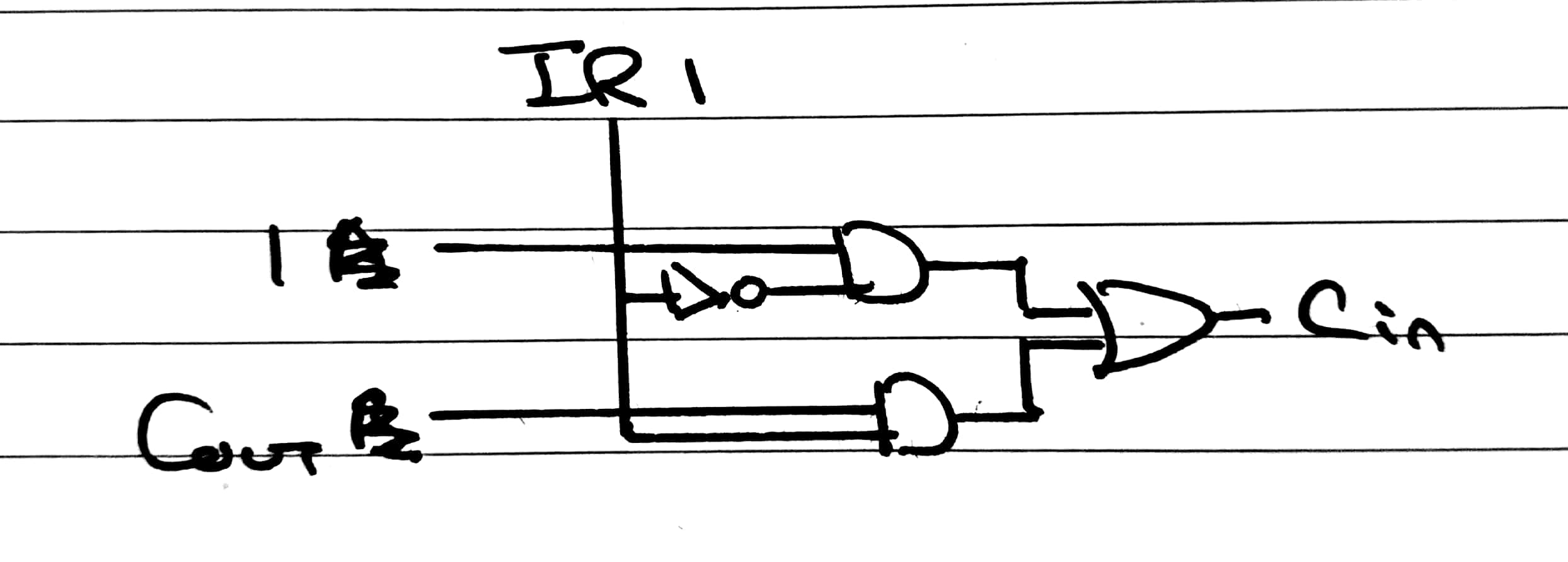
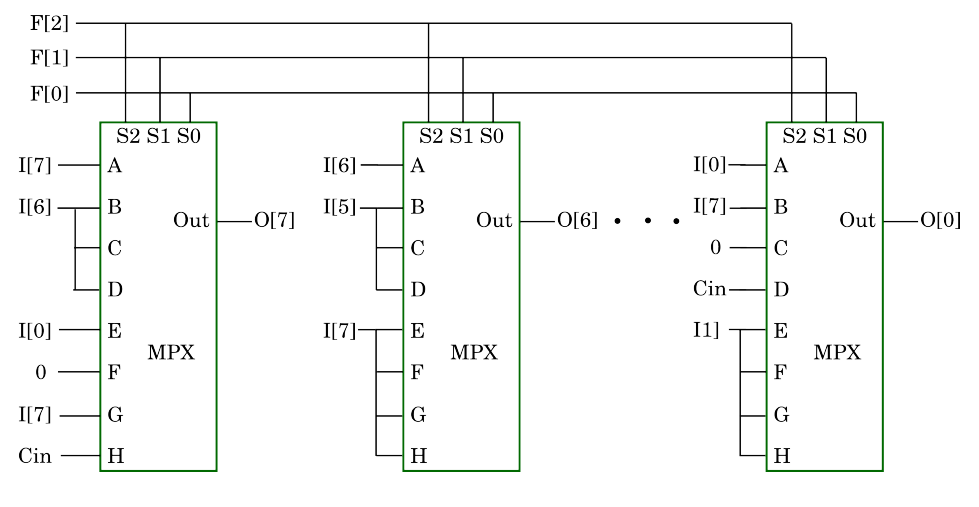
**w Question 1 - The Manual Processor**

(a)



(b)



(from <https://www.doc.ic.ac.uk/~dfg/hardware/HardwareLecture14.pdf>)

There is one multiplexer for each bit of the processor, and ceil(log2(no. bits of processor)) control bits to select the output bit.

If you’re still a bit confused like I was, basically I in this case just stands for input. O stands for output. There are 8 bits going in and you need 8 bits to come out. Each output bit has 1, 8 bit multiplexer that generates it. First of all focussing on the MSB i.e. O[7]. If control is 000, it’s unchanged so it should just be I[7]. If control is 001, 010 or 011 then O[7] should be I[6] shown above in the left most multiplexer.

(c)

We need to add A and B in the first cycle, then divide by two (arithmetic shift right) in the next cycle.

(If you're confused about the tables below, the explanation is on the first page here: <https://www.doc.ic.ac.uk/~dfg/hardware/HardwareLecture15.pdf>. The table always follows the strict pattern in the notes.)

|  |  |  |  |
| --- | --- | --- | --- |
| State | Source | Destination | Purpose |
| 1 | Data In | IR | First instruction byte to be loaded to set Cout to 0 and A to Data Iny |
| 2 | Data In | A | First number loaded into register A |
| 3 | Data In, Cout | B, C | Second number loaded into register B; C is set to 0 |
| 4 | Data In | IR | Second instruction byte to be loaded (A + B + 0) |
| 5 | ALU | RES, C | Results of computation stored in RES and C registers |

|  |  |  |  |
| --- | --- | --- | --- |
| State | Source | Destination | Purpose |
| 1 | Data In | IR | First instruction byte to be loaded to ensure ALU output remains (A + B) and A/RES select ALU result` |
| 2 | RES | A | Unchanged result from previous cycle loaded into A |
| 3 | Data In | B | Doesn't matter what B is; C set to contents of Cout |
| 4 | Data In | IR | Second instruction byte to be loaded (shift arithmetic right and RES selects shifter output) |
| 5 | Shifter | RES | Results of computation stored in RES (C will be 0 so not needed) |

Alternatively, you could load A and B in the first cycle, then both Add and Shift in the second cycle. This is how the same operation is attempted in the answers to Tutorial 2

(d) (To do this question, don't remember/follow the schematic for the instruction register in Duncan's notes, but instead see the diagram for the IR bits just in case the instruction format is modified in the paper [2014 paper])

For reference, bits from IR7 to IR0 respectively: 7 - IGNORE; 6 - SHIFTER/ALU; 5 - SHIFTER/ALU; 4 - SHIFTER/ALU; 3 - IGNORE; 2 - RES MPX; 1 - Cin MPX; 0 - A MPX

Cycle 1:

Steps 2-4: want to use ALU to set Cout to 0 and set A to Data In so X000XXX1

Step 5: want to add A and B with ALU, store results in RES/Cin so X011X11X

Cycle 2:

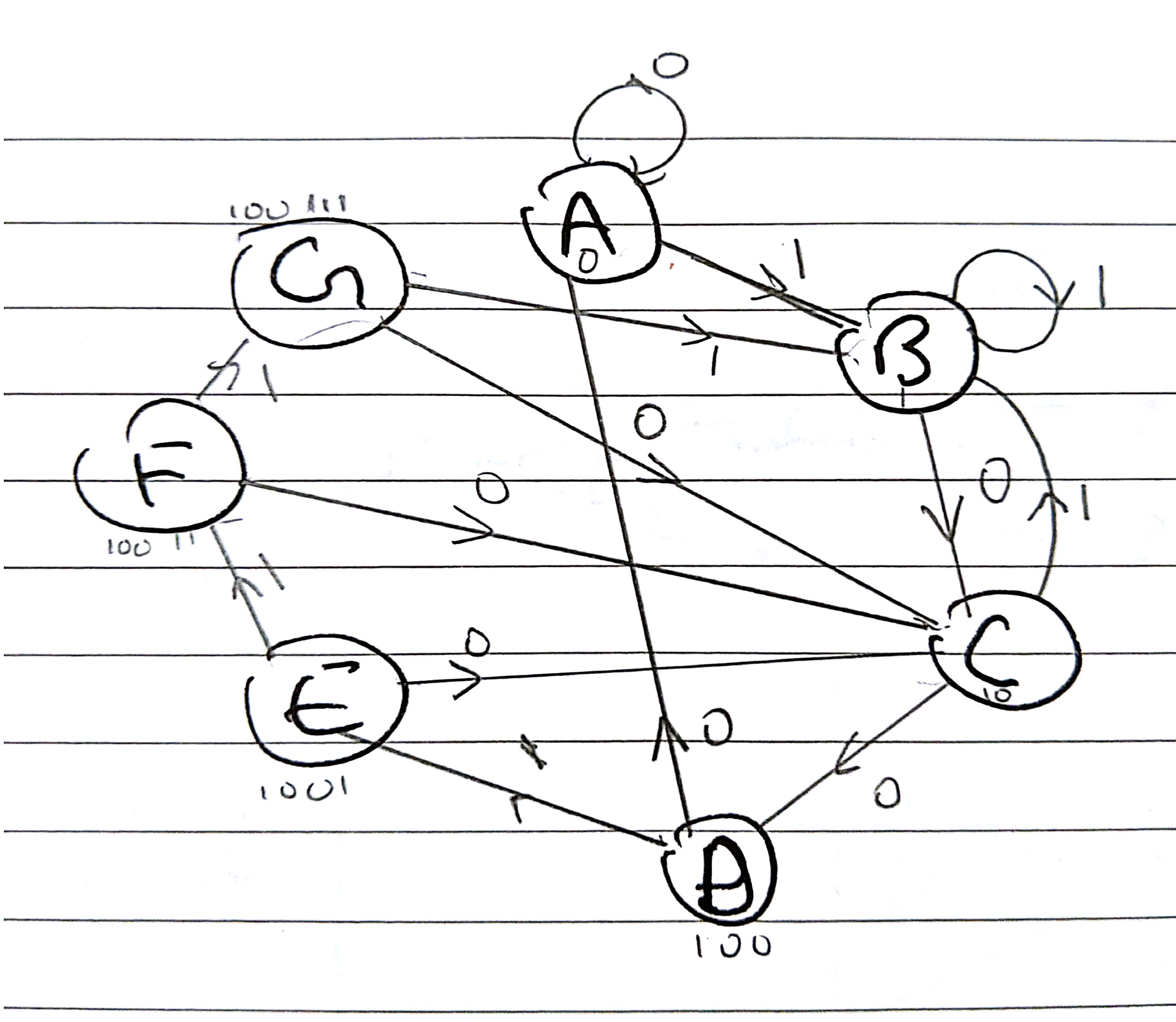
Steps 2-4: do A + B again and load result into A/RES so XXXXXXX0

Step 5: shift arithmetic right on A and load result into RES so X110X0XX

q

**Question 2 - A Sequence Recognition Circuit**

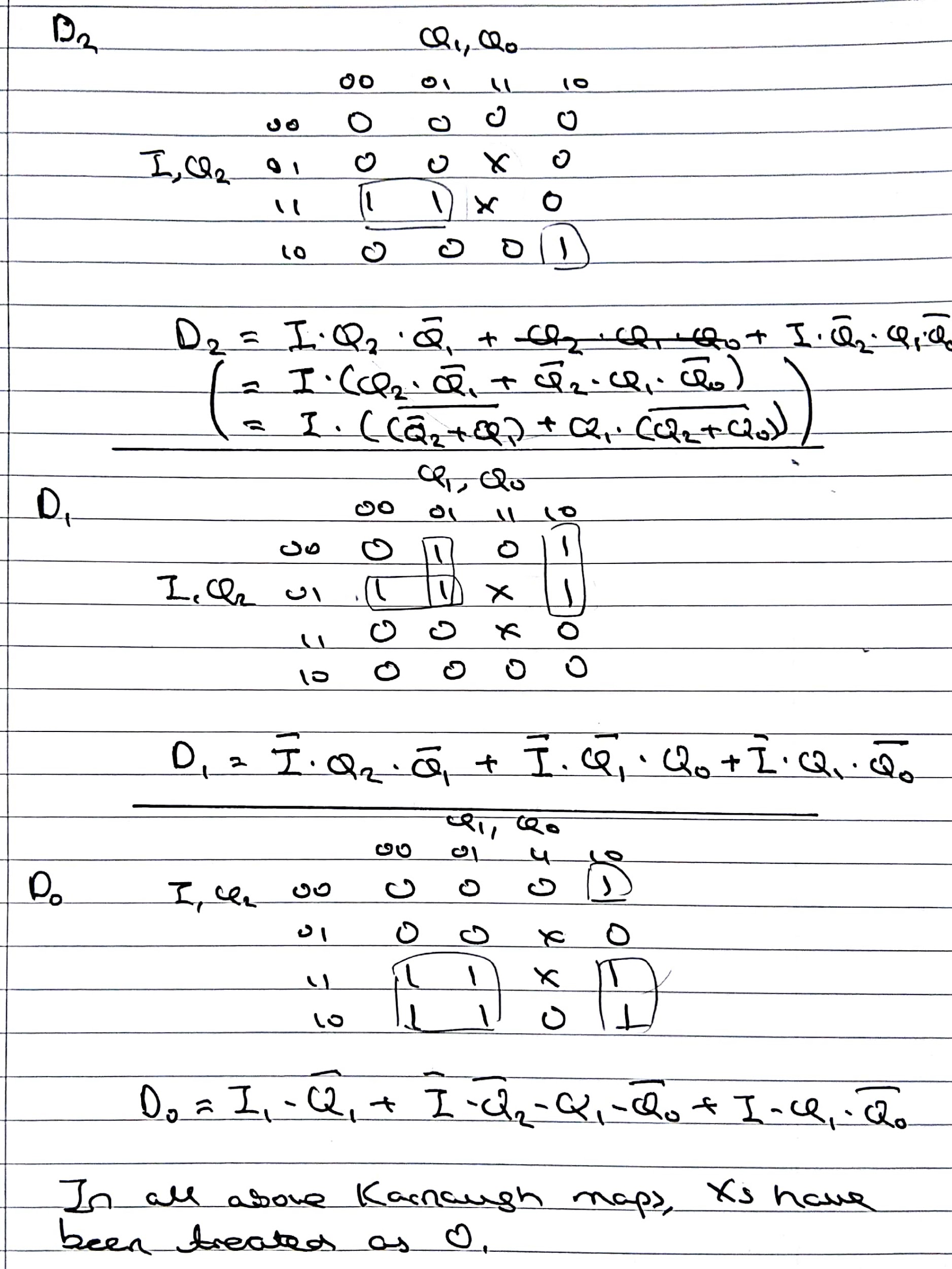
(a)



(b)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input | This State | Q2 | Q1 | Q0 | Next State | D2 | D1 | D0 |
| 0 | A | 0 | 0 | 0 | A | 0 | 0 | 0 |
| 0 | B | 0 | 0 | 1 | C | 0 | 1 | 0 |
| 0 | C | 0 | 1 | 0 | D | 0 | 1 | 1 |
| 0 | D | 0 | 1 | 1 | A | 0 | 0 | 0 |
| 0 | E | 1 | 0 | 0 | C | 0 | 1 | 0 |
| 0 | F | 1 | 0 | 1 | C | 0 | 1 | 0 |
| 0 | G | 1 | 1 | 0 | C | 0 | 1 | 0 |
| 0 | Unused | 1 | 1 | 1 | X | X | X | X |
| 1 | A | 0 | 0 | 0 | B | 0 | 0 | 1 |
| 1 | B | 0 | 0 | 1 | B | 0 | 0 | 1 |
| 1 | C | 0 | 1 | 0 | B | 0 | 0 | 1 |
| 1 | D | 0 | 1 | 1 | E | 1 | 0 | 0 |
| 1 | E | 1 | 0 | 0 | F | 1 | 0 | 1 |
| 1 | F | 1 | 0 | 1 | G | 1 | 1 | 0 |
| 1 | G | 1 | 1 | 0 | B | 0 | 0 | 1 |
| 1 | Unused | 1 | 1 | 1 | X | X | X | X |

(c)



K maps with corrections from the state transition table below (Please correct if wrong xx)

(d)

Output is when Q2Q1Q0 is state G, so output is when Q2Q1Q0 = 110

(e)

In all three Karnaugh maps, the Xs have been treated as 0. Therefore, when the circuit goes into the unused state, D2D1D0 becomes 000, so the circuit goes to state A. (The circuit does not enter a loop as input 1 will cause the circuit to go from state A to state B.)

(changed because kmaps changed?)

