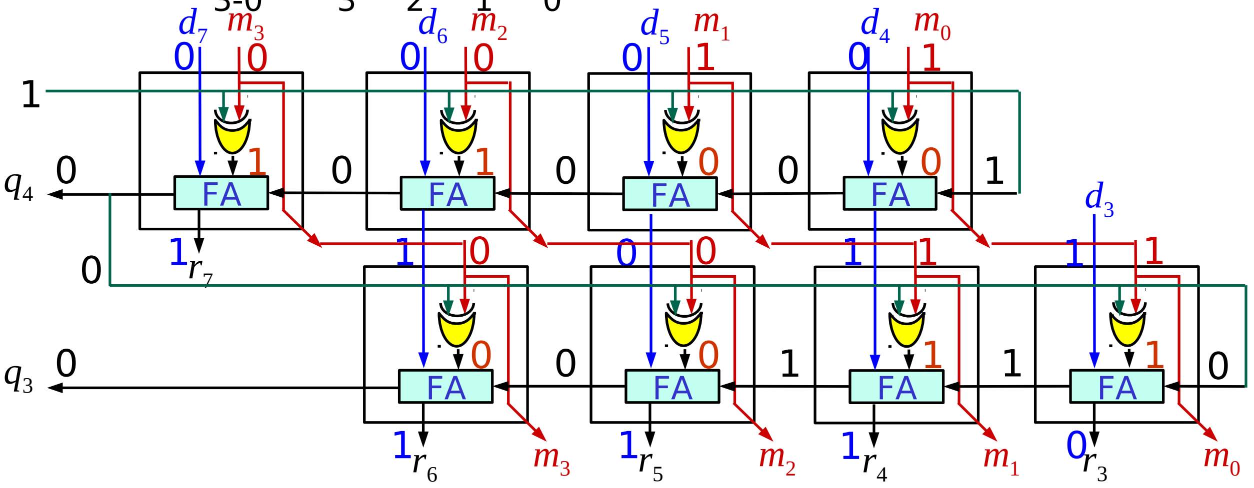
- D_{7-0}: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- M_{3-0}: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



Subtraction

4-bit CA Division: Non-restore Algorithm

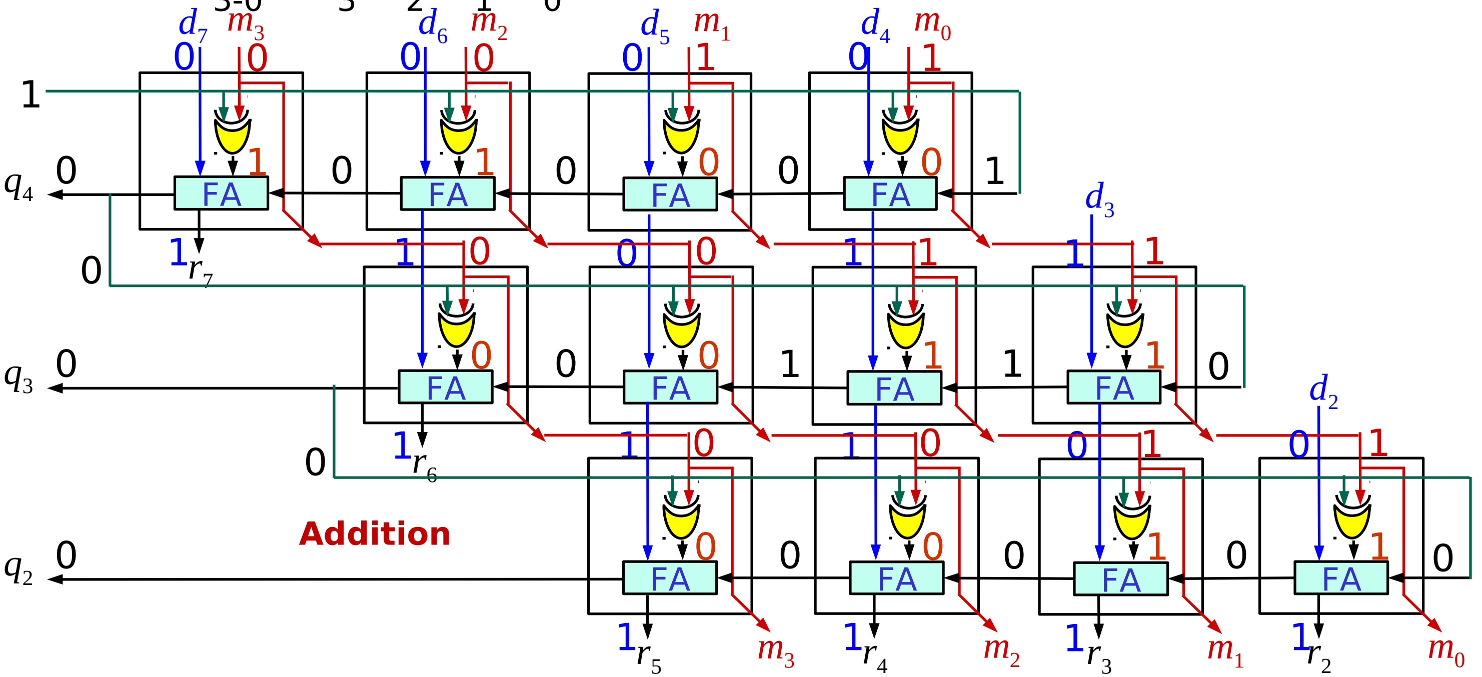
- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



Addition

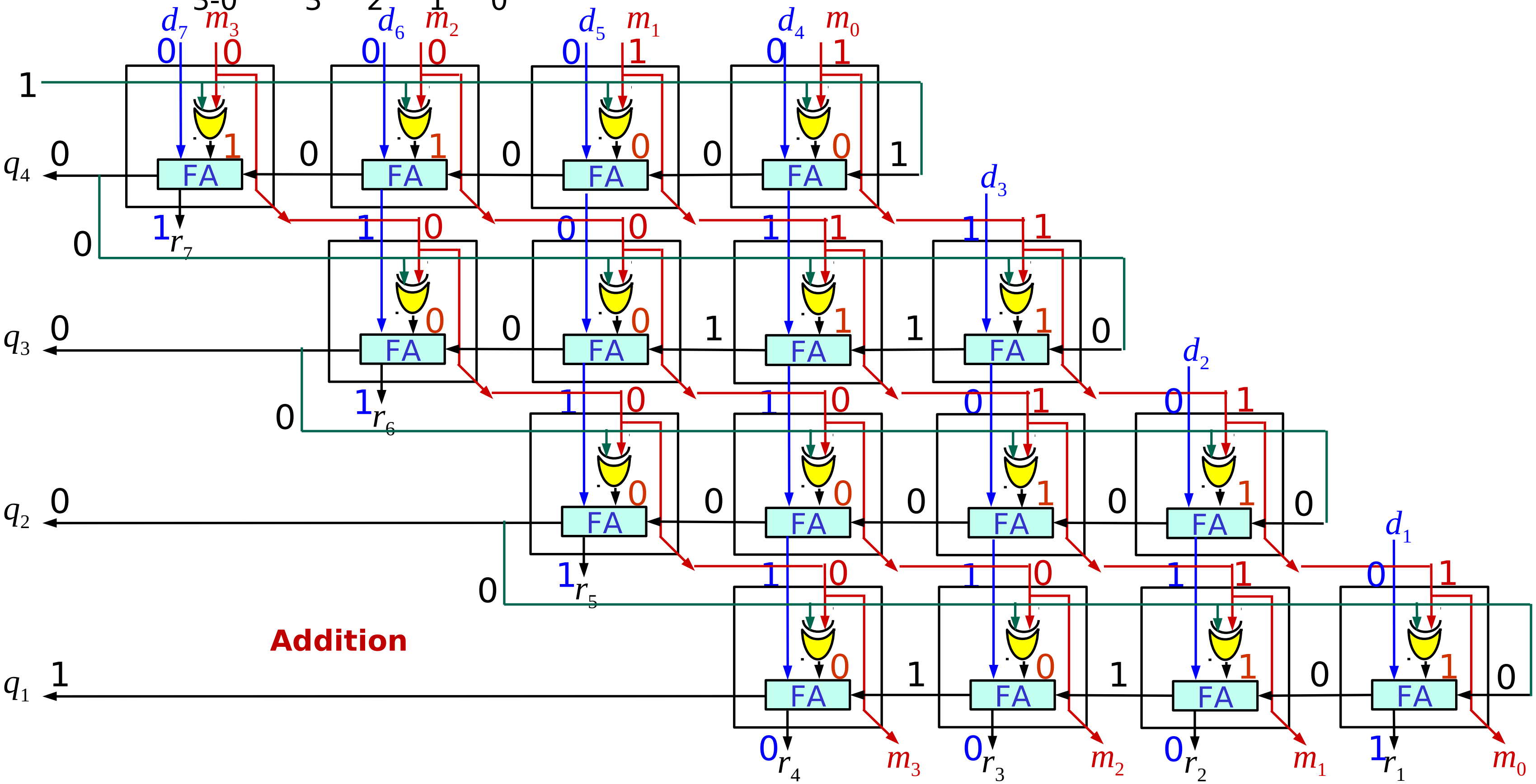
4-bit CA Division: Non-restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



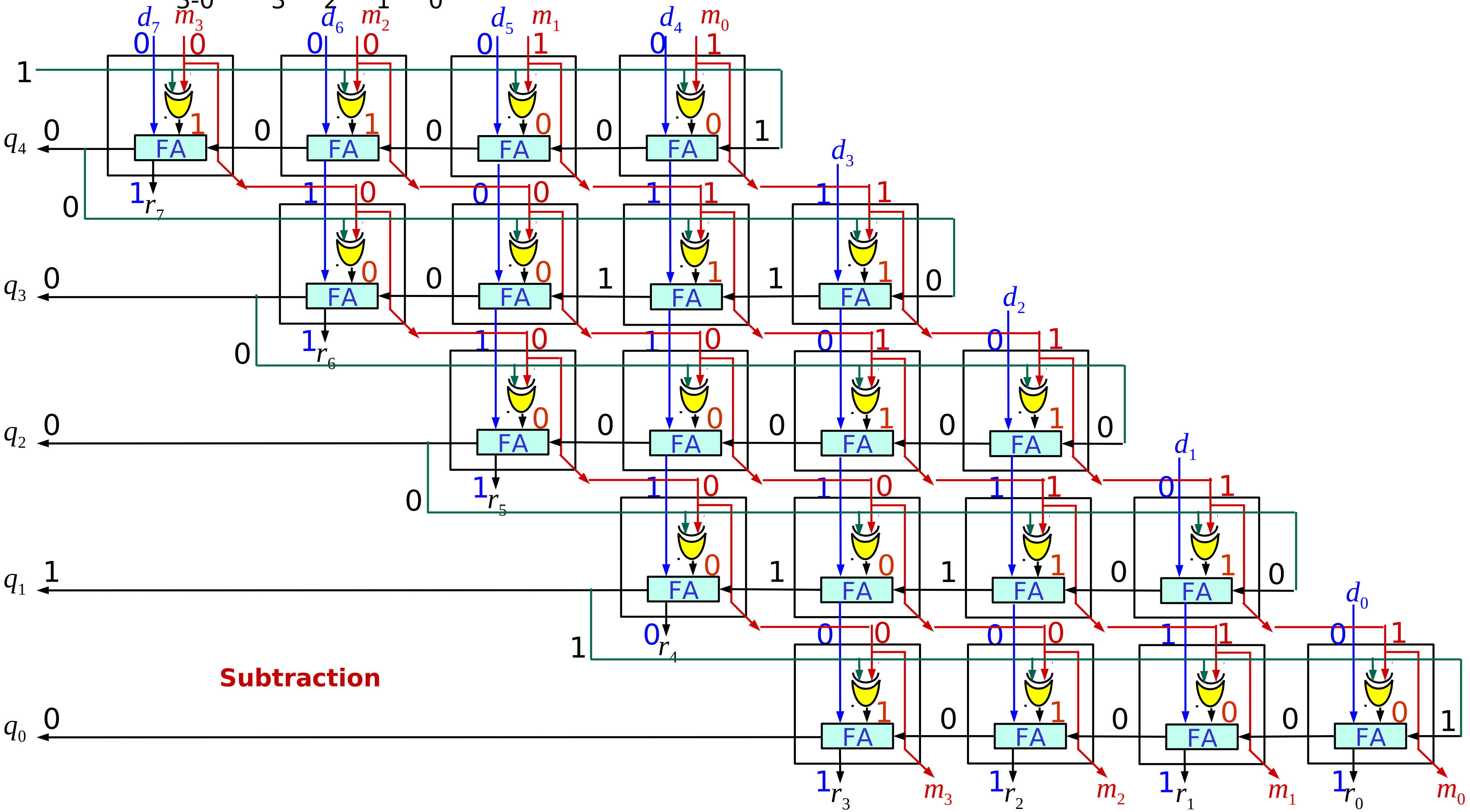
4-bit CA Division: Non-restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



4-bit CA Division: Non-restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ — Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ — Divisor: 0011

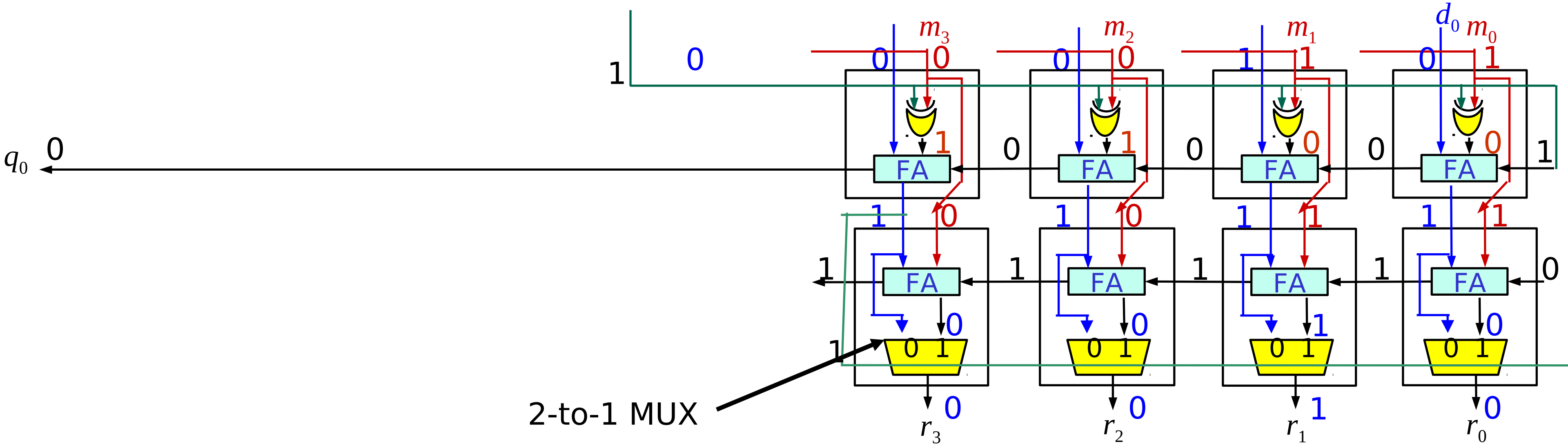


4-bit CA Division: Non-restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$

Dividend: 00001000

Divisor: 0011



CA Division Circuit – Restoring Algorithm

- A row of cells accepts the intermediate remainder and divisor as input
- Subtraction should be implemented i.e. 1-bit full subtractor to be implemented
- Depending upon the outcome of subtraction
  - the row output can be restored intermediate remainder input to the row or
  - the result of the subtraction

Basic Cell of a CA Division – Restoring Algorithm

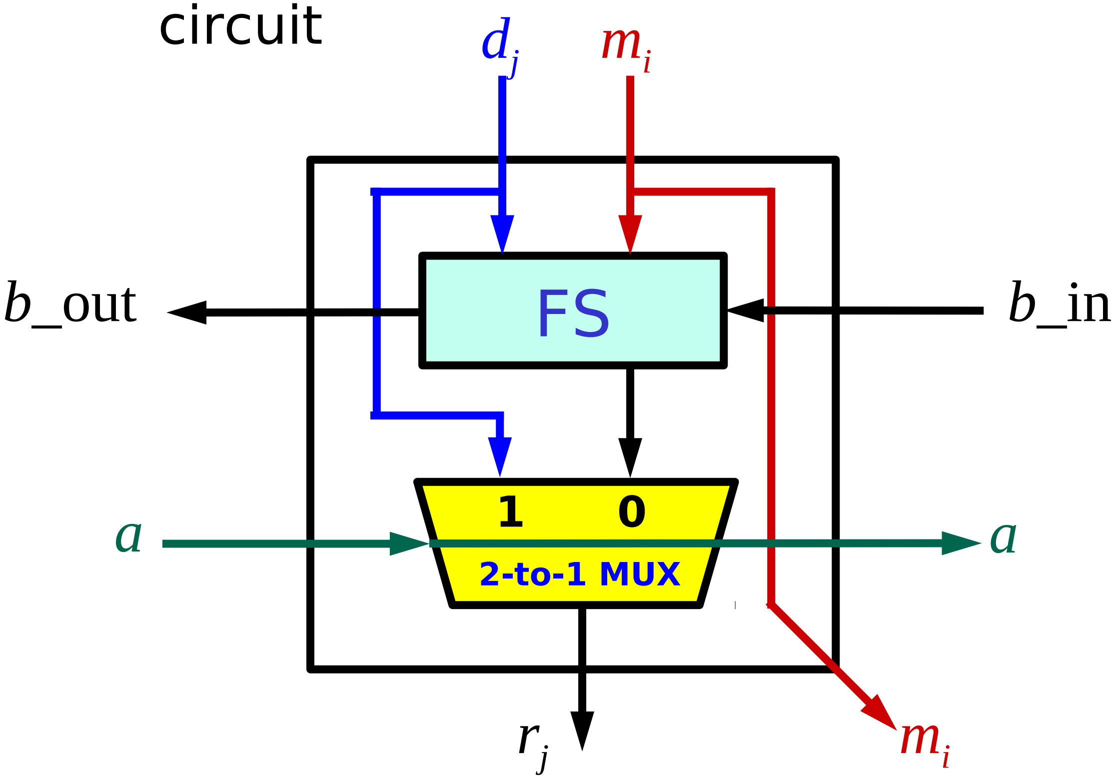
- Dividend is $2n$-bit length

  D = $d_{2n-1}\ d_{2n-2}\ \dots\ d_1\ d_0$

- Divisor is n-bit length

  M = $m_{n-1}\ m_{n-2}\ \dots\ m_1\ m_0$

- Basic cell is a full subtractor (FS) along with extra logic circuit



b_in : Barrow in

b_out : Barrow out

r_j : Remainder bit

when,

$a = 1,\ r_j = d_j$

$a = 0,\ r_j = d_j - m_i - b\_in$

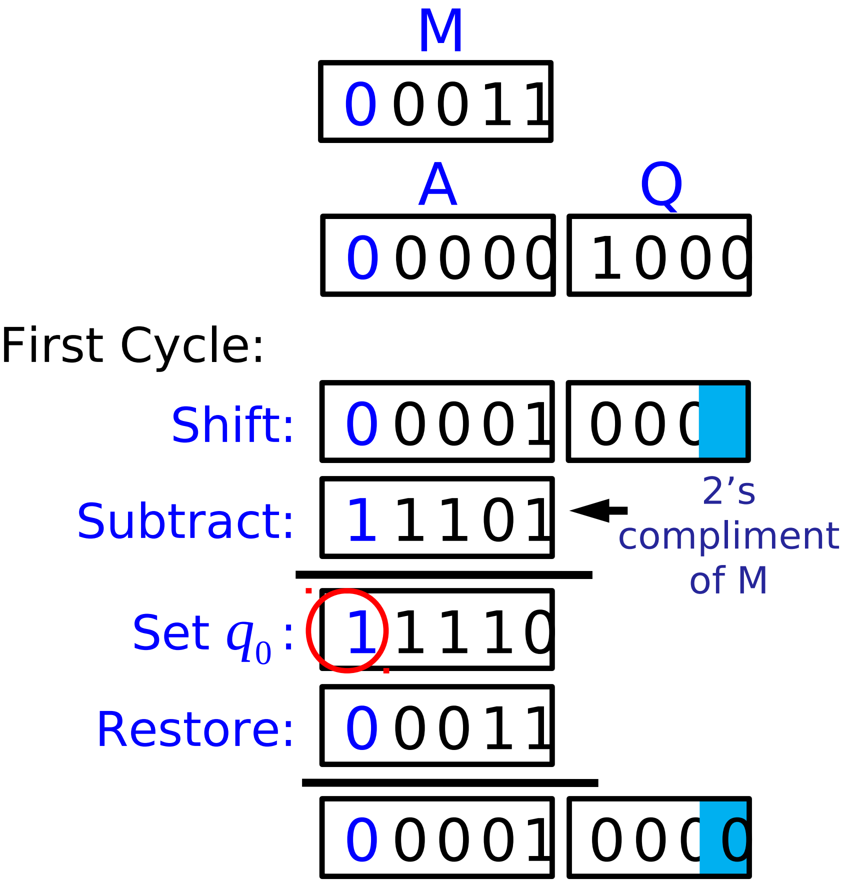
Restoring Division

- Algorithm:

Do the following steps n times

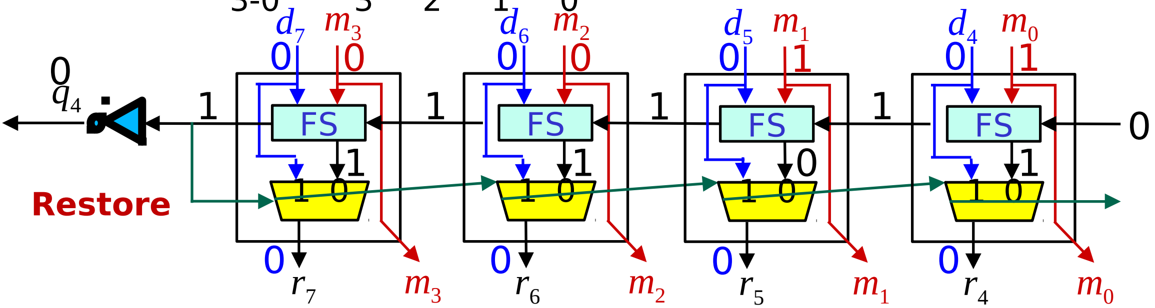
1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1
   1. Set q_0 to 0
   2. Add M back to A (i.e. restore A)
4. Otherwise, set q_0 to 1

- 4-bit division:



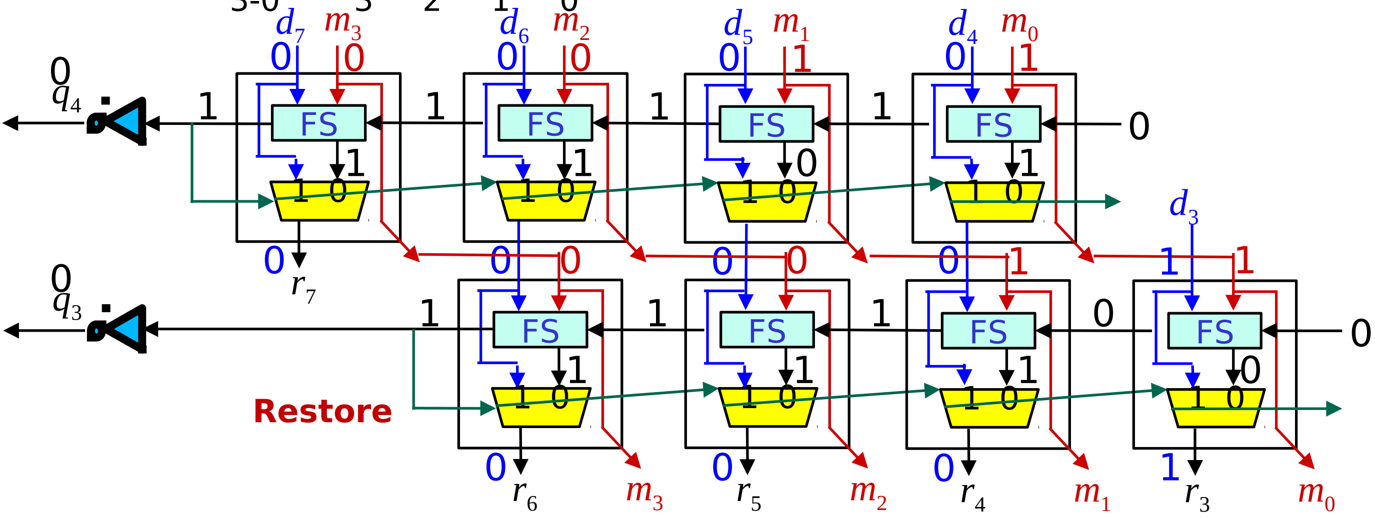
4-bit CA Division: Restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



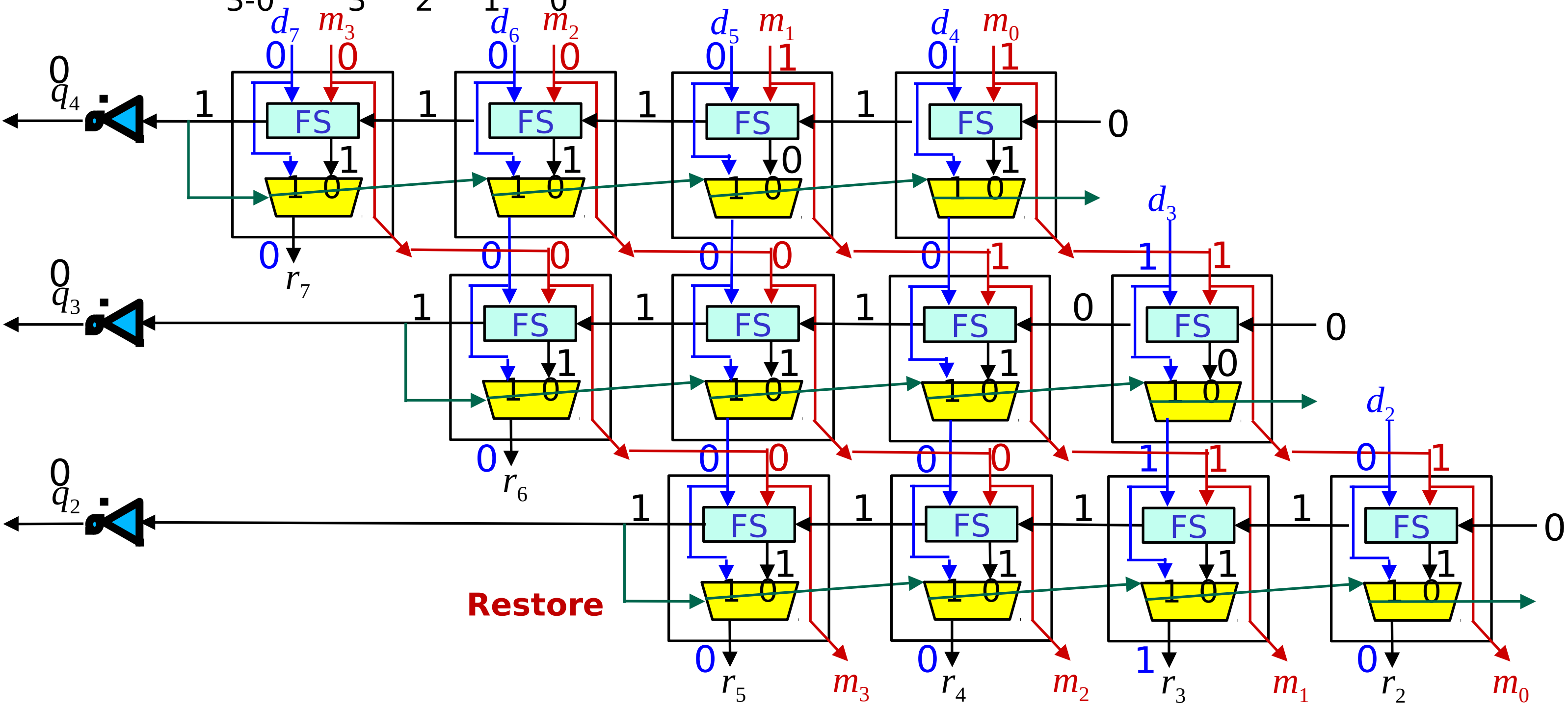
4-bit CA Division: Restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



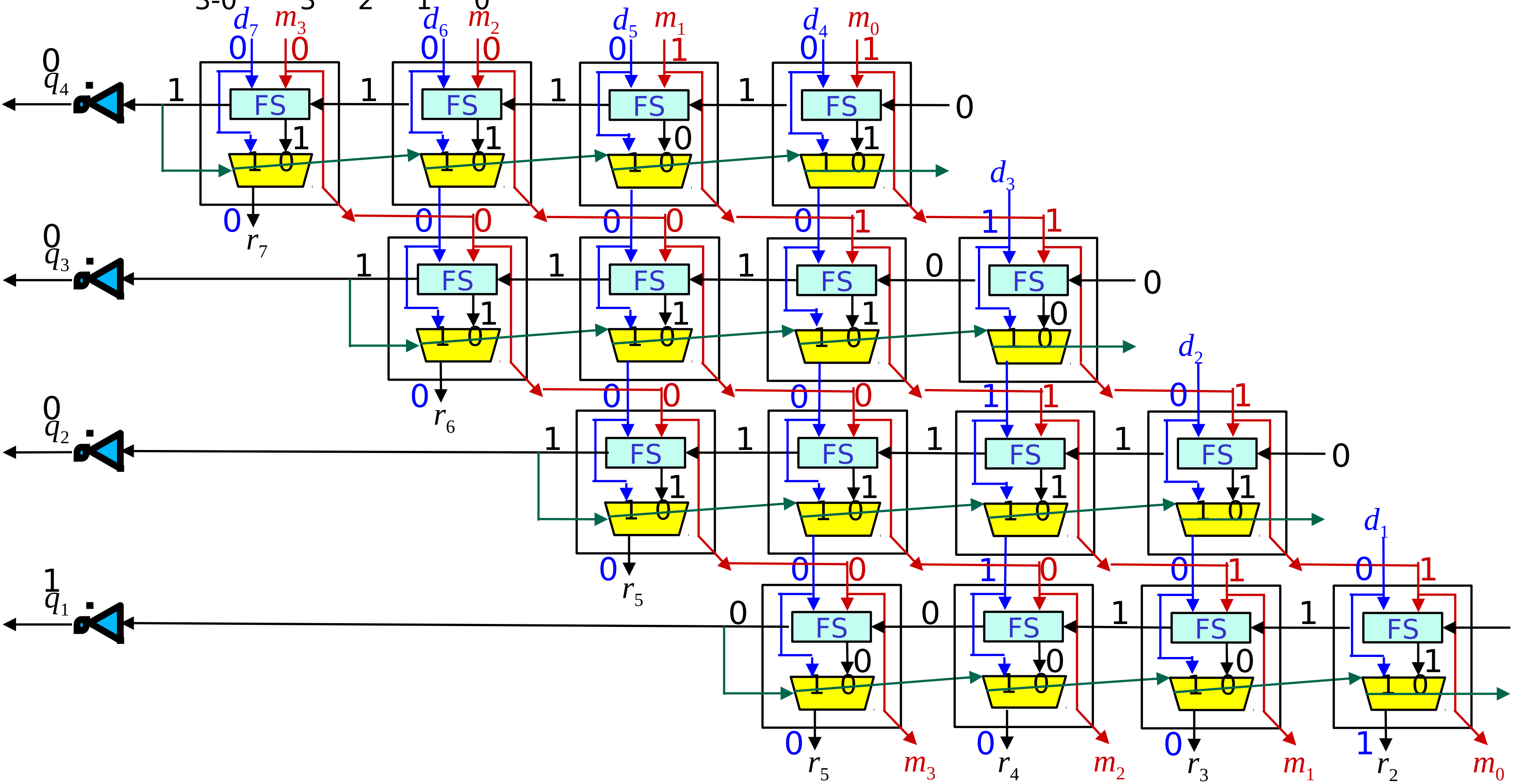
4-bit CA Division: Restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



4-bit CA Division: Restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011



4-bit CA Division: Restore Algorithm

- $D_{7\text{-}0}$: $d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0$ Dividend: 00001000
- $M_{3\text{-}0}$: $m_3\, m_2\, m_1\, m_0$ Divisor: 0011