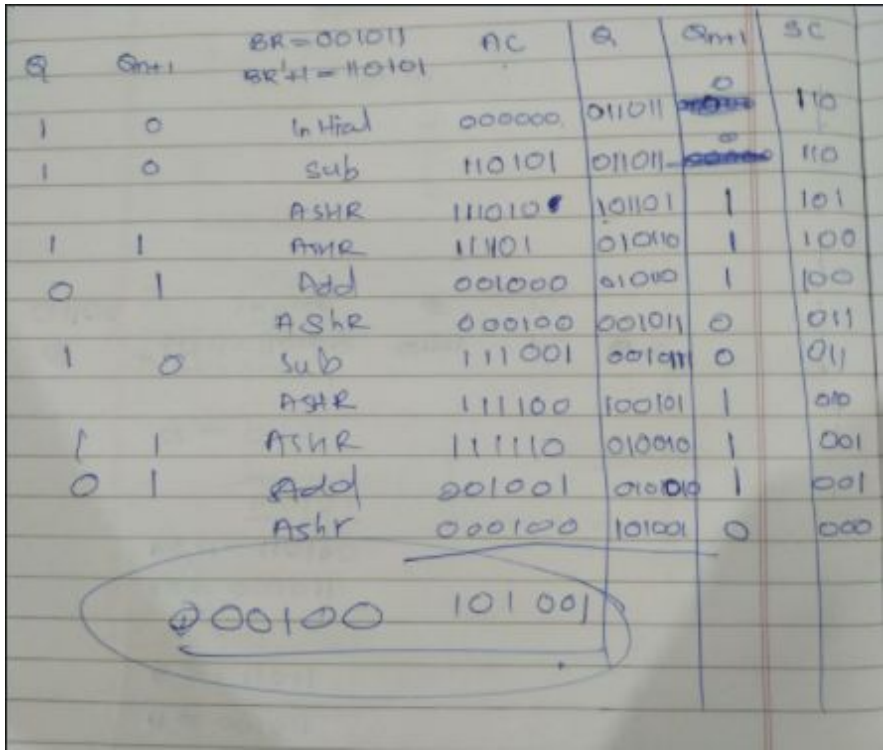
Prove that the worst case delay through an n × n array multiplier is 6(n − 1) −1 gate delays

The worst case signal propagation delay path is from the upper right corner of the array to the high-order product bit output at the bottom left corner of the array. The path consists of the staircase pattern that includes the two cells at the right end of each row, followed by all the cells in the bottom row. Assuming that there are two gate delays from the inputs to the outputs of a full adder block, the path has a total of $6(n-1)-1$ gate delays, including the initial AND gate delay in all cells, for the $n \times n$ array. (See Problem 6.12.) Only the AND gates are actually needed in the first row of the array because the incoming partial product PP0 is zero. This has been taken into account in developing the delay expression.

Multiply each of the following pairs of signed 2's-complement numbers using the Booth algorithm. In each case, assume that A is the multiplicand and B is the multiplier.

i) A = 001011 and B = 011011

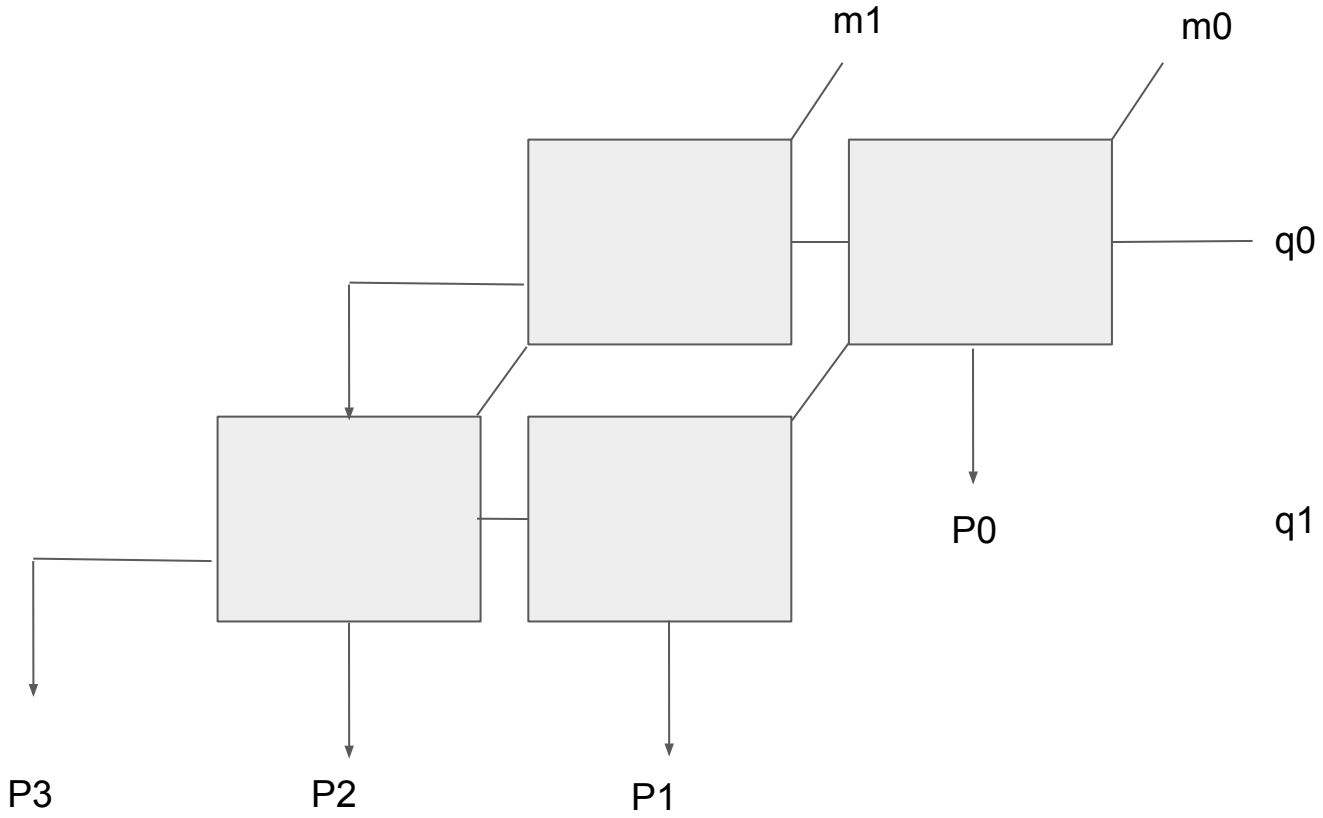
ii) A = 000111 and B = 000111

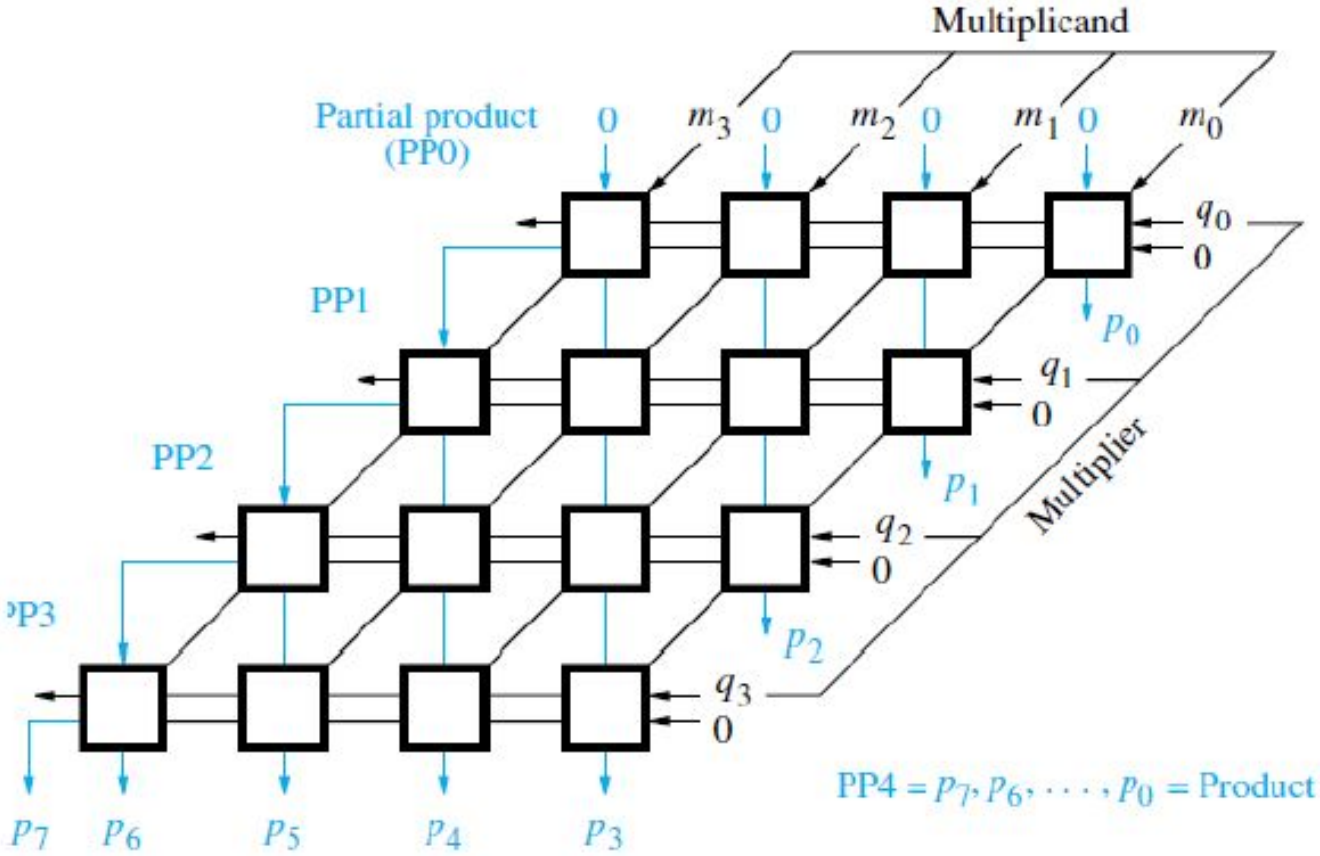


Q	Q_{n+1}	BR = 001011 BR' + 1 = 110101	AC	Q	Q_{n+1}	SC
					0	
1	0	In Hial	000000	011011	0	110
1	0	Sub	110101	011011	0	110
		ASHR	111010	101101	1	101
1	1	ASHR	111101	010110	1	100
0	1	Add	001000	010110	1	100
		ASHR	000100	001011	0	011
1	0	Sub	111001	001011	0	011
		ASHR	111100	100101	1	010
1	1	ASHR	111110	010010	1	001
0	1	Add	001001	010010	1	001
		Ashr	000100	101001	0	000

000100 101001

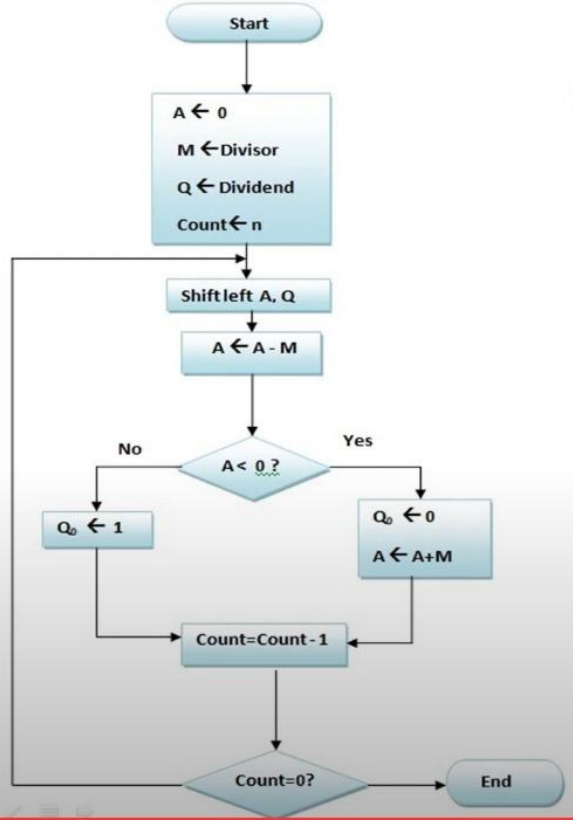
Design a 2x2 array multiplier.




Multiplicand
Partial product (PP0)
0
m_3
0
m_2
0
m_1
0
m_0
q_0
0
PP1
p_0
q_1
0
PP2
p_1
Multiplier
q_2
0
p_2
q_3
0
p_7
p_6
p_5
p_4
p_3
PP4 = $p_7, p_6, \cdots, p_0$ = Product

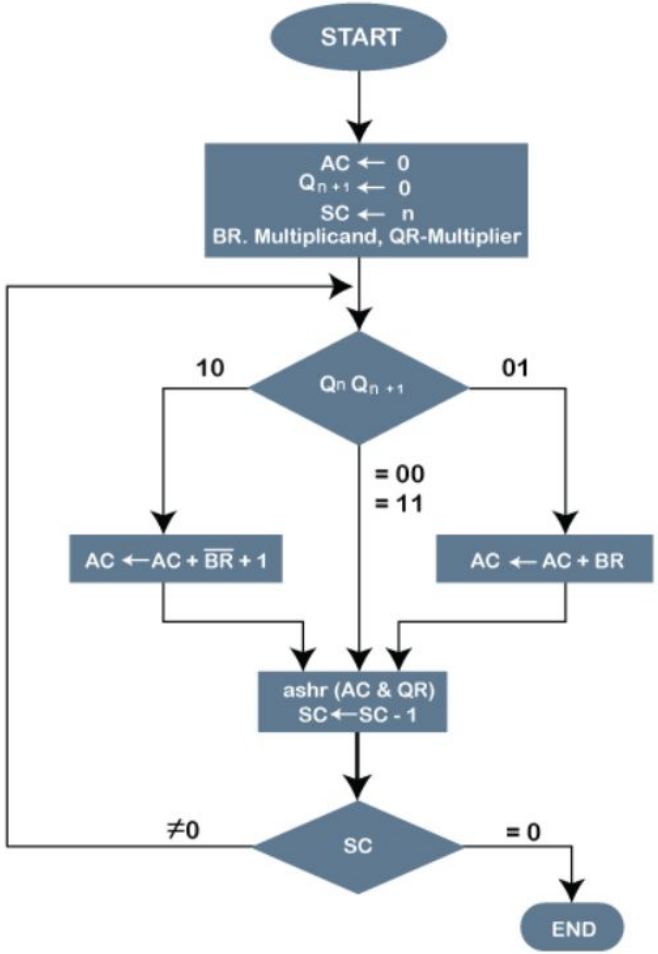
Explain restoring method of division with the help of a flow chart.

Algorithm for Division

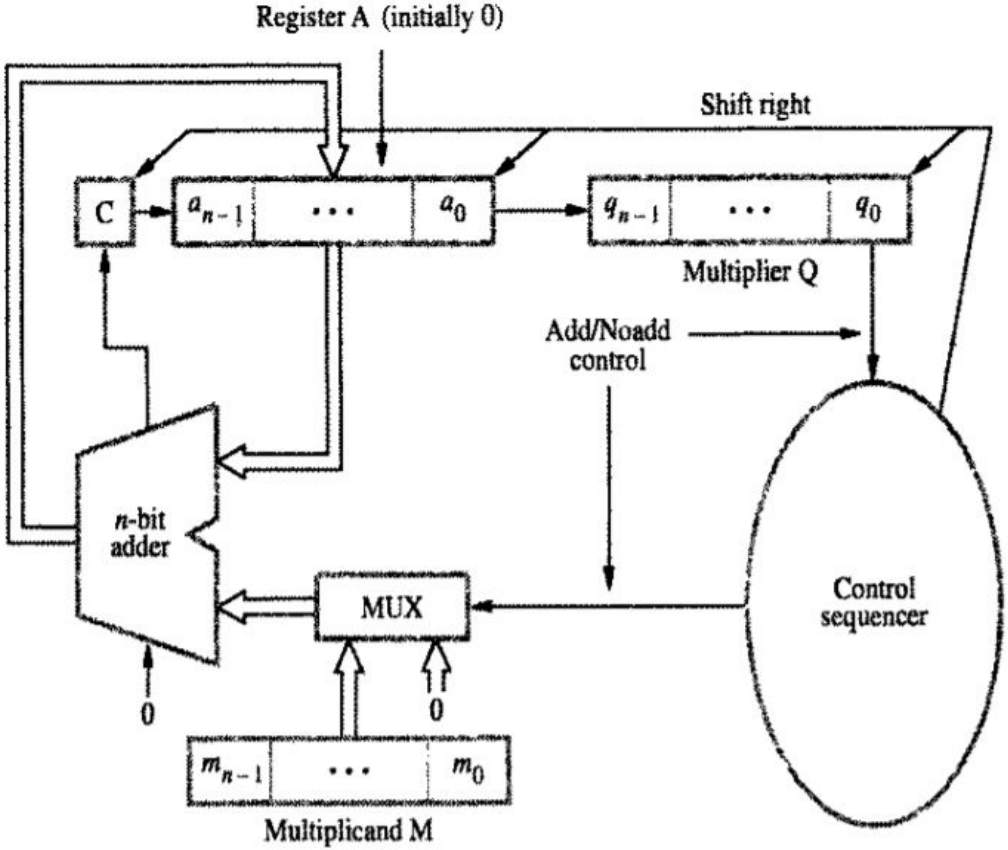


Count	A	Q	M	Operation
4	0000	0111	0011	Intialization
3	0000 1101 0000	1110 1110 1110	0011 0011 0011	Left Shift A ← A- M Restore (A←A +M) and $Q_0 \leftarrow 0$
2	0001 1110 0001	1100 1100 1100	0011 0011 0011	Left shift A ← A-M Restore (A←A +M) and $Q_0 \leftarrow 0$
1	0011 0000 0000	1000 1000 1001	0011 0011 0011	Left Shift A ← A-M Set $Q_0 \leftarrow 1$
0	0001 1110 0001 Remainder	0010 0010 0010 Quotient	0011 0011 0011	Left Shift A ← A- M Restore (A←A +M) and $Q_0 \leftarrow 0$

Draw the flow chart for Booth's Multiplication algorithm.

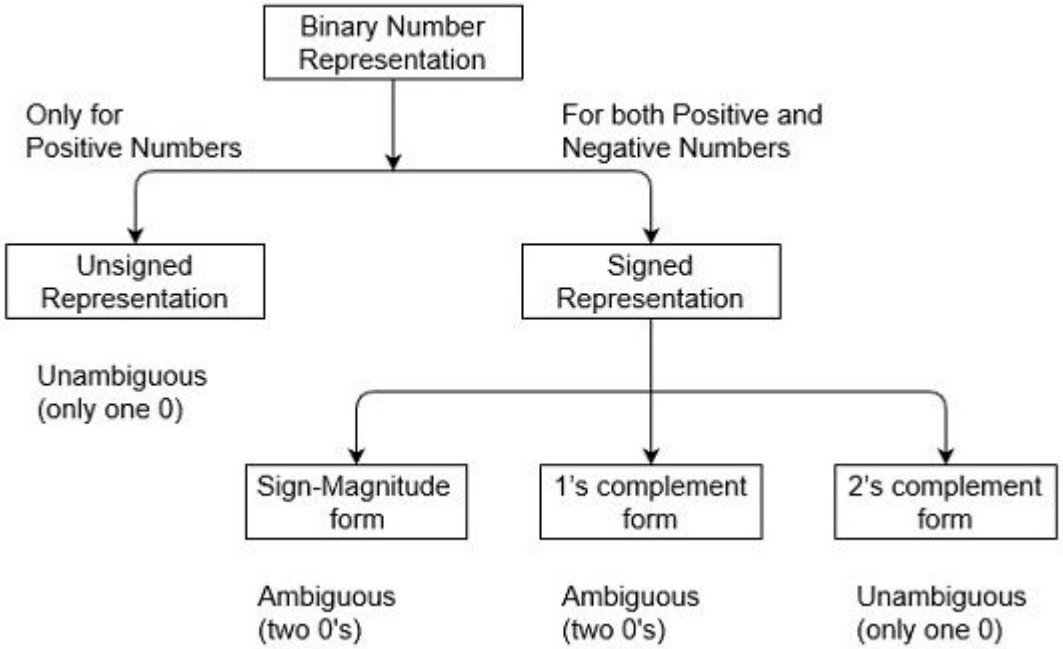


Draw and explain the circuit for implementing unsigned integer multiplication.



(a) Register configuration

Explain about signed and unsigned numbers representation in binary



Evaluate $(0010)_2$ with $(0011)_2$ using multiplication algorithm.

BOOTH ALGORITHM

Explain data hazards and control hazard with an example.

Data hazards occur when instructions depends on result of prior instruction still in the pipeline. Consider two instructions I and J, Each instruction can be considered as a mapping from a set of data objects to a set of data objects.

- The domain D(I) of an instruction I is the set of resource objects whose data objects affects the execution of the instruction I.
- The range R(I) of an instruction I is the set of resource objects whose data objects may be modified by the instruction I.
- The operands to be used for the execution of the instruction are read from its Domain and the results are written in its Range.
- Consider 2 instructions I and J. Instruction J appears after the instruction I in the program. Instruction J may enter the execution pie before or after the completion of execution of instruction I.

Data hazards can be classified into three types:

- WAR- Write after read. This occurs then instruction J tries to modify information that is already read by I.
- RAW- Read after write. Occurs when J tries to read information modified by I.
- WAW- Write after write, when I and J try to modify data at the same time.

This type of dependency occurs during the transfer of control instructions such as BRANCH, CALL, JMP, etc. On many instruction architectures, the processor will not know the target address of these instructions when it needs to insert the new instruction into the pipeline. Due to this, unwanted instructions are fed to the pipeline.
Consider the following sequence of instructions in the program:
100: I1
101: I2 (JMP 250)
102: I3
.
.
250: BI1
Expected output: I1 -> I2 -> BI1
NOTE: Generally, the target address of the JMP instruction is known after ID stage only.

Discuss about various types of pipeline hazards.

Structural Hazards- This occurs when there is a lack of resources. A resource conflict arises when more than one instruction tries to access the same resources in the same cycle. A resource can be a register, memory, or ALU. Consider the table:

Instruction / Cycle	1	2	3	4	5
I_1	IF(Mem)	ID	EX	Mem	
I_2		IF(Mem)	ID	EX	
I_3			IF(Mem)	ID	EX
I_4				IF(Mem)	ID

In the above scenario, in cycle 4, instructions I1 and I4 are trying to access same resource (Memory) which introduces a resource conflict.
To avoid this problem, we have to keep the instruction on wait until the required resource (memory in our case) becomes available. This wait will introduce **stalls** in the pipeline as shown below:

Stall : A stall is a cycle in the pipeline without new input.

Cycle	1	2	3	4	5	6	7	8
I_1	IF(Mem)	ID	EX	Mem	WB			
I_2		IF(Mem)	ID	EX	Mem	WB		
I_3			IF(Mem)	ID	EX	Mem	WB	
I_4				-	-	-	IF(Mem)	

Solution for structural dependency
To minimize structural dependency stalls in the pipeline, we use a hardware mechanism called Renaming.

Renaming : According to renaming, we divide the memory into two independent modules used to store the instruction and data separately called Code memory(CM) and Data memory(DM) respectively. CM will contain all the instructions and DM will contain all the operands that are required for the instructions.

Draw the flowchart for decimal multiplication.