

COA – MODULE 2

IEEE 754 Standard For
Floating Point
Representation

* Floating Point Representation :-

$$(5.625)_{10} \rightarrow (101.101)_2$$

\Downarrow \Downarrow

$0.\underline{5625} \times 10^1$

Mantissa Exponent

$0.\underline{101101} \times 2^3$

Mantissa Exponent

→ In computer, we used a fixed bit memory space for representation



* Normalization :-

→ Need of Normalization

$$(101.101)_2 \rightarrow (0.101101 \times 2^3)$$

$$(101.101)_2 \rightarrow (101101 \times 2^{-3})$$

$$(101.101)_2 \rightarrow (1.01101 \times 2^2)$$

∴ Various floating point representation is possible. So, Standardization is required.

This process called Normalization

Normalization

Explicit

→ Move radix point to the LHS of the most significant '1' in the bit sequence.

$$(101.101)_2 \rightarrow (0.101101 \times 2^3)$$

Implicit

→ Move the radix point to the RHS of the most significant '1' in the bit sequence

$$(101.101) \rightarrow (1.01101 \times 2^2)$$

* Biasing :-

→ To store a number, we required a fixed bit storing element like below



$$(101.101)_2 \rightarrow 101101 \times 2^{-3}$$

Now, question is how to represent those
-ve numbers → 2's complement?

→ exponent
can also
-ve & +ve
for different
representation

→ In, 2's compliment representation of n-bit number,
the range is $[-2^{n-1} \text{ to } (2^{n-1} - 1)]$.

→ For $n=4$, range of 2's comp. representation,

$$[-2^{4-1} \text{ to } (2^{4-1} - 1)] \Rightarrow [-8 \text{ to } 7]$$



-8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7

We need to convert it into unsigned representation

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Add 8
↓
excess
8

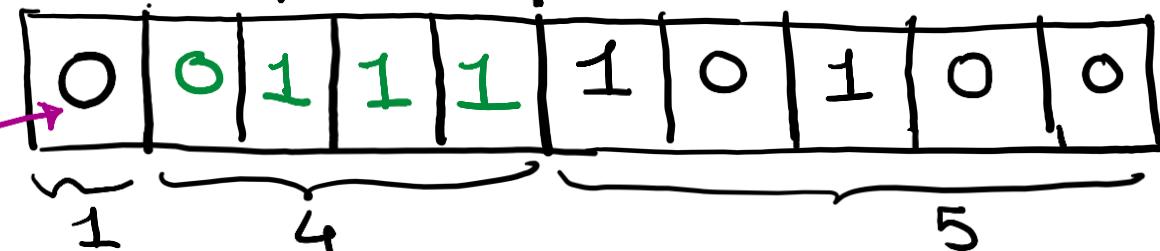
Ex:-

$$\therefore (0.0101)_2 \rightarrow 0.\underbrace{101}_{} \times 2$$

⇒ Explicit Representation

(10 bit) memory space.

S ← Exponent → | ← Mantissa →



$$ii) (\pm 01.101)_2 \rightarrow (1.01101 \times 2)$$

⇒ Implicit Representation

+2

10 ⇒ (1010)₂



← 10 bit memory space

→ To convert the memory stored number into
human readable form,

1. Explicit Normalization : $(-1)^S \times 0.M \times 2^{E-\text{bias}}$

2. Implicit Normalization : $(-1)^S \times 1.M \times 2^{E-\text{bias}}$

IEEE 754 Binary Floating Point Standard :-

Name	Common Name	Significant Bits (M+S)	Exponent Bits	Exponent Bias	S, M, E
Binary 16	Half Precision	11	5	15 (excess 15)	S → 1 M → 10 E → 5
Binary 32	Single Precision	24	8	127 (excess 127)	S → 1 M → 23 E → 8
Binary 64	Double Precision	53	11	1023 (excess 1023)	S → 1, M → 52 E → 11
Binary 128	Quadruple Precision	113	15	16383 (excess 16383)	S → 1, M → 112 E → 15
Binary 256	Octuple Precision	237	19	262143 (excess 262143)	S → 1, M → 236 E → 19

IEEE 754 - Single Precision (Binary 32)

S(1bit)	E(8bit)	M(23bit)	Representation
0 or 1	0000 0000 E = (0) ₁₀	0000 00...0000 (M = 0) ₁₀	± 0
0 or 1	1111 1111 E = (255) ₁₀	0000 00...0 (M = 0) ₁₀	$\pm \infty$
0 or 1	$1 \leq E \leq 254$	M = x x x x ... x	Implicit Normalized Form $(-1)^S \times 1.M \times 2^{E-127}$
0 or 1	E = (0) ₁₀	M ≠ 0	Denormalized or Fractional Form :- $(-1)^S \times 0.M \times 2^{-126}$
0 or 1	E = (255) ₁₀	M ≠ 0	NAN (not a number) e.g:- % ; overflow.

IEEE 754 - Double Precision (Binary 32)

S(1bit)	E(11bit)	M(52bit)	Representation
0 or 1	00000000000 E = (0) ₁₀	000000000...0 M = (0) ₁₀	± 0
0 or 1	11111111111 E = (255) ₁₀	000000000...0 M = (0) ₁₀	$\pm \infty$
0 or 1	$1 \leq E \leq 2046$	M = xxxx...x	Implicit Normalized Form $(-1)^S \times 1.M \times 2^{E-1023}$
0 or 1	E = (0) ₁₀	M ≠ 0	Denormalized or Fractional Form :- $(-1)^S \times 0.M \times 2^{-1022}$
0 or 1	E = (2047) ₁₀	M ≠ 0	NAN (not a number) e.g:- % ; overflow.

* Suppose a 16 bit register of the following format is used for storing binary point numbers :-

[Mantissa (M) is denoted using normalized sign-magnitude fraction. Exponent (E) is expressed in excess 64 form]

1. How many bits used for the fractional mantissa.
2. What should be the expression for the decimal value?
3. What is largest number (in Base 10) that can be represented using this register.
4. What should be the bit patterns for $(7.5)_{10}$ & $(-16.125)_{10}$?
5. Determine the difference between the smallest +ve number and 2nd smallest positive number in the presentation.

1) Excess 64 is used,

$$\therefore \text{Maximum -ve value} = -64 = -2^6 = -2^{-7-1}$$

↳ 2's Comp. lowest range.

$$\therefore \text{Exponent bits (E)} = 7$$

$$[-2^{n-1} \text{ to } + (2^{n-1} - 1)]$$

$$\therefore \text{Sign bit (s)} = 1$$

Max -ve value

$$\therefore \text{Mantissa (M)} = 8 \text{ bit}$$



2) As nothing is mentioned, so we assume implicit normalization because it gives better precision over explicit normalization.

∴ Implicit Normalization Format (Human Readable)

$$(-1)^S \times 1.M \times 2^{E-\text{bias}}$$

∴ For excess bias -64

$$(-1)^S \times 1.M \times 2^{E-64}$$

Expression for
the decimal
value.

Q) The largest number (in base 10) that can be represented using this register :-



Max value of Mantissa
 $(1111111)_2$

Max value of exponent

$(1111111)_2$

For max value
sign bit must be
+ve; $S=0$

$$\begin{aligned} & (2^7 - 1)_{10} = (128 - 1)_{10} \\ & = (127)_{10} \end{aligned}$$

→ Implicit Normalization,

$$\text{Value} \Rightarrow (-1) \times 1.\underbrace{11111111}_{m} \times 2^{\text{E}}$$

$$\Rightarrow (1.\underbrace{11111111}_{\text{Mantissa}} \times 2^{63})_2$$

$$\Rightarrow (11111111 \times 2^{55})_2 \quad (\text{Shift radix point RS})$$

$$\Rightarrow (11111111 \underbrace{000000}_{\text{Fraction}} \times 2^{50})_2$$

Converting into Hexa.

Face values of Hexadecimal

$$\begin{array}{c} 0011 \\ \downarrow \\ 3 \\ \hline F(15) \quad E(14) \quad 0 \\ \hline (3FE0)_{16} \end{array}$$

$\downarrow \text{E}$
127 - 64
 $\downarrow \text{Bias}$

$$2^{10} \approx 10^3$$

$$2^{50} \approx 10^{15}$$

∴ Largest decimal number can store inside
of 16 bit register .

$$\text{Value} \Rightarrow (3 \times 16^3 + 15 \times 16^2 + 14 \times 16^1 + 0 \times 16^0) \times 10^{15}$$
$$\Rightarrow (16352 \times 10^{15})_{10} \quad \dots \text{(Ans)}$$

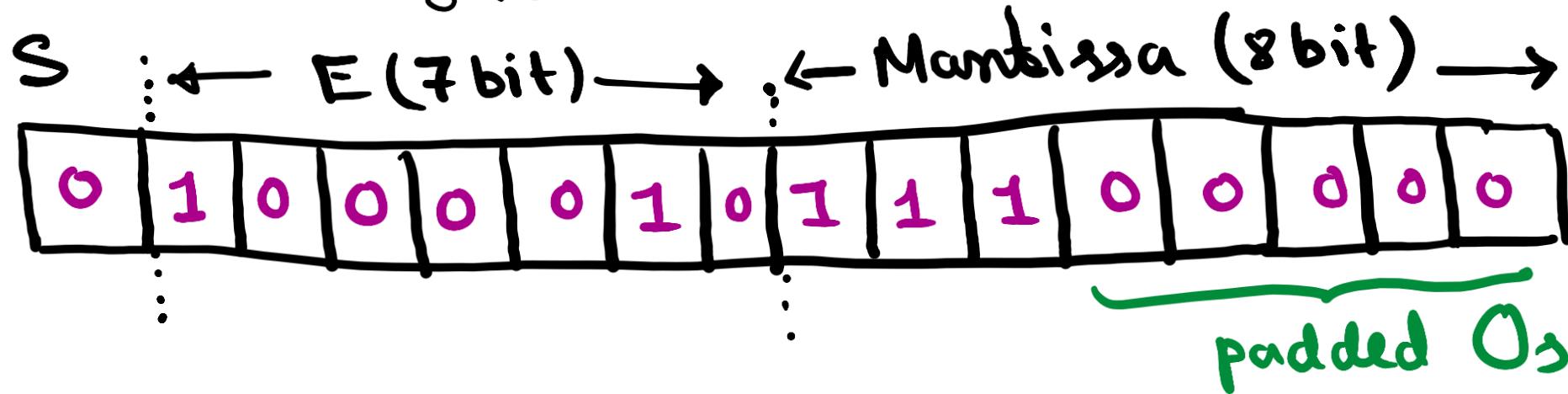
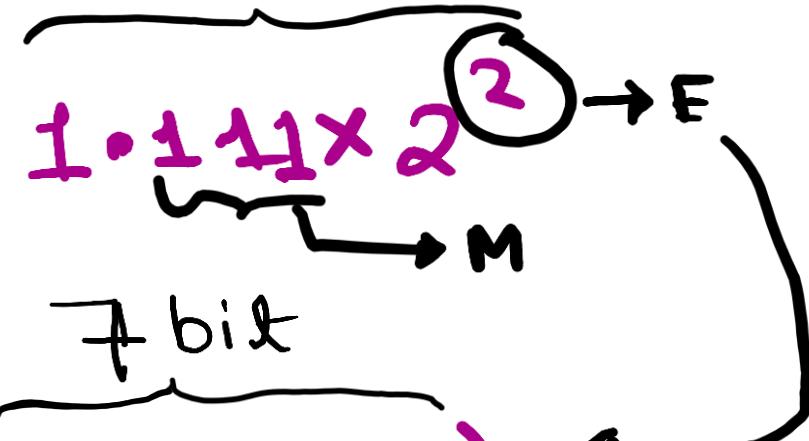
4:
a) Bit Pattern of $(7.5)_{10}$:-

$$(7.5)_{10} \rightarrow (111.1)_2 = 1 \cdot \underbrace{111}_{\text{M}} \times 2^{\circled{2} \rightarrow E}$$

S = 0 \Rightarrow +ve no.

$$E = 2 + \underbrace{64}_{\substack{\text{excess 64} \\ \text{given}}} = (66)_{10} = (1000010)_2$$

Implicit form



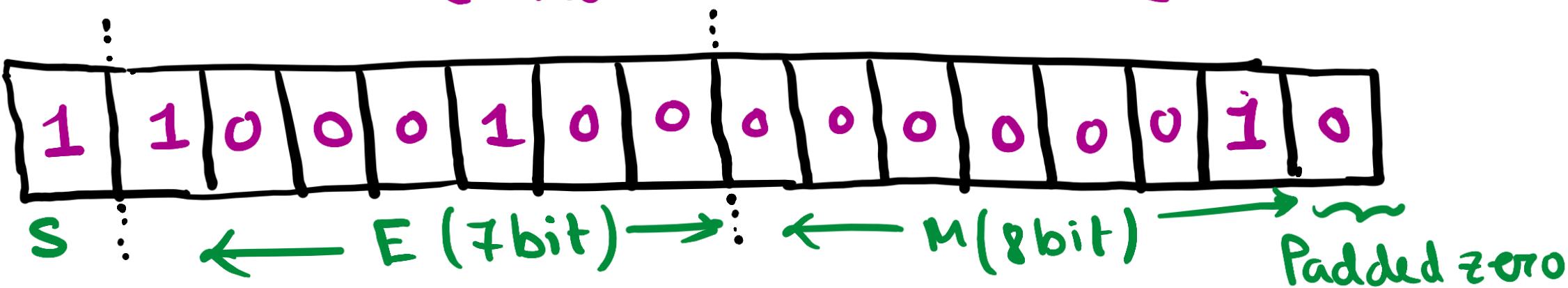
4. b) Bit pattern of $(-16.125)_{10}$:-

$$(-16.125)_{10} \rightarrow (10000.001)_2$$

$$\Rightarrow 1.\underbrace{0000001}_M \times 2^4$$

S = 1 \Rightarrow -ve no.

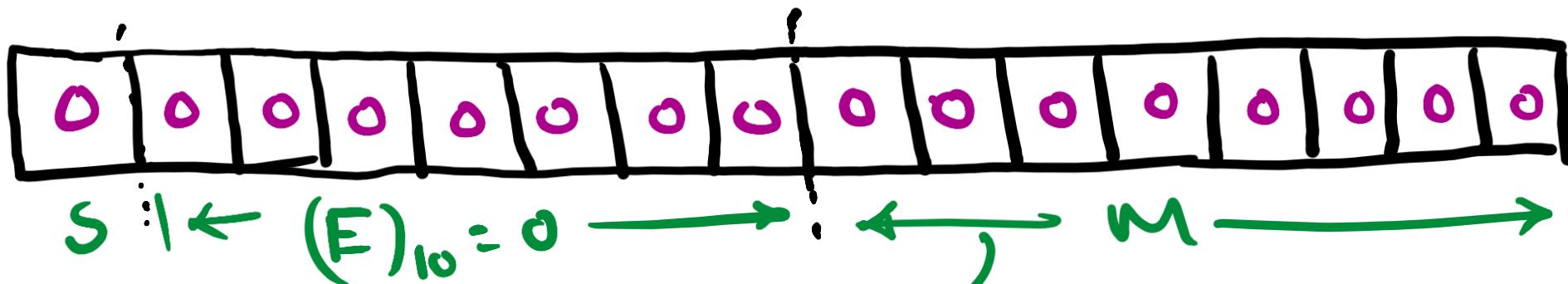
$$E = 4 + 64 = (68)_{10} = (1000100)_2$$



5. Smallest (+ve) Number:-

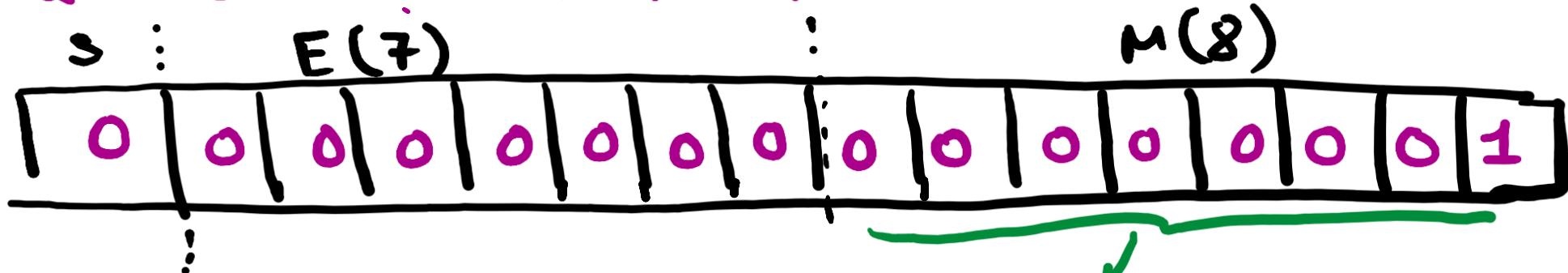
→ Let take a implicit normalization form with excess 64 bias,

$$(-1)^s \times 1.M \times 2^{E-64}$$



Smallest Value $\Rightarrow (-1)^0 \times 1.\underbrace{00000000}_{M} \times 2^{0-64} = 1.0 \times 10^{-64}$

→ 2nd Smallest Number :-



2nd Smallest value $\Rightarrow (-1)^0 \times 1. \underbrace{00000001}_{\text{fraction}} \times 2^{0-64}$
 $\Rightarrow 1.00000001 \times 2^{-64}$

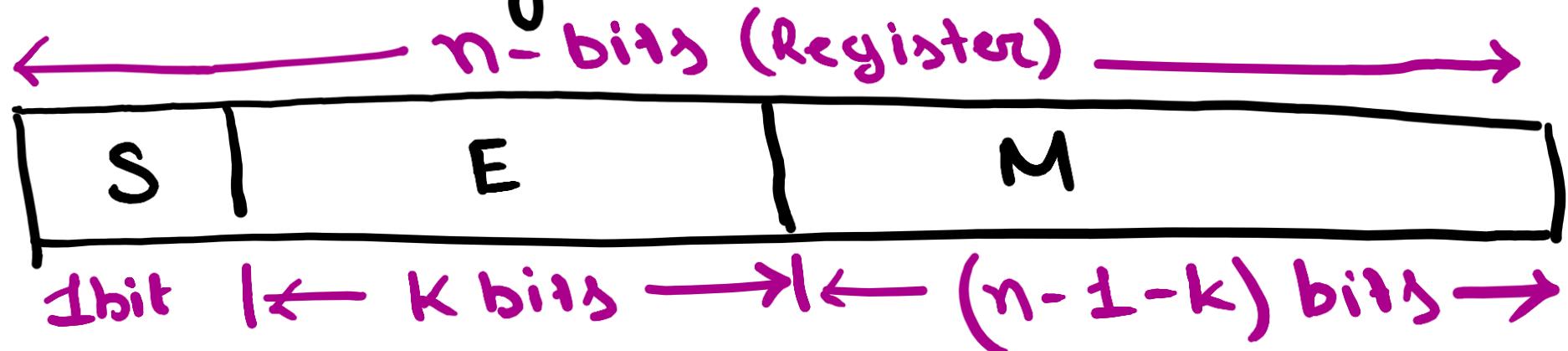
Smallest +ve no. :- $1.00000000 \times 2^{-64}$

2nd Smallest +ve no. :- $1.00000001 \times 2^{-64}$

Difference :- $0.00000001 \times 2^{-64}$

Shift Radix Point $\Rightarrow 1 \times 2^{-8} \times 2^{-64} \Rightarrow 1 \times 2^{-72} \dots (R)$

Note:: Floating Point Numbers (Generally)



Biased exponent : $0 \leq E \leq 2^{k-1}$

Bias : $(2^k - 1)$

\therefore True Exponent = (Biased Exponent - Bias)

Design of Adder

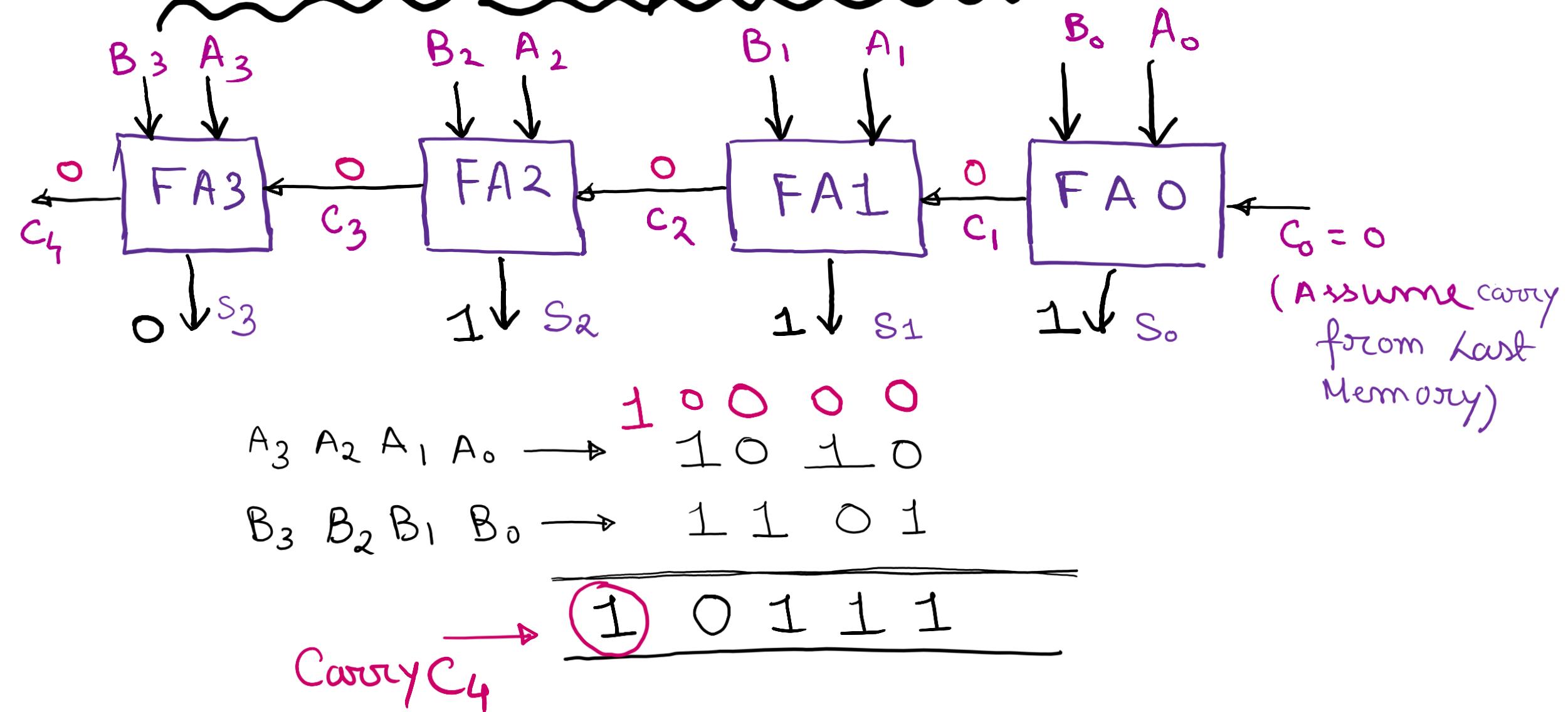
→ Ripple Carry Adder

→ Carry Look Ahead Adder

Note:- Both are comes under

“ Parallel Adder”

1. Ripple Carry Adder:-



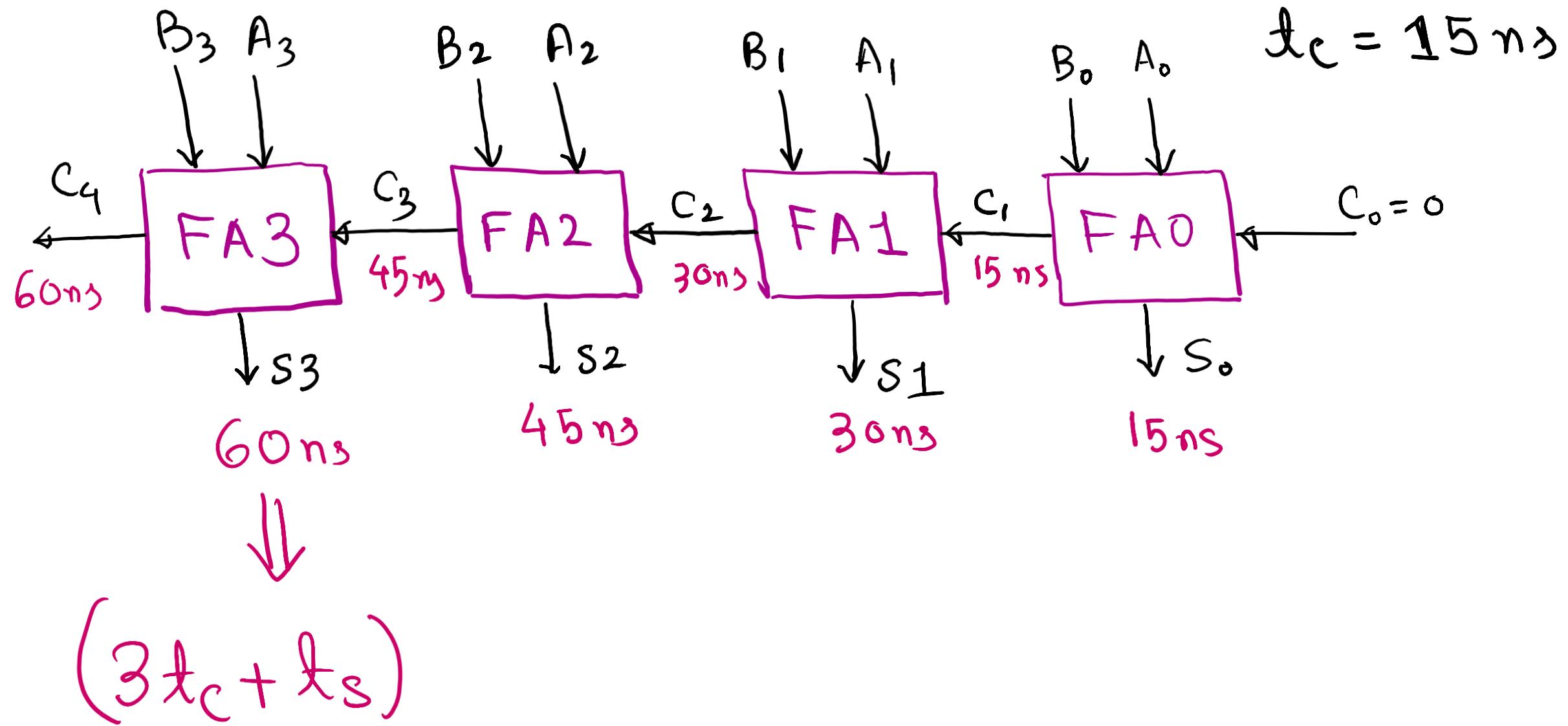
$t_p \Rightarrow$ Propagation Delay

\Rightarrow Time required to get output in
a stage.

$\therefore t_p$ can be divided into two parts

- $\rightarrow t_s$: Propagation delay for Sum output.
- $\rightarrow t_c$: Propagation delay for Carry output.

→ Let consider a 4 bit adder with $t_s = 15 \text{ ns}$



\therefore if $t_s \neq t_c$, then for n-bit adder

\therefore Total Propagation Delay for
Sum output = $(n-1)t_c + t_s$

\therefore Total Propagation Delay for = nt_c
Carry output

Note:-

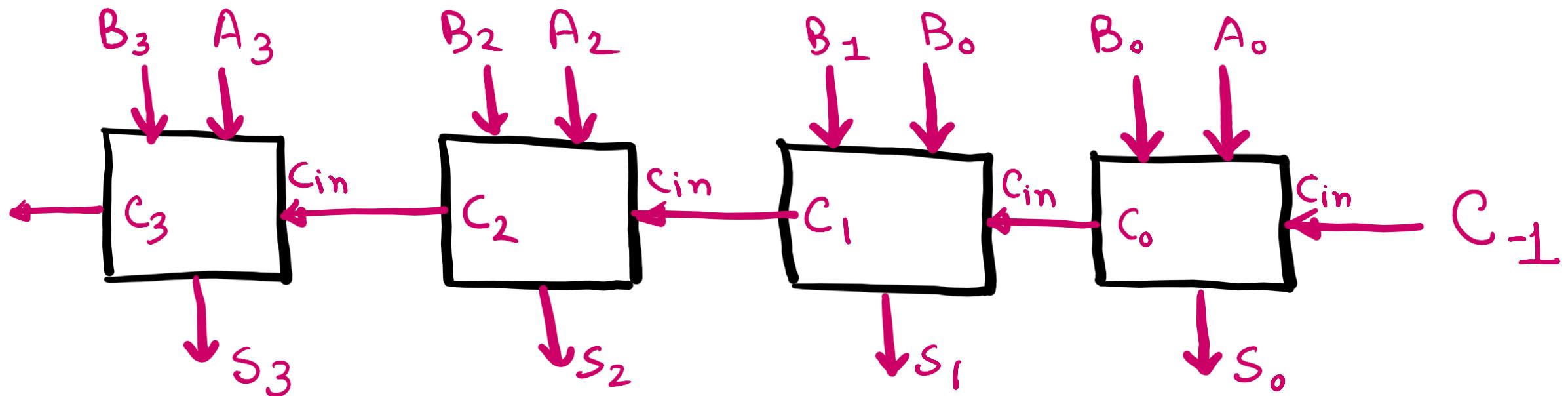
→ The carry propagation delay of the ripple carry adder is the main disadvantage. ⇒ To resolve it



We go for ⇒ "Look Ahead Carry Adder"

≡ Carry Look Ahead Adder (CLA) :-

- Let predict carry of next stage before it will generate
- Let consider a 4-bit parallel adder :-



A_i	B_i	C_{in}	C_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$A_i \oplus B_i = 1$$

or

$$C_{in} = 1$$

when

$$\begin{cases} A_i = 1 \\ B_i = 1 \end{cases} \rightarrow C_i = 1$$

→ Logic for Curry predictor,

$$C_i = \underbrace{A_i B_i}_{\text{Carry Generator } (G_i)} + \underbrace{(A \oplus B) \cdot C_{in}}_{\text{Carry Propagator } (P_i)}$$

$$\therefore C_i = G_i + P_i C_{i-1}$$

where
 $i = 0, 1, 2, \dots$

$$\xrightarrow{i=0} C_0 = G_0 + P_0 \underbrace{C_{-1}}_{\dots \dots (i)}$$

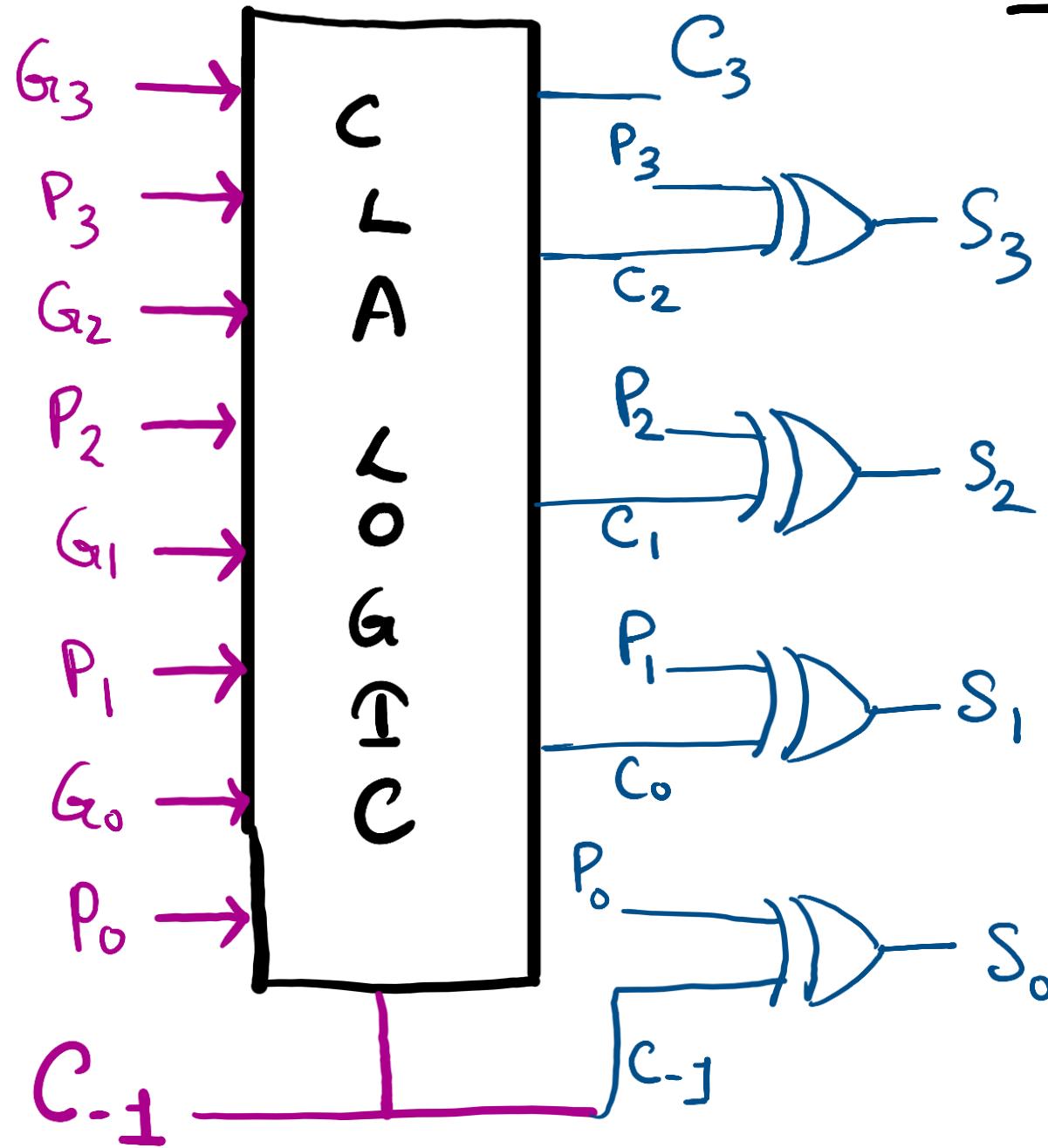
$$\xrightarrow{i=1} C_1 = G_1 + P_1 C_0$$

$$C_1 = G_1 + P_1 G_0 + P_1 P_0 \underbrace{C_{-1}}$$

$$\xrightarrow{i=2} C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 \underbrace{C_{-1}}$$

$$\xrightarrow{i=3} C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 \underbrace{C_{-1}}$$

* if we know C_{-1} (initial value), then we can easily predict C_1, C_2 in advance.



→ 4 bit CLA, carry predictor

$$G_0 = A_0 B_0, G_1 = A_1 B_1$$

$$G_2 = A_2 B_2, G_3 = A_3 B_3$$

$$P_0 = A_0 \oplus B_0, P_1 = A_1 \oplus B_1$$

$$P_2 = A_2 \oplus B_2, P_3 = A_3 \oplus B_3$$

$$S_3 = P_3 \oplus C_2$$

$$S_2 = P_2 \oplus C_1$$

$$S_1 = P_1 \oplus C_0$$

$$S_0 = P_0 \oplus C_{-1}$$

Design of Multiplier

- Shift and add method
- Booth Multiplier
- Carry Save Multiplier (Home Assignment)

* Shift and Add Method:-

→ Pen and Paper Method

e.g:- $(14) \times (10) = 140 \leftarrow \text{decimal}$

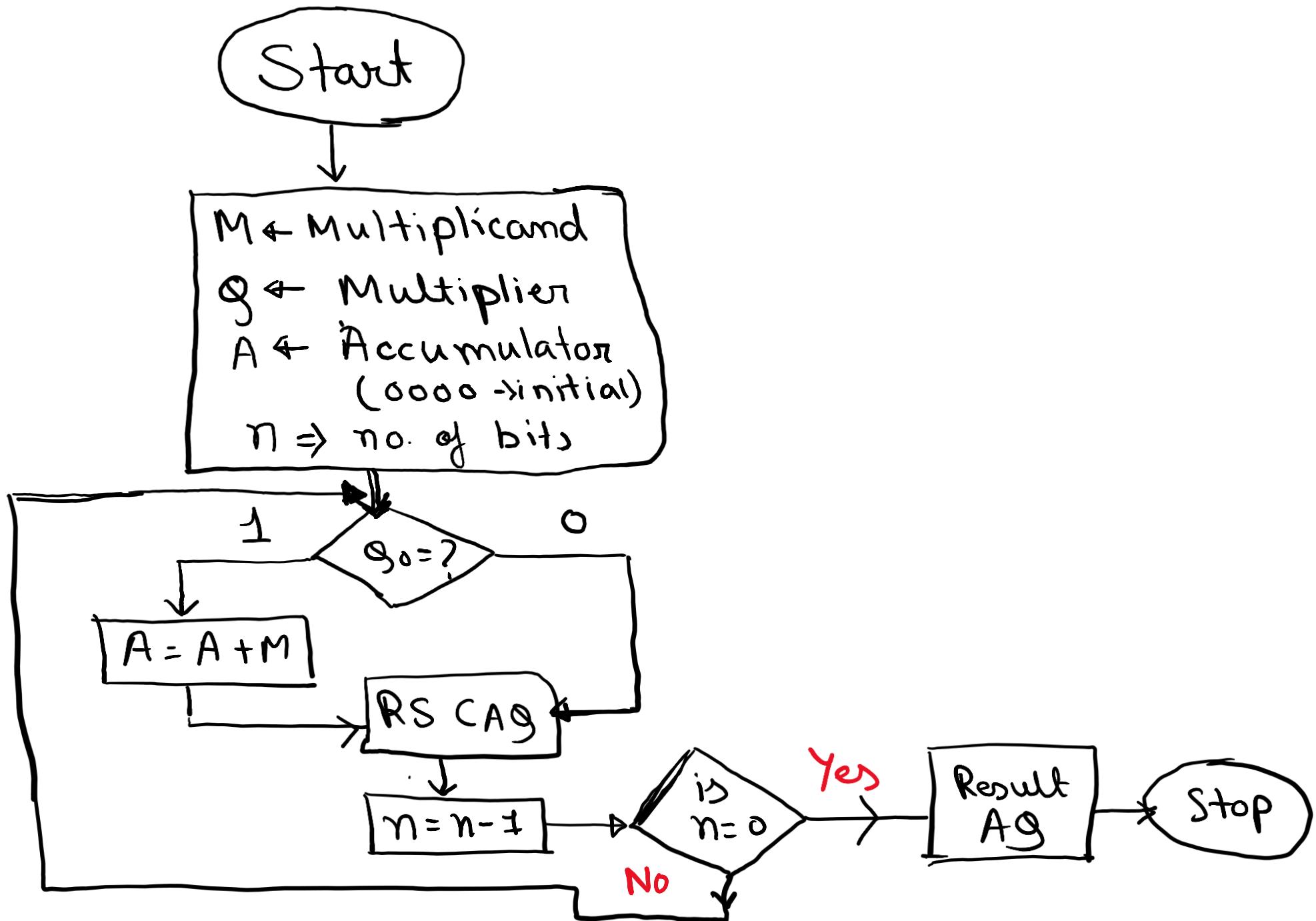
$14 \rightarrow 1110 \Rightarrow (M:- \text{Multiplicand})$

$10 \rightarrow 1010 \Rightarrow (Q:- \text{Multiplier})$

$$\begin{array}{r} & 1110 \\ \times & 1010 \\ \hline & 0000 \\ 1110 \\ 0000 \\ \hline 1110 \end{array} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{Partial Product} \quad (10001100)_2 \longrightarrow (140)_{10}$$

Let, $\underbrace{11}_{2} \times \underbrace{13}_{2} = (143)_{10} \rightarrow$ Final Ans.

$$(1011)_2 \times (1101)_2 = (?)_2$$
$$= (1000\ 1111)_2$$



n	M	C	A	Q	Operation
4	1011	0	0000	$q_3 q_2 q_1 q_0$ 1101	Initialization
		Carry of $A+M$	1011 0101 1110	1101	If $q_0=1$, then $A = A + M$ Shift Right CAG
3	1011	0	0010	1111	If $q_0=0$, then Shift Right CAG
2	1011	0	1101 0110 1111	1111	If $q_0=1$, then $A = A + M$ Shift Right CAG
1	1011	1	0001 1000	1111	$q_0=1$, then $A = A + M$ Shift Right CAG

$$A \otimes g = \left(\begin{array}{cc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \downarrow & & & & & & & \\ = & (128 + 15)_{10} \end{array} \right)_2$$

$$A \otimes g = (143)_{10}$$

$M \rightarrow 4\text{bit or } n\text{-bit}$

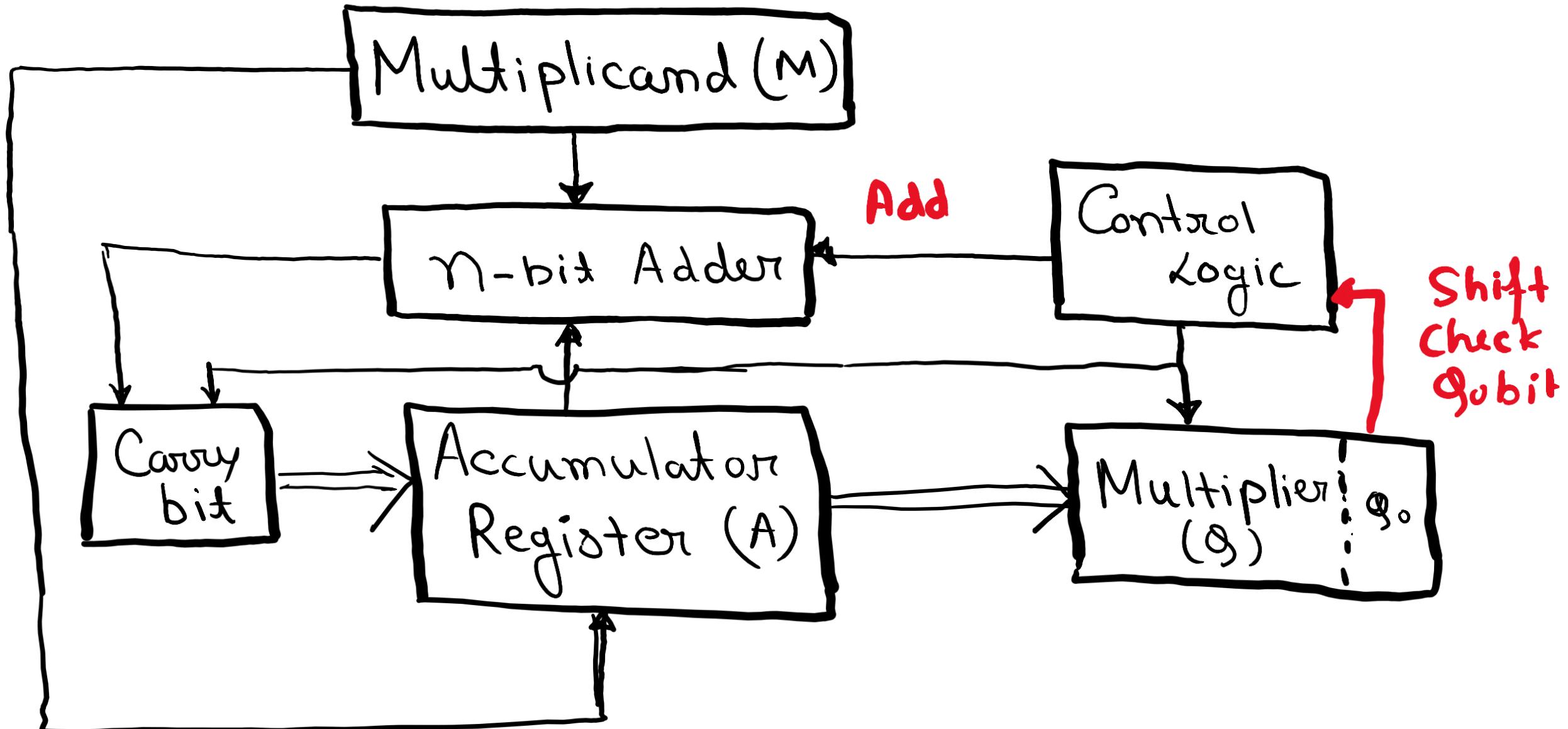
$g \rightarrow 4\text{bit or } n\text{-bit}$

$A \rightarrow 4\text{bit or } n\text{-bit (Accumulator)}$

- No. of Addition Operation = No. of 1's present in Q.
- No. of Shift Operation = No. of bits in MEG Perform

Here Addition = 3 & Shift = 4

Hardware:-



Booth's Algorithm (Signed Multiplication)

e.g:-

$$(-7)_{10} \times (+3)_{10} = (-21)_{10}$$

\downarrow \downarrow \swarrow Product

Multiplicand (M) Multiplier (Q) $\rightarrow (0_4 0_3 0_2 0_1)_2$

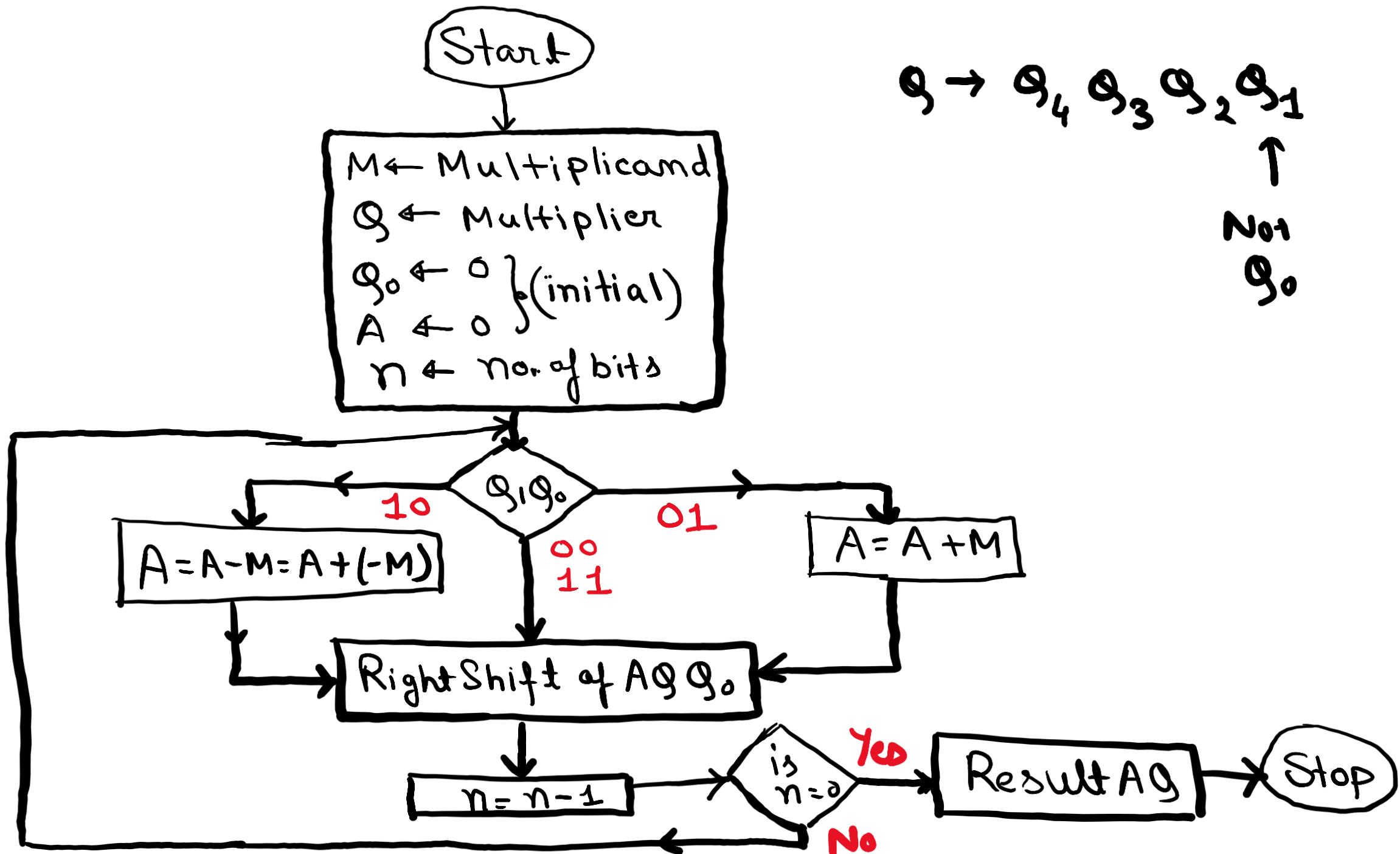
Now,

$$M = (-7)_{10} \rightarrow -M = (7)_{10} = (0111)_2$$

\downarrow

2's Complement form

$$M = (-7)_{10} = (1001)_2$$



n	A	Q_1	Q_0	Operation
4	0000	$Q_4 Q_3 Q_2 Q_1$ 0 0 1 1	0	Initialization
	0111	0011	0	$AS(Q_1 Q_0) \rightarrow (10); A = A - M = A + (-M)$
	0011	1001	1	Right Shift (RS) $\rightarrow A Q Q_0$
3	0001	1100	1	$AS(Q_1 Q_0) \rightarrow (11) \Rightarrow RS \rightarrow A Q Q_0$
2	1010	1100	1	$AS(Q_1 Q_0) \rightarrow (01) \Rightarrow A = A + M$ $0001 + 1001$
	1101	0110	0	RS $\rightarrow A Q Q_0$
1	1110	1011	0	$AS(Q_1 Q_0) \rightarrow (00) \Rightarrow RS \rightarrow A Q Q_0$
0	1110	1011	0	Stop

$AQ \rightarrow \underline{1}110 \quad 1011$

→ if MSB is '1' → it is a -ve number → 2's Complement is required to get final result

→ if MSB is '0' → it is a +ve number → AQ is final result

$AQ \rightarrow \underline{1}110 \quad 1011$

$\cdot \begin{matrix} 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{matrix}$

$(0001 \quad 0101)_2 \rightarrow (-21)_2$ for MSB is 1

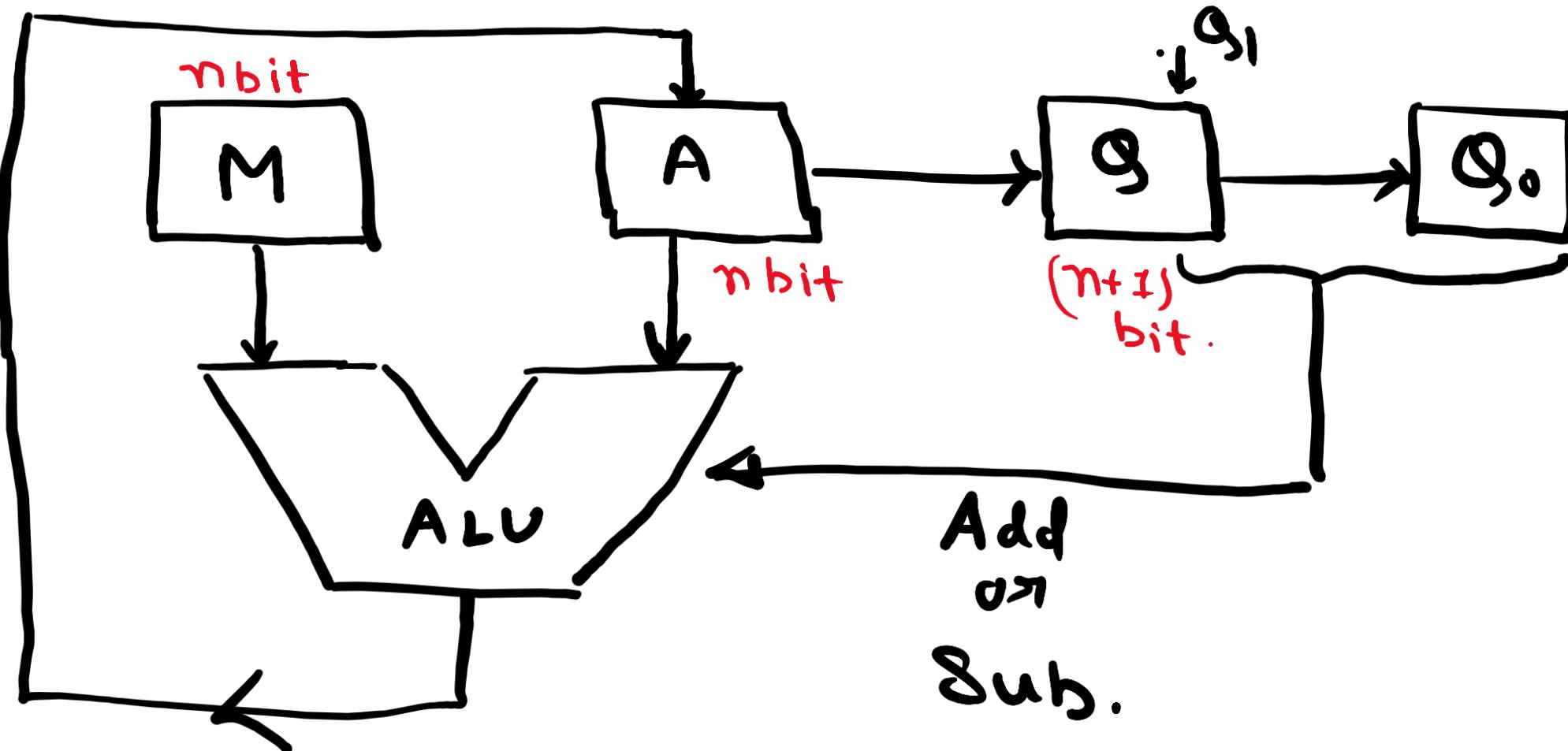
Using Booth's Algorithm for Multiplication,
reduce no. of addition operation
Compare to Add & Shift method.

Here, Addition

Operation Required = 2

2 Shift operation Required = 4

Hardware implementation



Design of Divisor

Slow Algorithms

[Iterative Operation
: Subtraction]

→ Restoring Division

→ Non-Restoring Division

Fast Algorithms

[Iterative Operation
: Multiplication]

→ Division by series expansion

→ Newton - Raphson Method

* The Concept of Restoring Division:- (Unsigned no.)

- Dividend (d) = 4537
- Divisor (v) : 3
- Quotient (q) : 1512
- Remainder (r) : 1

$$\begin{array}{r} 1512 \\ \hline 3 \overline{)4537} \\ -3 \quad | \\ \hline 15 \quad | \\ \quad 15 \downarrow \\ \hline \quad 3 \quad | \\ \quad 3 \downarrow \\ \hline \quad 7 \\ \hline \quad 6 \quad | \\ \quad 1 \end{array}$$

$$q_v \rightarrow \begin{matrix} q_3 & q_2 & q_1 & q_0 \\ 1 & 5 & 1 & 2 \end{matrix}$$

$$V \rightarrow 3 \left| \begin{array}{cccc} 4 & 5 & 3 & 7 \\ -3 & 0 & 0 & 0 \\ \hline 1 & 5 & 3 & 7 \end{array} \right. \Rightarrow V \times q_3 \times 10^3$$

$$\pi(3) \rightarrow \begin{array}{cccc} 1 & 5 & 0 & 0 \\ \hline 1 & 5 & 0 & 0 \end{array} \Rightarrow V \times q_2 \times 10^2$$

$$\pi(2) \rightarrow \begin{array}{cccc} 0 & 0 & 3 & 7 \\ 0 & 0 & 3 & 0 \\ \hline 0 & 0 & 0 & 7 \end{array} \Rightarrow V \times q_1 \times 10^1$$

$$\pi(1) \rightarrow \begin{array}{cccc} 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \end{array} \Rightarrow V \times q_0 \times 10^0$$

$$1^{\text{st}} \text{ iteration} : 4537 - 3 \times 1 \times 10^3 = 1537$$

$$2^{\text{nd}} \text{ iteration} : 1537 - 3 \times 5 \times 10^2 = 0037$$

$$3^{\text{rd}} \text{ iteration} : 0037 - 3 \times 1 \times 10^1 = 0007$$

$$4^{\text{th}} \text{ iteration} : 0007 - 3 \times 2 \times 10^0 = 0001$$

∴ Reminder function for i^{th} iteration,

$$r(i) = r(i+1) - q_i \times v \times 10^i$$

→ Restoring division process,

dividend (d) = 4537, Divisor (v) = 3, Quotient =
(q_1)

$q_3 \ q_2 \ q_1 \ q_0$

$$4537 - 3 \times 10^3 = 1537; \ q_3 = 1$$

$$1537 - 3 \times 10^3 = \boxed{-1463}; \ q_3 = 2$$

$$-1463 + 3 \times 10^3 = \boxed{1537}; \ q_3 = 1$$

$$1537 - 3 \times 10^2 = 1237; \ q_2 = 1 \Rightarrow$$

$$1237 - 3 \times 10^2 = 937; \ q_2 = 2$$

$$937 - 3 \times 10^2 = 637; \ q_2 = 3$$

-ve remainder
not possible,
so restore the
value by adding

q_i also implies
no. of subtraction
is performed

$$637 - 3 \times 10^2 = 337, q_2 = 4$$

-ve remainder
is not possible.
so restore the value

$$337 - 3 \times 10^2 = 37, q_2 = 5$$

$$37 - 3 \times 10^2 = \boxed{-263}, q_2 = 6$$

$$-263 + 3 \times 10^2 = 37, \boxed{q_2 = 5} \rightarrow \text{Restoring}$$

$$37 - 3 \times 10^1 = 7, q_1 = 1$$

$$7 - 3 \times 10^1 = \boxed{-23}, q_1 = 2$$

$$-23 + 3 \times 10^1 = 7, \boxed{q_1 = 1} \rightarrow \text{Restoring}$$

$$7 - 3 \times 10^0 = 4, q_0 = 1$$

$$4 - 3 \times 10^0 = 1, q_0 = 2$$

$$1 - 3 \times 10^0 = \boxed{-2}, q_0 = 3$$

$$-2 + 3 \times 10^0 = 1, q_0 = 2$$

Restoring

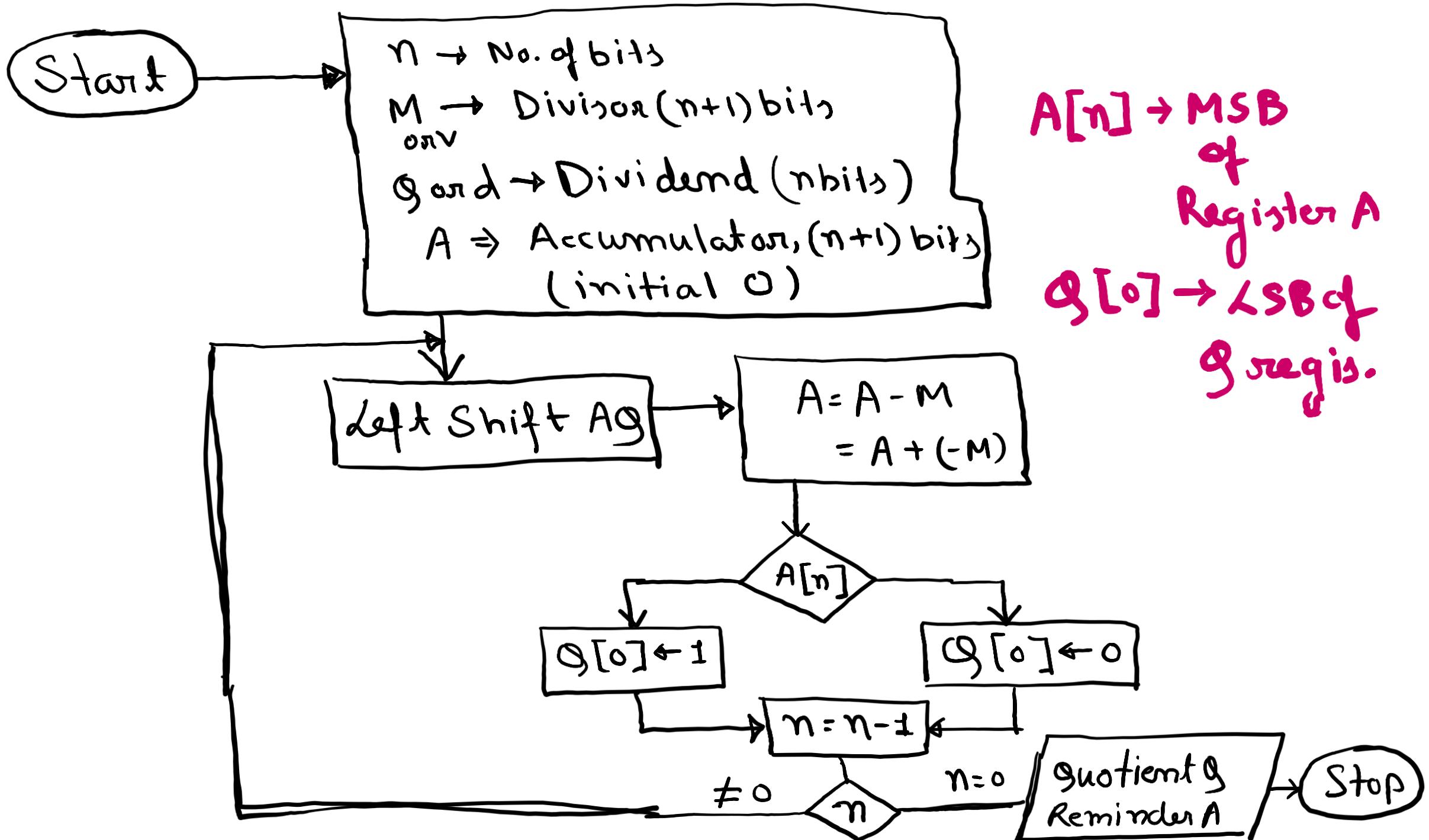
Final remainder.

$$d = 4537$$

$$v = 3$$

$$\pi = 1$$

$$q_3 q_2 q_1 q_0 \\ Q = 1512$$



Example :- Dividend (Q or d) = $(11)_{10}$
Divisor (M or v) = $(3)_{10} \rightarrow$

$$\begin{array}{r} 3 \\ \overline{)11} \\ 9 \\ \hline 2 \end{array}$$

Quotient (**Result stored at Q**)
Reminder (**Result stored at A**)

$$M \rightarrow (3)_{10} \rightarrow (11)_2 \rightarrow (00011) \Rightarrow (n+1) \Rightarrow (4+1) \text{ bit}$$

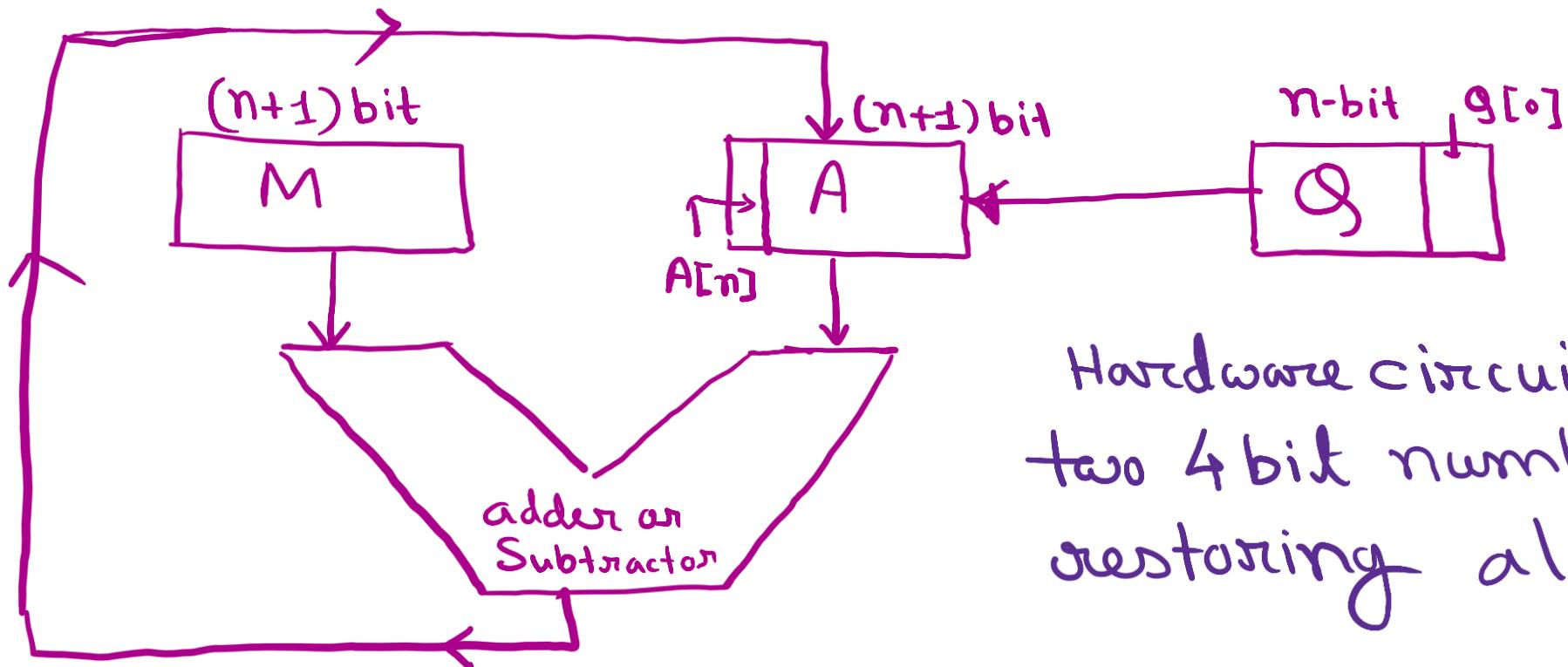
$$Q \rightarrow (11)_{10} \rightarrow (1011)_2 \Rightarrow 4 \text{ bit}$$

$$-M \rightarrow 2's \text{ comp of } M \rightarrow (11101)_2 \Rightarrow 5 \text{ bit}.$$

n	M	A	Q	Operation
4	00011	00000 00001 11110	1011 011? 011?	Initialization $LS \rightarrow AQ$
	$A[n]$			$A = A - M = A + (-M)$
3	Restore A	00001	0110	$A \rightarrow A[n] = 1, Q[0] \leftarrow 0$ Restore A
		00010	110?	$LS \rightarrow AQ$
	$A[n]$	11111	110?	$A = A - M$
2		00010	1100	$A \rightarrow A[n] = 1, Q[0] \leftarrow 0, $ Restore A
		00101	100?	$LS \rightarrow AQ$
	0	00010	100?	$A = A - M$
1		00010	1001	$A \rightarrow A[n] = 0, Q[0] \leftarrow 1, $ No Restore
		00101	001?	$LS \rightarrow AQ$
	0	00010	001?	$A = A - M$
0		00010	0011	$A \rightarrow A[n] = 0, Q[0] = 1, $ No Restore

\therefore Remainder, $R = [A] = (00010)_2 = (2)_{10} \cdot \cdot (A)$

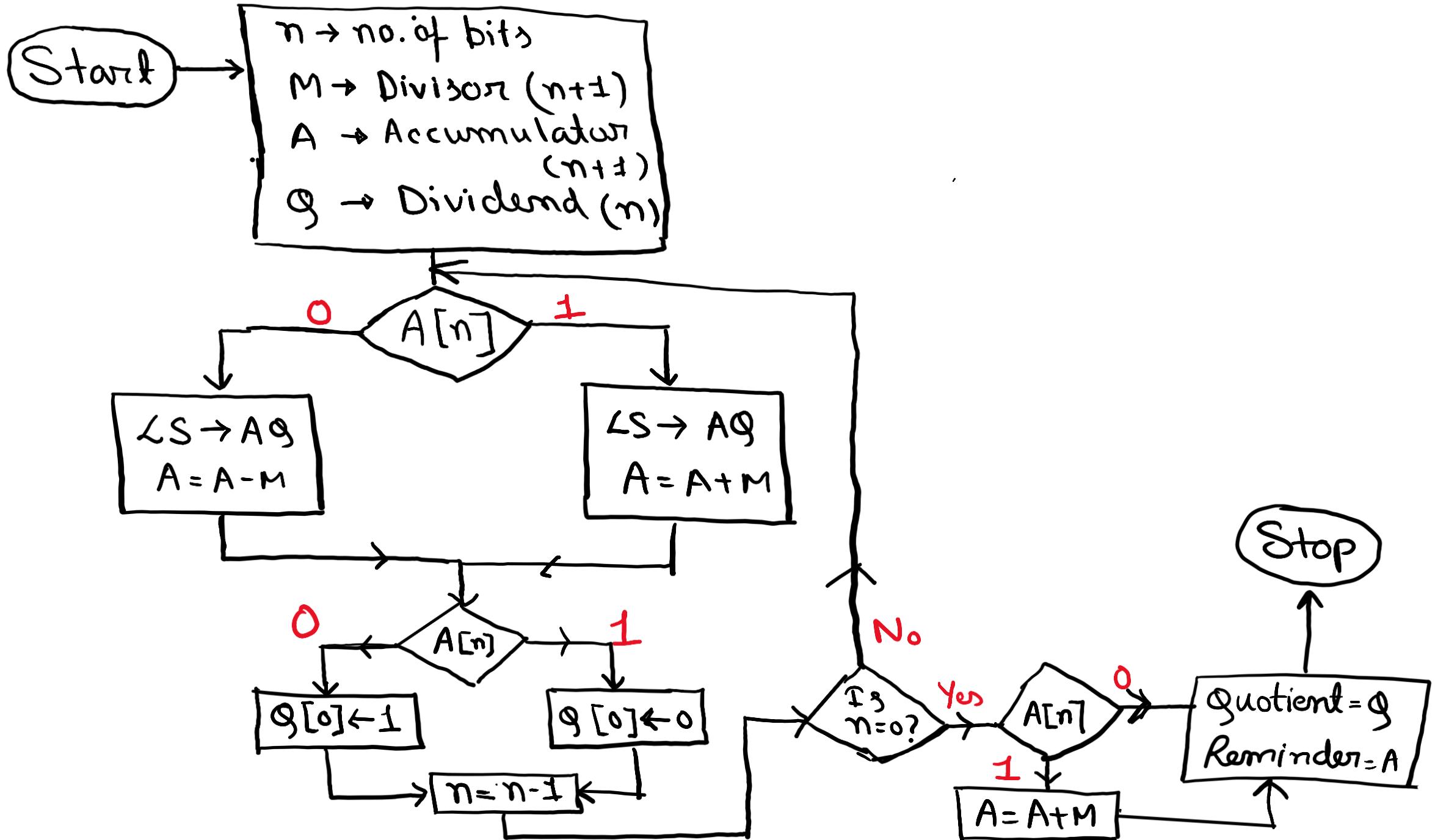
\therefore Quotient, $Q = (0011)_2 = (3)_{10} \cdot \cdot (A)$



Hardware circuit to divide
two 4 bit numbers using
restoring algorithm

* Non-Restoring Division Algorithm of Unsigned Number

- Dividend (ϱ) = $(11)_{10} = (1011)_2 \rightarrow n$ bit
- Divisor (M) = $(3)_{10} = (00011)_2 \rightarrow (n+1)$ bit
- Quotient (q) = $(3)_{10} = (0011)_2 \rightarrow n$ -bit (ϱ)
- Remainder (r) = $(2)_{10} = (00010)_2 \rightarrow (n+1)$ bit (A)
- $-M = 2^k$'s Compliment form = $(11101)_2$ of M



n	M	A	Q	Operation
4	00011	00000 A[n] → 0	1011 011?	Initialization $LS \rightarrow AQ$
.		11110 A[n] → 1	011?	$A[n] = 0 \rightarrow A = A - M = A + (-M)$
3		11100 A[n] = 1 ; $Q[0] = 0$	0110 110?	$LS \rightarrow AQ$
2		11111 A[n] = 1 ; $A = A + M$	1100 100?	$A[n] = 1 ; Q[0] = 0$
1	00010 00010 00101 00010	100? 1001 001? 001?	100? 1001 001? 001?	$LS \rightarrow AQ$ $A[n] = 1 ; A = A + M$ $A[n] = 0 ; Q[0] \leftarrow 1$ $LS \rightarrow AQ$ $A[n] = 0 ; A = A - M$ $A[n] = 0 ; Q[0] \leftarrow 1$

$$[A] = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix}_2 = (2)_{10} \Rightarrow \text{Reminder}$$

↑
 $A[n] = 0$

$$[g] = \begin{pmatrix} 0 & 0 & 1 & 1 \end{pmatrix}_2 = (3)_{10} \Rightarrow \text{Quotient.}$$

