# ECE 594: Circuit Simulation

Jonathan K
Ravi
Eric
Kai
2013 March 7

# Topics

- Definition
- Introduction
- Gate Simulation
- “Breadboard” Simulation
- Code Compiled Simulation
- Timing Problems
- Event-Driven Simulation
- Summary

# Simulation

**sim·u·la·tion** [sìmmyə láysh'n]
(*plural* **sim·u·la·tions**)
*noun*

1. **reproduction of features of something:** the reproduction of the essential features of something, e.g. as an aid to study or training

2. **false appearance:** the imitation or feigning of something

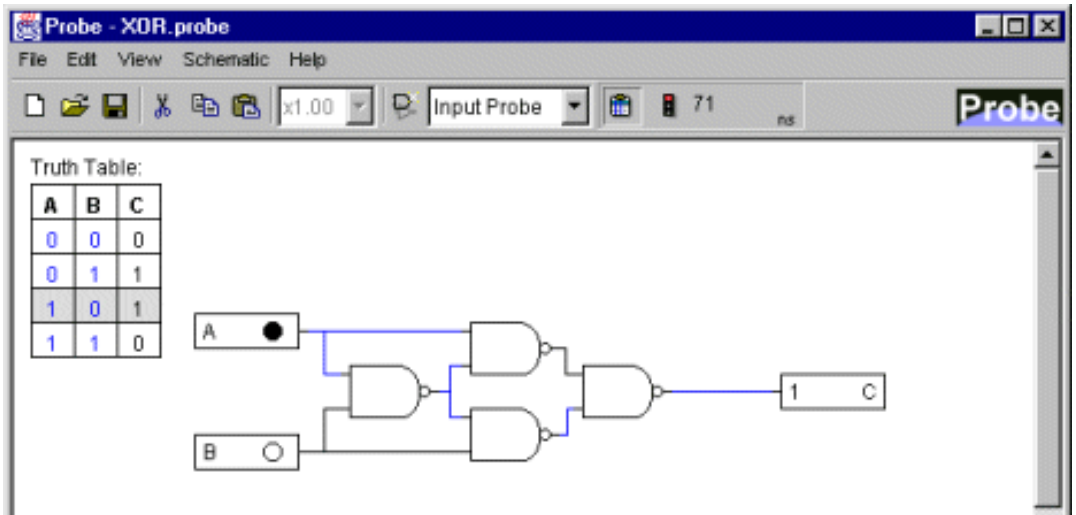
3. **fake:** an artificial or imitation object

4. COMPUT **construction of mathematical model:** the construction of a mathematical model to reproduce the characteristics of a phenomenon, system, or process, often using a computer, in order to infer information or solve problems

5. COMPUTER GAMES **computer game:** a computer game that simulates a real activity such as flying

**imitation**, reproduction, replication, recreation, mock-up, model, virtual reality





# Dress Example

Once the cloth is cut, it will be too late to discover any errors in the design.

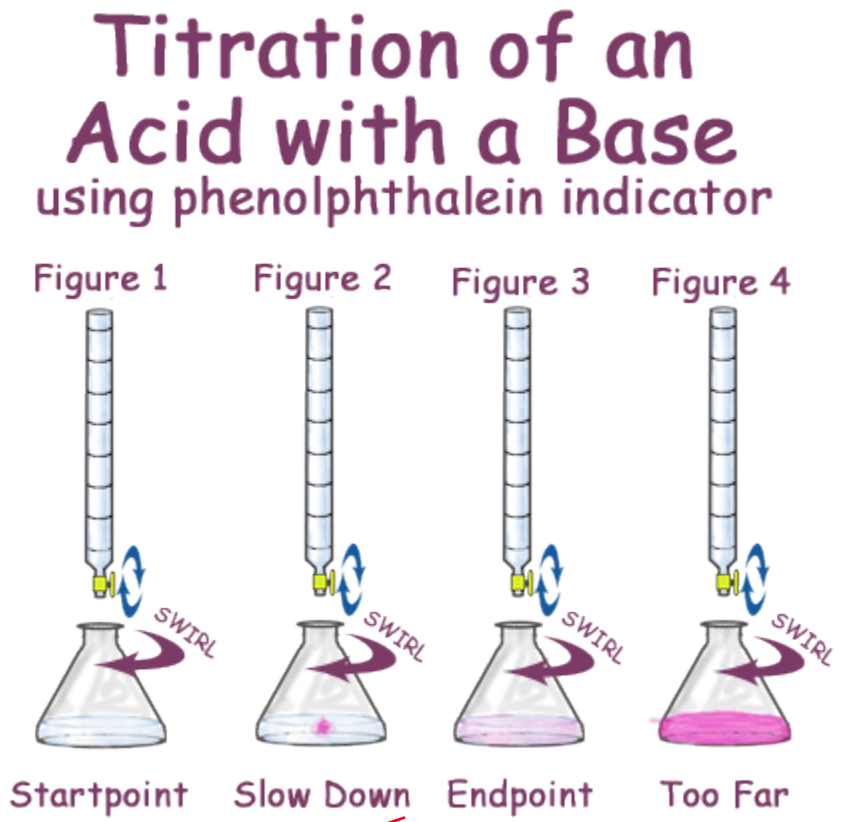
# Everyday Example

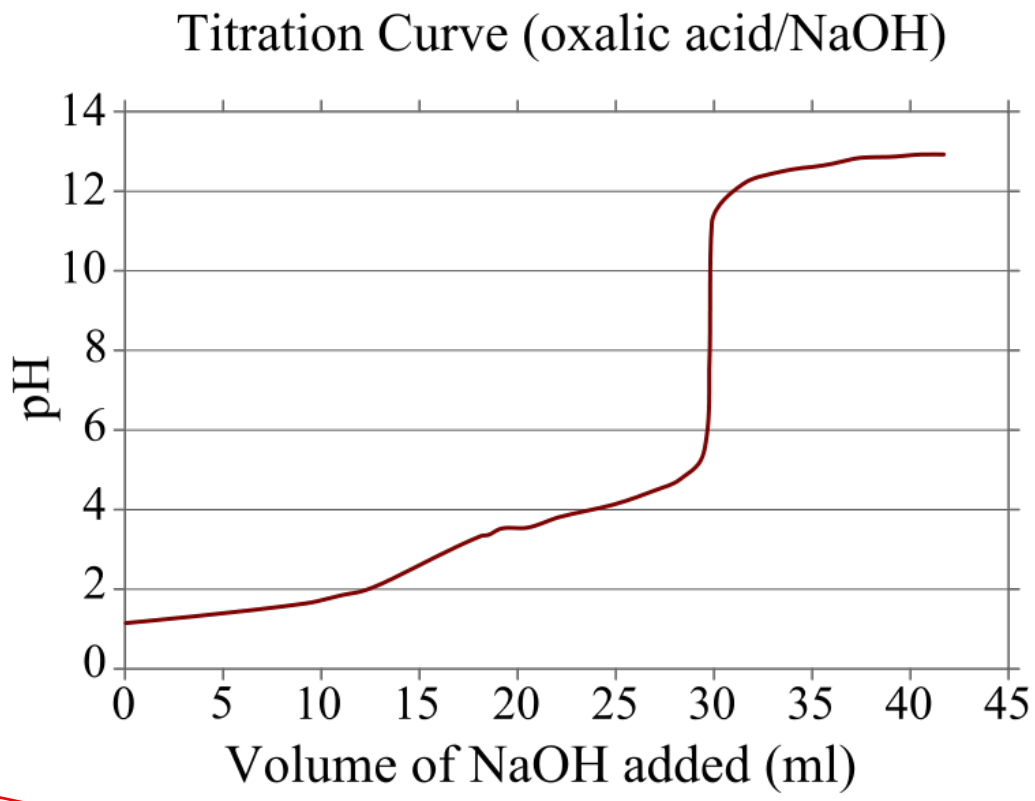
Salt is difficult to remove but easy to add in food

Therefore, add a little bit at a time, try it, and then add more if not salty enough

Trying a spoonful should simulate how to rest of the serving taste

# Chemistry Example



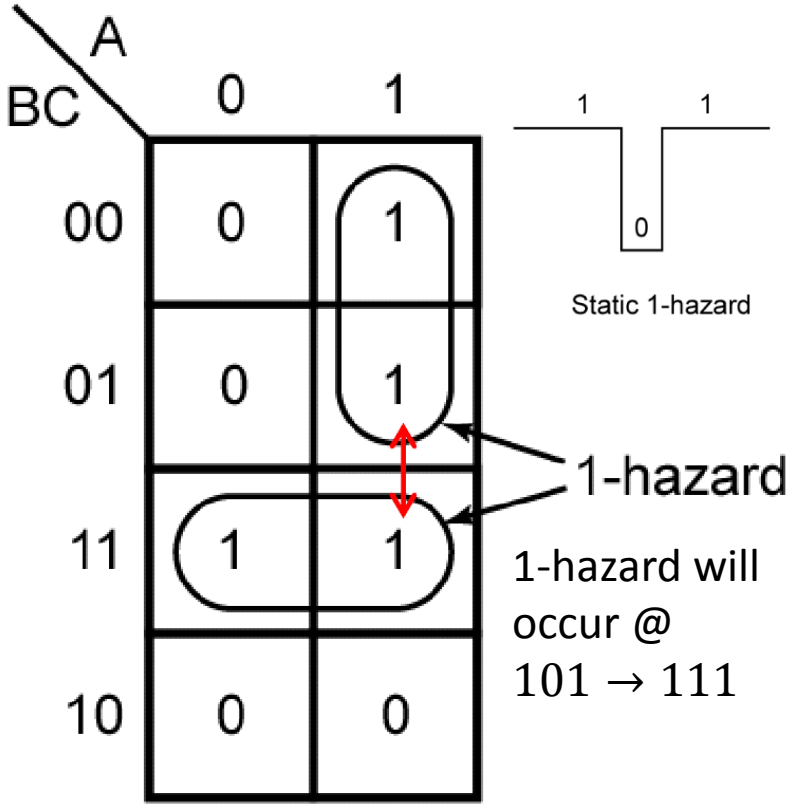




Already took a while a get to this point

Although you could slow down here, but it will still take a while to finish the titration based on graph

# Why Simulation?

- Validate assumptions
- Verify logic
- Verify performance
- Find tests for the circuit

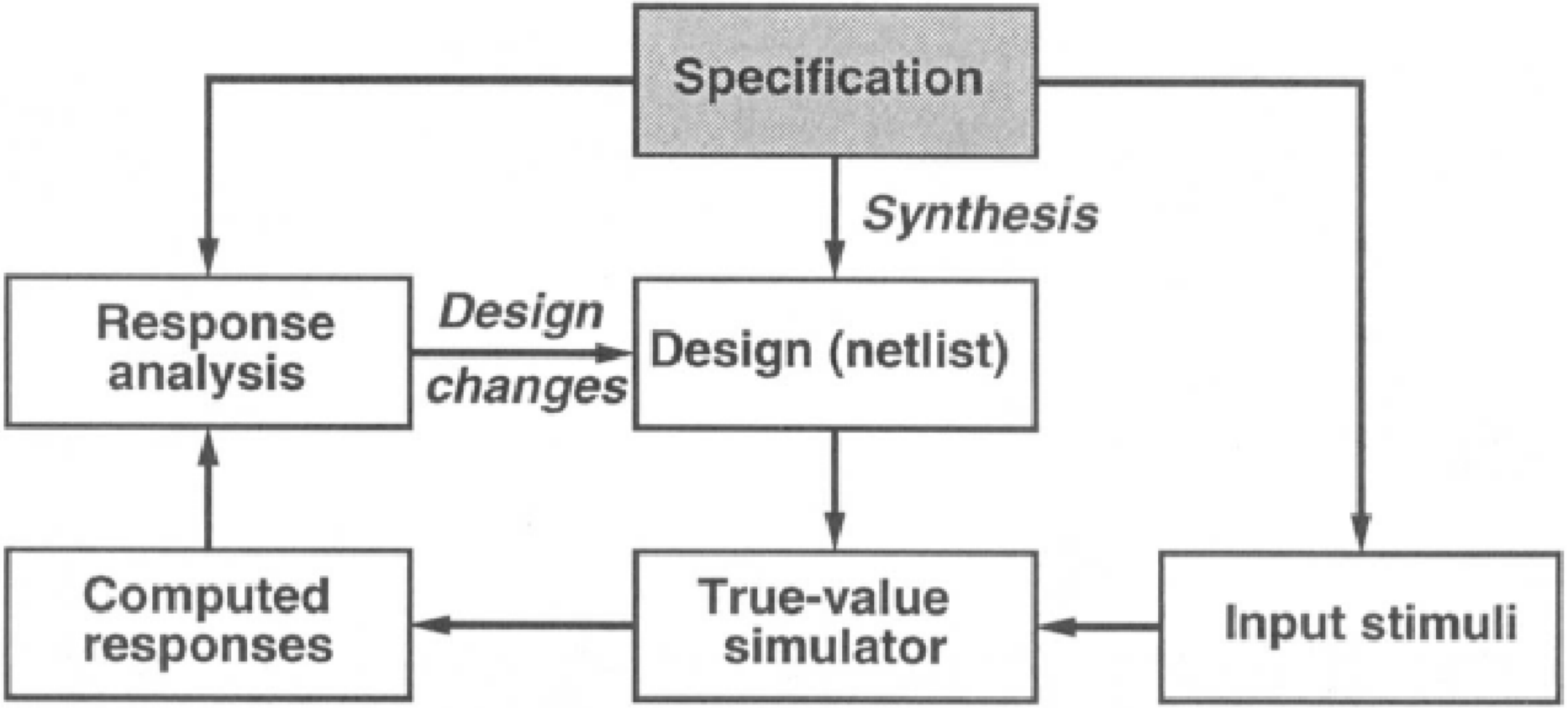




| A | B | Z | A sa0 | A sa1 | B sa0 | B sa1 | Z sa0 | Z sa1 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | x | | x | | x |
| 0 | 1 | 1 | | x | x | | x | |
| 1 | 0 | 1 | x | | | x | x | |
| 1 | 1 | 0 | x | | x | | | x |

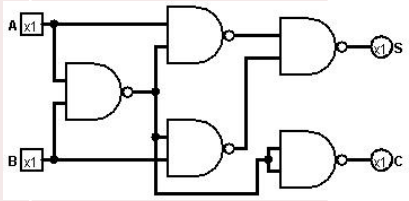
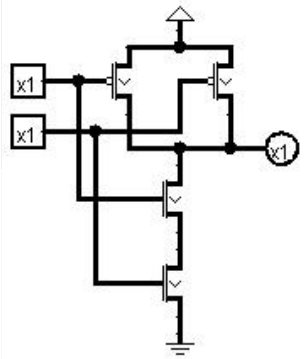
1. Shown is a XOR-model truth table and corresponding stuck-at coverage.

# Simulation for Design Verification

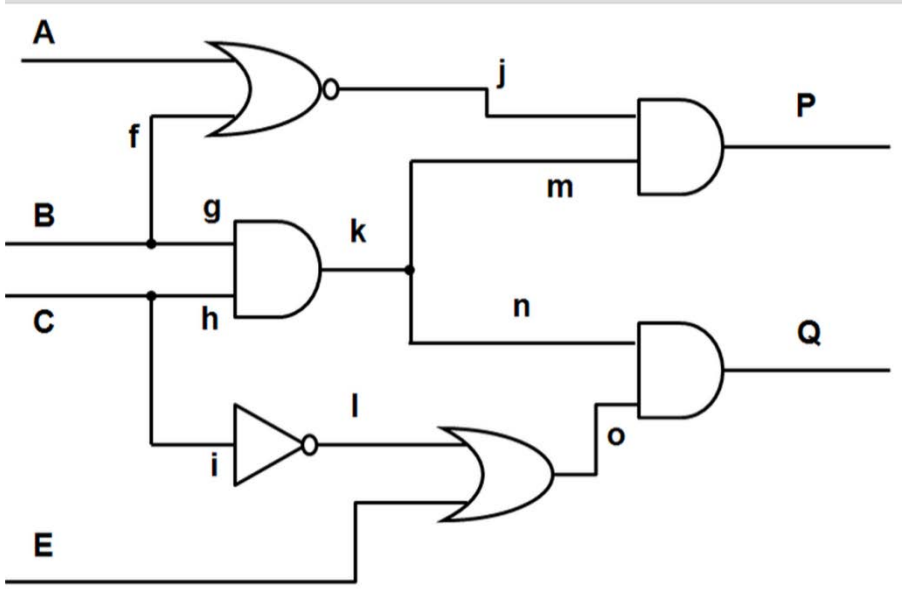



Design Verification basically compares the output of the simulator versus the specification to determine design changes

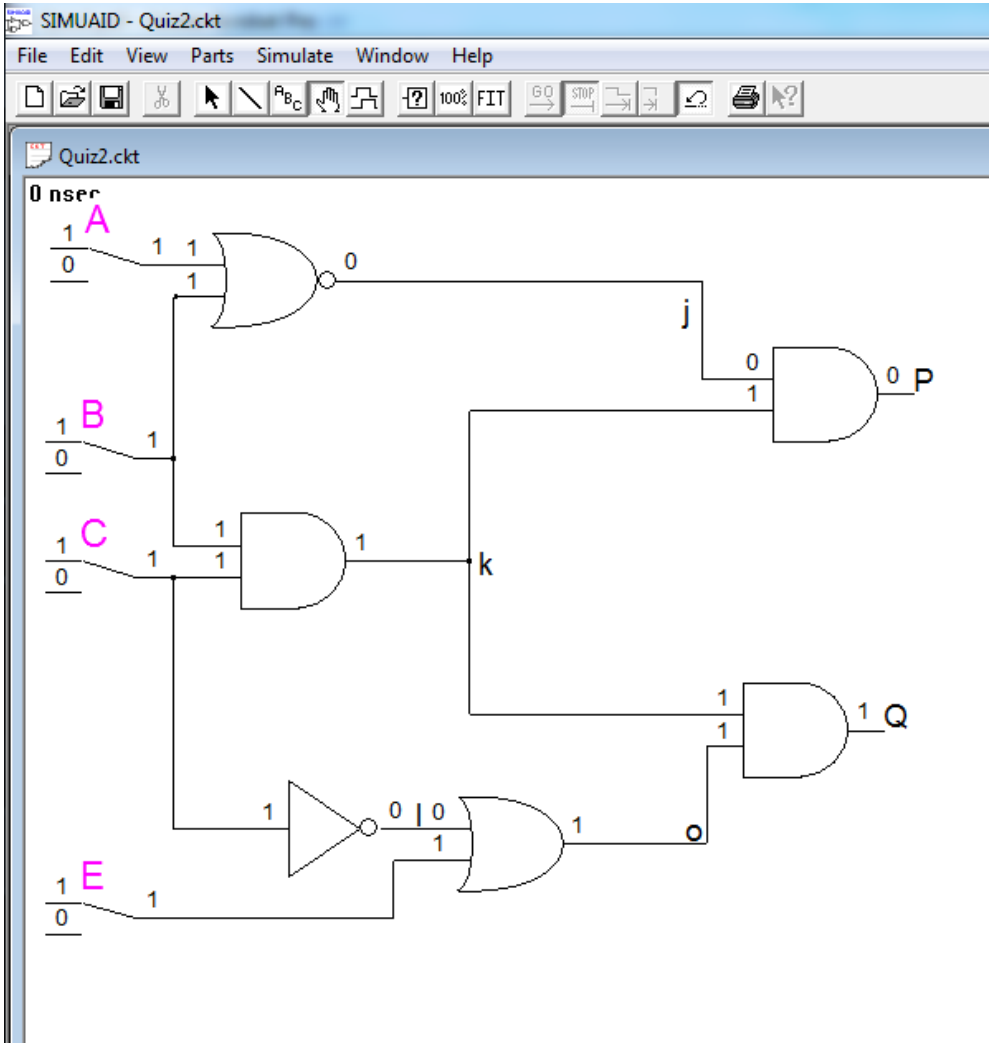
# Simulation Model Levels

| Modeling Level | Circuit Description | Signal Values | Timing | Application | |
|---|---|---|---|---|---|
| functional | high level programming languages | 0, 1 | clock values | design verification |   |
| logic | boolean gates and flip-flops | 0, 1 & X | zero-delay, unit or multiple delay | logic verification |   |
| switch | MOS transistors | 0, 1, X & Z | zero-delay | logic verification |   |
| timing | transistors with tech data | analog voltage | fine-grain time | timing verification | |
| circuit | active & passive components | analog voltage & current | continuous time | circuit verification | |

# Gate Simulation: SimUAid

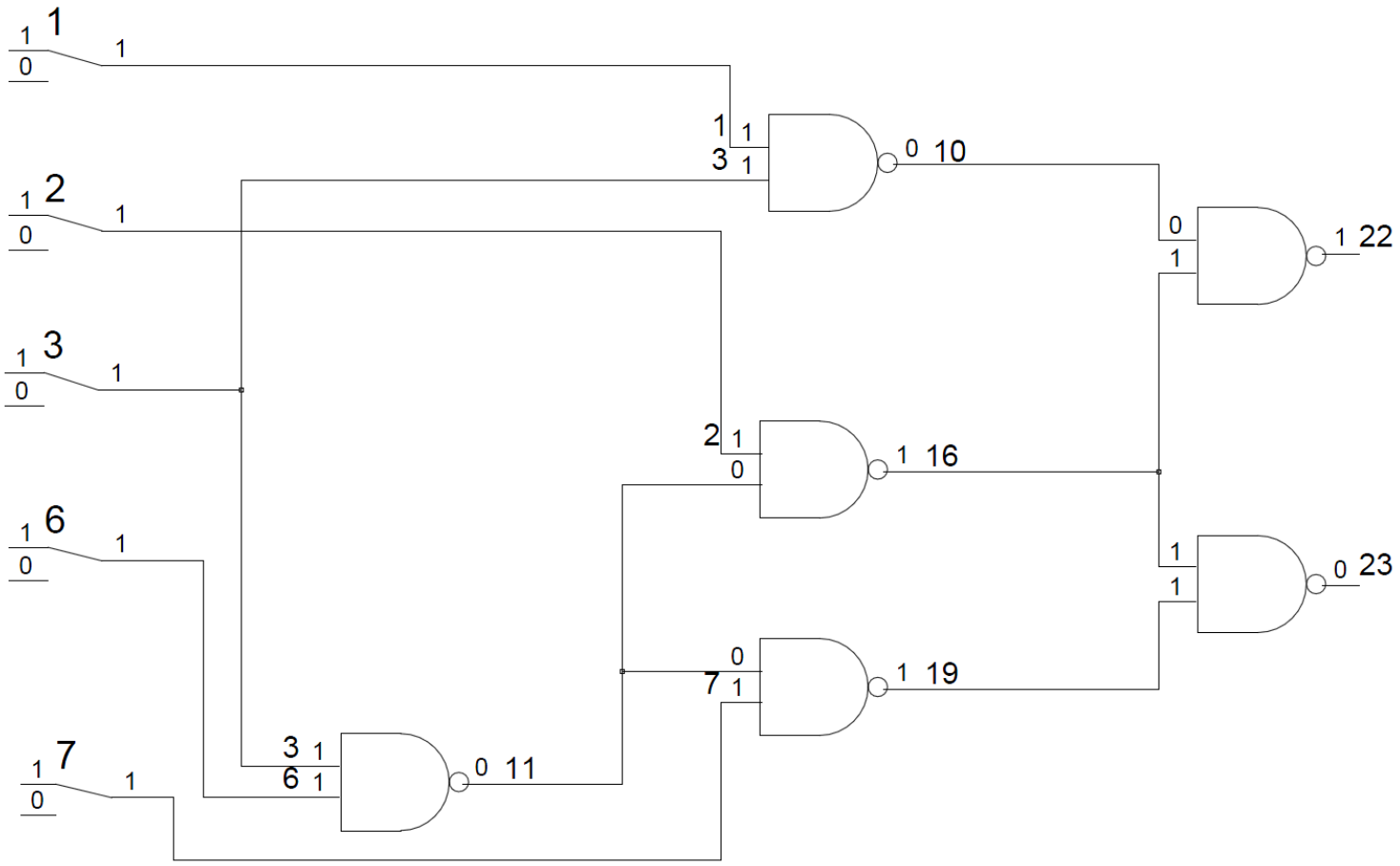



This is the circuit we have been using on every quiz so far.



# C17.bench

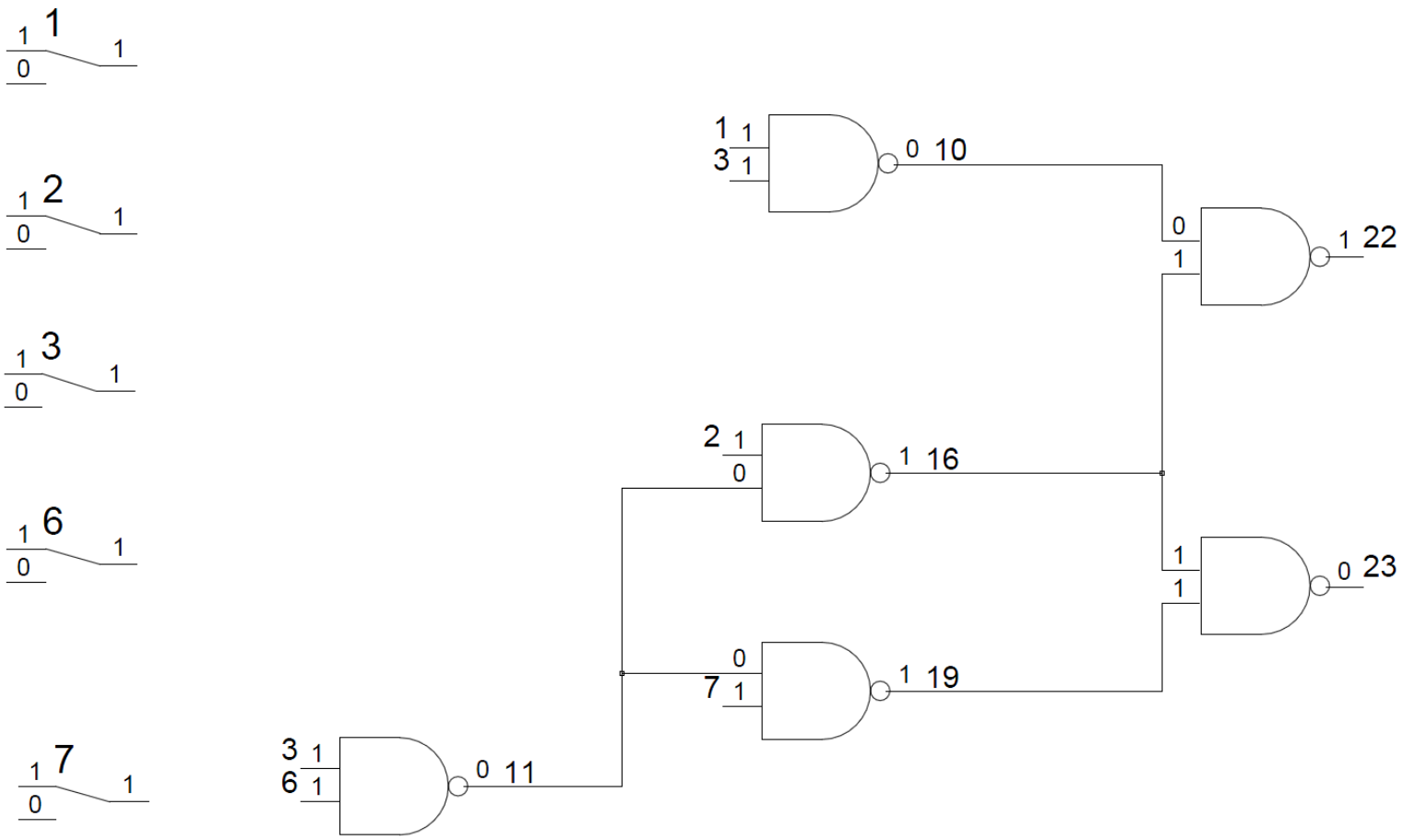
```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```



# C17.bench

```
1  # c17
2  # 5 inputs
3  # 2 outputs
4  # 0 inverter
5  # 6 gates ( 6 NANDs )
6
7  INPUT(1)
8  INPUT(2)
9  INPUT(3)
10 INPUT(6)
11 INPUT(7)
12
13 OUTPUT(22)
14 OUTPUT(23)
15
16 10 = NAND(1, 3)
17 11 = NAND(3, 6)
18 16 = NAND(2, 11)
19 19 = NAND(11, 7)
20 22 = NAND(10, 16)
21 23 = NAND(16, 19)
```

## Virtual Connections



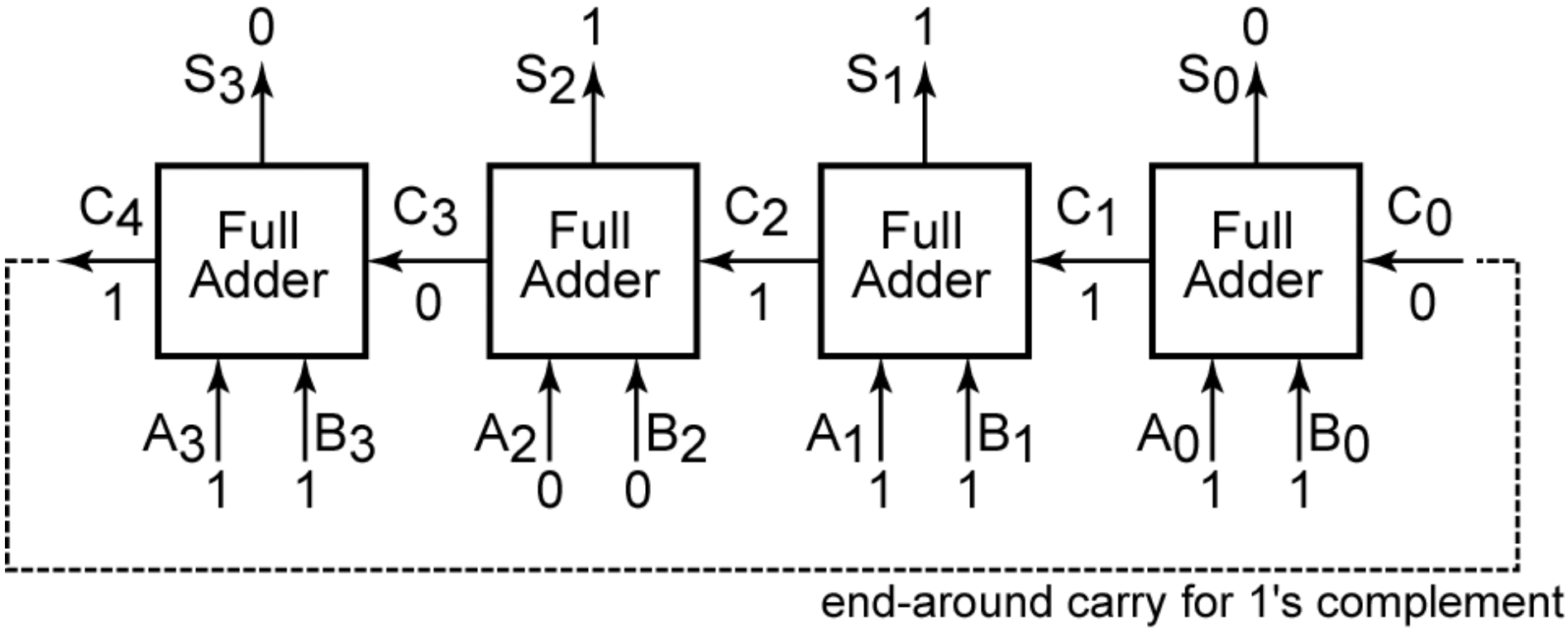
1. What will the output be for 3+(-3) in 1's complement?

# (a) 0000, (b) 1111, or (c) depends

| | + | - | |
|---|---|---|---|
| 0 | 0000 | 1111 | —Note that +0 and −0 return TRUE when tested for zero, FALSE when tested for non-zero. |
| 1 | 0001 | 1110 | |
| 2 | 0010 | 1101 | |
| 3 | 0011 | 1100 | |
| 4 | 0100 | 1011 | |
| 5 | 0101 | 1010 | |
| 6 | 0110 | 1001 | |
| 7 | 0111 | 1000 | |

This is just a reminder of 1's complement

$$0011_{1's\ C}$$
$$+1100_{1's\ C}$$
$$=?_{1's\ C}$$



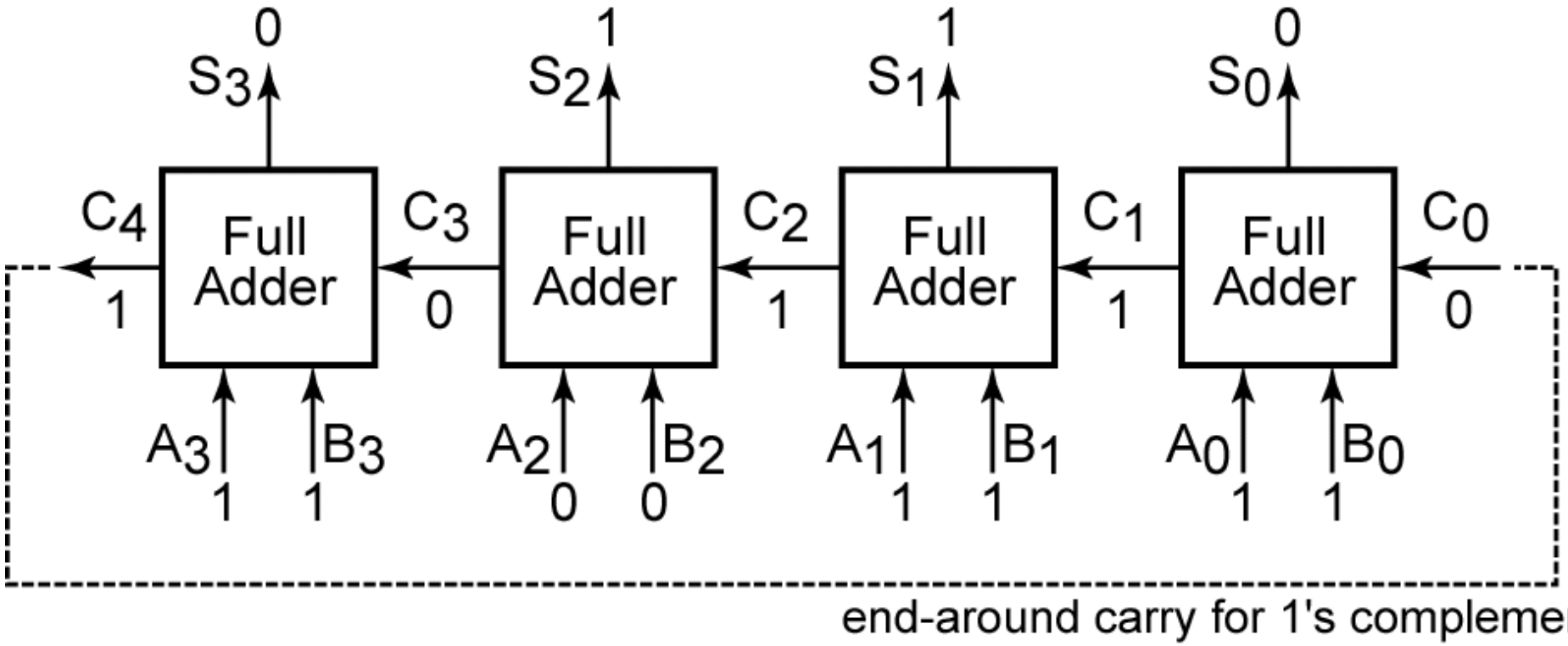
# 2. If it depends, then on what?

## (a) 3+(-3) vs (-3)+3, (b) order of input, or (c) something else

| | + | - | |
|---|---|---|---|
| 0 | 0000 | 1111 | —Note that +0 and −0 return TRUE when tested for zero, FALSE when tested for non-zero. |
| 1 | 0001 | 1110 | |
| 2 | 0010 | 1101 | |
| 3 | 0011 | 1100 | |
| 4 | 0100 | 1011 | |
| 5 | 0101 | 1010 | |
| 6 | 0110 | 1001 | |
| 7 | 0111 | 1000 | |

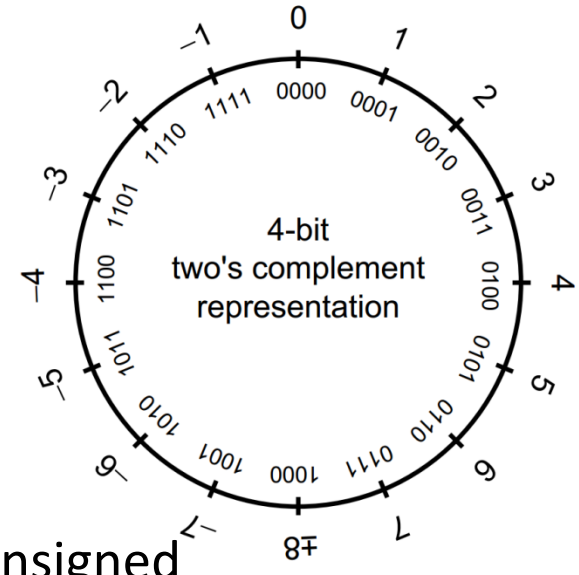
This is just a reminder of 1's complement

$$0011_{1's\ C}$$
$$+1100_{1's\ C}$$
$$=?_{1's\ C}$$

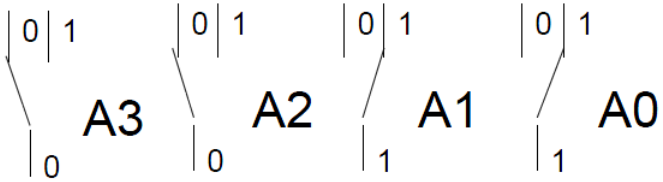


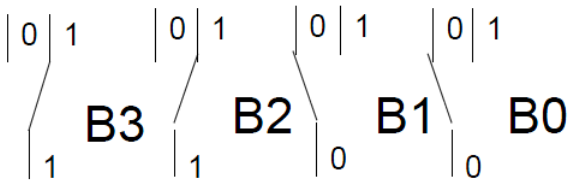
$$\begin{array}{r} 1100 \\ +\ 0011 \\ =\ 1111 \end{array}$$

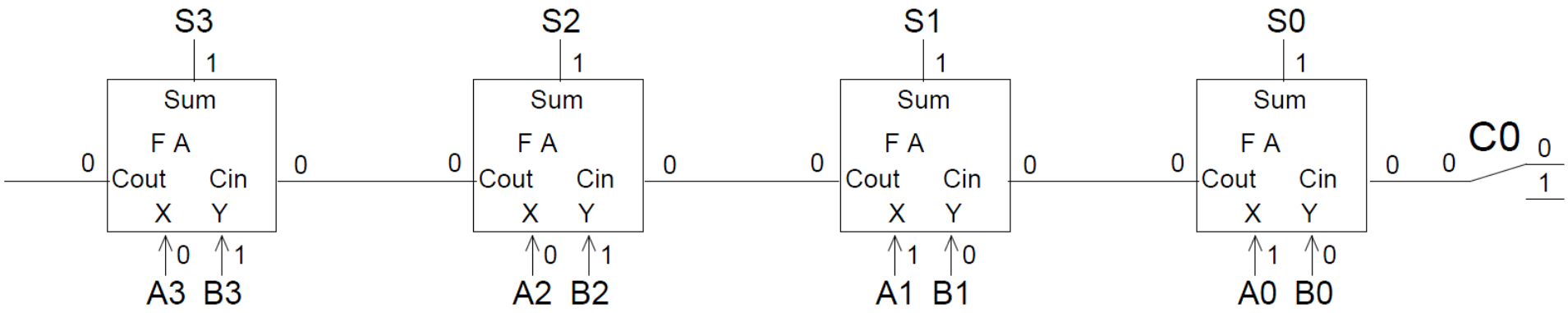
# SimUAid Simulation



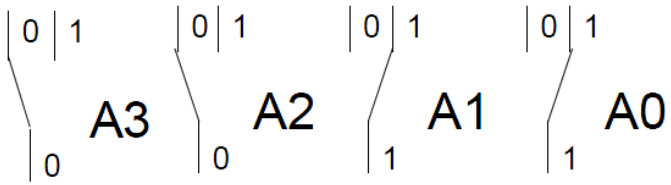

- This is NOT 1's complement
- In Unsigned Value, 3+12=15=$1111_{\text{Unsigned}}$
- In 2's complement, 3+(-4)=-1=$1111_{\text{2's C}}$

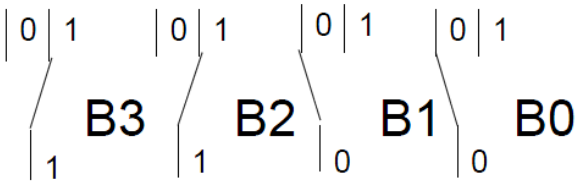


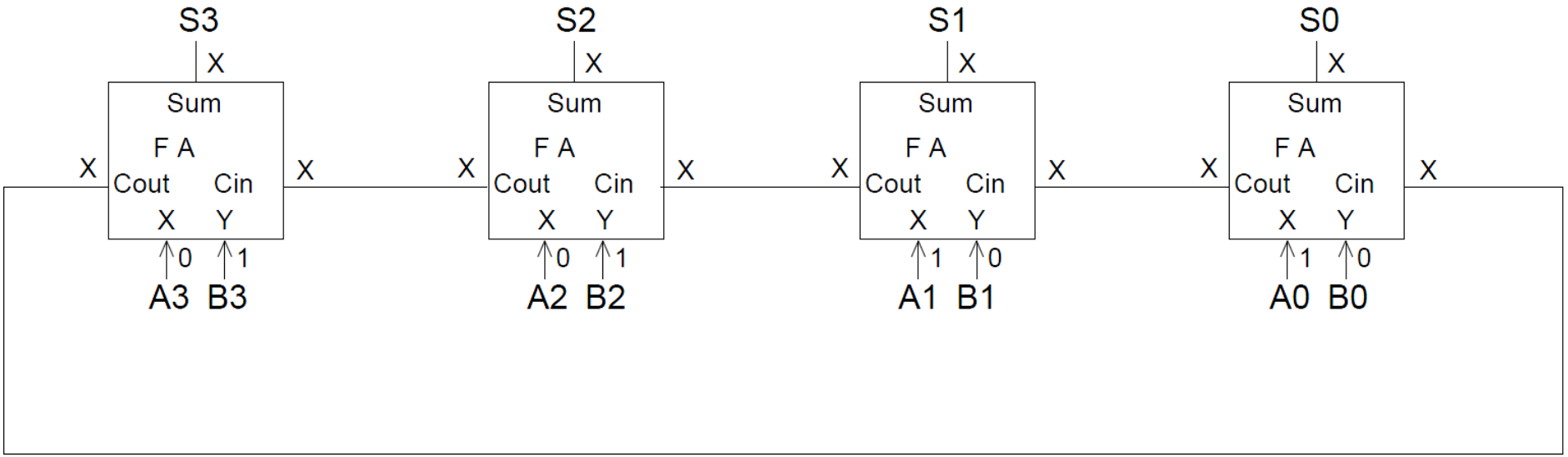



# SimUAid Simulation



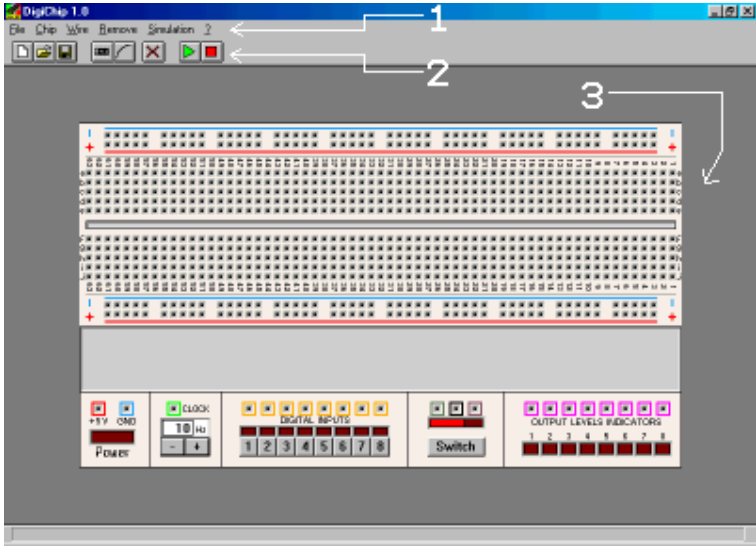

SimUAid failed to simulate an actual output

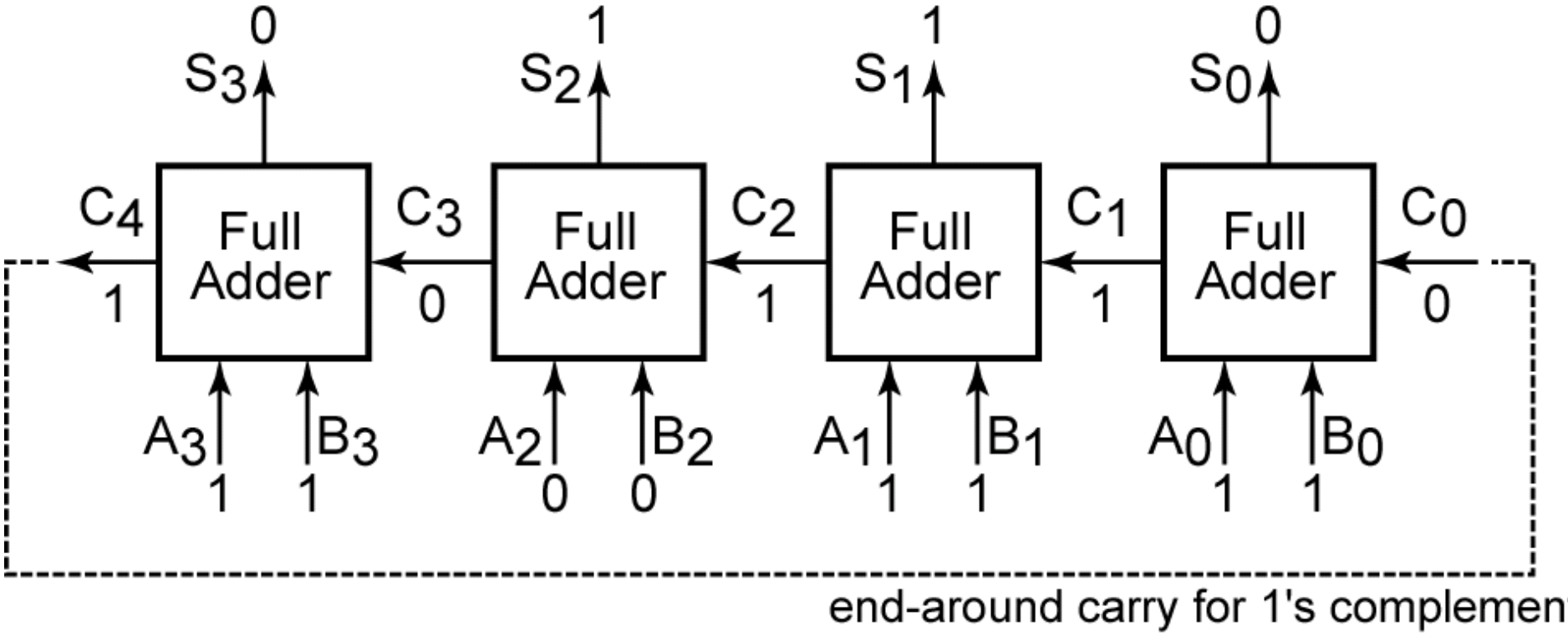
(reason for failure will be explained later)

SimUAid cannot simulate everything

# “Breadboard” Simulation

What can go wrong with a “Breadboard” simulation?

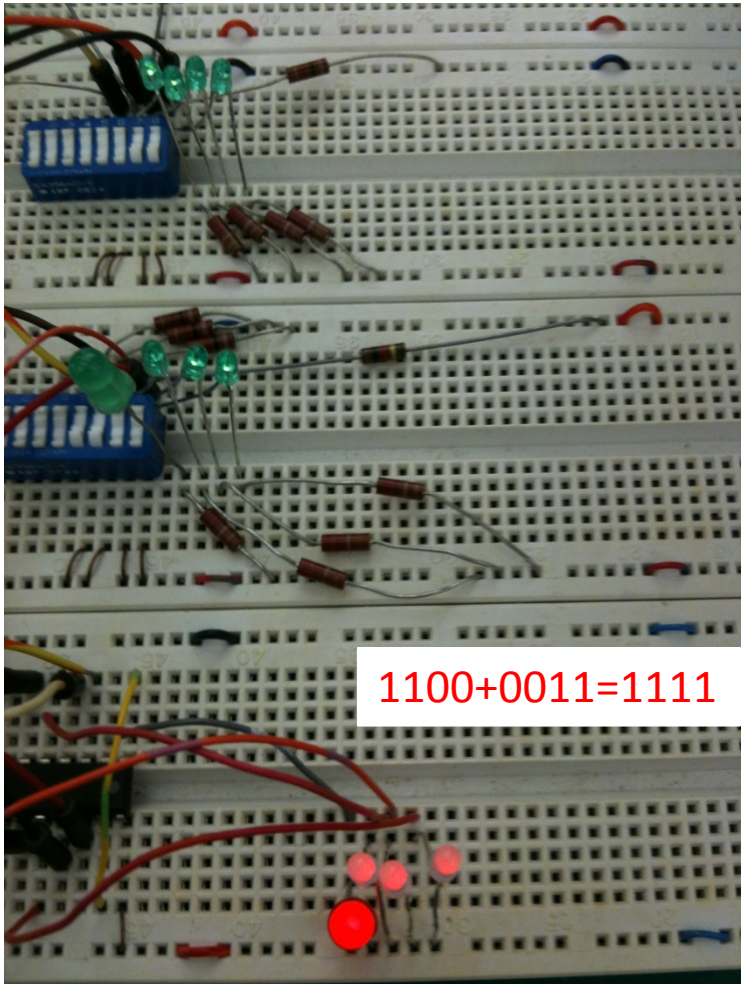


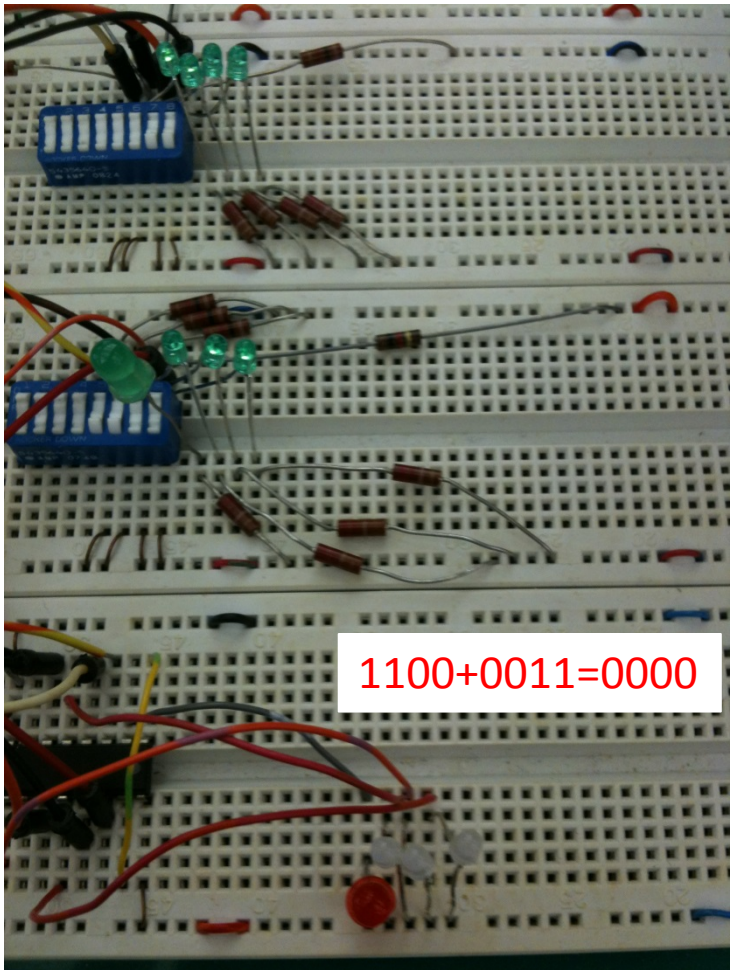



# “Breadboard” Simulation

Results

- If you have 3+(-3) set up as inputs before you supply the power, then the $C_0 = 0$ and output becomes 1111.
- Otherwise, it depends on the previous value of $C_0$, just like a flip-flop



# “Breadboard” Simulation Problems

- Faulty wires
- Power supply (low batteries)
- Short circuit (wiring error)
- Damaged IC
- Incorrect logic (truth tables, K-maps)
- Off by 1 column
- Time consuming (bug)

So many problems with “Breadboard” simulation.

# CODE COMPILED SIMULATION (PART 1)

# Quiz Problem 1 Example1

- Given a circuit or a bench file and an input, compute the outputs by hand

"1" = input = 0

"2" = input = 0

"3" = input = 0

"6" = input = 0

"7" = input = 0

benchFile

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

inputFile

```
00000

```

# Quiz Problem 1 Example1

- Given a circuit or a bench file and an input, compute the outputs by hand

“1” = input = 0

“2” = input = 0

“3” = input = 0

“6” = input = 0

“7” = input = 0

“10” = NAND(“1”, “3”) = NAND(0, 0) = 1

“11” = NAND(“3”, “6”) = NAND(0, 0) = 1

benchFile

```
1  # c17
2  # 5 inputs
3  # 2 outputs
4  # 0 inverter
5  # 6 gates ( 6 NANDs )
6
7  INPUT(1)
8  INPUT(2)
9  INPUT(3)
10 INPUT(6)
11 INPUT(7)
12
13 OUTPUT(22)
14 OUTPUT(23)
15
16 10 = NAND(1, 3)
17 11 = NAND(3, 6)
18 16 = NAND(2, 11)
19 19 = NAND(11, 7)
20 22 = NAND(10, 16)
21 23 = NAND(16, 19)
```

inputFile

```
1  00000
2
```

# Quiz Problem 1 Example1

- Given a circuit or a bench file and an input, compute the outputs by hand

benchFile

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

inputFile

```
00000

```

"1" = input = 0

"2" = input = 0

"3" = input = 0

"6" = input = 0

"7" = input = 0

"10" = NAND("1", "3") = NAND(0, 0) = 1

"11" = NAND("3", "6") = NAND(0, 0) = 1

"16" = NAND("2", "11") = NAND(0, 1) = 1

"19" = NAND("11", "7") = NAND(1, 0) = 1

# Quiz Problem 1 Example1

- Given a circuit or a bench file and an input, compute the outputs by hand

benchFile

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

inputFile

```
00000

```

"1" = input = 0

"2" = input = 0

"3" = input = 0

"6" = input = 0

"7" = input = 0

"10" = NAND("1", "3") = NAND(0, 0) = 1

"11" = NAND("3", "6") = NAND(0, 0) = 1

"16" = NAND("2", "11") = NAND(0, 1) = 1

"19" = NAND("11", "7") = NAND(1, 0) = 1

"22" = NAND("10", "16") = NAND(1, 1) = 0

"23" = NAND("16", "19") = NAND(1, 1) = 0

Doing the computations by hand is still slow.

# Pseudo Code

```
int numberOfInputs = 0;
int numberOfOutputs = 0;
int numberOfGates = 0;
int maxNode = -1;
Scanner benchFile;
while (benchFile.hasNextLine())
{
    grab next line
    parse line to increment number of inputs, outputs, or gates // to determine array size
    check maxNode                                               // to determine array size
}
// DATA STRUCTURES // numberOfOutputs is a subset of numberOfGates ////////////////////////////////////
int[] outputs = new int[numberOfOutputs];                       // used to store/print out outputs
int[] outputTracker = new int[numberOfOutputs];                 // a sorted array to keep track of outputs
int[] nodes = new int[numberOfInputs+numberOfGates];            // a array to store each input/gate // this could be a boolean as well
int[] nodePointer = new int[maxNode+1];                         // used for direct access to input values
// end of data structures ////////////////////////////////////////////////////////////////////////////
int pointerWalker = 0;
benchFile.reset();                                              // go back to begining of bench file
while (benchFile.hasNextLine())
{
    if (input or gate)
        pointer[Integer.parseInt(intValue)] = pointerWalker++;  // fill out pointer array by using the pointerWalker
}
```

O(b)

O(b)

benchFile

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

5 inputs

2 outputs

6 gates

outputs = {_, _}

outputTracker = {_,_}

nodes = {_,_,_,_,_,_,_,_,_,_,_}

nodePointer = {_,0,1,2,_,_,3,4,_,_,5,6,_,_,_,_,7,_,_,8,_,_,9,10}

5+6 = 11 nodes

maxNode = 23

inputFile

```
00000

```

```
Scanner inputFile;
while (inputFile.hasNextLine())
{
    for (int i = 0; i<numberOfInputs; i++)                  // fill out nodes
        nodes[i] = inputString.charAt(i)-'0';
    int nodeWalker = numberOfInputs;                        // do not mess with user input
    int outputWalker = 0;                                   // walker to fill out the output array
    benchFile.reset();
    while (benchFile.hasNextLine())
    {
        if (output)
            add to outputTracker
        else if (gate)
        {
            parse string to determine which gate
            parse string to determine gate inputs
            ArrayList<Integer> gateInputValues = new ArrayList<Integer>(); // use a dynamix sized array list
            for (int i = 0; i<numberOfGateInputs; i++)
                gateInputValues.add(nodes[nodePointer[gateInputs.get(i)]]);

            if (NOT)
                results = !gateInputValues.get(0);          // results = negation of first input
            if (BUFF)
                results = gateInputValues.get(0);           // results = first input
            if (OR)
                results = any input is 1
            if (XOR)
                results = odd # of 1's
            if (NOR)
                results = any input is 1 then negation
            if (AND)
                results = any input is 0 then negation
            if (NAND)
                results = any input is 0

            if (outputTracker.contains(nodeLabel))          // check if it is an output
                outputs[outputWalker++] = results;
            nodes[nodeWalker] = results;
            nodeWalker++;
        }
    }
    print out results
}
```

x

O(i)

outputs = {_, _}

outputTracker = {22,23}

nodes = {0,0,0,0,0,_,_,_,_,_,_}

nodePointer = {_,0,1,2,_,_,3,4,_,_,5,6,_,_,_,_,7,_,_,8,_,_,9,10}

O(1)

O(g)

$"10" = NAND("1","3") = NAND(0,0) = 1$

$"11" = NAND("3","6") = NAND(0,0) = 1$

outputs = {0, 0}

outputTracker = {22,23}

nodes = {0,0,0,0,0,1,1,1,1,0,0}

nodePointer = {_,0,1,2,_,_,3,4,_,_,5,6,_,_,_,_,7,_,_,8,_,_,9,10}

$"16" = NAND("2","11") = NAND(0,1) = 1$

$"19" = NAND("11","7") = NAND(1,0) = 1$

$"22" = NAND("10","16") = NAND(1,1) = 0$

$"23" = NAND("16","19") = NAND(1,1) = 0$

O(1)

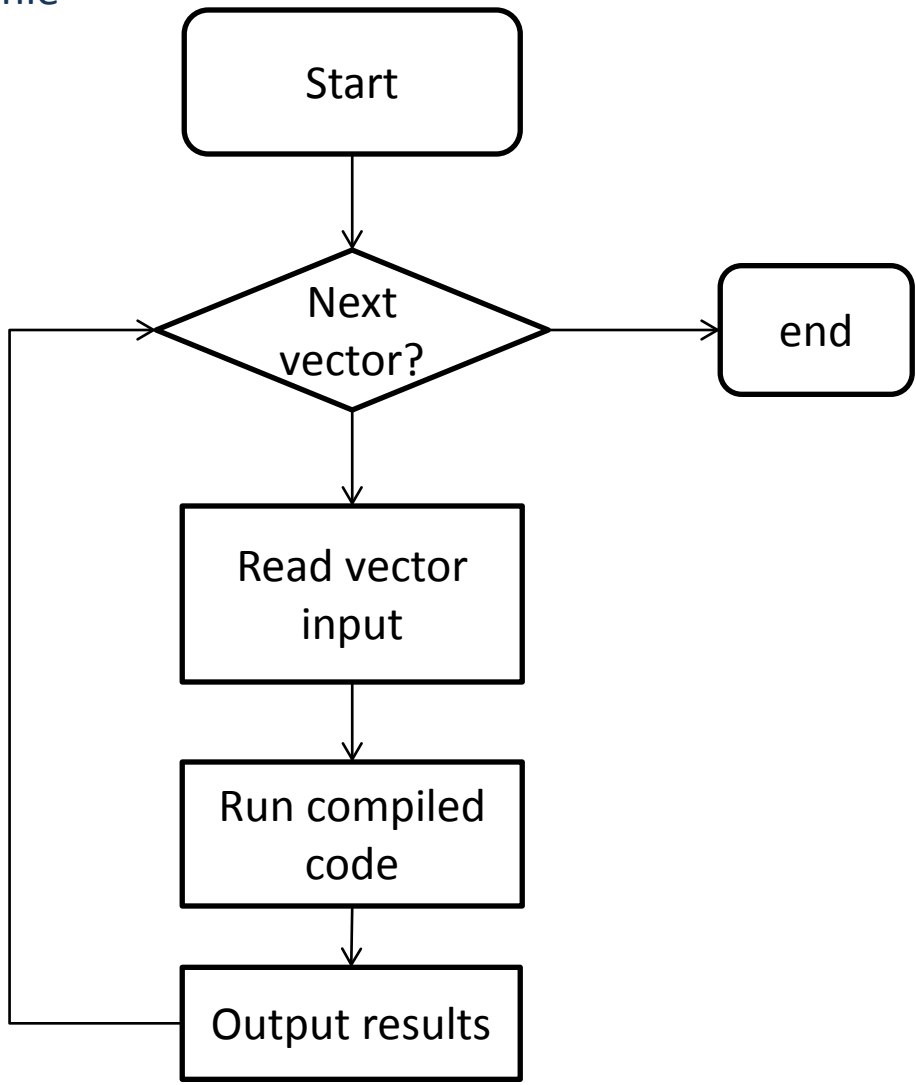
Total *runtime*: $O(2 \times b + x \sum_{node\ i} b_i g_i + x \times o)$

# c17.bench Input/Output Example

Bench file

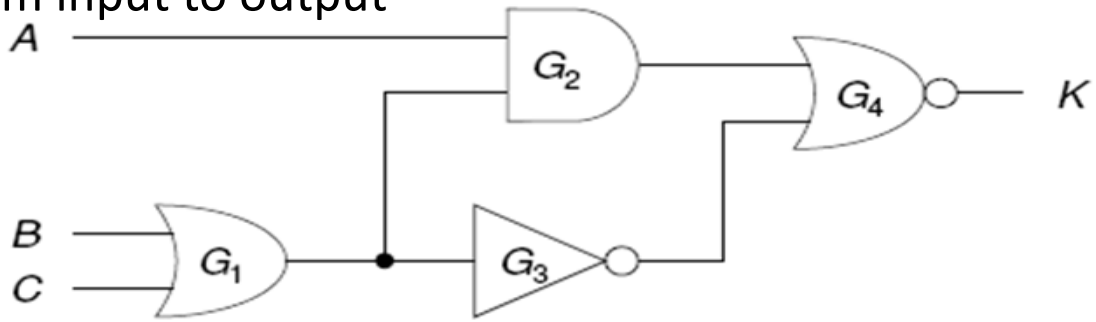
```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )
INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)
OUTPUT(22)
OUTPUT(23)
10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

| Input file | Output file |
|---|---|
| 00000 | 00 |
| 01010 | 11 |
| 10101 | 11 |
| 11000 | 11 |
| 00111 | 00 |
| 11111 | 10 |



# Logic Levelization

- Determine the order of the gate evaluations
- Help get from input to output




start

assign level 0 to all PI's

put all PI fanout gates in *Q*

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |

*Q* empty?

no

yes

end

pop next gate *g* from *Q*

append *g* to *Q*

ready to levelize *g*?

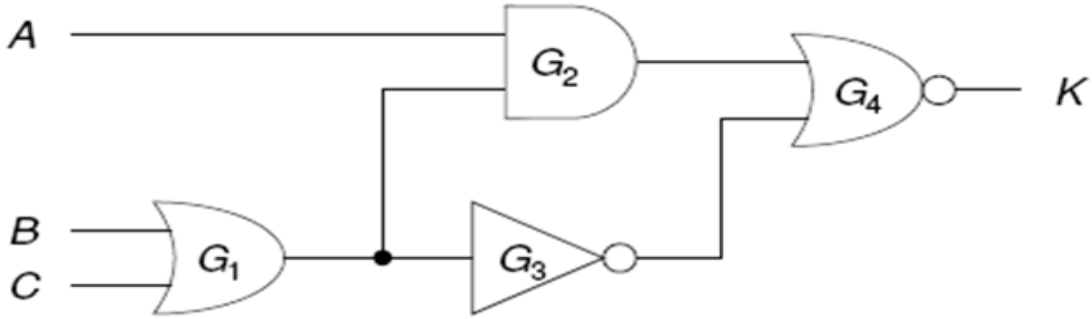
no

yes

1. *l* = maximum of *g*'s driving gate levels
2. assign *l*+1 to *g*

append *g*'s fanout gates to *Q*

# Logic Levelization (Quiz Problem 1 Example)




start

assign level 0 to all PI's

put all PI fanout gates in *Q*

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |

*Q* empty?

no

yes

end

pop next gate *g* from *Q*

append *g* to *Q*

ready to levelize *g*?

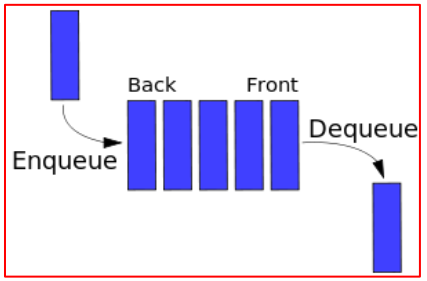
no

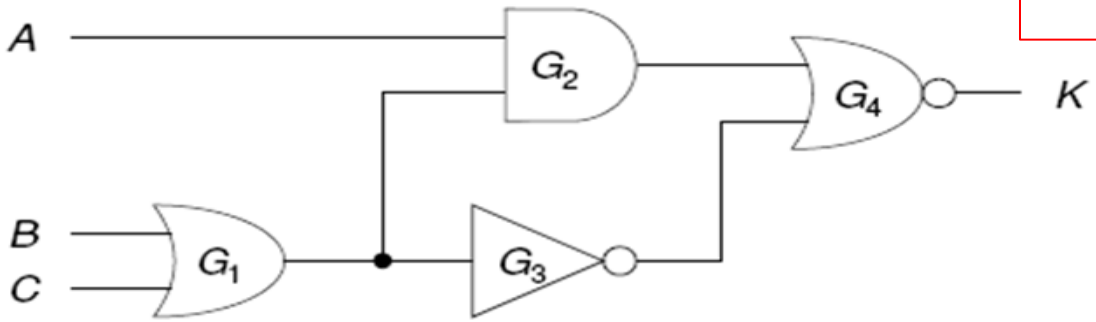
yes

1. *l* = maximum of *g*'s driving gate levels
2. assign *l*+1 to *g*

append *g*'s fanout gates to *Q*

# Logic Levelization




start

assign level 0 to all PI's

put all PI fanout gates in *Q*

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | $<G_2, G_1>$ |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |

Alternative solution will be presented later if you placed $<G_1, G_2>$ in Q

*Q* empty?

no

yes

pop next gate *g* from *Q*

append *g*'s fanout gates to *Q*

1. *l* = maximum of *g*'s driving gate levels
2. assign *l*+1 to *g*

ready to levelize *g*?

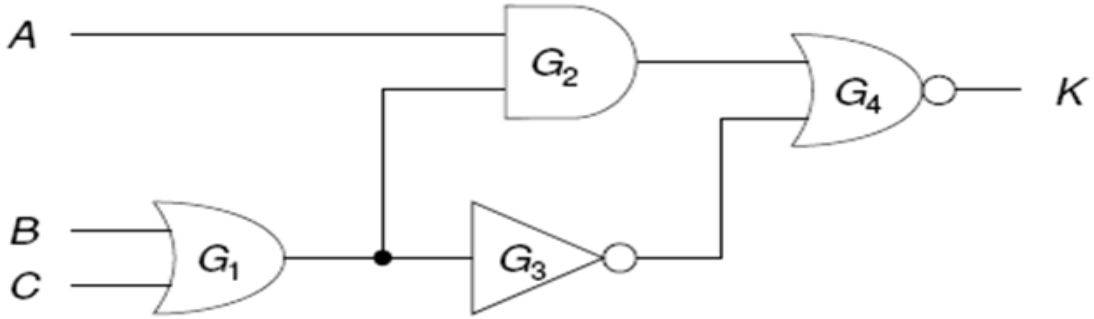
no

yes

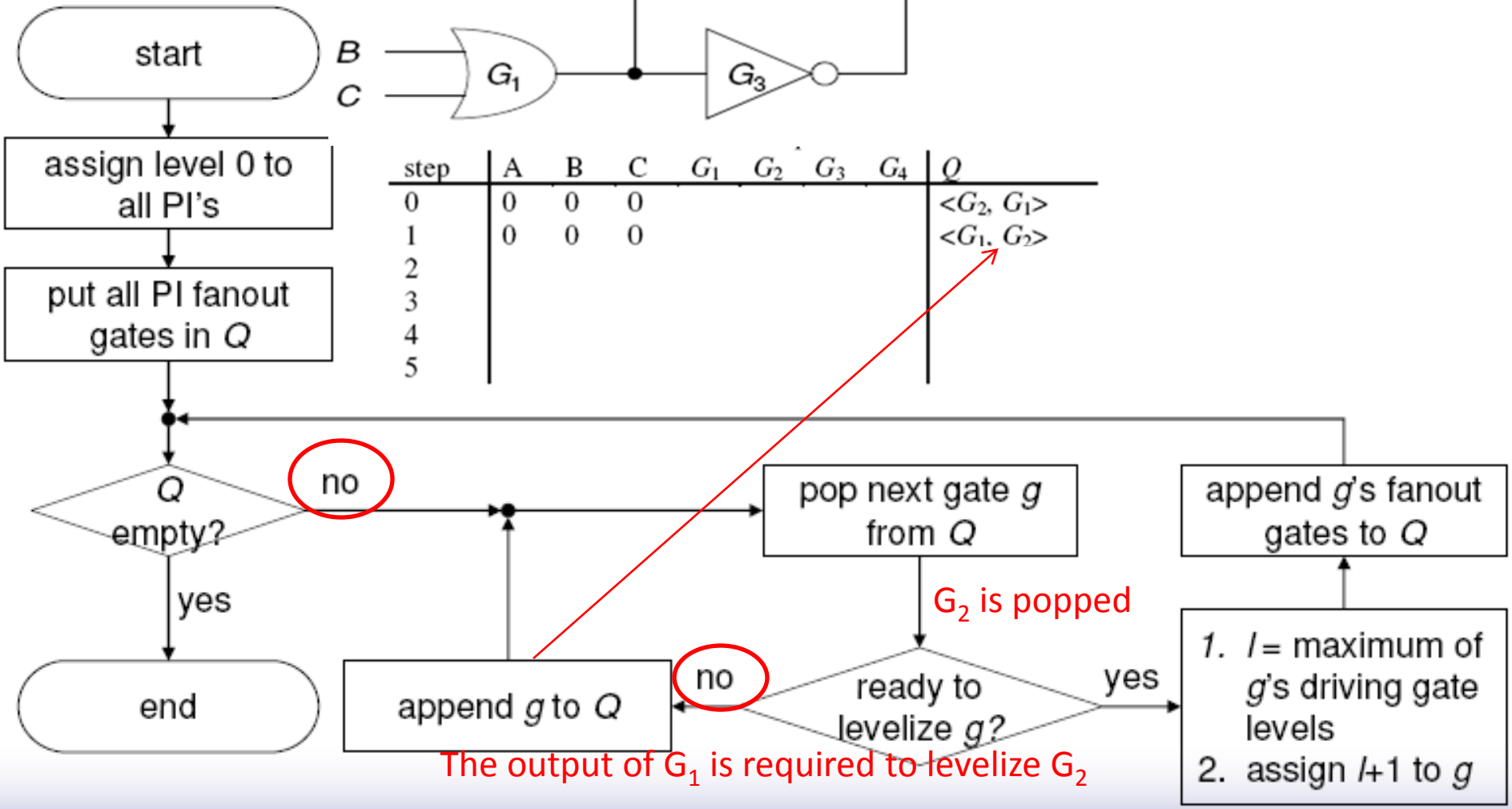
append *g* to *Q*

end

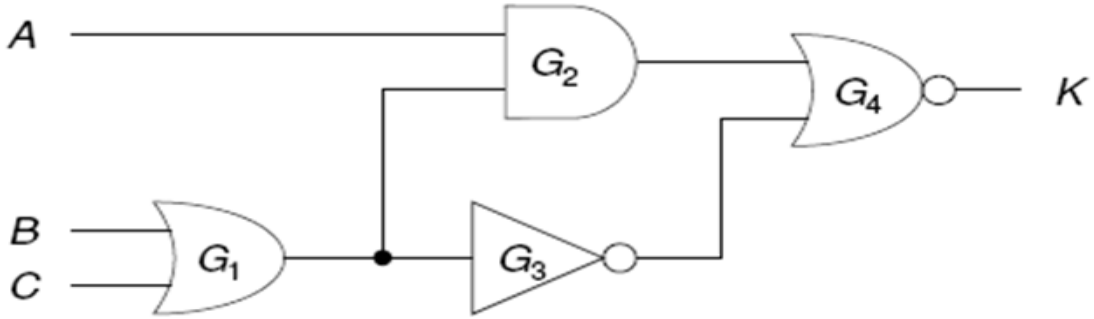
# Logic Levelization



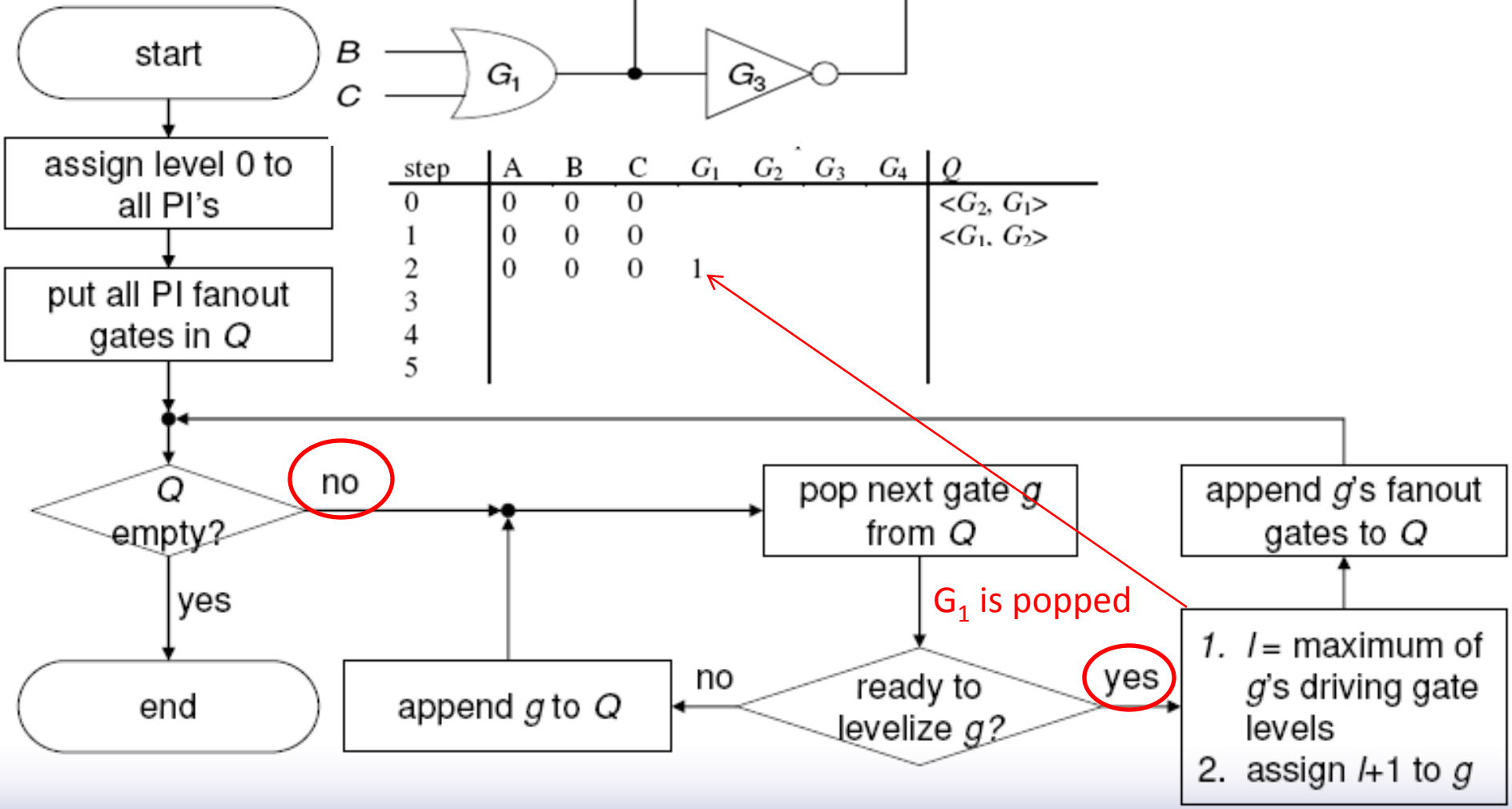

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |



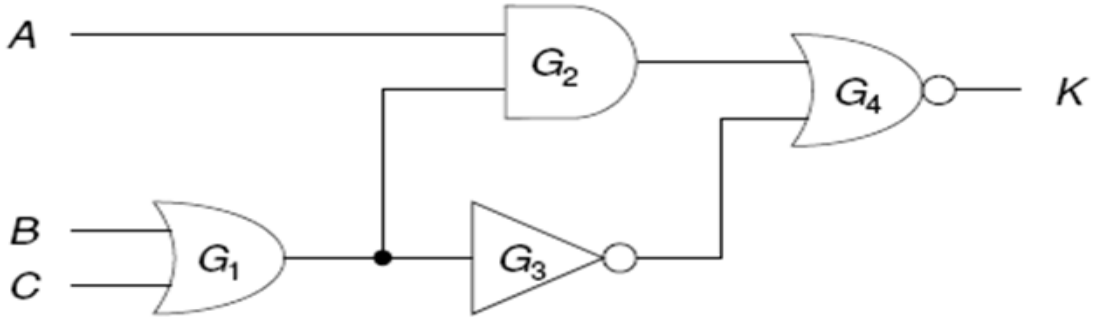
# Logic Levelization



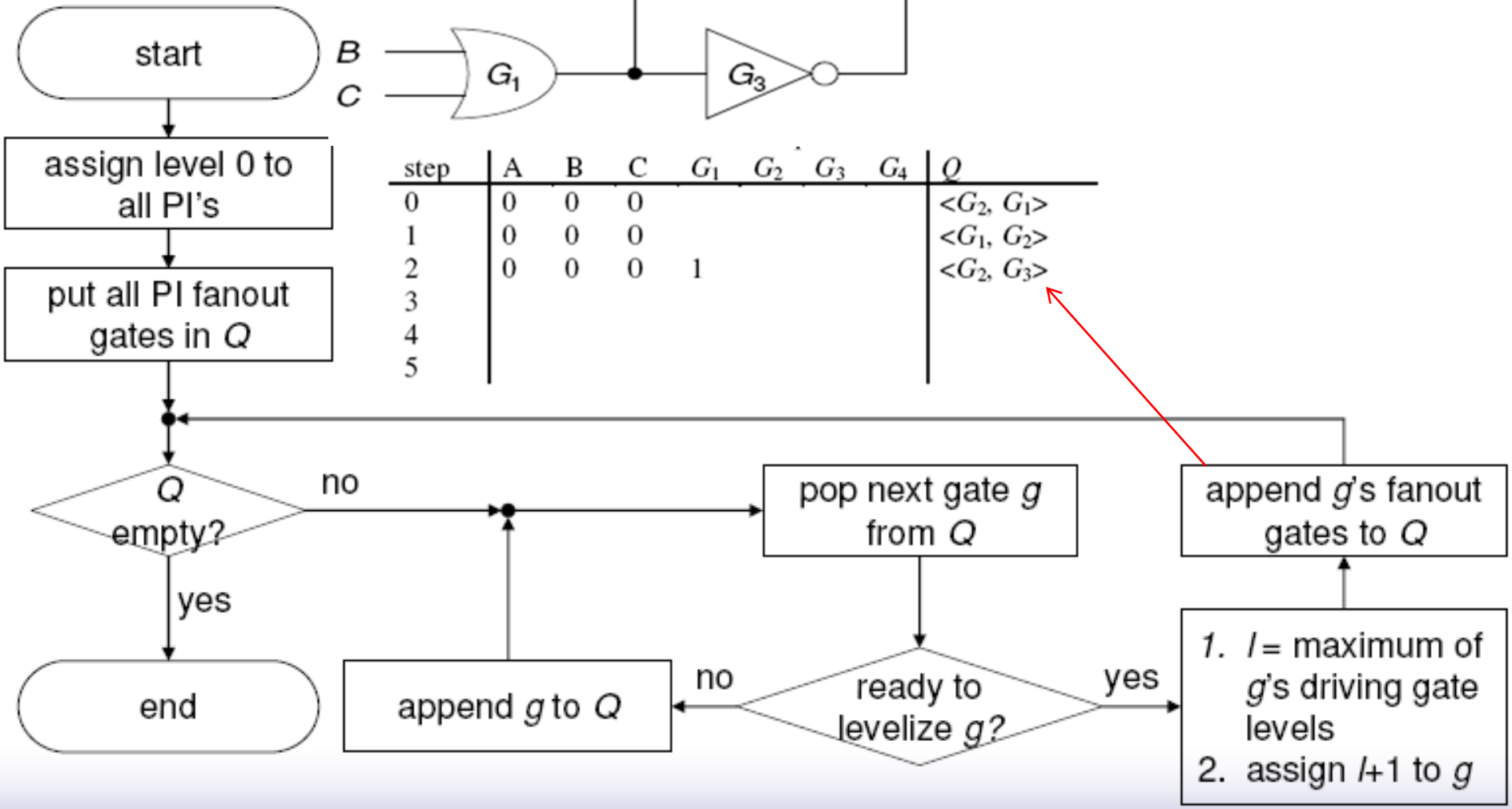

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | $<G_2, G_1>$ |
| 1 | 0 | 0 | 0 | | | | | $<G_1, G_2>$ |
| 2 | 0 | 0 | 0 | 1 | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |



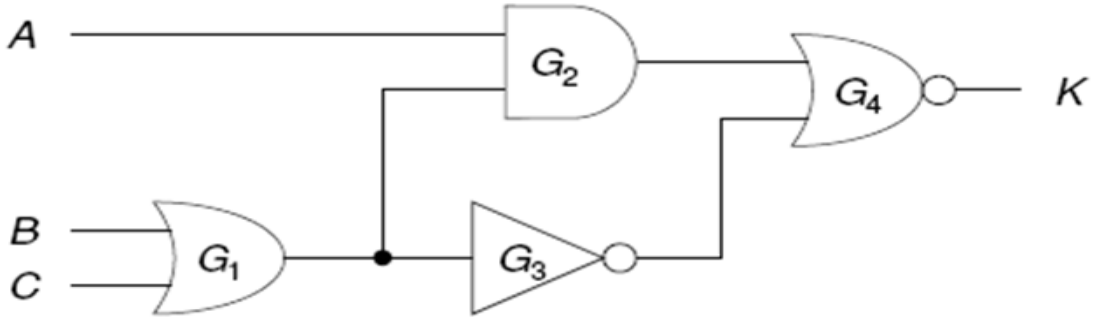
# Logic Levelization



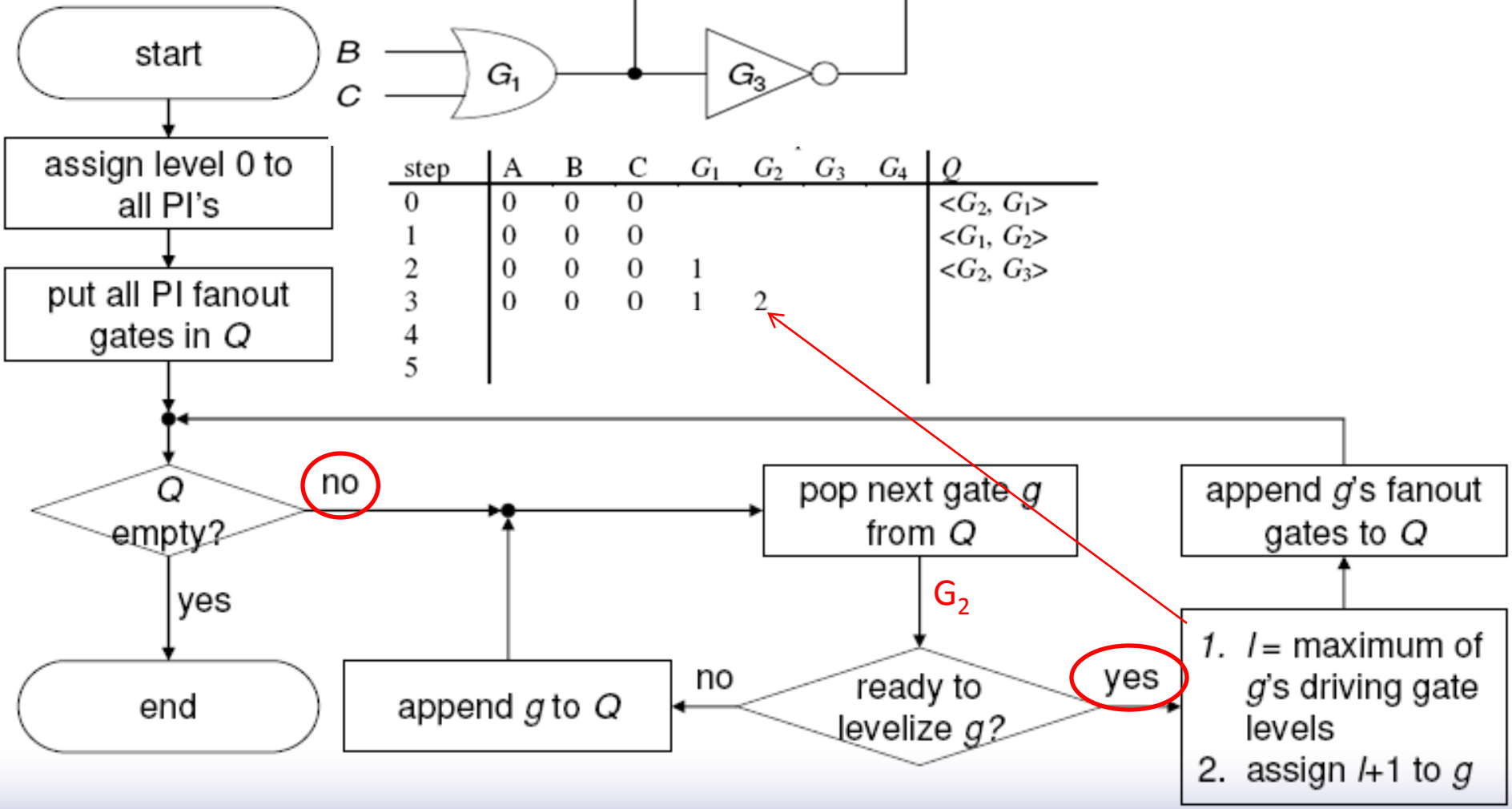

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |



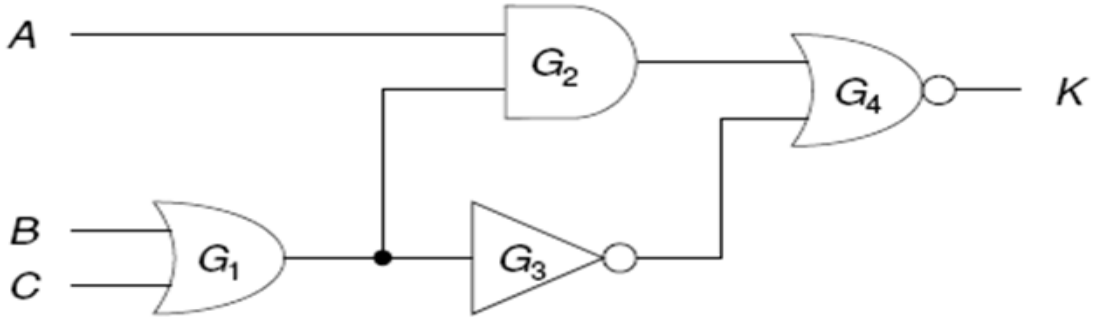
# Logic Levelization



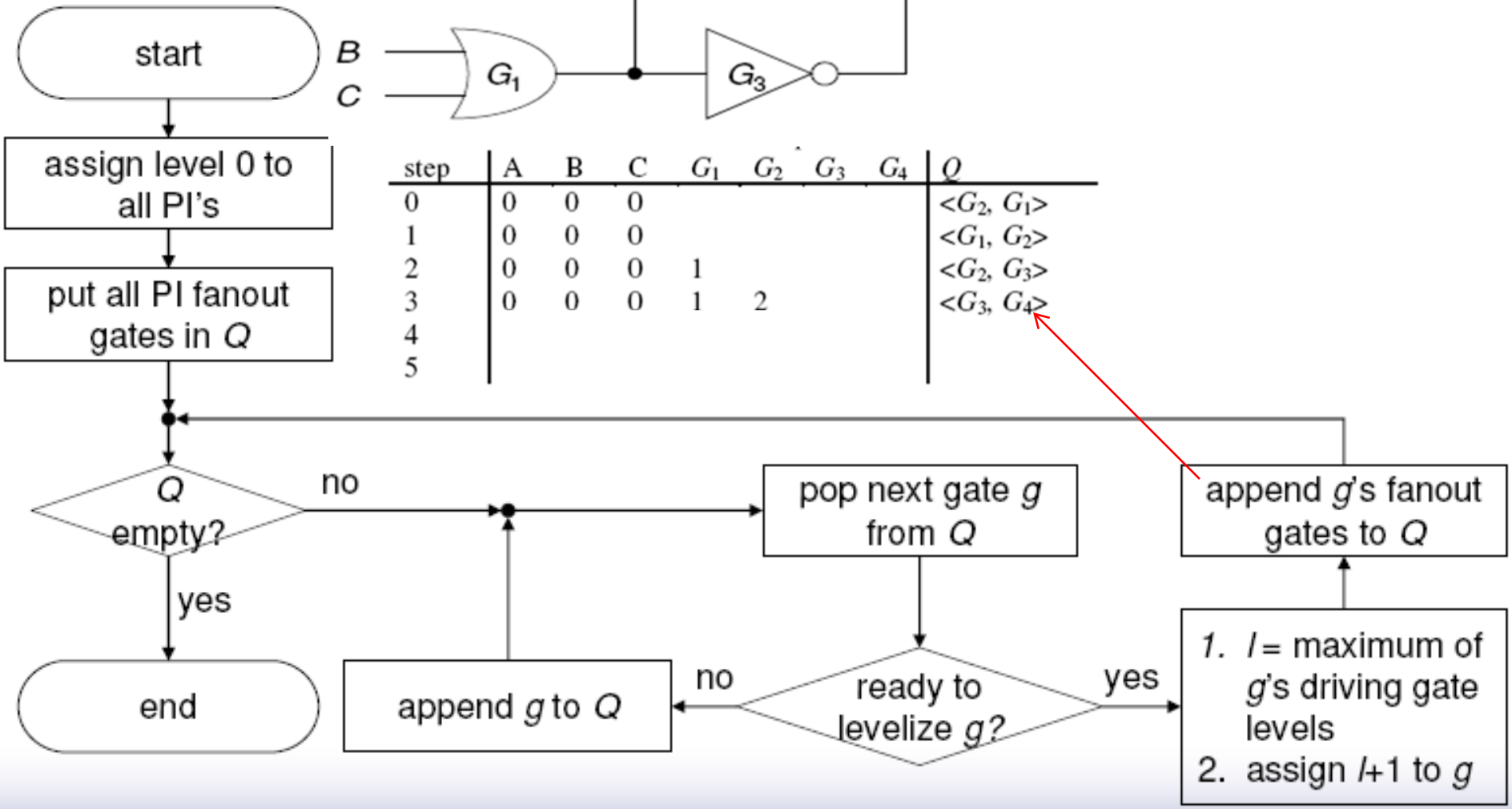

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |



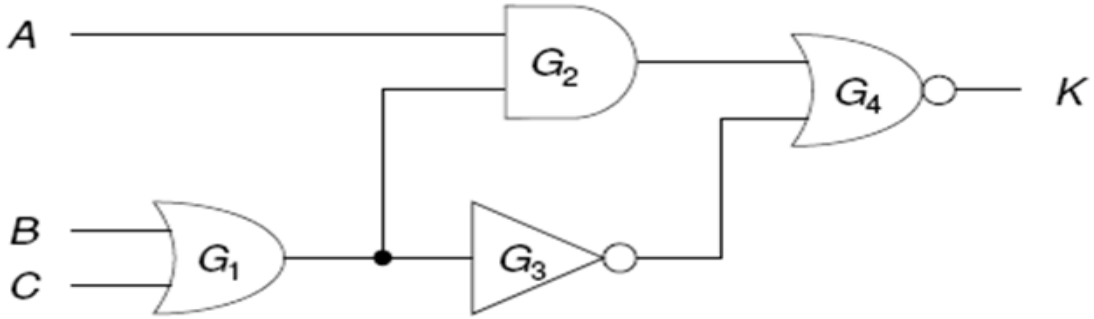
# Logic Levelization



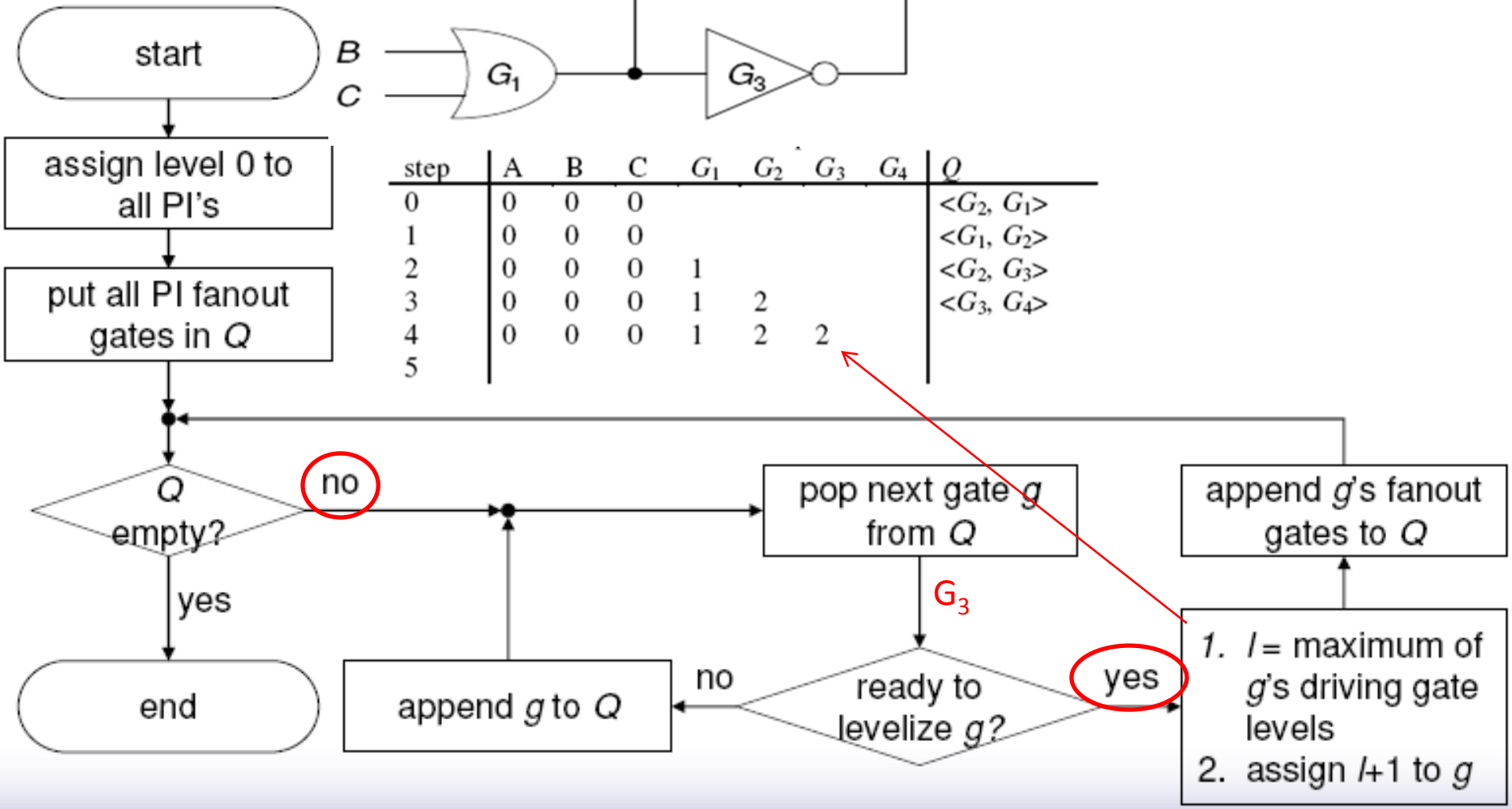

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | | | | | | | | |
| 5 | | | | | | | | |



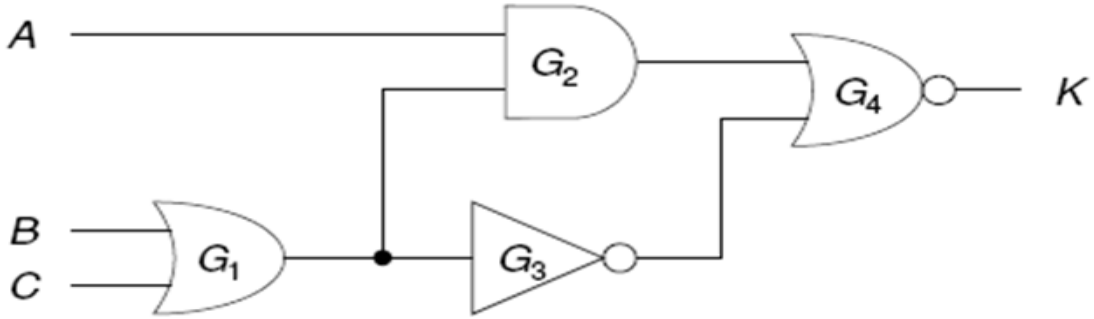
# Logic Levelization



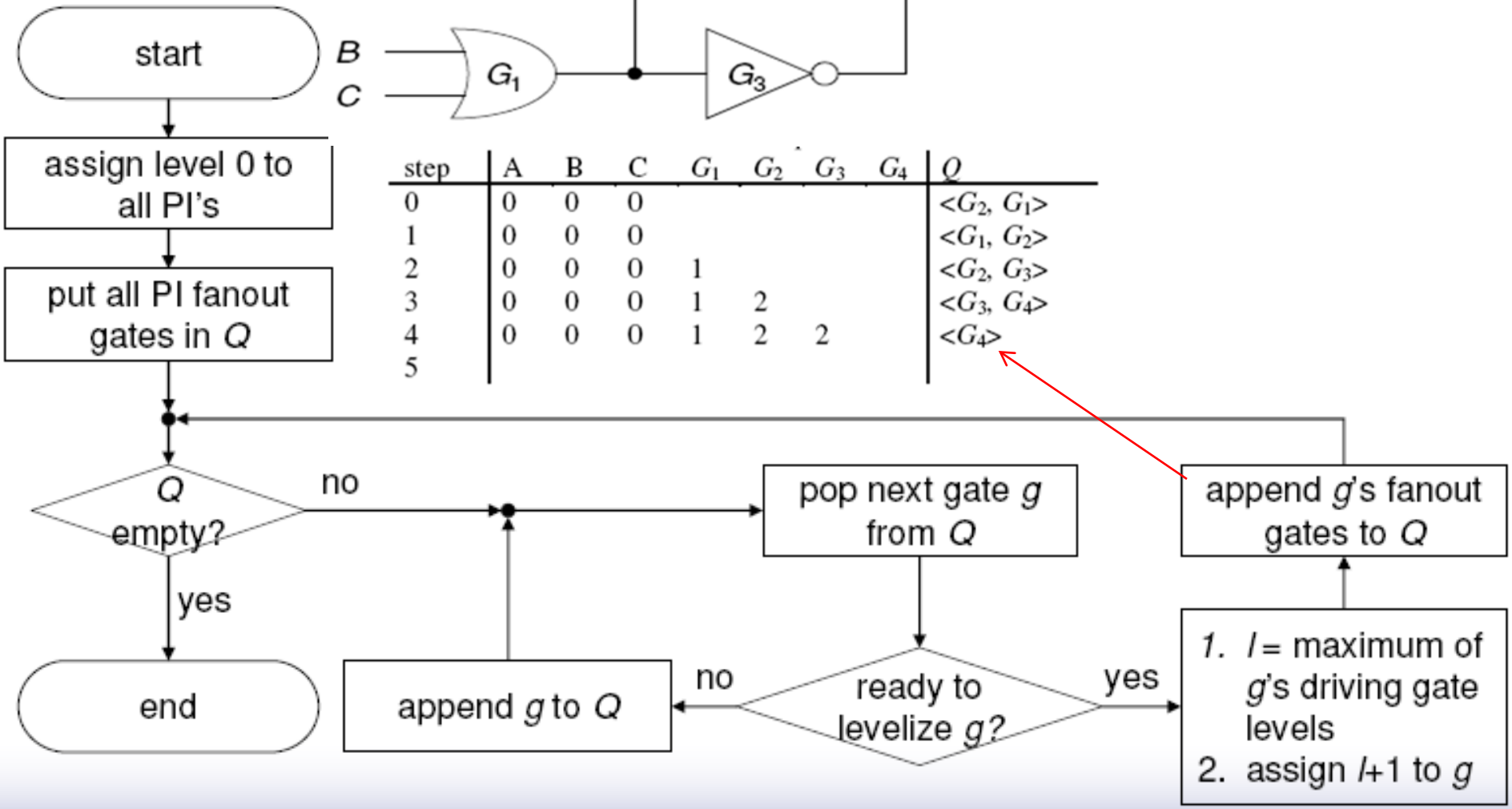

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | |
| 5 | | | | | | | | |



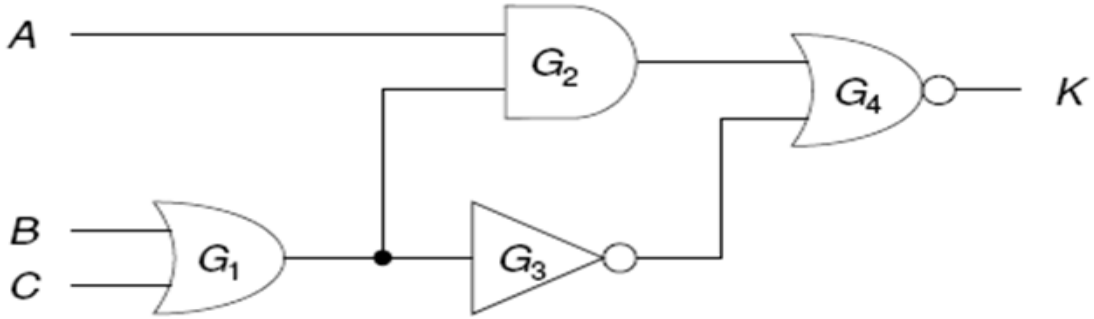
# Logic Levelization



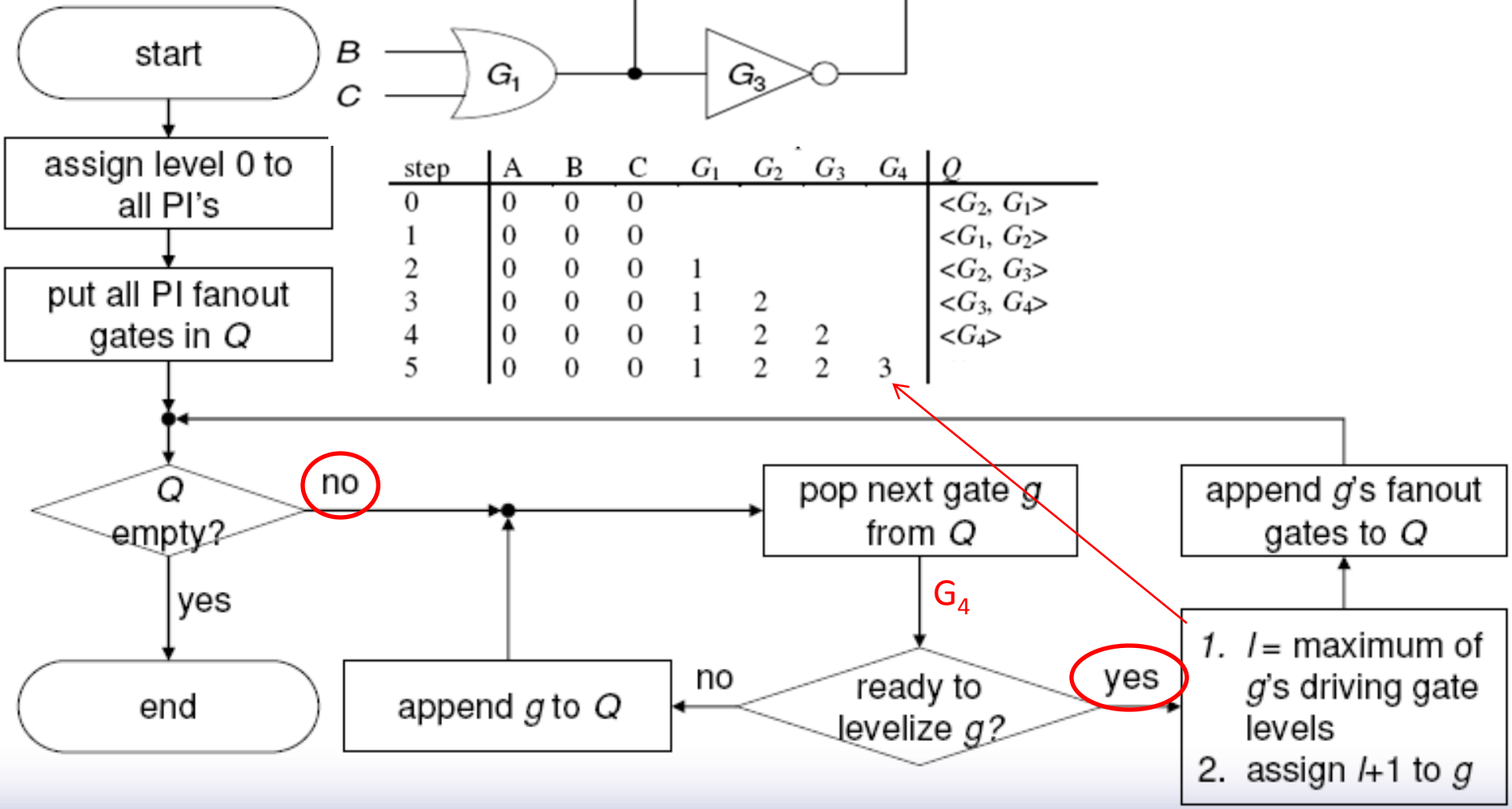

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 5 | | | | | | | | |



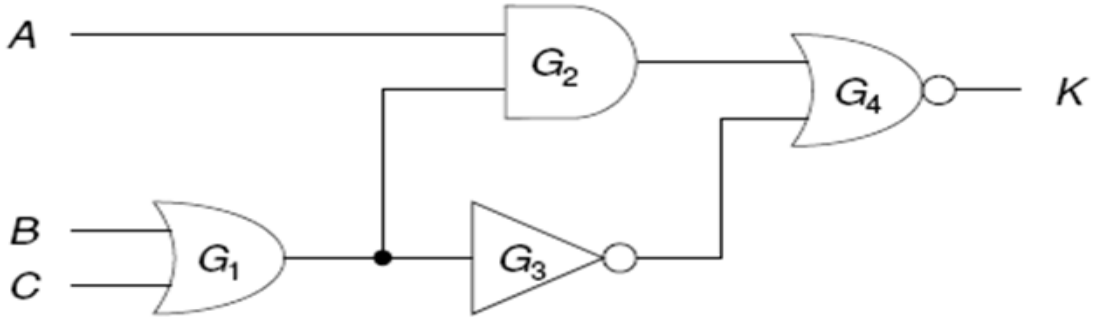
# Logic Levelization



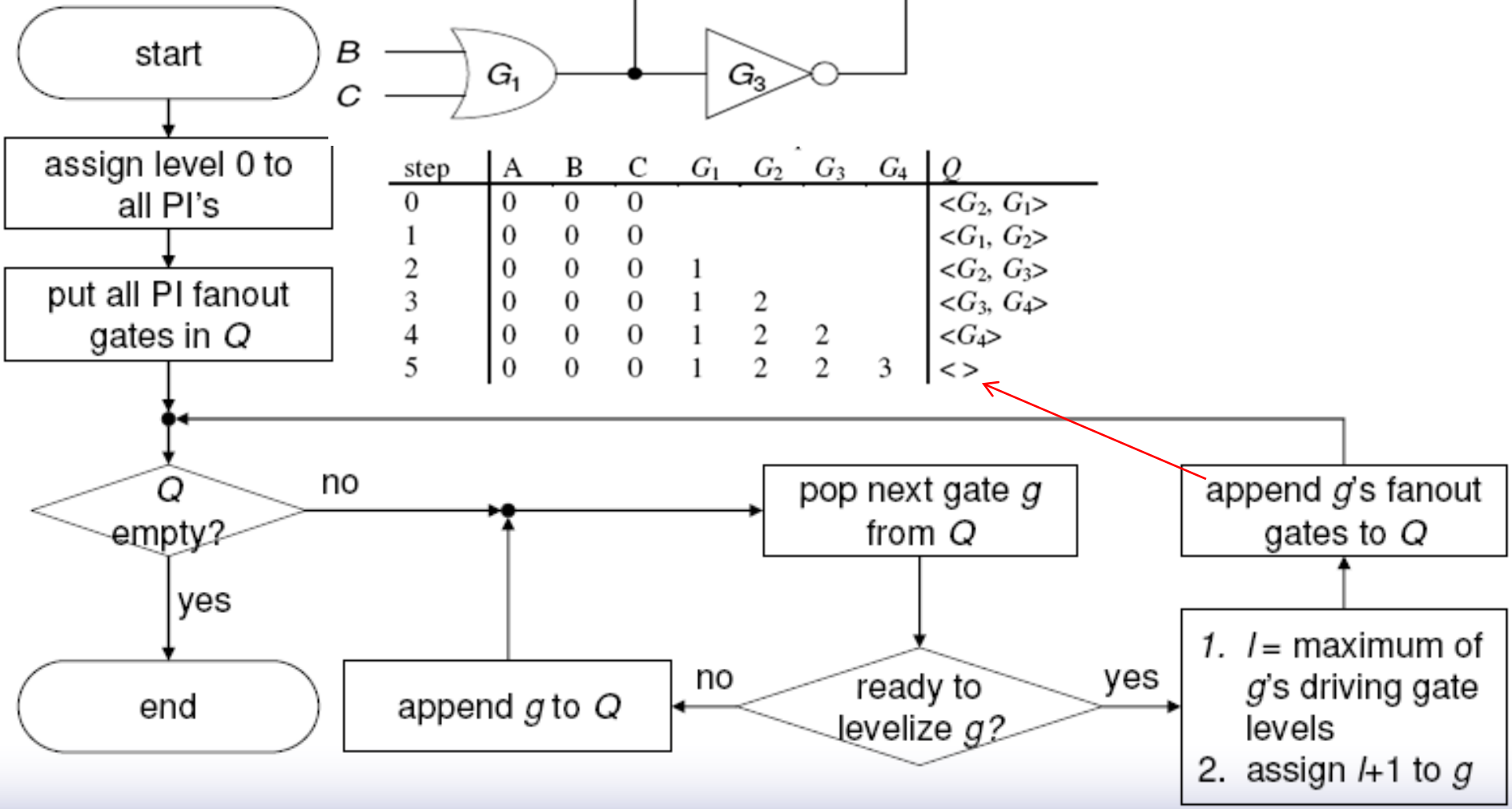

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 5 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | |



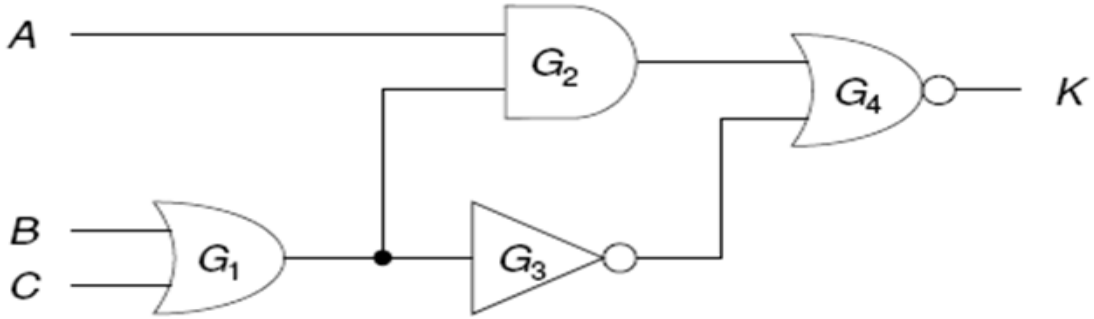
# Logic Levelization



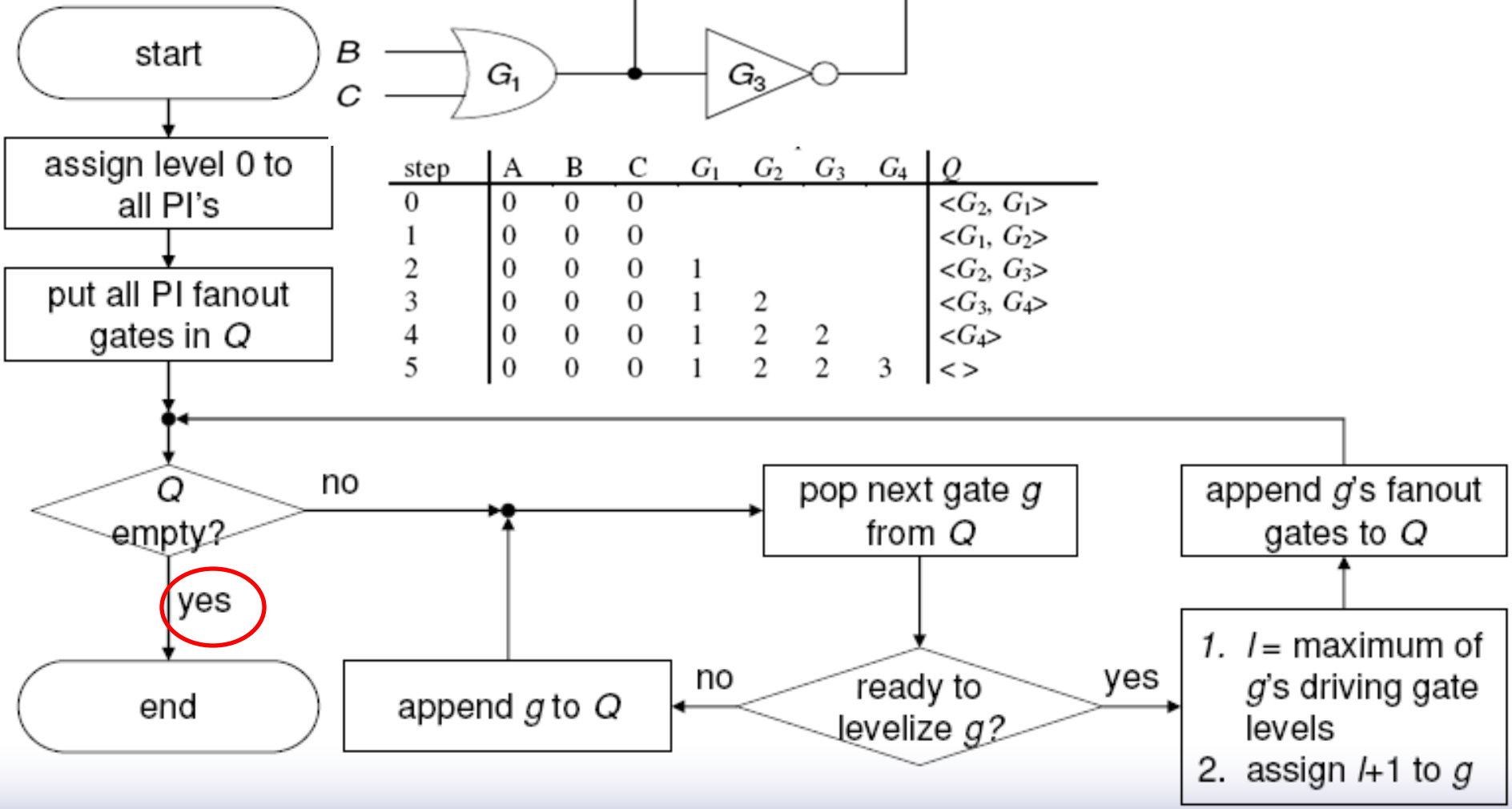

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 5 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | < > |



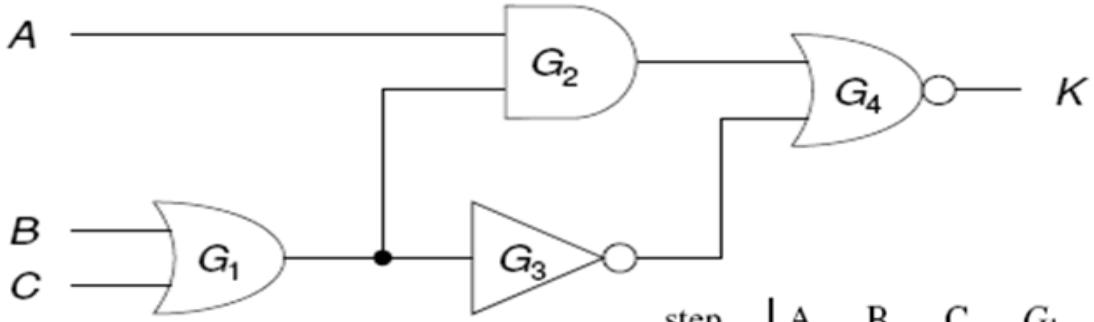
# Logic Levelization




| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 5 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | < > |



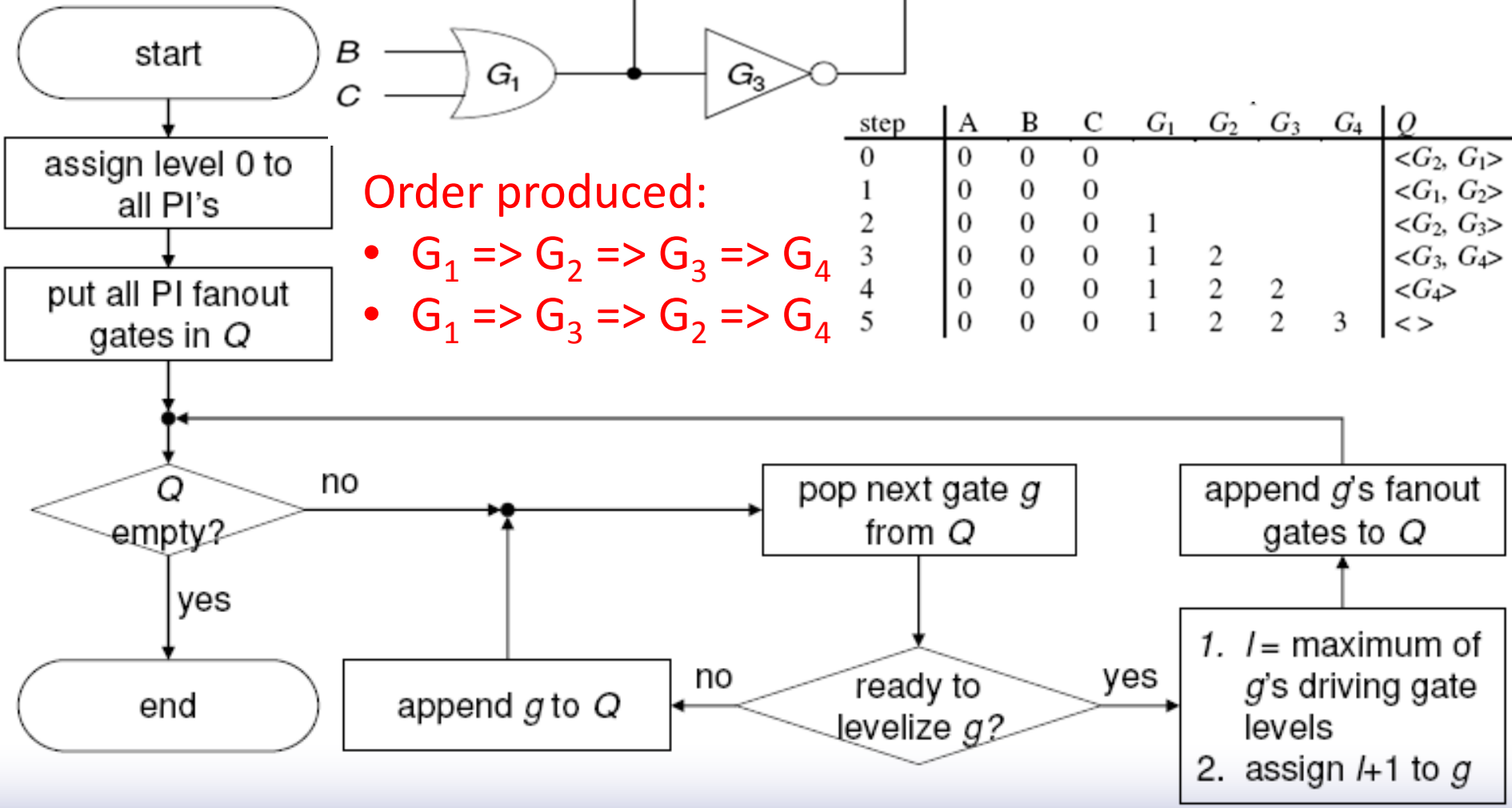
# Logic Levelization




| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 5 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | < > |

Order produced:

- $G_1$ => $G_2$ => $G_3$ => $G_4$
- $G_1$ => $G_3$ => $G_2$ => $G_4$

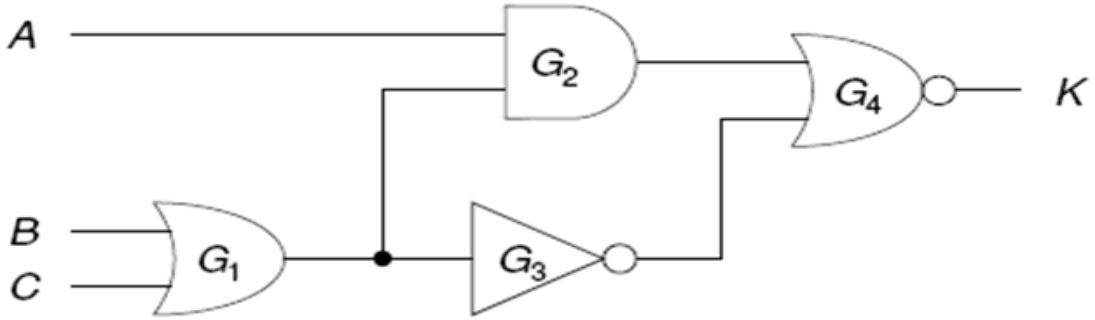


# Logic Levelization

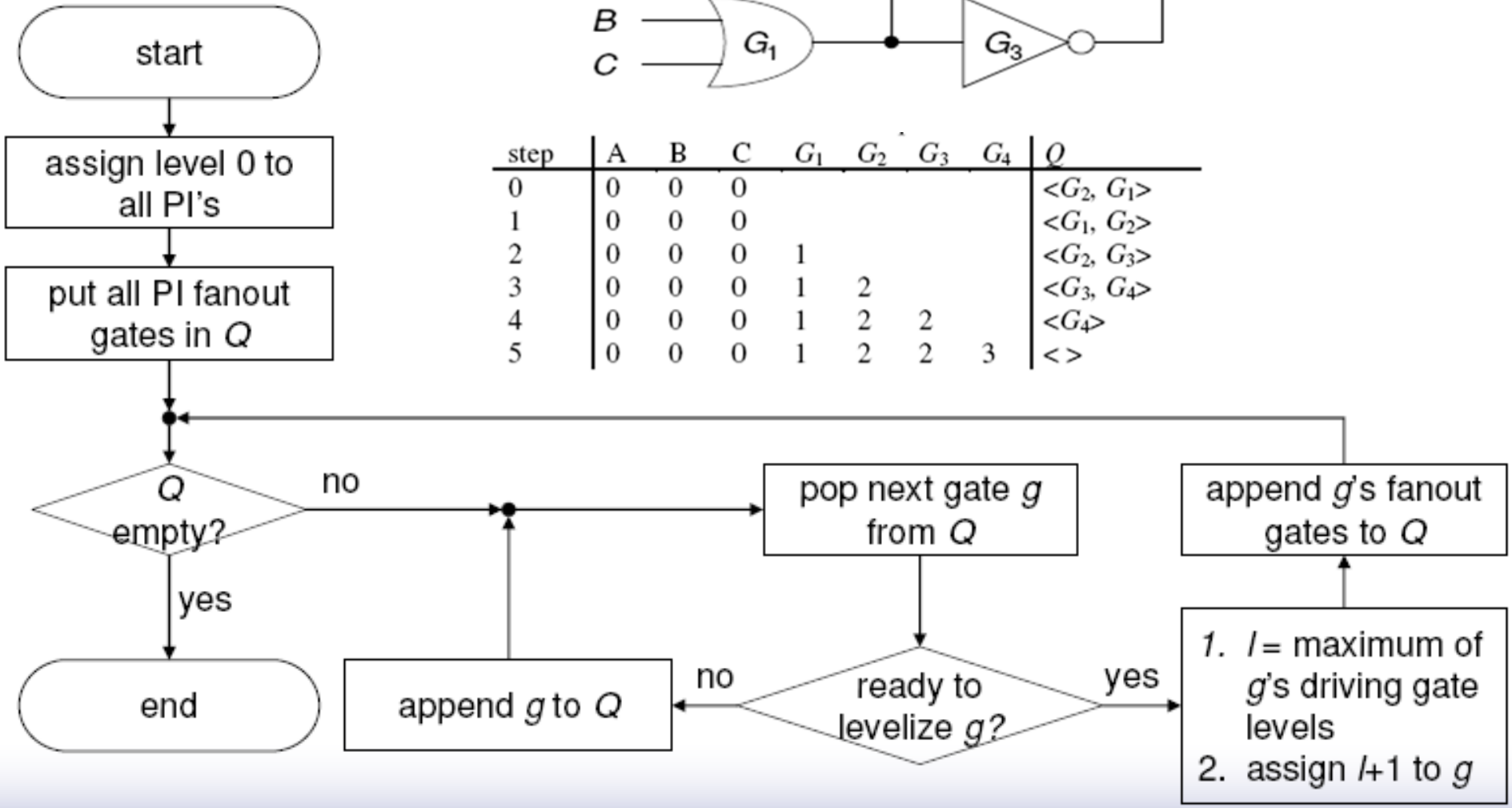
Order produced:

- $G_1 \Rightarrow G_2 \Rightarrow G_3 \Rightarrow G_4$
- $G_1 \Rightarrow G_3 \Rightarrow G_2 \Rightarrow G_4$

The order lets the machine know in which order to compute the gates.



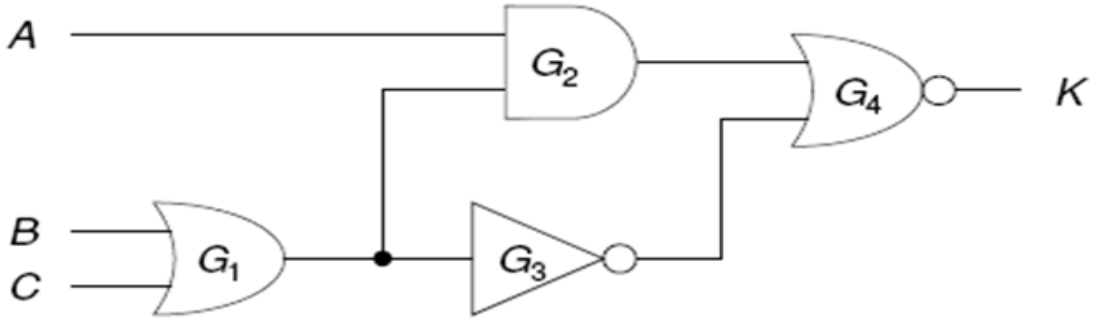

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $Q$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_2$, $G_1$> |
| 1 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 2 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 3 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 5 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | < > |



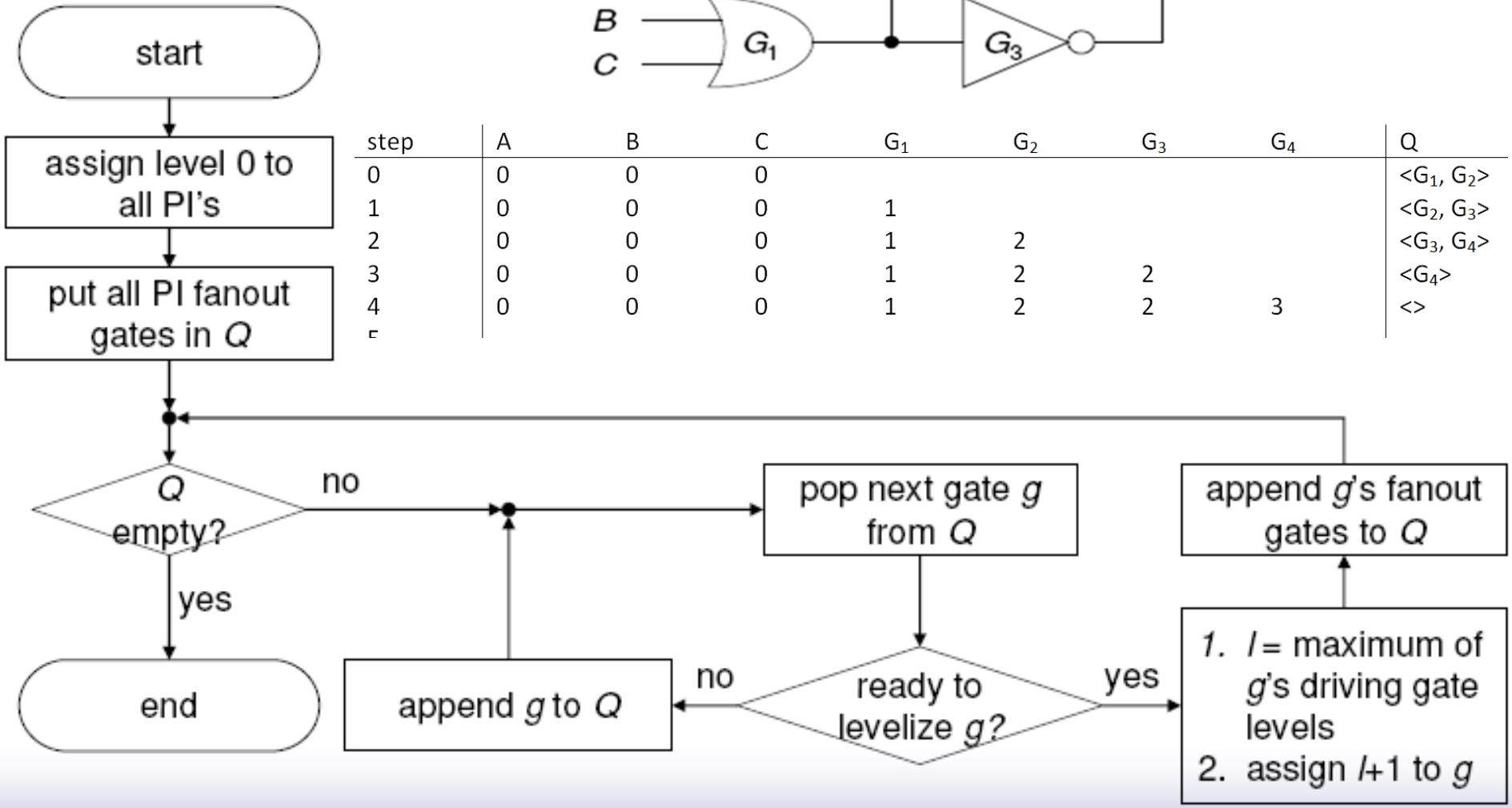
# Logic Levelization

If you started with <$G_2$, $G_1$> in Q, then you will end up with the same order, but with 1 step less

- $G_1$ => $G_2$ => $G_3$ => $G_4$
- $G_1$ => $G_3$ => $G_2$ => $G_4$

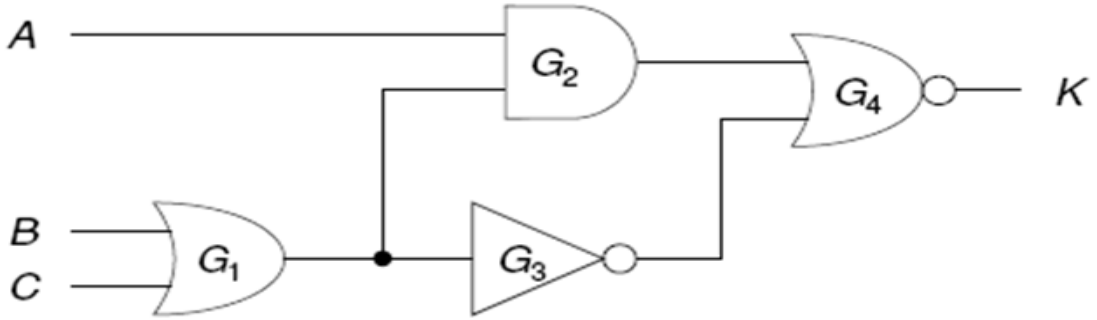


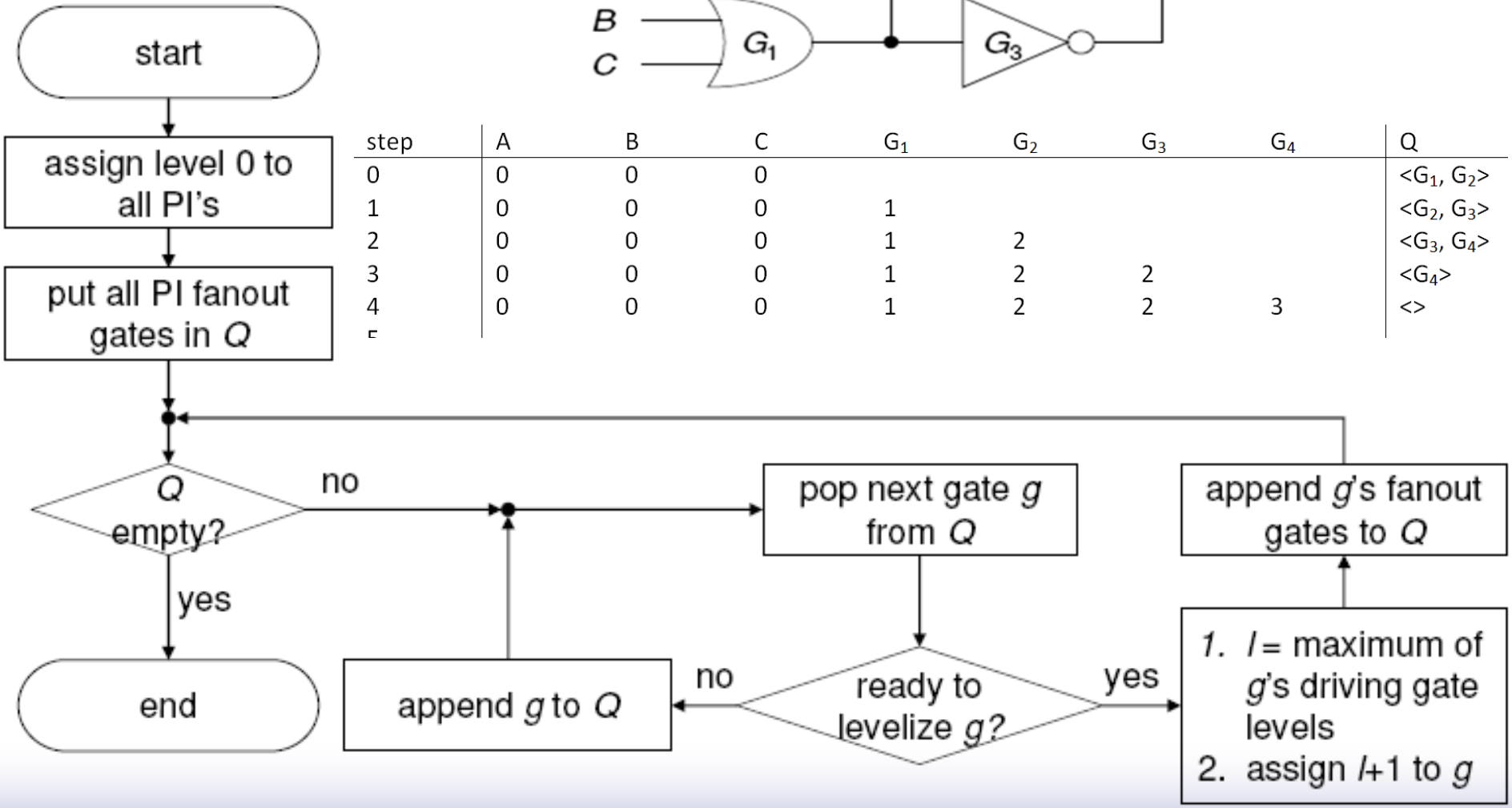

| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 1 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 2 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 3 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | <> |



# Logic Levelization

Questions?
Algorithm?

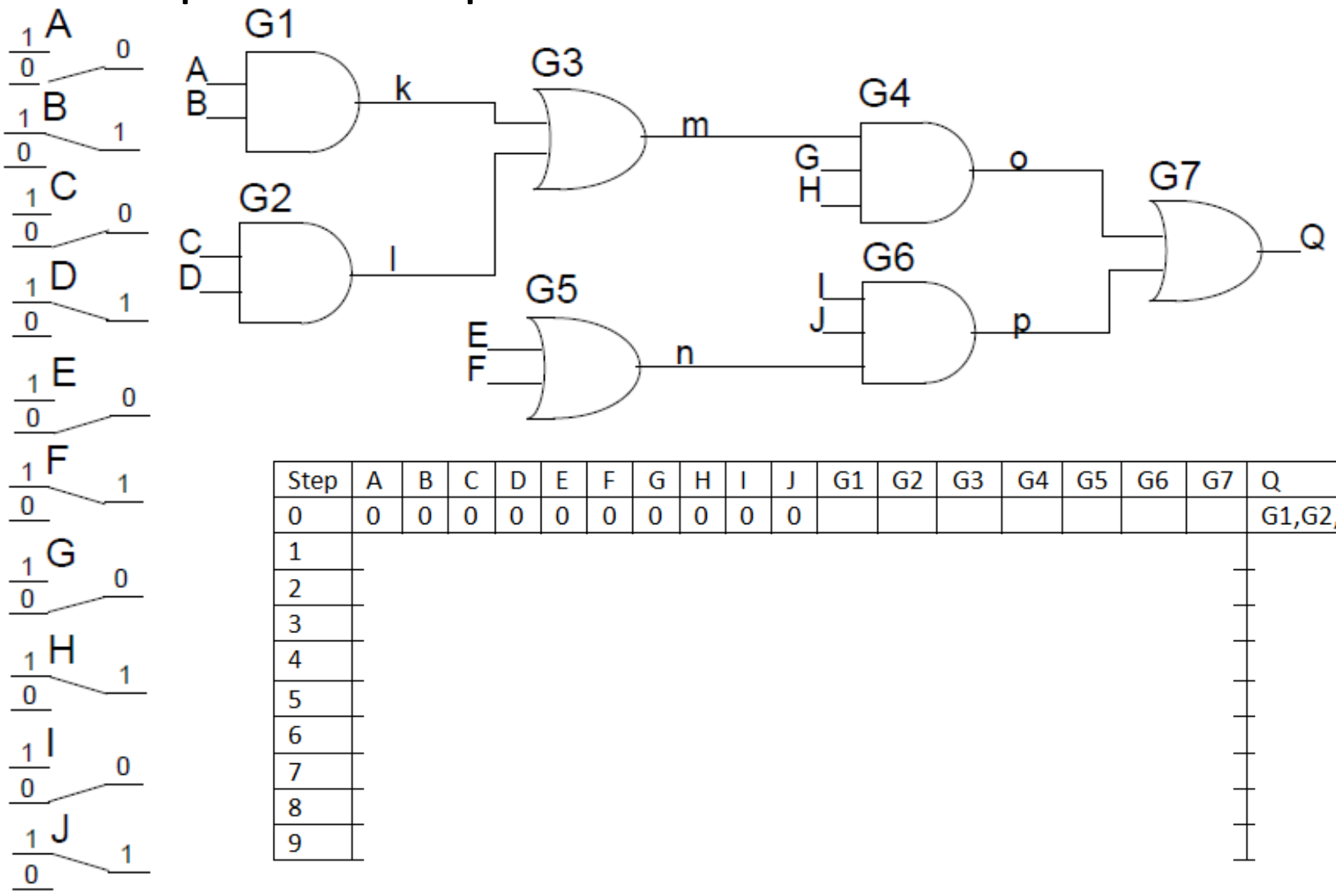



| step | A | B | C | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 1 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$> |
| 2 | 0 | 0 | 0 | 1 | 2 | | | <$G_3$, $G_4$> |
| 3 | 0 | 0 | 0 | 1 | 2 | 2 | | <$G_4$> |
| 4 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | <> |

# Quiz Problem 1 Practice 1

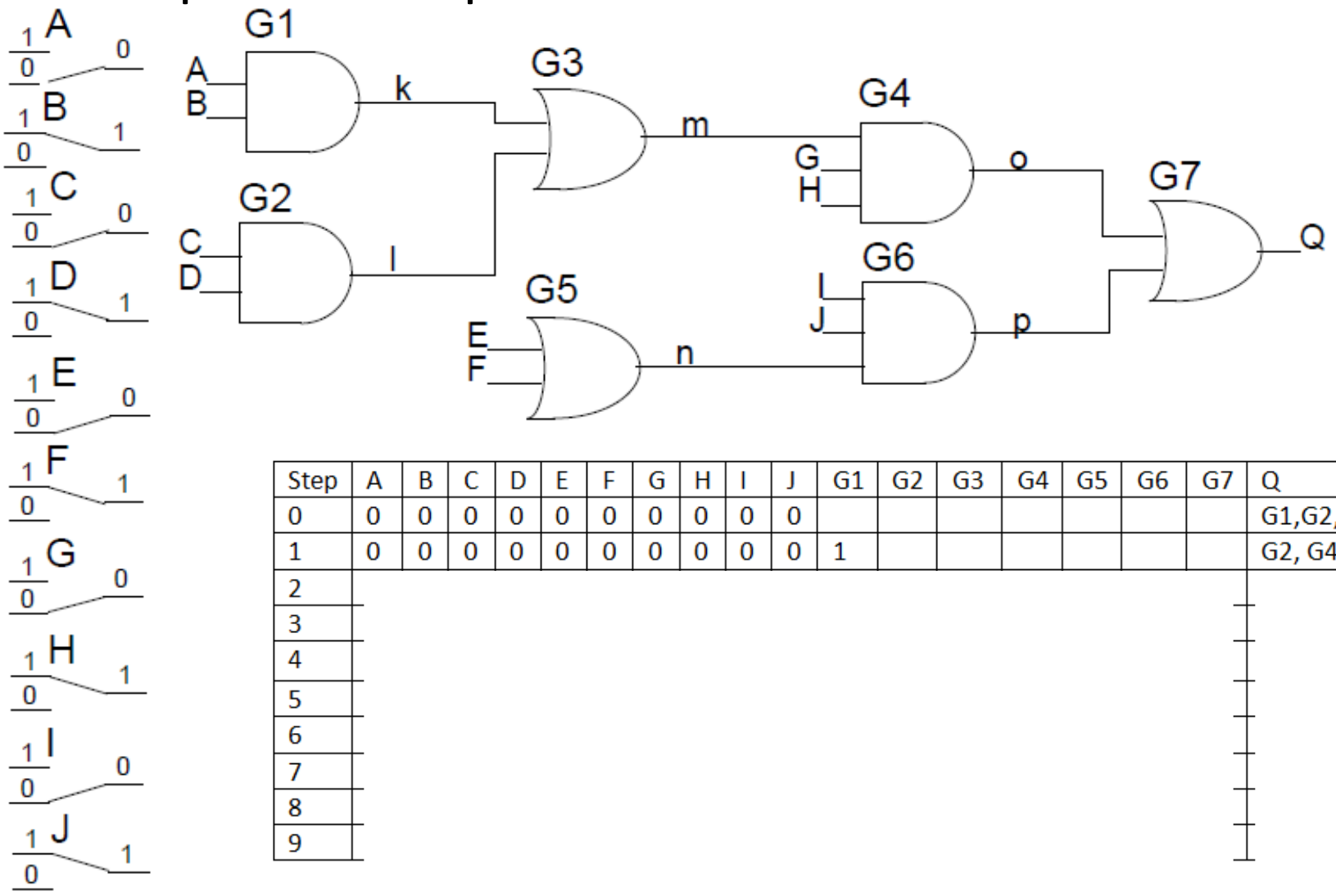
Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101




| Step | A | B | C | D | E | F | G | H | I | J | G1 | G2 | G3 | G4 | G5 | G6 | G7 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | G1,G2, G4, G5, G6 |
| 1 | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | |

# Quiz Problem 1 Practice 1

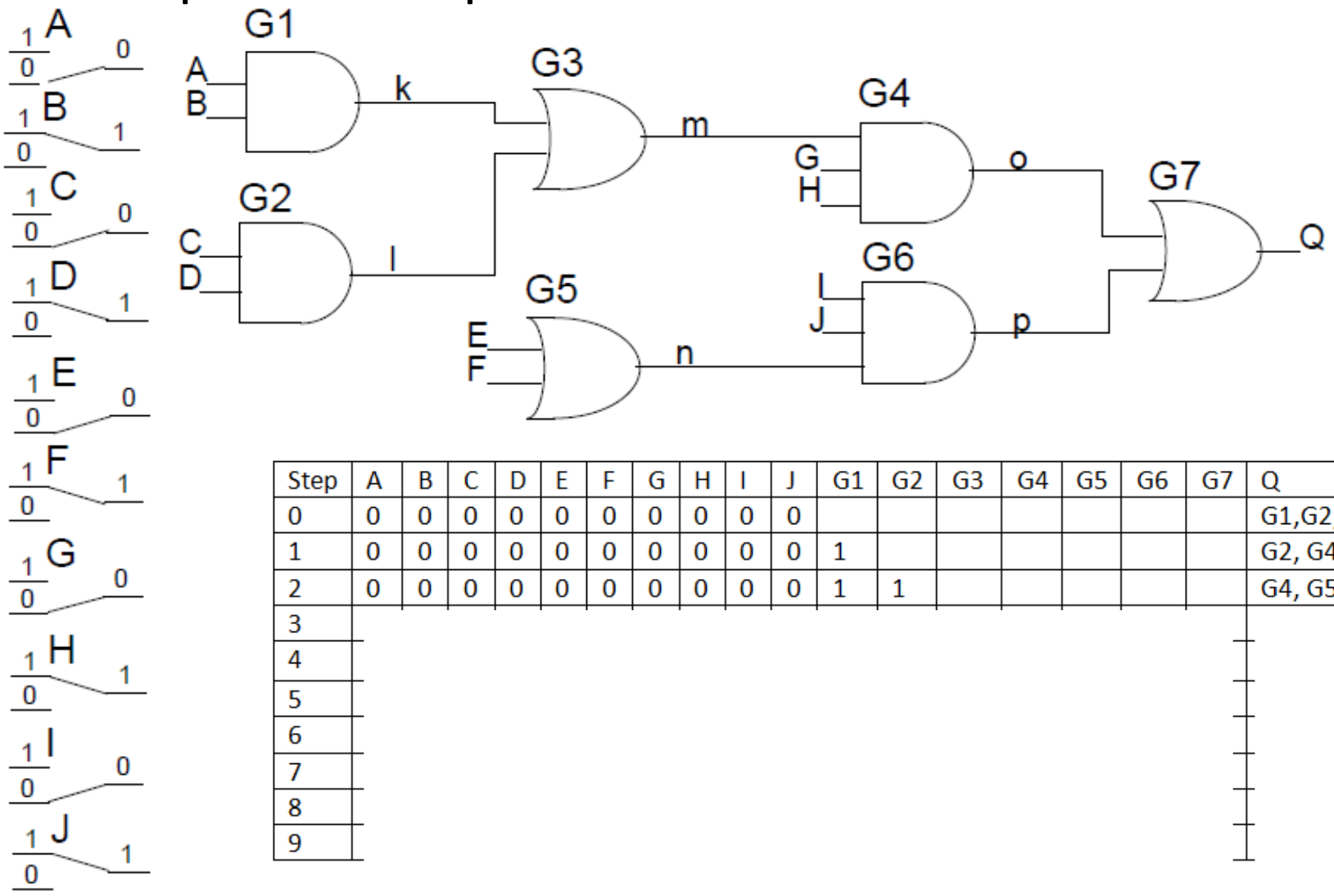
Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101




| Step | A | B | C | D | E | F | G | H | I | J | G1 | G2 | G3 | G4 | G5 | G6 | G7 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | G1,G2, G4, G5, G6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | G2, G4, G5, G6, G3 |
| 2 | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | |

# Quiz Problem 1 Practice 1

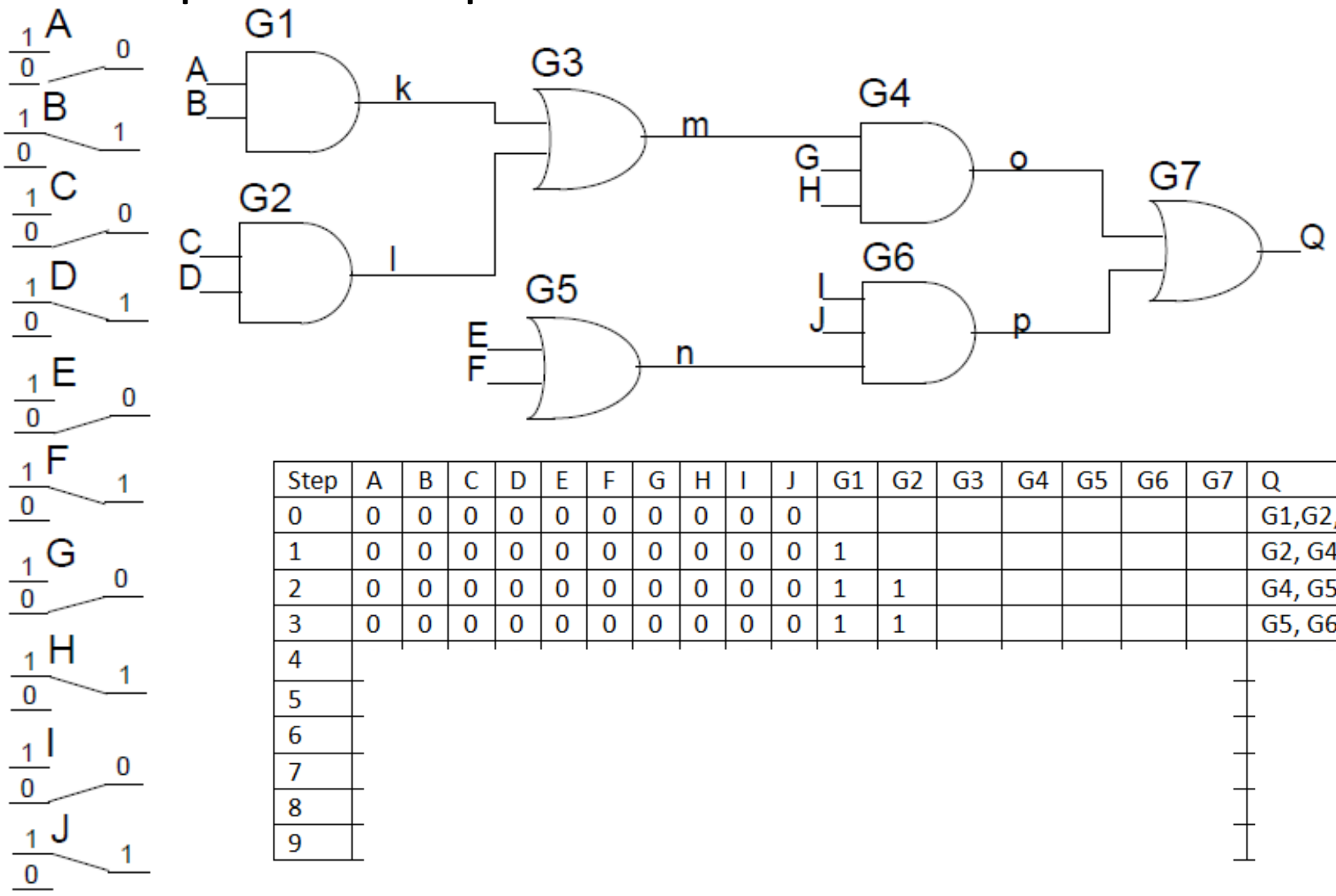
Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101




| Step | A | B | C | D | E | F | G | H | I | J | G1 | G2 | G3 | G4 | G5 | G6 | G7 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | G1,G2, G4, G5, G6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | G2, G4, G5, G6, G3 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G4, G5, G6, G3 |
| 3 | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | |

# Quiz Problem 1 Practice 1

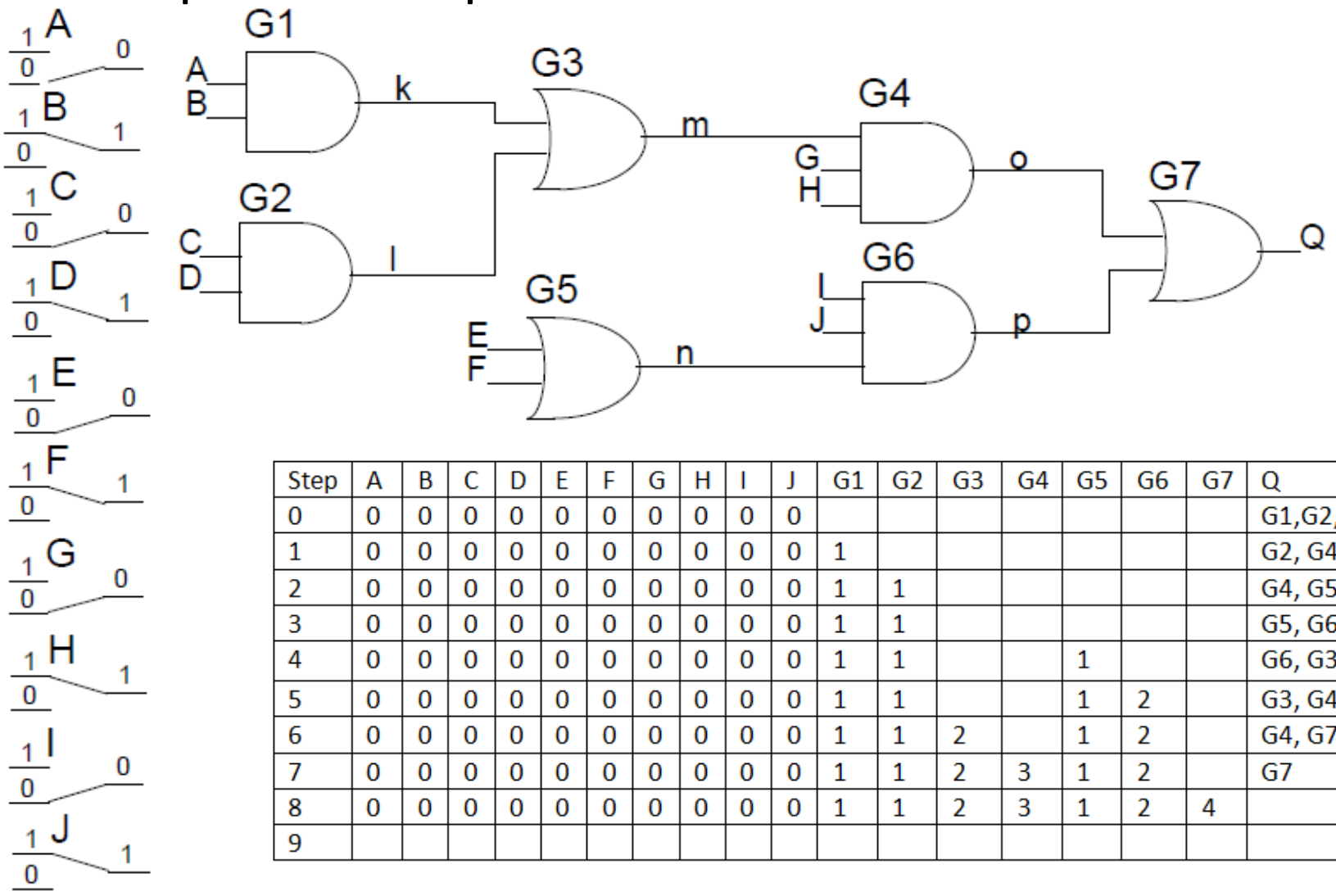
Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101




| Step | A | B | C | D | E | F | G | H | I | J | G1 | G2 | G3 | G4 | G5 | G6 | G7 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | G1,G2, G4, G5, G6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | G2, G4, G5, G6, G3 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G4, G5, G6, G3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G5, G6, G3, G4 |
| 4 | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | |

# Quiz Problem 1 Practice 1

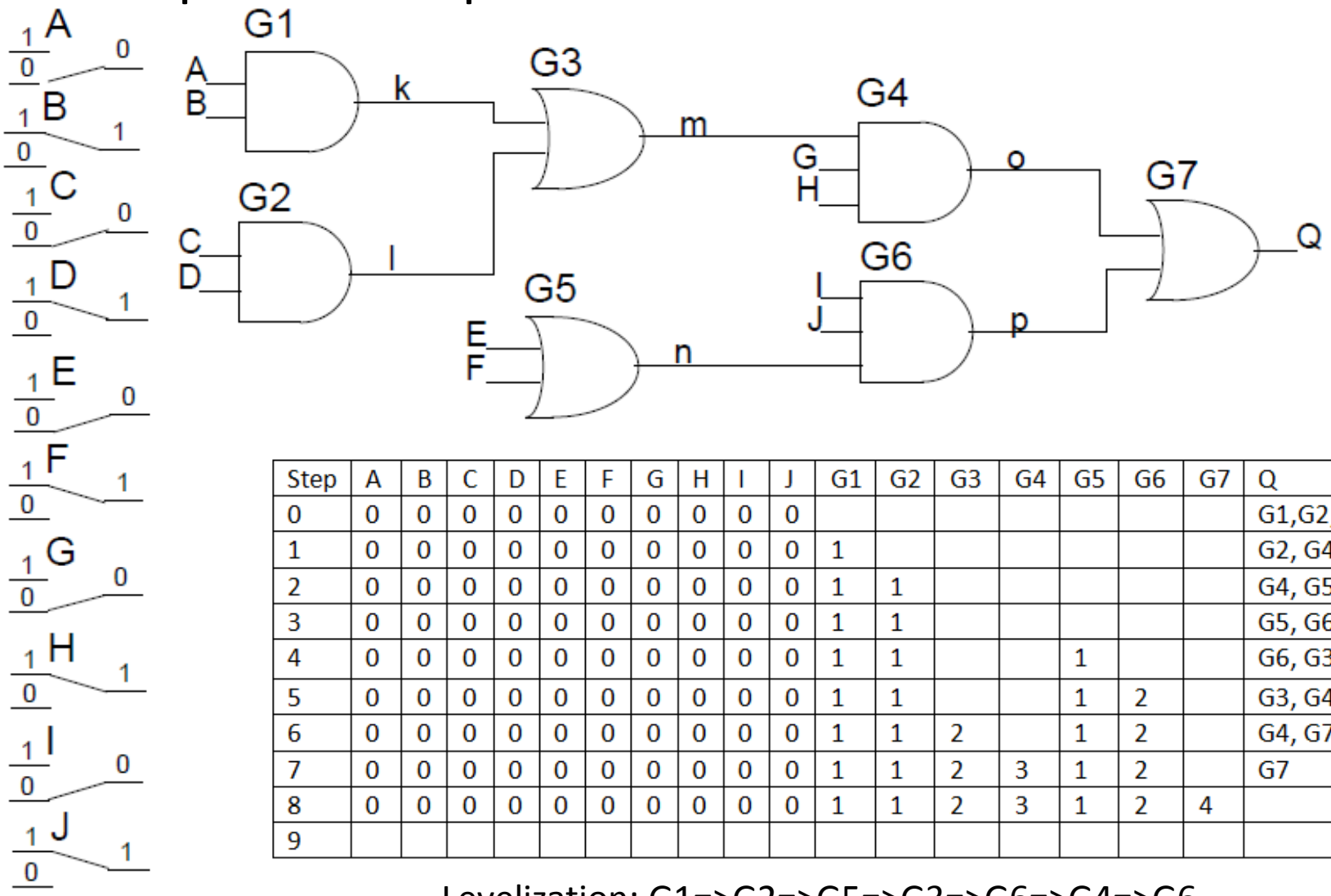
Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101




| Step | A | B | C | D | E | F | G | H | I | J | G1 | G2 | G3 | G4 | G5 | G6 | G7 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | G1,G2, G4, G5, G6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | G2, G4, G5, G6, G3 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G4, G5, G6, G3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G5, G6, G3, G4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | 1 | | | G6, G3, G4 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | 1 | 2 | | G3, G4, G7 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | | 1 | 2 | | G4, G7 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 1 | 2 | | G7 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 1 | 2 | 4 | |
| 9 | | | | | | | | | | | | | | | | | | |

# Quiz Problem 1 Practice 1

Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101

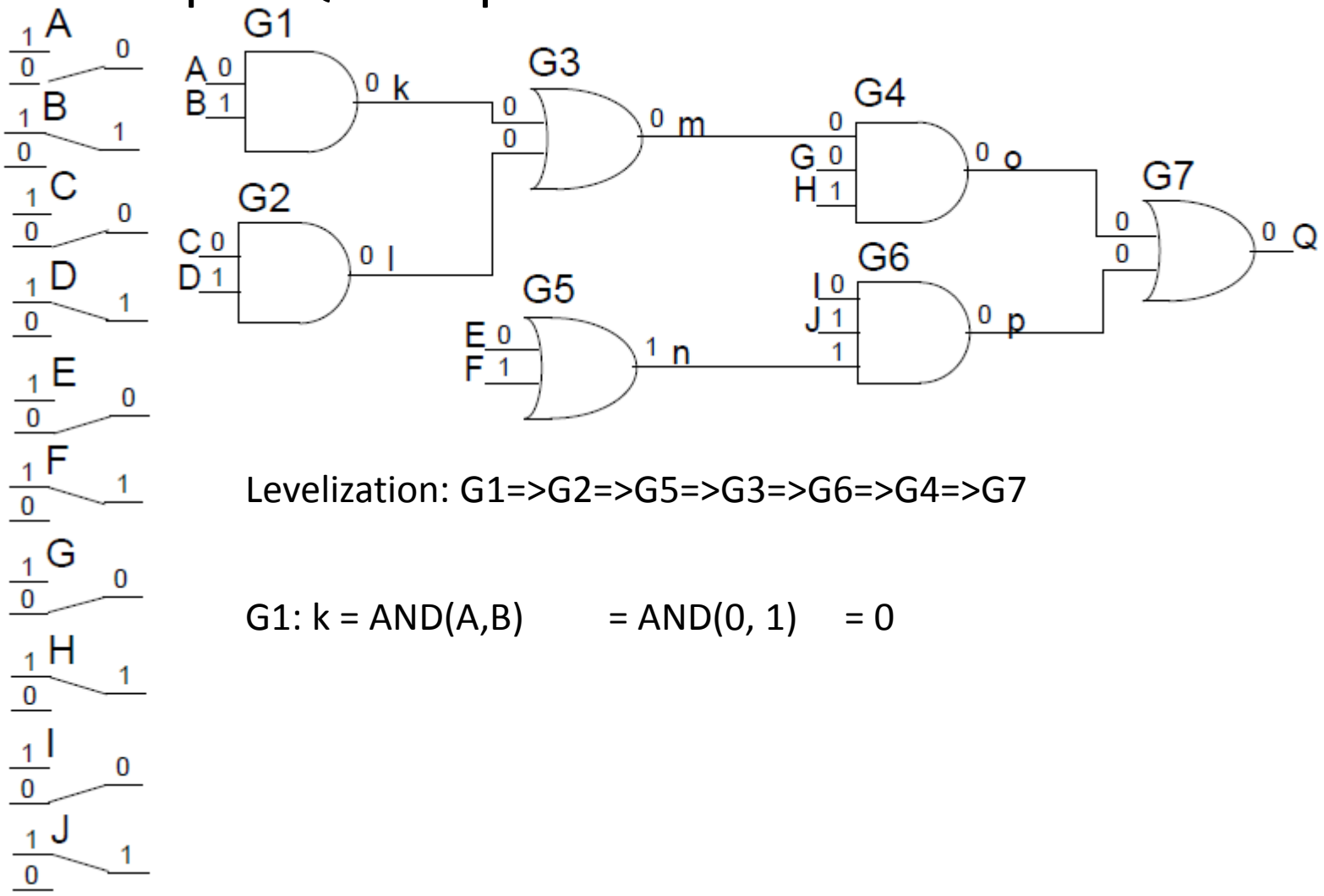



| Step | A | B | C | D | E | F | G | H | I | J | G1 | G2 | G3 | G4 | G5 | G6 | G7 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | G1,G2, G4, G5, G6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | G2, G4, G5, G6, G3 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G4, G5, G6, G3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | G5, G6, G3, G4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | 1 | | | G6, G3, G4 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | 1 | 2 | | G3, G4, G7 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | | 1 | 2 | | G4, G7 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 1 | 2 | | G7 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 1 | 2 | 4 | |
| 9 | | | | | | | | | | | | | | | | | | |

Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G6

# Quiz Problem 1 Practice 1

Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101

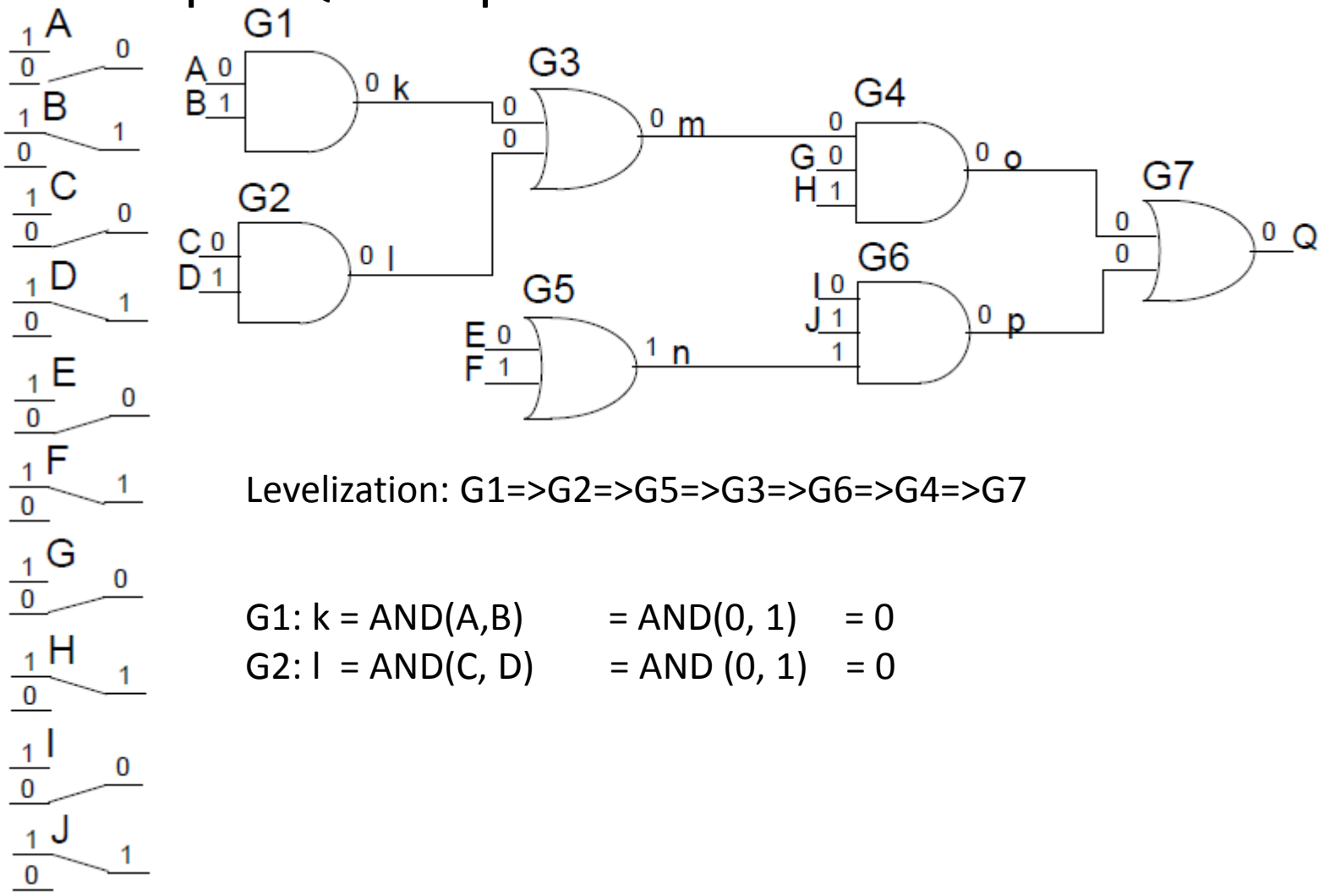



Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G7

G1: k = AND(A,B) = AND(0, 1) = 0

# Quiz Problem 1 Practice 1

Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101



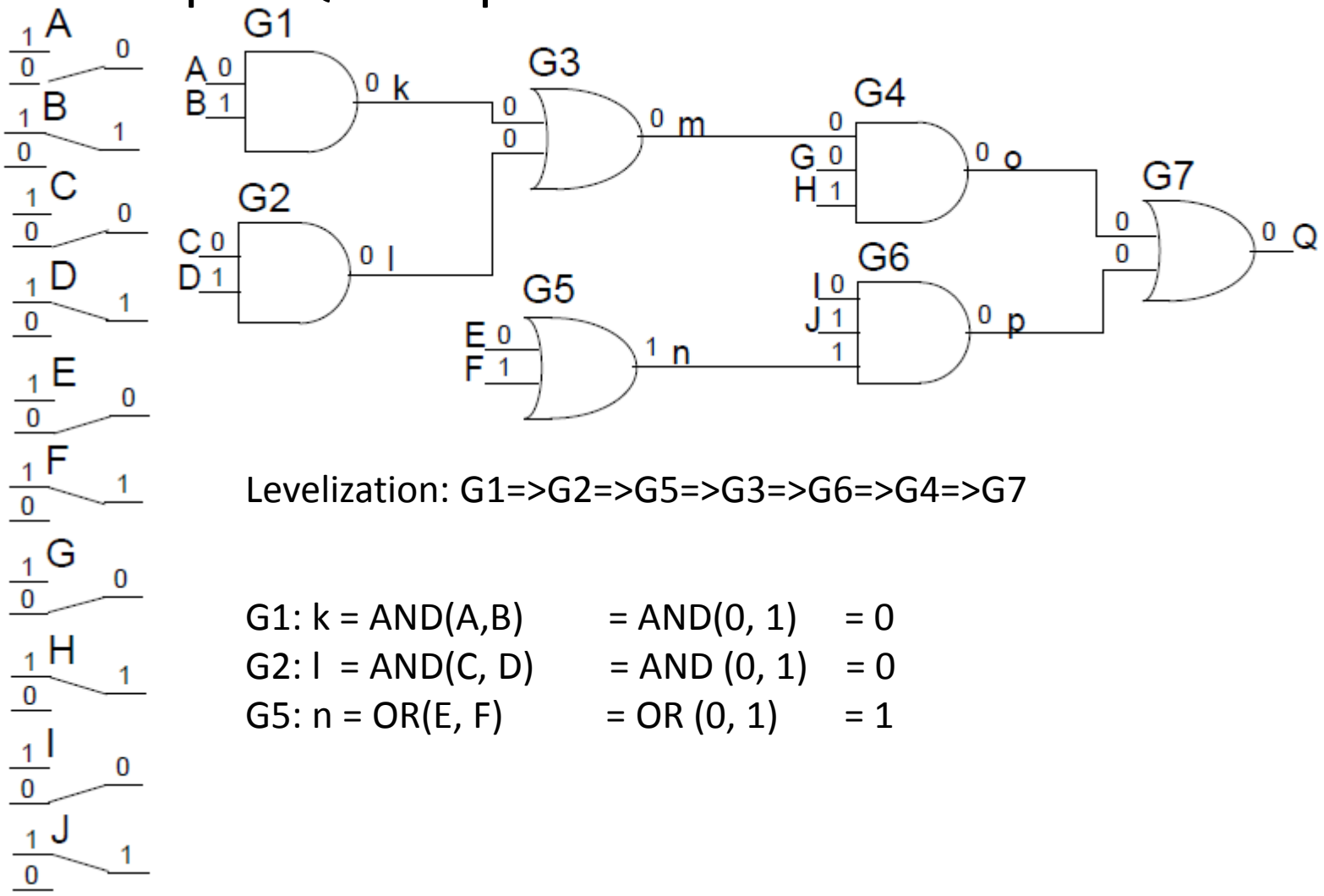

Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G7

G1: k = AND(A,B) = AND(0, 1) = 0

G2: l = AND(C, D) = AND (0, 1) = 0

# Quiz Problem 1 Practice 1

Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101

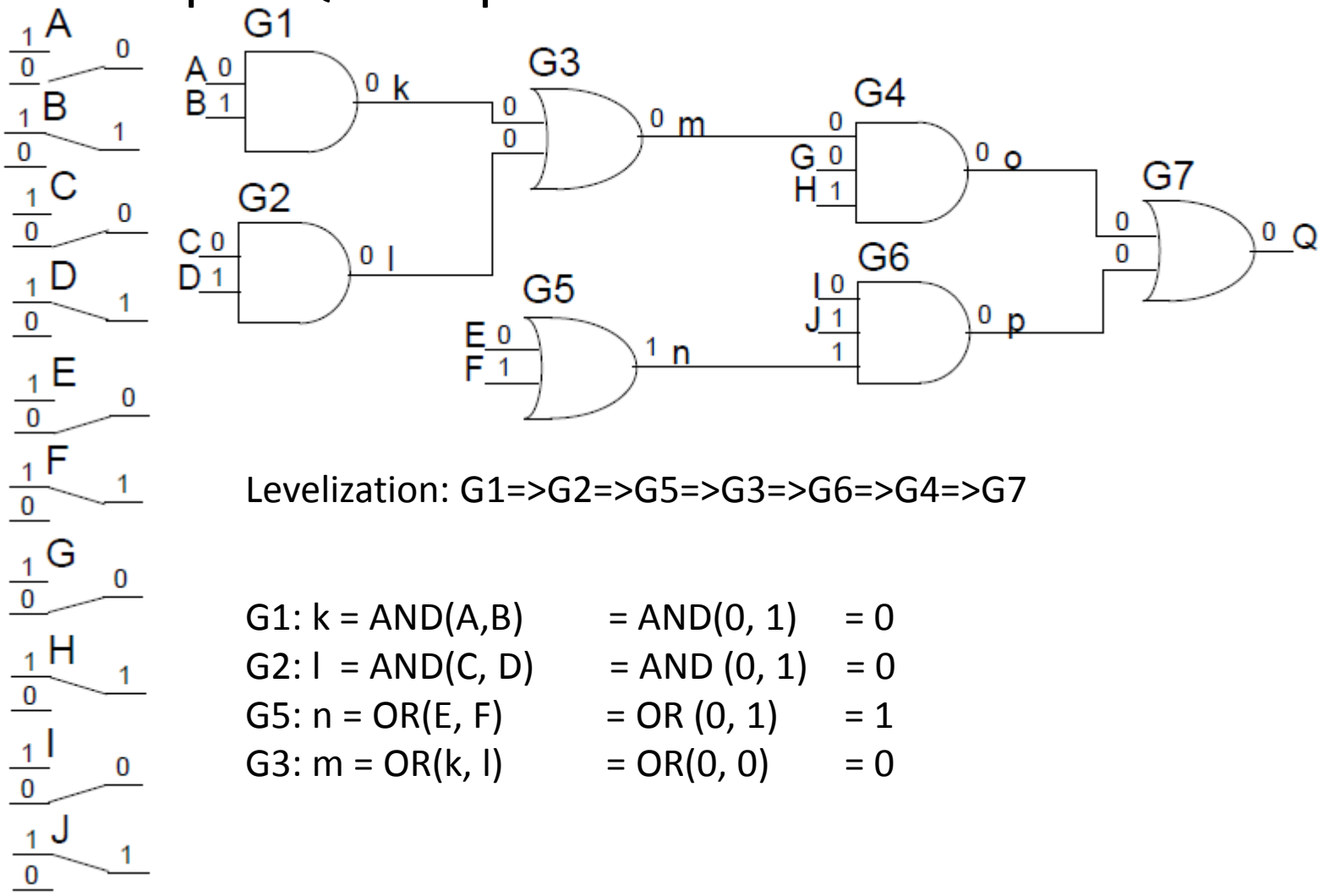



Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G7

G1: k = AND(A,B) = AND(0, 1) = 0
G2: l = AND(C, D) = AND (0, 1) = 0
G5: n = OR(E, F) = OR (0, 1) = 1

# Quiz Problem 1 Practice 1

Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101

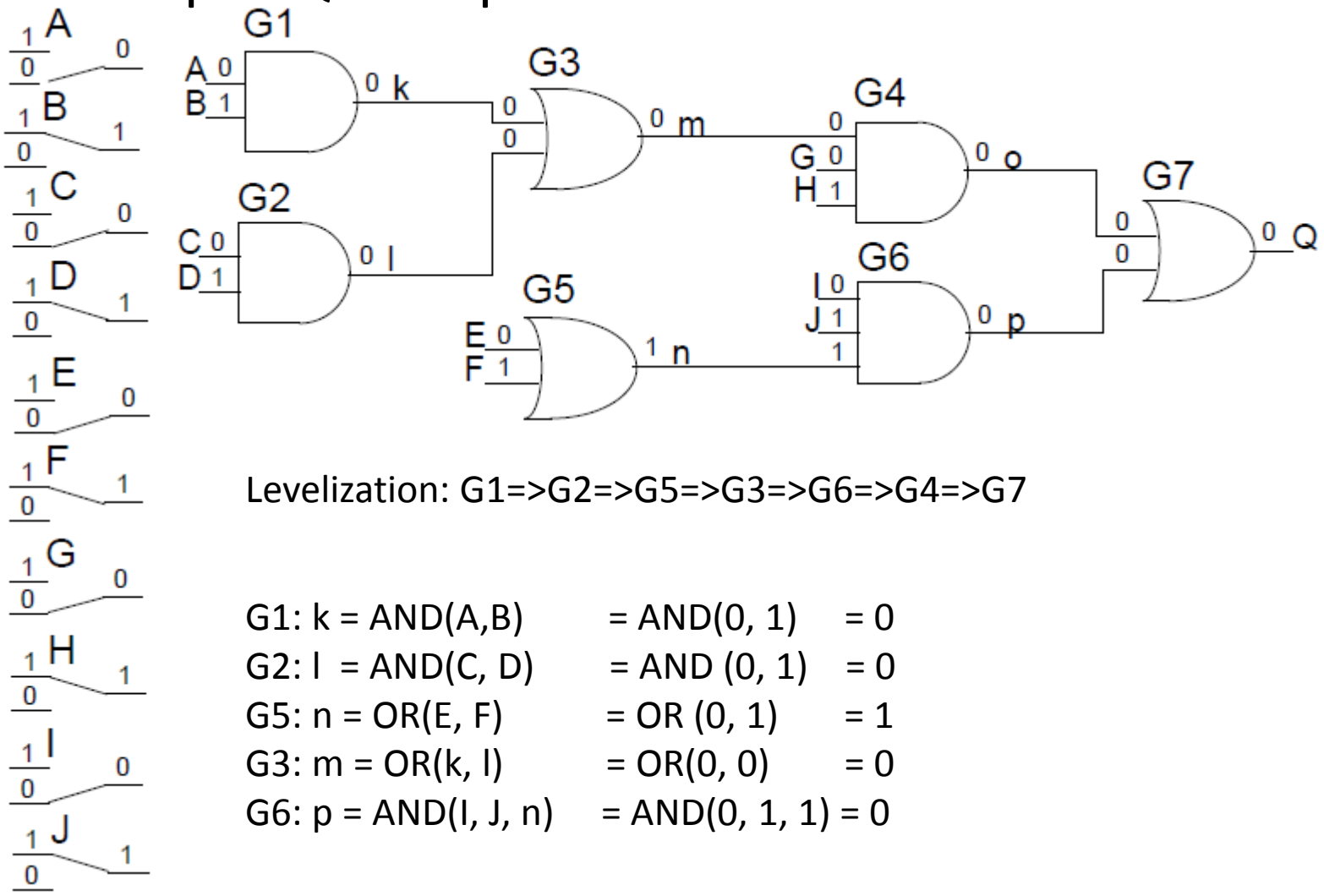



Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G7

G1: k = AND(A,B) = AND(0, 1) = 0
G2: l = AND(C, D) = AND (0, 1) = 0
G5: n = OR(E, F) = OR (0, 1) = 1
G3: m = OR(k, l) = OR(0, 0) = 0

# Quiz Problem 1 Practice 1

Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101

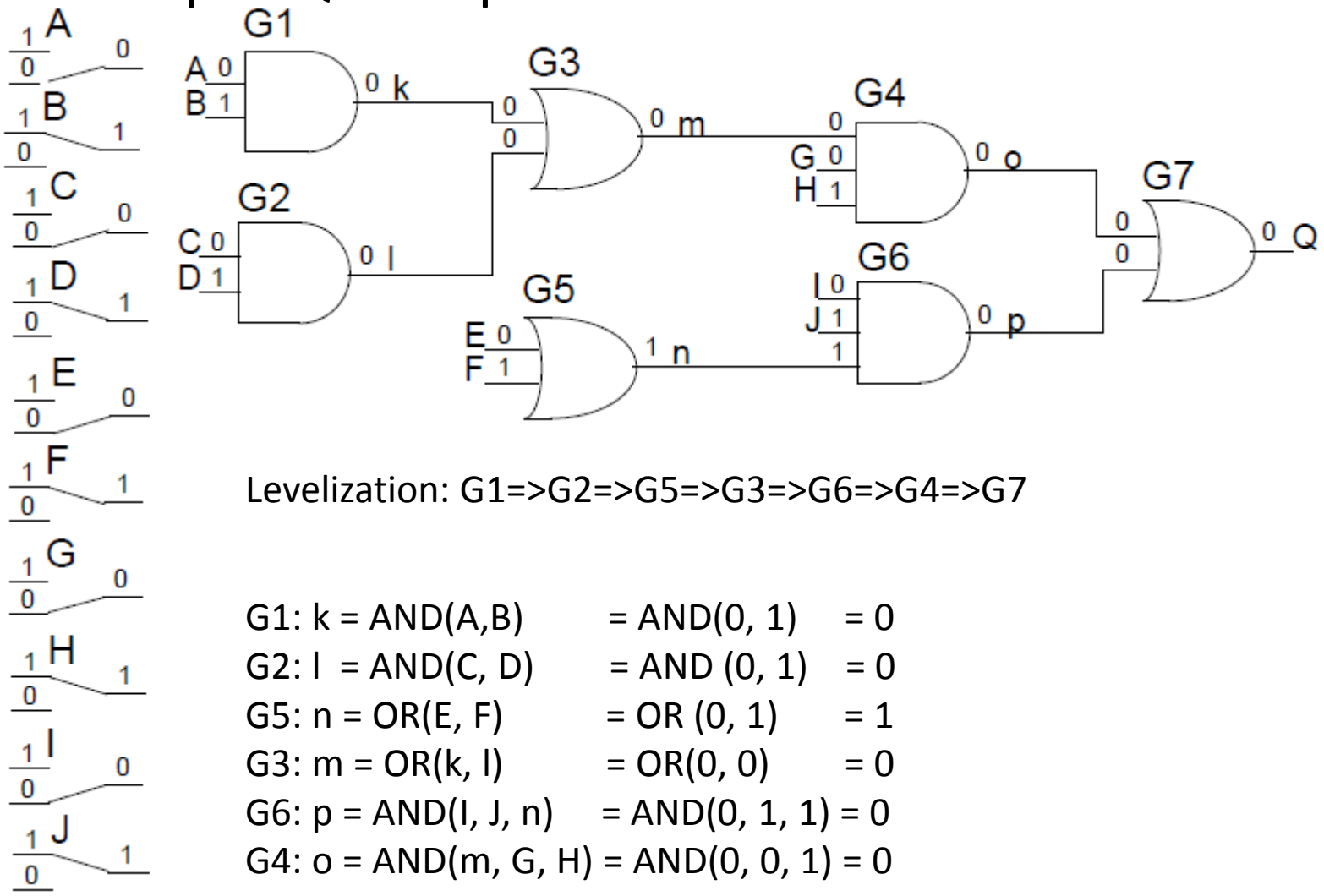



Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G7

G1: k = AND(A,B) = AND(0, 1) = 0
G2: l = AND(C, D) = AND (0, 1) = 0
G5: n = OR(E, F) = OR (0, 1) = 1
G3: m = OR(k, l) = OR(0, 0) = 0
G6: p = AND(I, J, n) = AND(0, 1, 1) = 0

# Quiz Problem 1 Practice 1

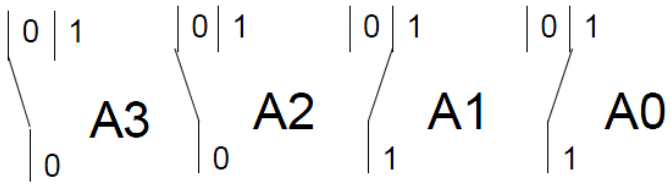
Give an gate levelization to simulate the circuit. Compute output Q for input 0101010101

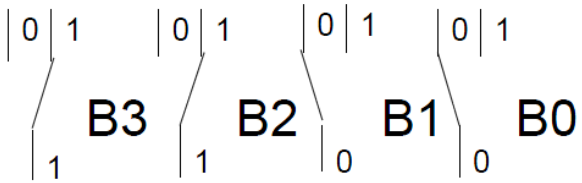


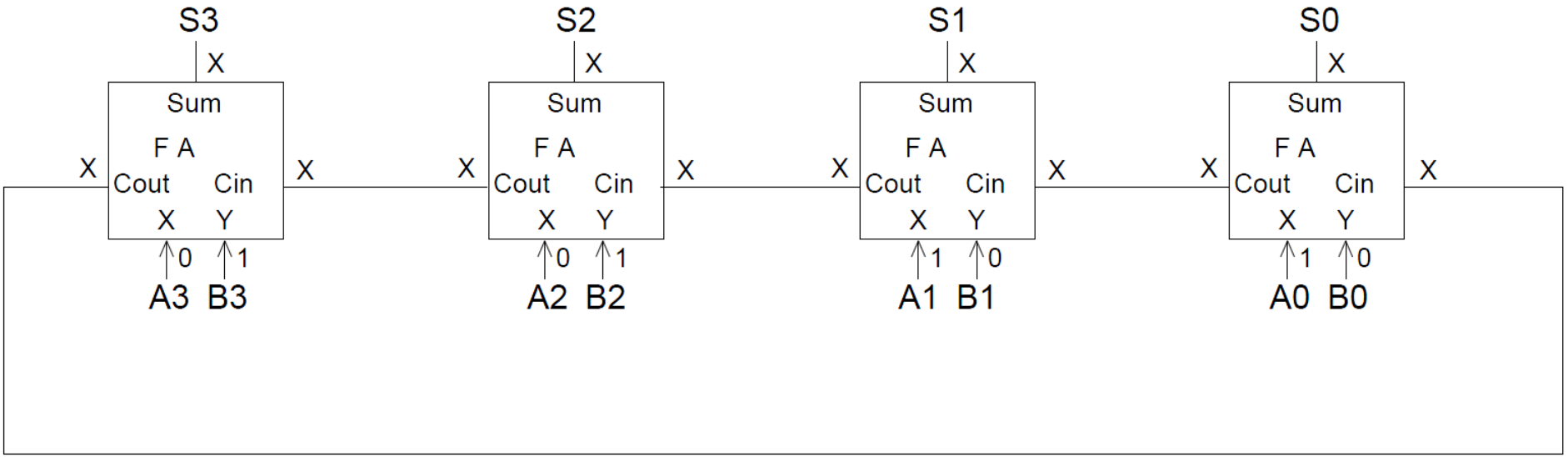

Levelization: G1=>G2=>G5=>G3=>G6=>G4=>G7

G1: k = AND(A,B) = AND(0, 1) = 0
G2: l = AND(C, D) = AND (0, 1) = 0
G5: n = OR(E, F) = OR (0, 1) = 1
G3: m = OR(k, l) = OR(0, 0) = 0
G6: p = AND(I, J, n) = AND(0, 1, 1) = 0
G4: o = AND(m, G, H) = AND(0, 0, 1) = 0

# SimUAid Simulation Revisited




SimUAid failed to simulate an actual output because without any initial values on the feedback, it cannot levelize the gate.

# Levelizing an Asynchronous Circuit

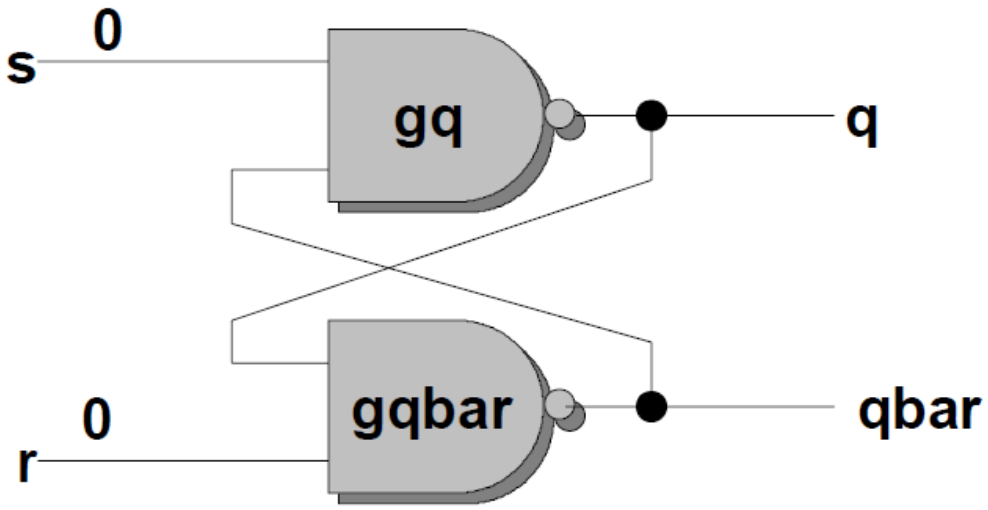
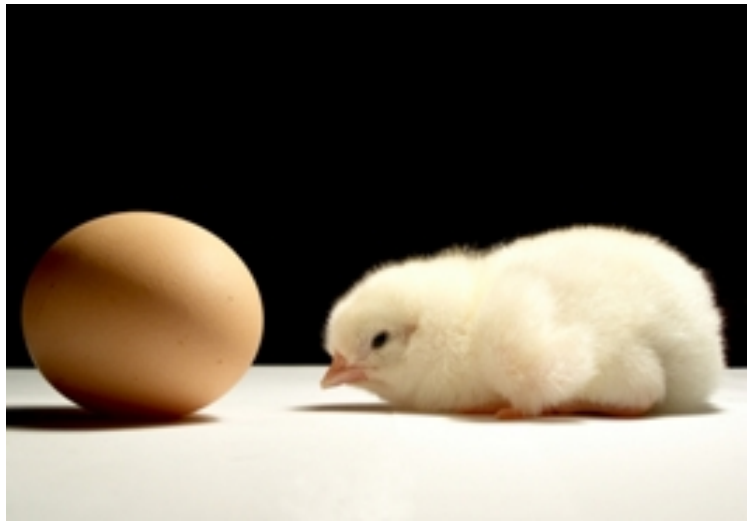



Figure 2-5. An Unlevelizable Circuit.



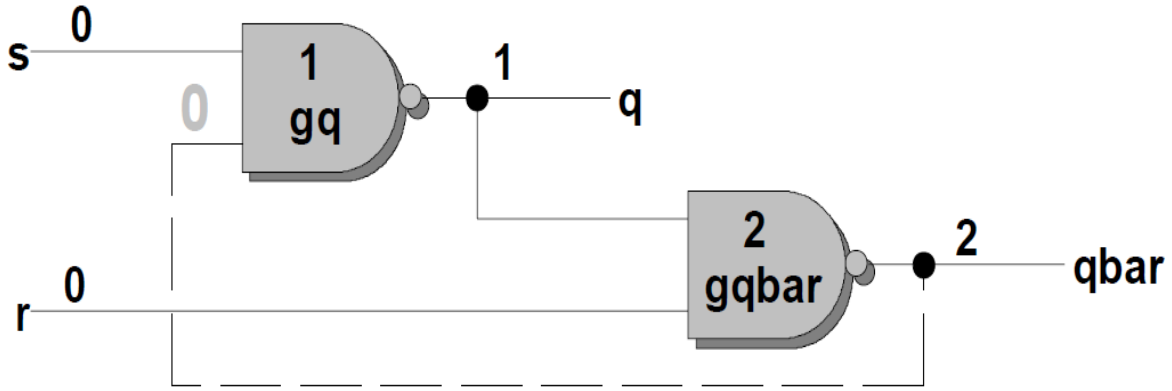
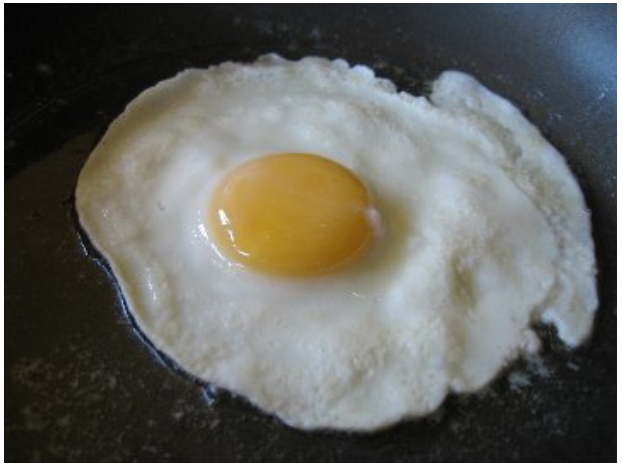
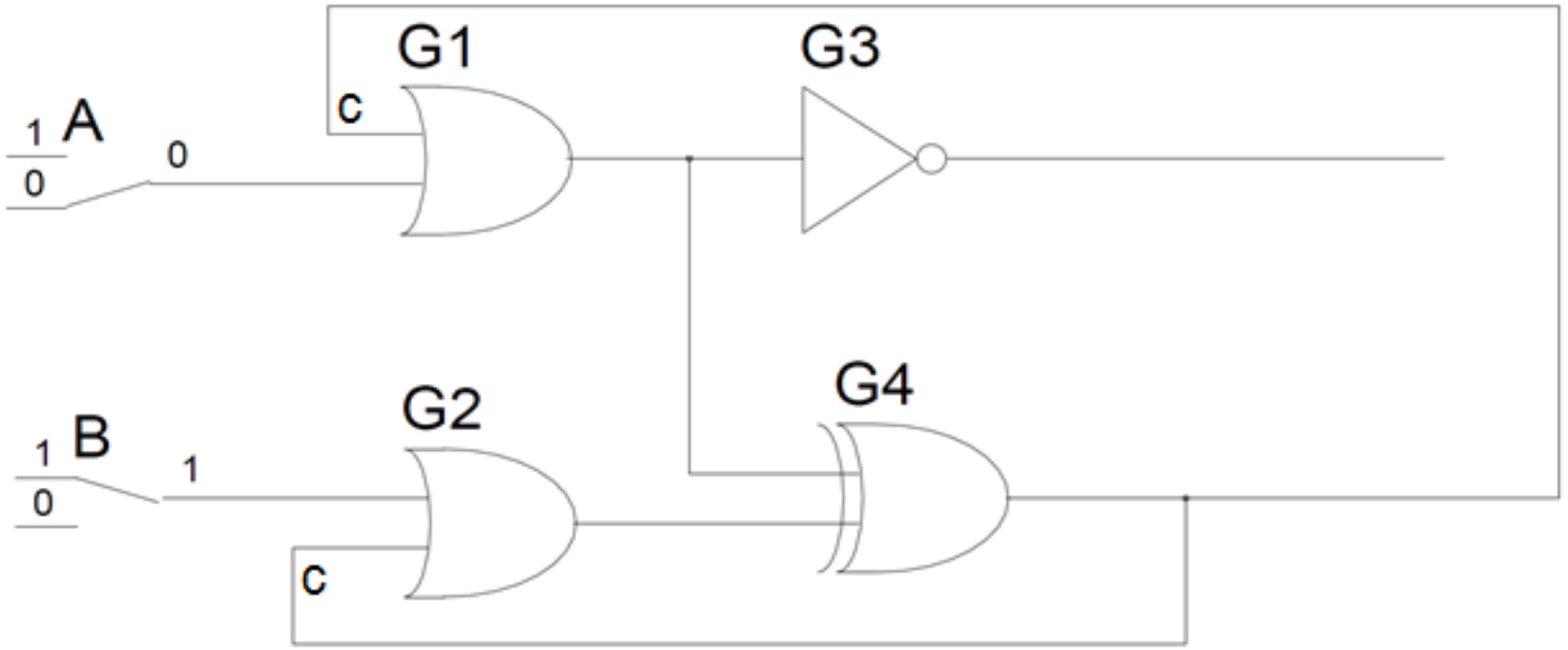



Figure 2-7. Levelizing an Asynchronous Circuit.



# Quiz Problem 1 Practice 2

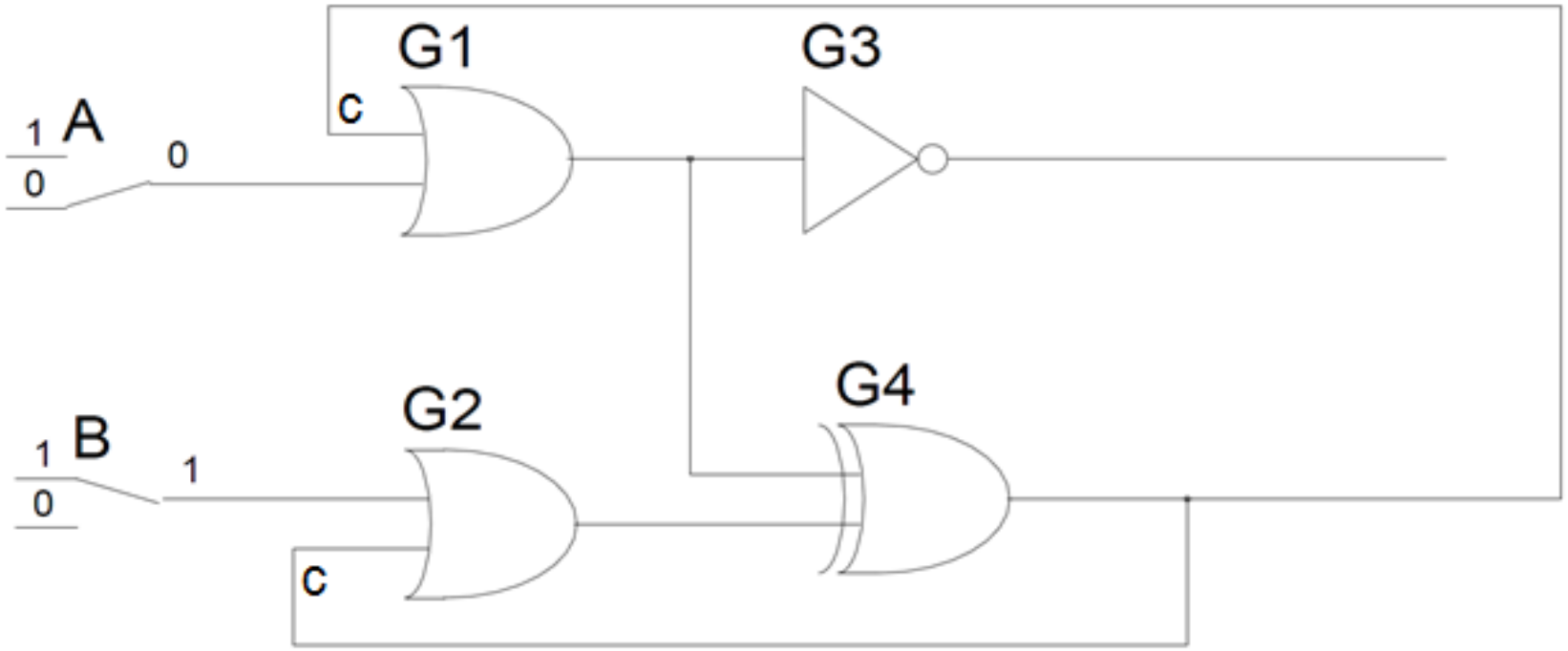
Determine the Gate Levelization and compute $G_3$ if $A = 0, B = 1, C = 0$




| step | A | B | c | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | $<G_1, G_2>$ |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |

# Quiz Problem 1 Practice 2

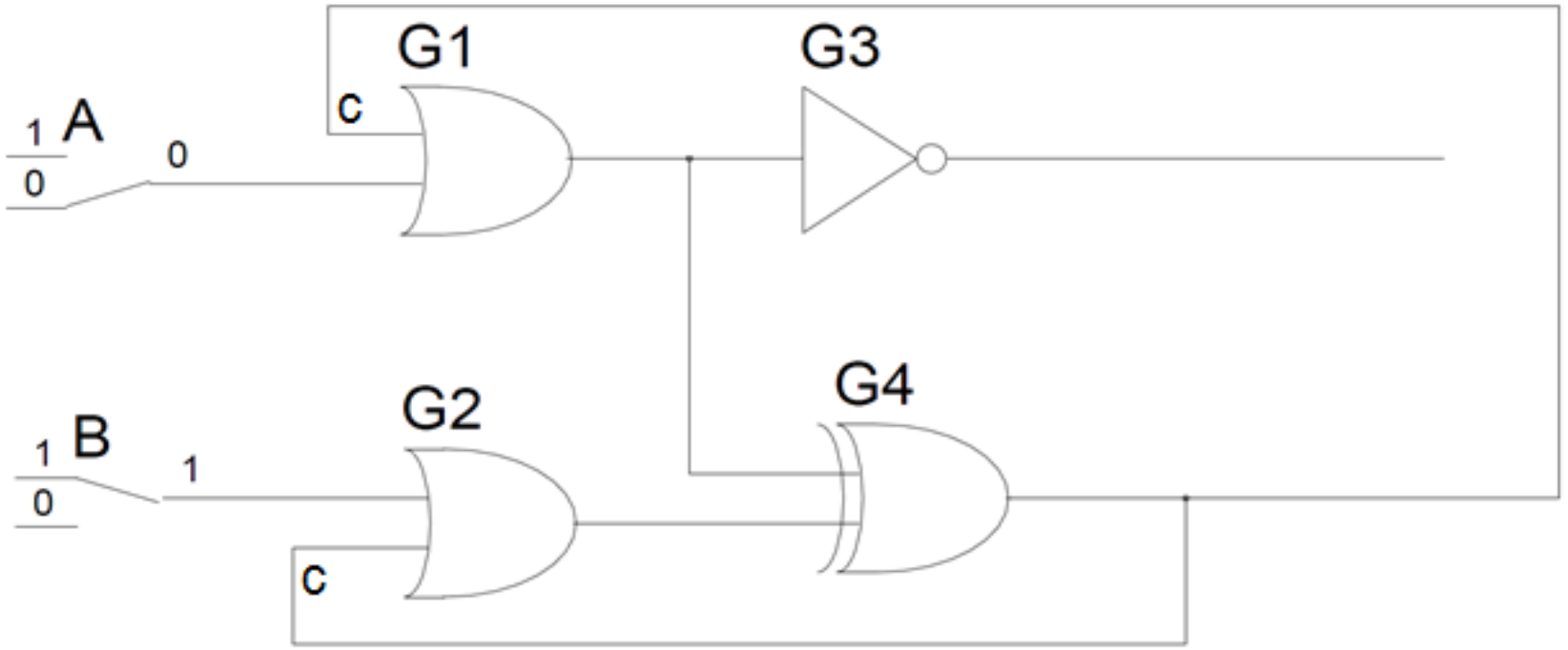
Determine the Gate Levelization and compute $G_3$ if $A = 0, B = 1, C = 0$




| step | A | B | c | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 1 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$, $G_4$> |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |

# Quiz Problem 1 Practice 2

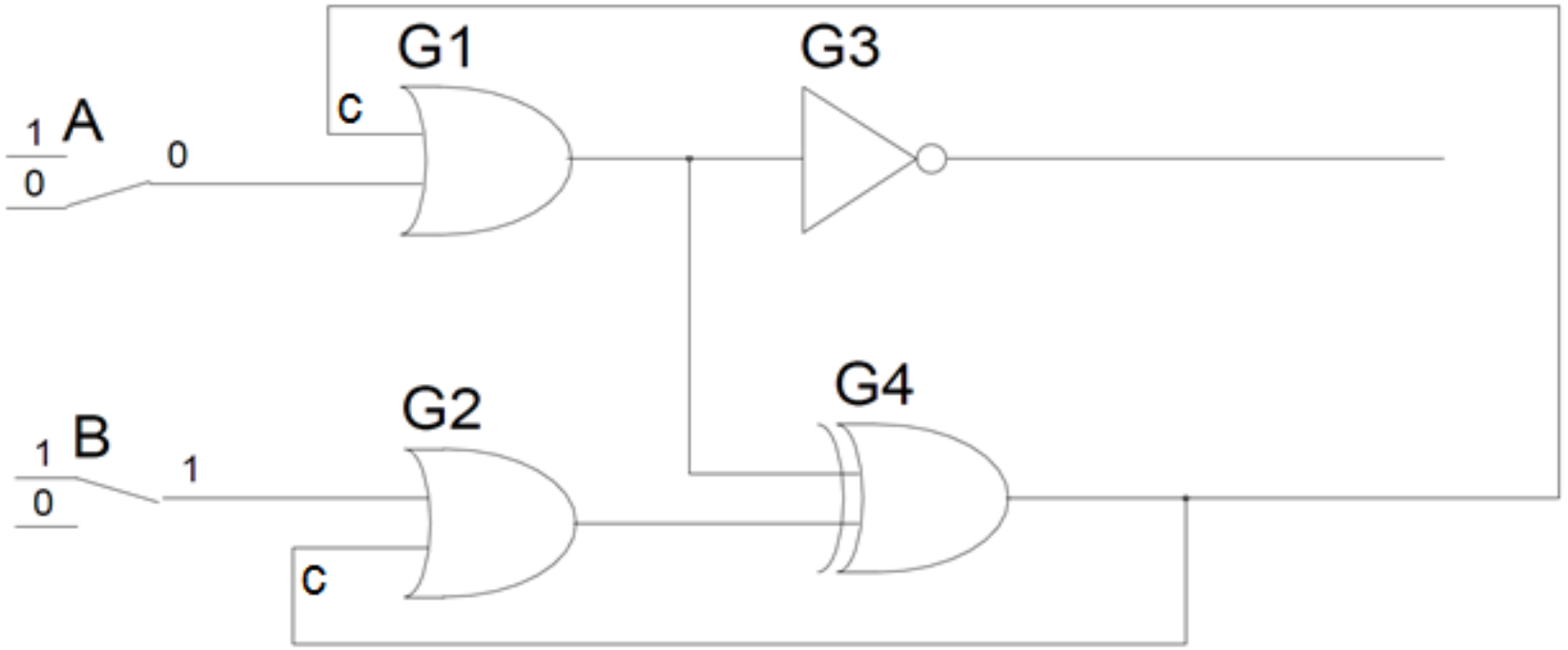
Determine the Gate Levelization and compute $G_3$ if $A = 0, B = 1, C = 0$




| step | A | B | c | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | $<G_1, G_2>$ |
| 1 | 0 | 0 | 0 | 1 | | | | $<G_2, G_3, G_4>$ |
| 2 | 0 | 0 | 0 | 1 | 1 | | | $<G_3, G_4>$ |
| 3 | | | | | | | | |
| 4 | | | | | | | | |

# Quiz Problem 1 Practice 2

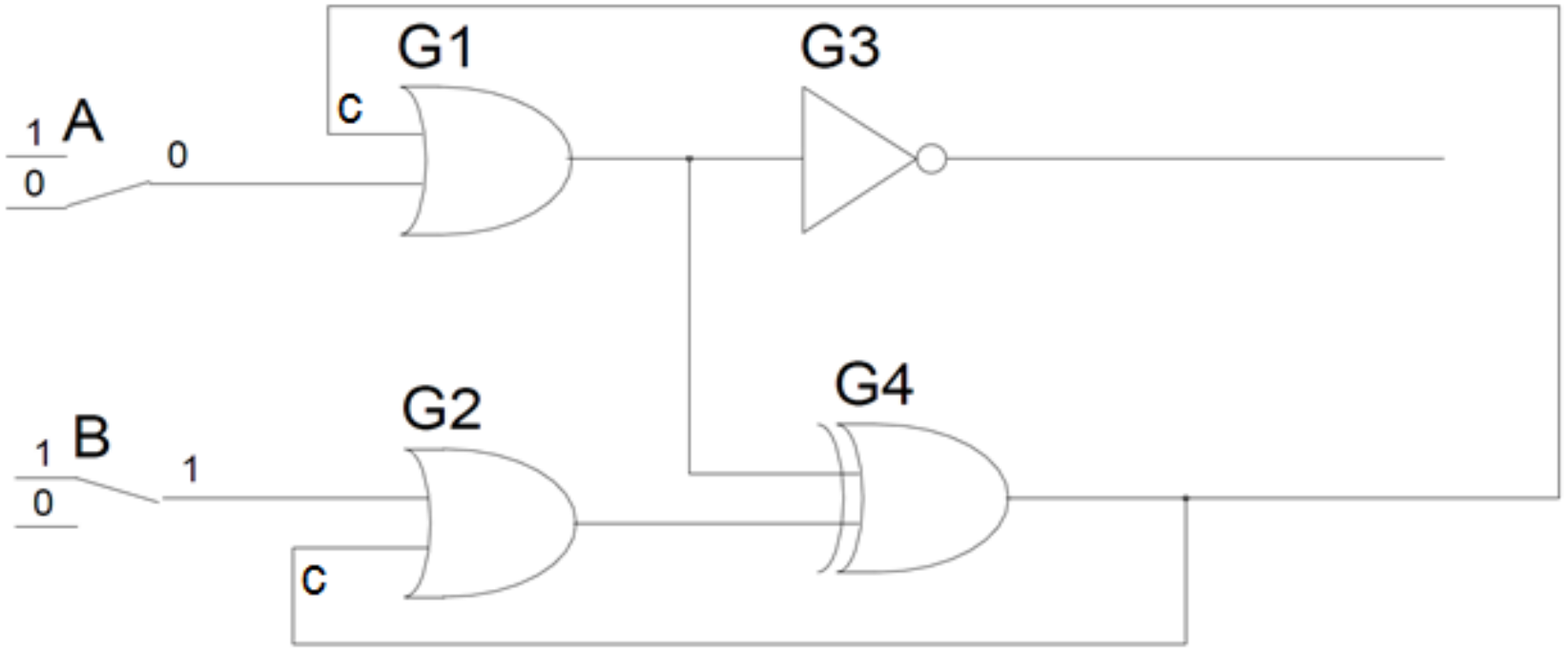
Determine the Gate Levelization and compute $G_3$ if $A = 0, B = 1, C = 0$




| step | A | B | c | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | $<G_1, G_2>$ |
| 1 | 0 | 0 | 0 | 1 | | | | $<G_2, G_3, G_4>$ |
| 2 | 0 | 0 | 0 | 1 | 1 | | | $<G_3, G_4>$ |
| 3 | 0 | 0 | 0 | 1 | 1 | 2 | | $<G_4>$ |
| 4 | | | | | | | | |

# Quiz Problem 1 Practice 2

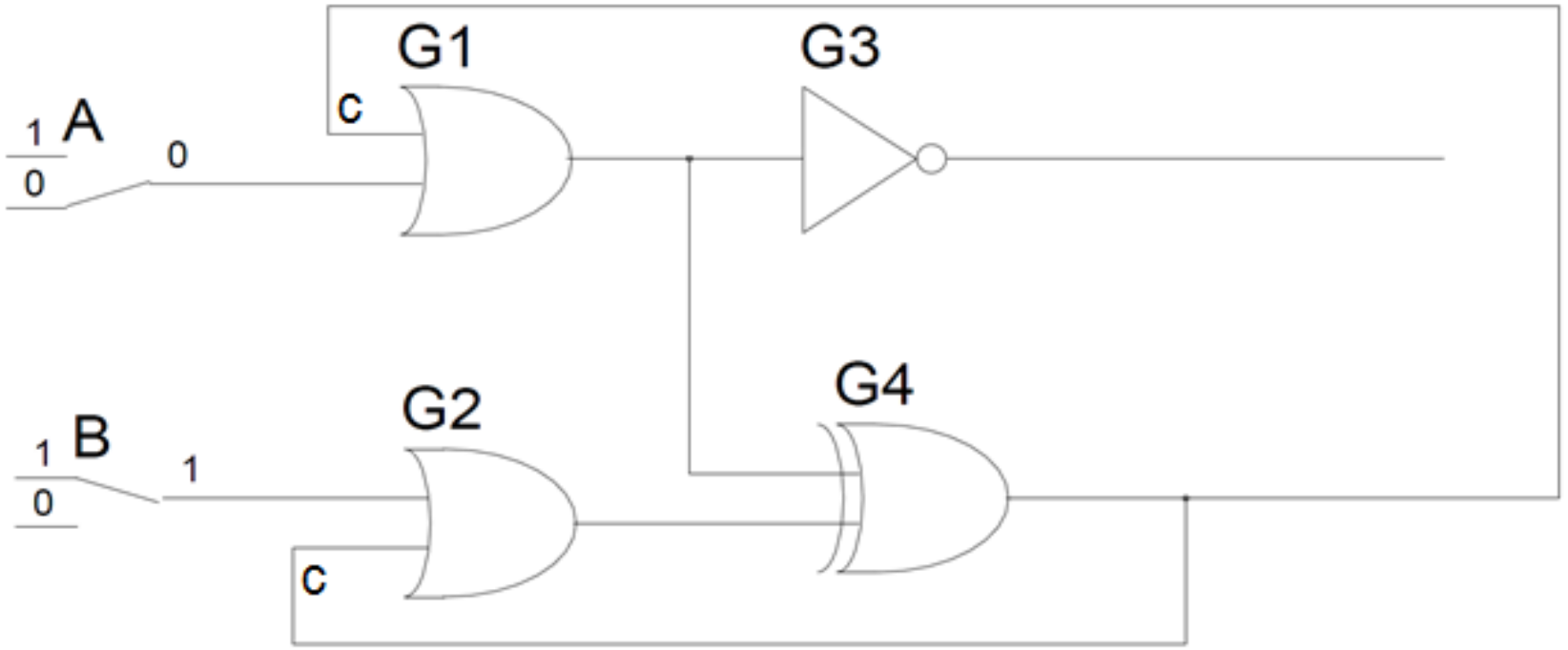
Determine the Gate Levelization and compute output of $G_3$ if $A = 0, B = 1, C = 0$




| step | A | B | c | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 1 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$, $G_4$> |
| 2 | 0 | 0 | 0 | 1 | 1 | | | <$G_3$, $G_4$> |
| 3 | 0 | 0 | 0 | 1 | 1 | 2 | | <$G_4$> |
| 4 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | <> |

# Quiz Problem 1 Practice 2

Determine the Gate Levelization and compute output of $G_3$ if $A = 0, B = 1, C = 0$




| step | A | B | c | $G_1$ | $G_2$ | $G_3$ | $G_4$ | Q |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | <$G_1$, $G_2$> |
| 1 | 0 | 0 | 0 | 1 | | | | <$G_2$, $G_3$, $G_4$> |
| 2 | 0 | 0 | 0 | 1 | 1 | | | <$G_3$, $G_4$> |
| 3 | 0 | 0 | 0 | 1 | 1 | 2 | | <$G_4$> |
| 4 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | <> |

Levelization: $G_1$=>$G_2$=>$G_3$=>$G_4$

$G_1$ = OR(A, c) = OR(0,0) = 0

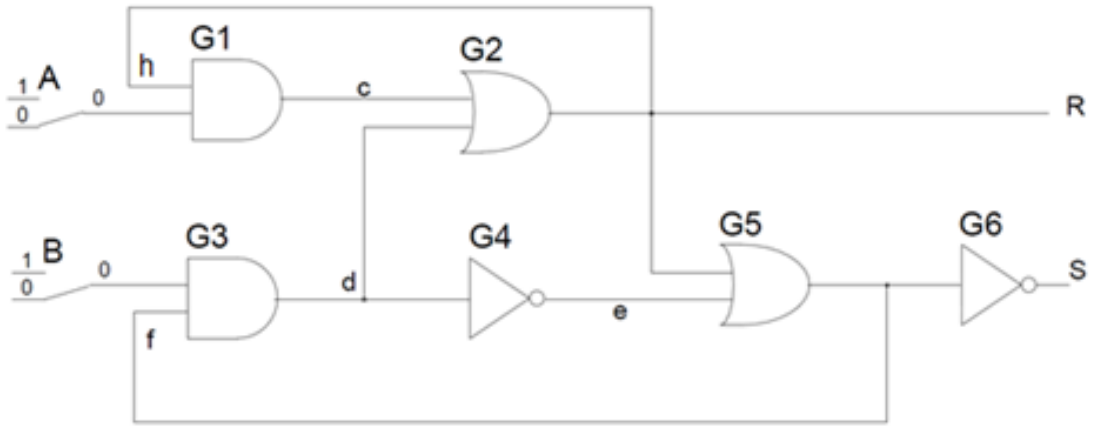
$G_2$ = OR(B, c) = OR(1,0) = 1

$G_3$ = NOT($G_1$) = NOT(0) = 1 = $G_3$

$G_4$ = XOR($G_1$, $G_2$) = XOR(0, 1) = 1

# Quiz Question 1

**Q1.**




**a)** Complete the gate levelization table and determine gate levelization

| step | A | B | h | f | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | <$G_3$, $G_1$> |
| 1 | | | | | | | | | | | |
| 2 | | | | | | | | | | | |
| 3 | | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | | | | | | | | | | |
| 6 | | | | | | | | | | | |
| 7 | | | | | | | | | | | |
| 8 | | | | | | | | | | | |
| 9 | | | | | | | | | | | |

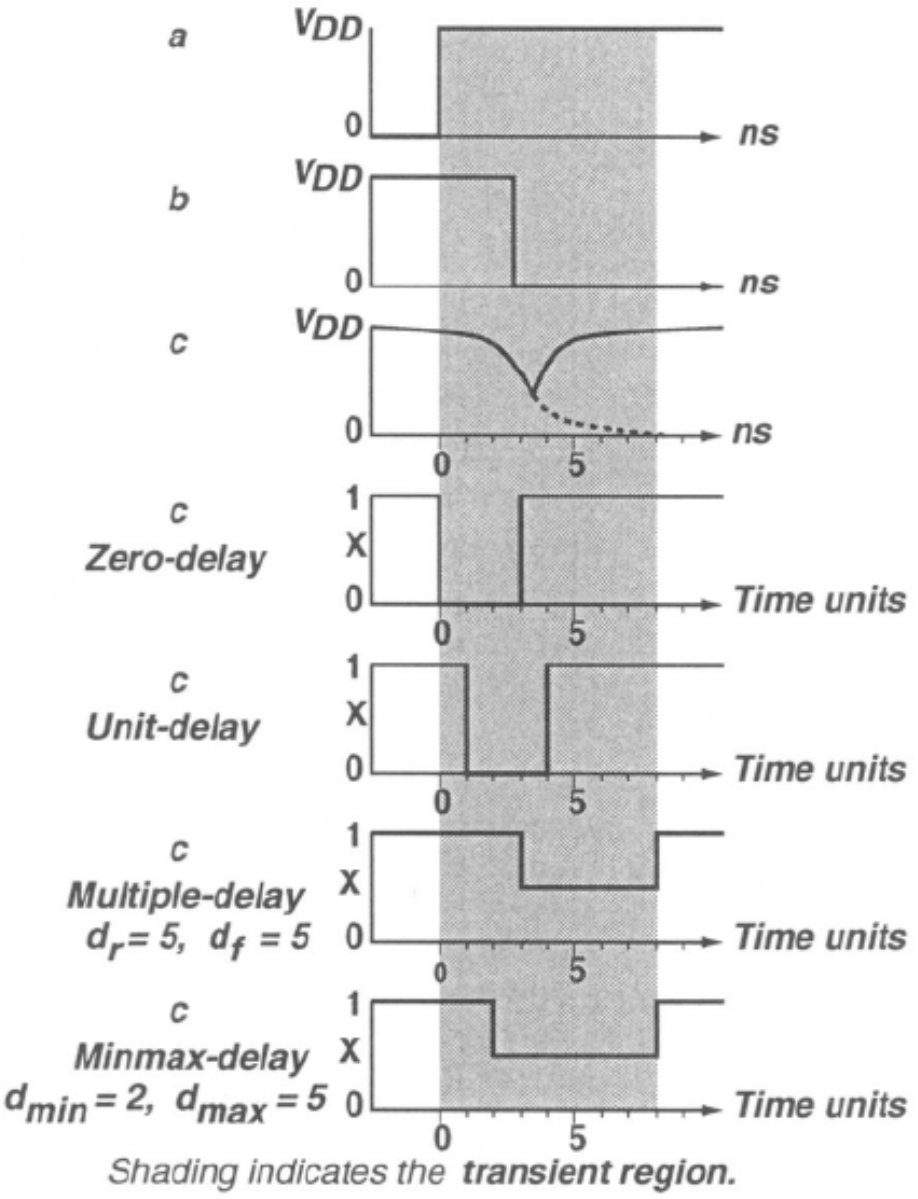
**b)** Compute output R and S using the levelization when $A = 0, B = 1, f = 1, h = 0$

**c)** Compute output R and S using the levelization when $A = 1, B = 0, f = 0, h = 1$

Show all the steps clearly.

# Timing

- Timing problems with code compiled – glitches, race conditions, etc., are not modeled
- There are two types of delay: inertial delay (input change to output change) and propagation delay (gate output to input via fanout)
- Simulators produces responses upon how delays are modeled.
  - Zero-delay
  - Unit-delay
  - Multiple-delay
  - Minmax-delay

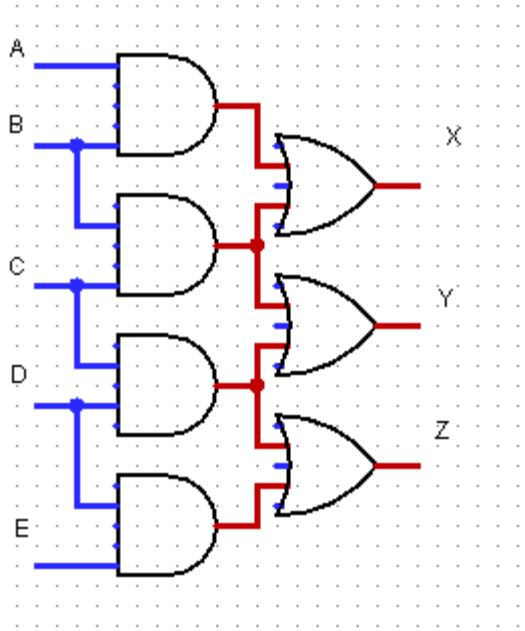



# EVENT DRIVEN SIMULATION (PART 2)

# Event Driven Simulation:

## Example Circuit:

A=B=C=D=0
E=1



# Event Driven Simulation:

[Not event-driven]

```
If(primary input is changed) then {
        for (all gates)
        {
                compute outputs of all gates from scratch
        }
}
```

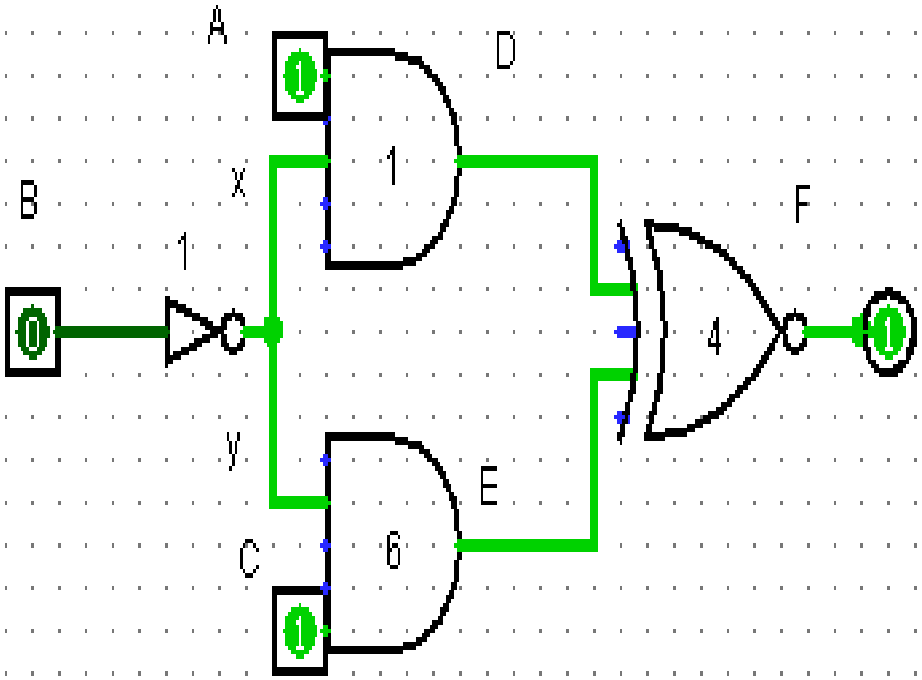
[event-driven]

```
If(primary input is changed) then {
        for (only affected gates)
        {
           compute new outputs of only affected gates while using old
           outputs of unaffected gates before input was changed.
        }
}
```

# Event Driven Simulation:

- “Event Driven” for circuit simulation refers to digital logic computing algorithms that take into account which logic gates affect the primary outputs of the circuit.
- Why use it? -> Eliminate un-needed/redundant gate level computation without adding too much extra computation.

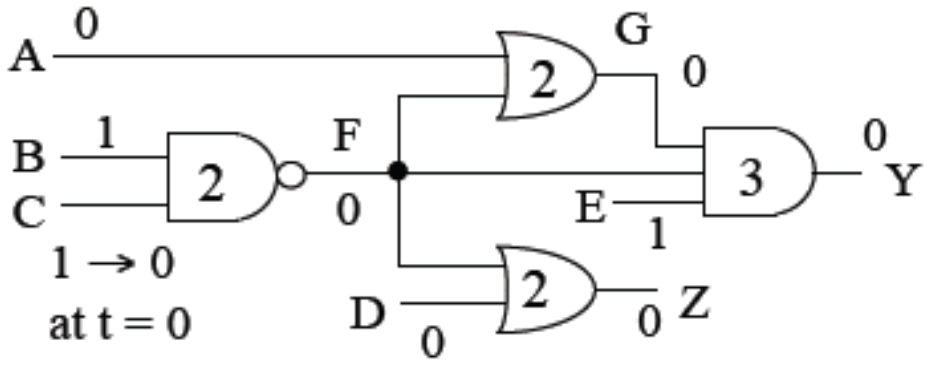
- Consider the following circuit:




Given initial Values
A=1 B=0 C=1
What would the event Queue look like for each "event" (step representing each level of circuit)
What would be the "flickering" of values for event driven at the primary output F before It settles down to a single value when the event queue is empty?

# Event-Driven Simulation (Example)

Event: the switching of signal's value




t = 0

| Event | Time |
|---|---|
| F = 1 | t = 2 |

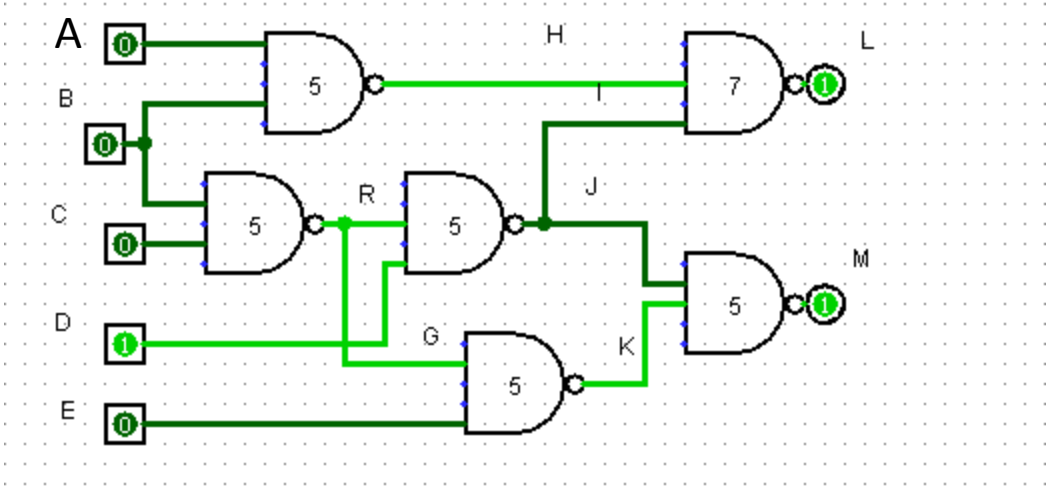
t = 2

| Event | Time |
|---|---|
| G = 1 | t = 4 |
| Z = 1 | t = 4 |

t = 4

| Event | Time |
|---|---|
| Y = 1 | t = 7 |

# Event-Driven Simulation (Example)




T = 0: D = 1->0
T = 1 …….

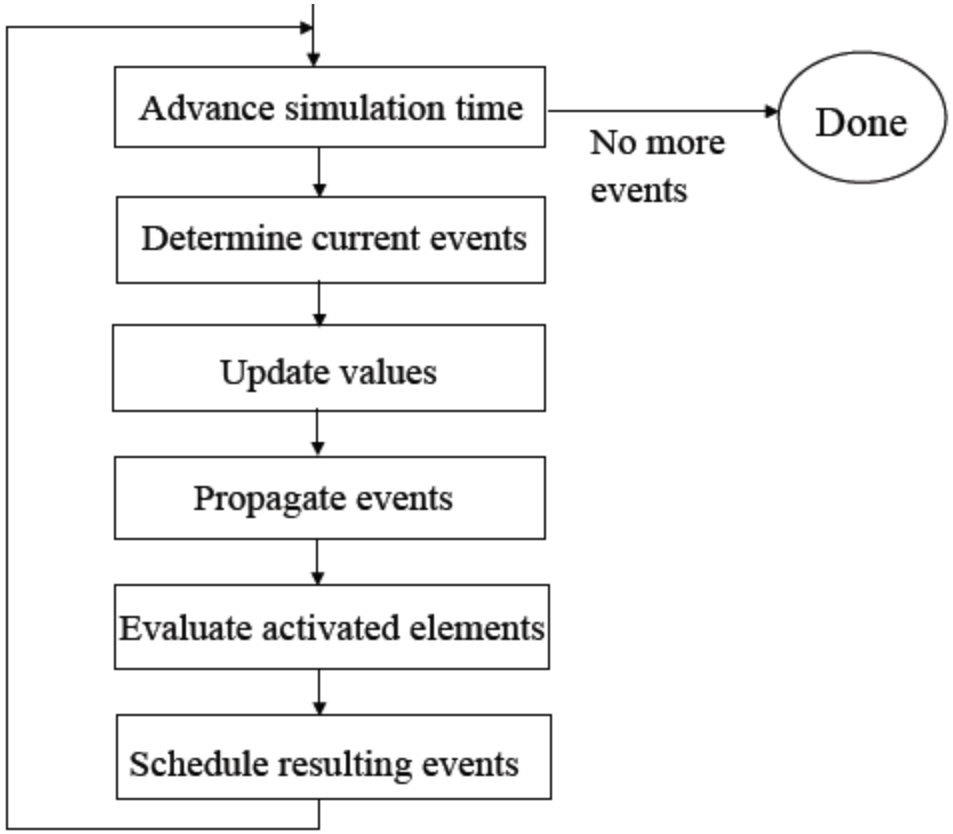
Or, try T = 0: D= 1->0 and B= 0->1 and A = 0 ->1

# Solution to Example

| time | event | activity |
|---|---|---|
| T = 0 | D = 0 | I,J |
| T = 5 | I = J = 1 | L,M |
| T = 10 | M = 0 | |
| T = 12 | L = 0 | |

| time | event | activity |
|---|---|---|
| T = 0 | D = 0, A = 1, B = 1 | I,J,H |
| T = 5 | I = J = 1, H = 0 | L,M |
| T = 10 | M = 0 | |

# Event-Driven Simulation



# Event-Driven Simulation (What we know so far)

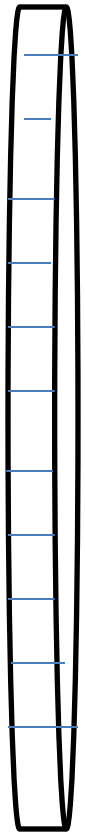
- Evaluate gate when inputs change
  - use logic model to compute new output value
  - use timing model to compute when output will change
- Schedule an output change *event*
  - store the event on a time-sorted *event queue*
- Process events from the queue
  - output change evaluated at scheduled time
  - causes new events to be scheduled
- We define an “event” to be a change in value in a network of gates (nodes).
- Event driven simulators detect “events” and store them into a data structure (usually some type of queue) for parsing.

# Zero Delay Event Driven Simulation

```
Input vector[0] = initial condition
For (all input vectors from 1 to n)
{
        Push onto event queue input_vector[i] primary input fanout gates
        While (Event queue is not empty)
        {
                Evaluate front queue logic gate "g"
                If (not output change)
                {
                        Push g's fanout gates onto queue
                }
        }
}
```

# Time Wheel (Advanced algorithms

MAX = Length of execution of events (based all gate delays put together)



T = 0 [ EVENT LIST for this time stamp]

T = 1 [ EVENT LIST for this time stamp]

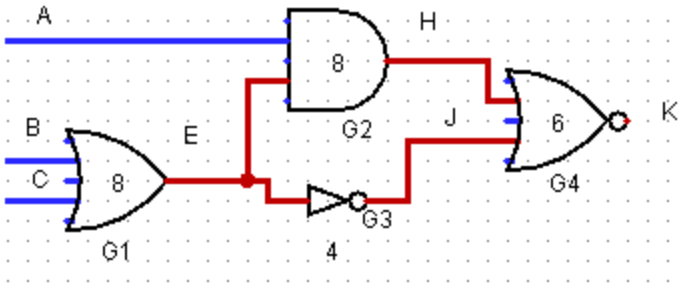
T = 2 [ EVENT LIST for this time stamp]

.....

T = max

# Advanced Algorithm Example

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| | | | |

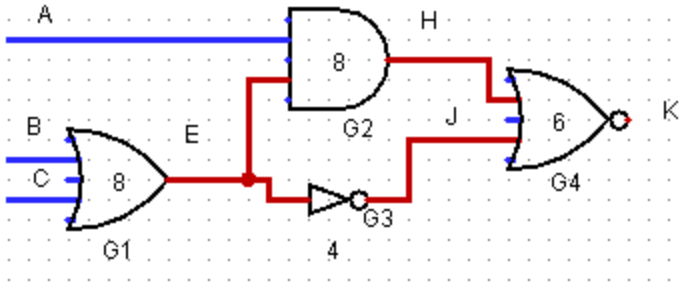



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | $\{(\mathbf{A},\mathbf{1})\}$ | $\{G_2\}$ | $\{(H,1,8)\}$ |

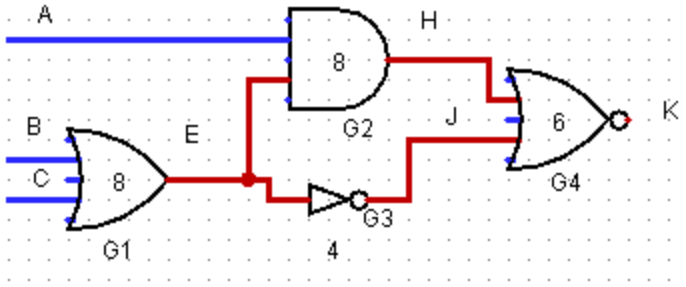



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | $\{(\mathbf{A},\mathbf{1})\}$ | $\{G_2\}$ | $\{(H,1,8)\}$ |
| 2 | $\{(\mathbf{C},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,1,10)\}$ |

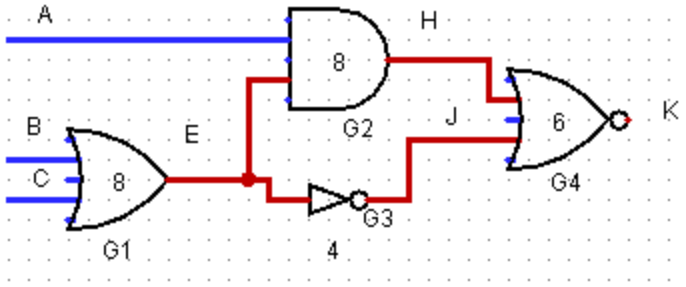



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | $\{(\mathbf{A},\mathbf{1})\}$ | $\{G_2\}$ | $\{(H,1,8)\}$ |
| 2 | $\{(\mathbf{C},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,1,10)\}$ |
| 4 | $\{(\mathbf{B},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,0,12)\}$ |

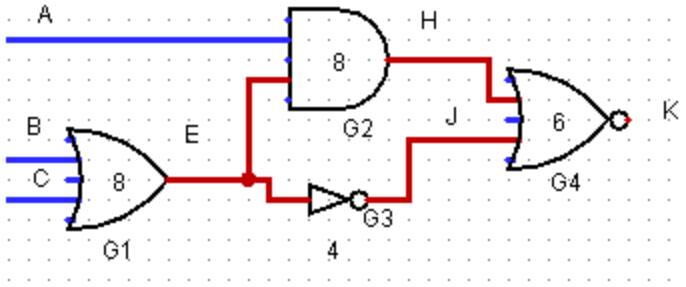



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | $\{(\mathbf{A},\mathbf{1})\}$ | $\{G_2\}$ | $\{(H,1,8)\}$ |
| 2 | $\{(\mathbf{C},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,1,10)\}$ |
| 4 | $\{(\mathbf{B},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,0,12)\}$ |
| 8 | $\{(\mathbf{A},\mathbf{0}),(H,1)\}$ | $\{G_2,G_4\}$ | $\{(H,0,16),(K,0,14)\}$ |

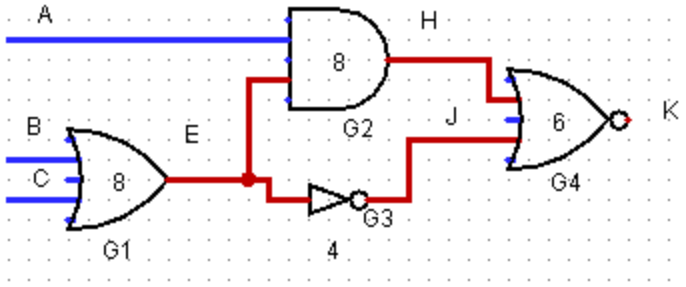



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | $\{(\mathbf{A},\mathbf{1})\}$ | $\{G_2\}$ | $\{(H,1,8)\}$ |
| 2 | $\{(\mathbf{C},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,1,10)\}$ |
| 4 | $\{(\mathbf{B},\mathbf{0})\}$ | $\{G_1\}$ | $\{(E,0,12)\}$ |
| 8 | $\{(\mathbf{A},\mathbf{0}),(H,1)\}$ | $\{G_2,G_4\}$ | $\{(H,0,16),(K,0,14)\}$ |
| 10 | $\{(E,1)\}$ | | |

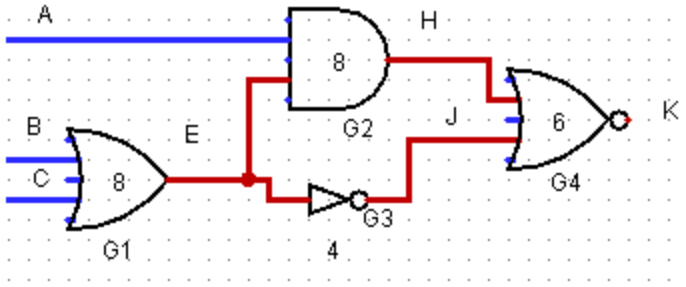



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | {(**A**,**1**)} | {$G_2$} | {($H$,1,8)} |
| 2 | {(**C**,**0**)} | {$G_1$} | {($E$,1,10)} |
| 4 | {(**B**,**0**)} | {$G_1$} | {($E$,0,12)} |
| 8 | {(**A**,**0**),($H$,1)} | {$G_2$,$G_4$} | {($H$,0,16),($K$,0,14)} |
| 10 | {($E$,1)} | | |
| 12 | {($E$,0)} | { $G_2$,$G_3$} | {($H$,0,20),($J$,1,16)} |

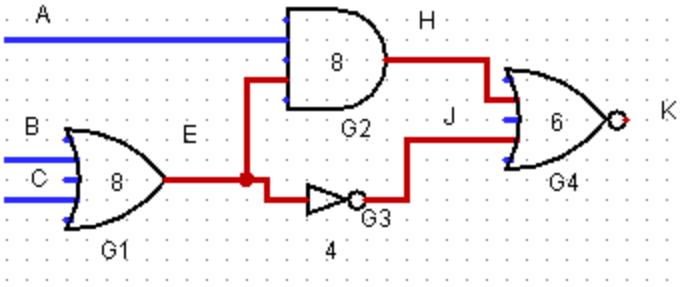



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | {(**A**,**1**)} | {$G_2$} | {($H$,1,8)} |
| 2 | {(**C**,**0**)} | {$G_1$} | {($E$,1,10)} |
| 4 | {(**B**,**0**)} | {$G_1$} | {($E$,0,12)} |
| 8 | {(**A**,**0**),($H$,1)} | {$G_2$,$G_4$} | {($H$,0,16),($K$,0,14)} |
| 10 | {($E$,1)} | | |
| 12 | {($E$,0)} | { $G_2$,$G_3$} | {($H$,0,20),($J$,1,16)} |
| 14 | {($K$,0)} | | |

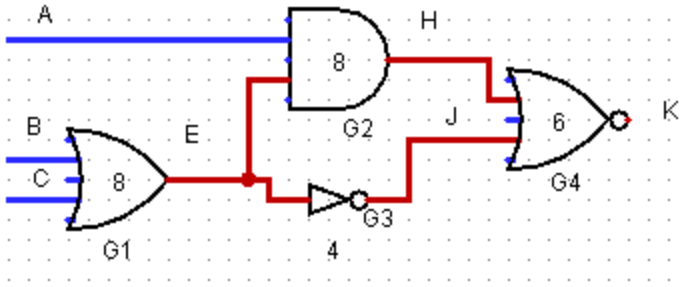



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | {(**A**,**1**)} | {$G_2$} | {($H$,1,8)} |
| 2 | {(**C**,**0**)} | {$G_1$} | {($E$,1,10)} |
| 4 | {(**B**,**0**)} | {$G_1$} | {($E$,0,12)} |
| 8 | {(**A**,**0**),($H$,1)} | {$G_2$,$G_4$} | {($H$,0,16),($K$,0,14)} |
| 10 | {($E$,1)} | | |
| 12 | {($E$,0)} | { $G_2$,$G_3$} | {($H$,0,20),($J$,1,16)} |
| 14 | {($K$,0)} | | |
| 16 | {($H$,0),($J$,1)} | {$G_4$} | {($K$,0,22)} |

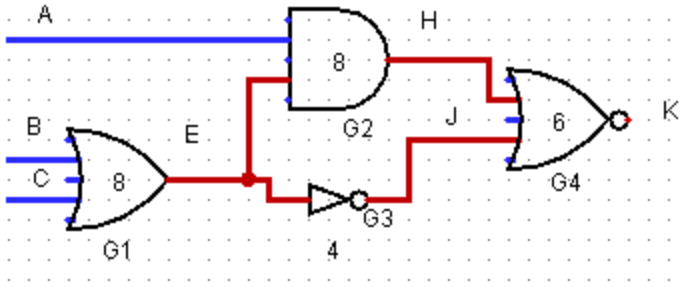



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | {(**A**,**1**)} | {$G_2$} | {($H$,1,8)} |
| 2 | {(**C**,**0**)} | {$G_1$} | {($E$,1,10)} |
| 4 | {(**B**,**0**)} | {$G_1$} | {($E$,0,12)} |
| 8 | {(**A**,**0**),($H$,1)} | {$G_2$,$G_4$} | {($H$,0,16),($K$,0,14)} |
| 10 | {($E$,1)} | | |
| 12 | {($E$,0)} | { $G_2$,$G_3$} | {($H$,0,20),($J$,1,16)} |
| 14 | {($K$,0)} | | |
| 16 | {($H$,0),($J$,1)} | {$G_4$} | {($K$,0,22)} |

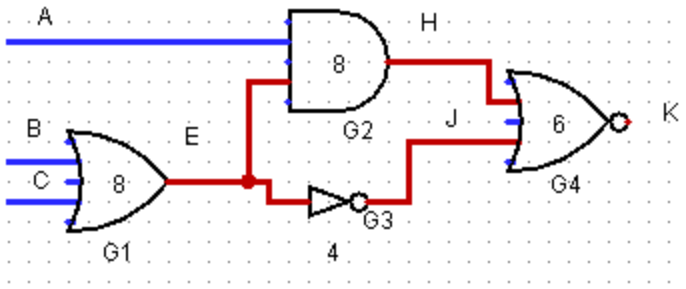



four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

8,8,4 and 6 are
The gate delays

# Example continued…

| Time | $L_E$ | $L_A$ | Scheduled events |
|---|---|---|---|
| 0 | {(**A**,**1**)} | {$G_2$} | {($H$,1,8)} |
| 2 | {(**C**,**0**)} | {$G_1$} | {($E$,1,10)} |
| 4 | {(**B**,**0**)} | {$G_1$} | {($E$,0,12)} |
| 8 | {(**A**,**0**),($H$,1)} | {$G_2$,$G_4$} | {($H$,0,16),($K$,0,14)} |
| 10 | {($E$,1)} | | |
| 12 | {($E$,0)} | { $G_2$,$G_3$} | {($H$,0,20),($J$,1,16)} |
| 14 | {($K$,0)} | | |
| 16 | {($H$,0),($J$,1)} | {$G_4$} | {($K$,0,22)} |
| 20 | {($H$,0)} | | |
| 22 | {($K$,0)} | | |



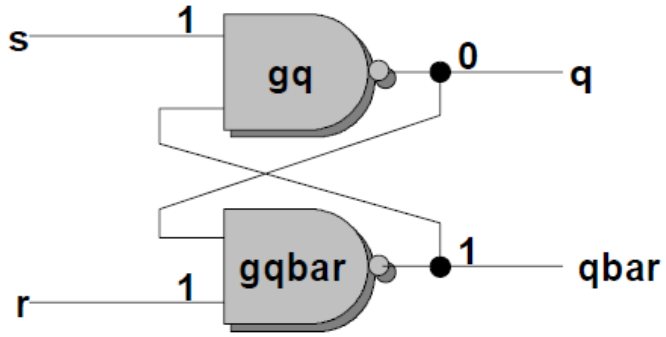

four input events:
(A, 1, 0),
(C, 0, 2), (B, 0, 4),
and (A, 0, 8),

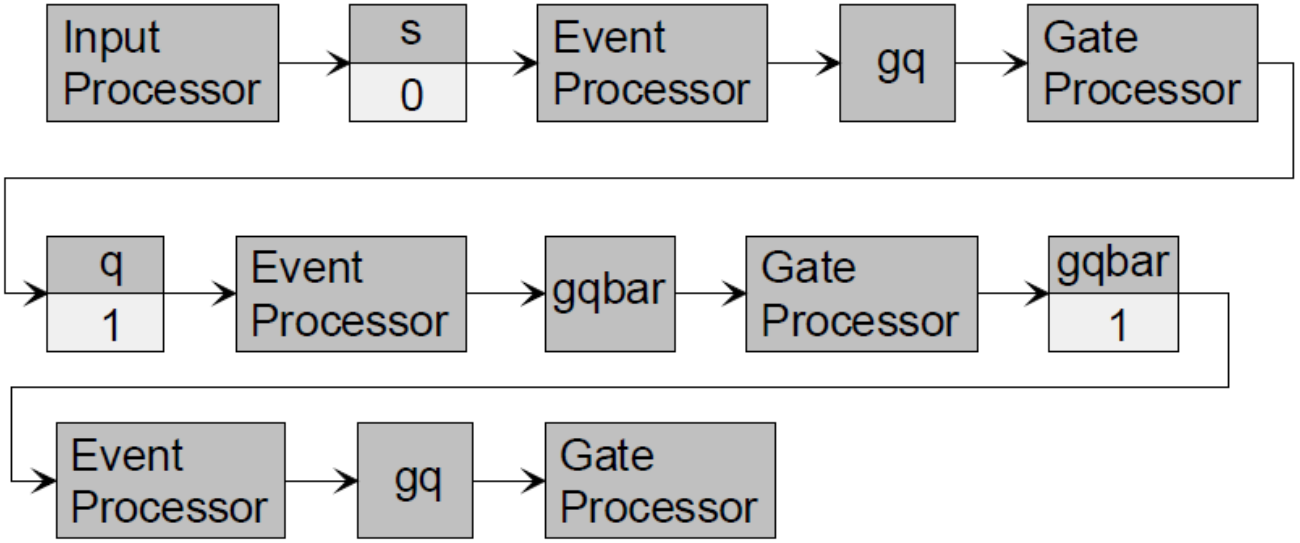
8,8,4 and 6 are
The gate delays

# Advanced Algorithm: Two-Pass EDS

```
Initialize with L_e (event list) and activity list L_A
Initialize with delay array d(# gates)
For (all time stamps t[0...n])
{
        Get event list corresponding to t
        while (event_list L_e is NOT empty)
        {
                Get event (g, v_g^+) from event list
                If (v_g^+ == v_g)
                {
                        v_g = v_g^+
                        Append g's fanout gates to activity list L_A
                }
        }
        While (activity list L_A is NOT empty)
        {
                Get next gate g
                Evaluate gate and schedule at t + d(g)
        }

}
```

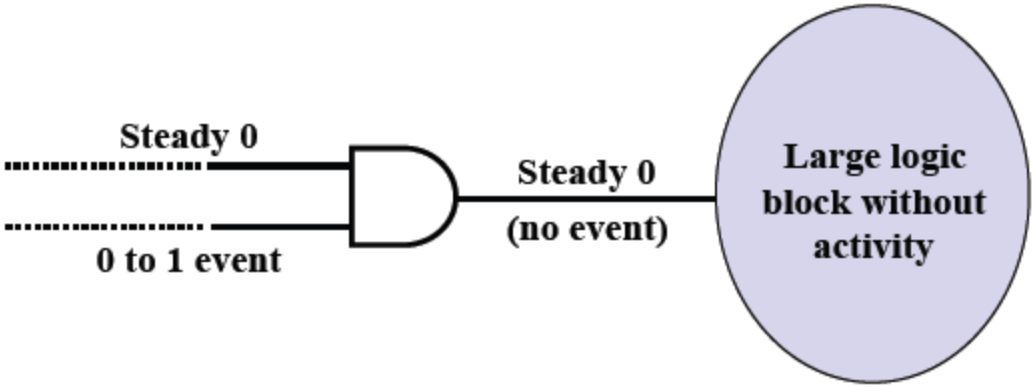
# Sequential Circuit



# Efficiency of Event-Driven Simulator

- Simulates events (value changes) only
- Speed up over compiled-code can be ten times or more; in large logic circuits about 0.1 to 10% gates become active for an input change



# Comparison

- Performance evaluation for event driven simulation
- Number of gates simulated per number of gates in circuit is a way of measuring the usage of event driven simulation and code compiled.
- More generally, for multiple input vectors:

  Activity Rate = Gates Simulated / ( # input vectors * # total circuit gates)

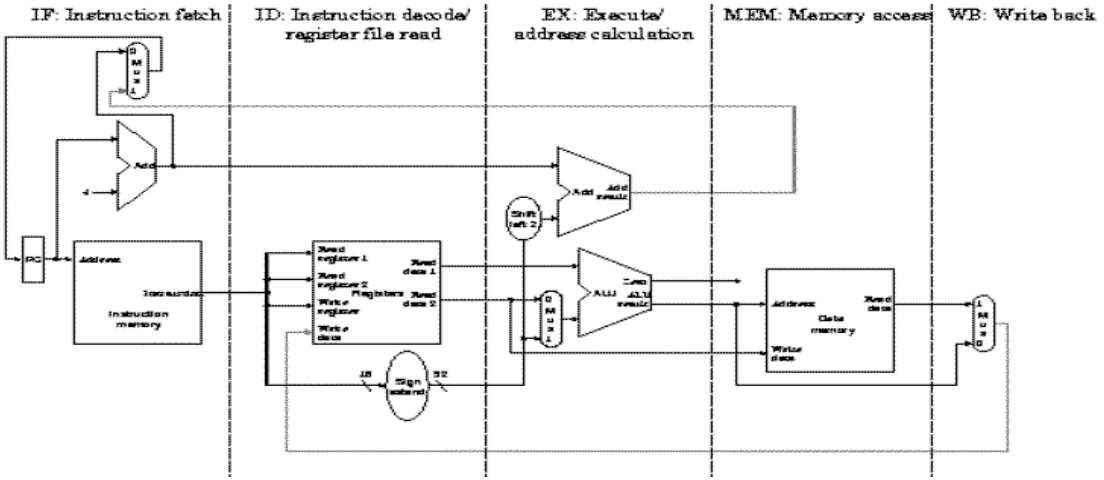
- The closer the activity rate is to 100% -> levelized
- The closer the activity rate is to 0% -> event driven

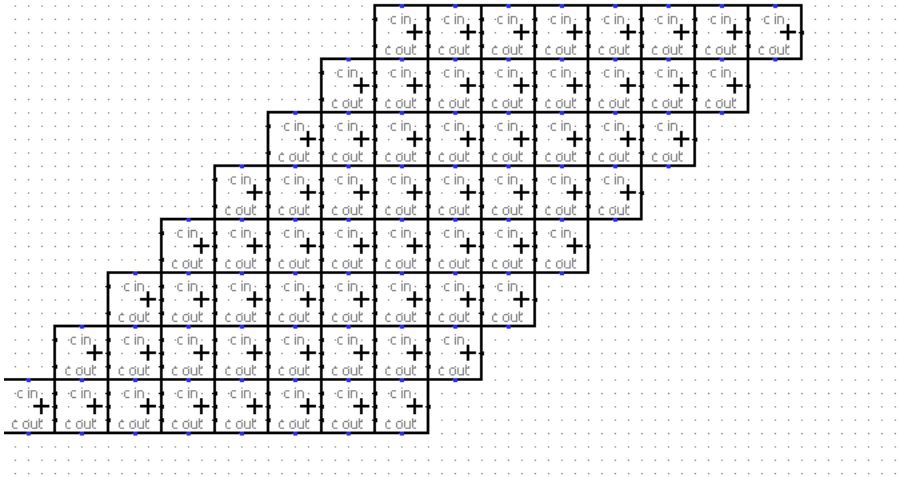
# Comparison CONTINUED

- The break-even point is NOT around 50% for when both perform equally, but rather at around 2-3%.

- Quick discussion:
- Which would be better for event-driven simulation and which better for levelized-code-compiled and why?
- 1) MIPS processor
- 2) Large array multiplier

# Comparison CONTINUED

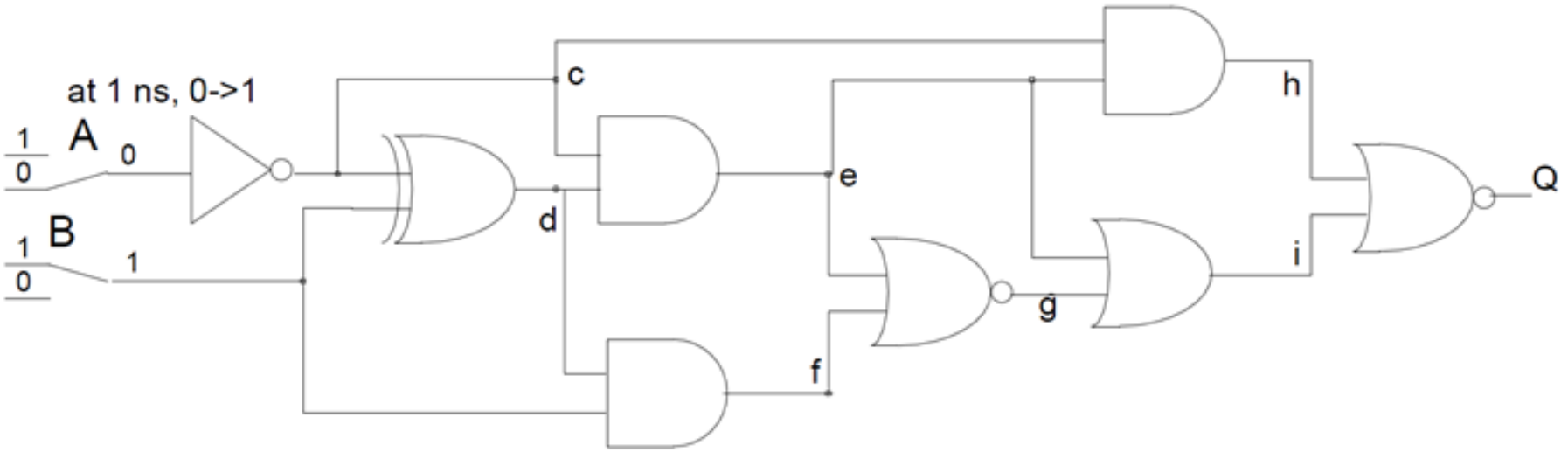


connectivity varies depending on type of binary multiplier.

# Quiz Question 2

**Q2.**




Initially $A = 0, B = 1; A\ changes\ to\ 1\ at\ time\ 1\ ns$. Inverters have a delay of 1 $ns$, OR and NOR gates have a delay of 2 $ns$, AND gates have a delay of 3 $ns$, and XOR gates haves a delay of 5 $ns$.

**a)** Compute event-driven simulation table

**b)** Plot timing diagram up until steady state