CS211 Summer 2020

Final Sample Questions

The final will be cumulative, meaning it will include all material learned since the beginning of the course. That being said, make sure to review the midterm questions and previous topics as well as use the sample questions below as a guide when reviewing for the topics covered post midterm.

# Storage and Caching

### 1. What is the definition of RAM? and what is the difference between SRAM and DRAM.

RAM – Random access memory. Packaged as a chip, basic storage is a cell.

SRAM – Faster than DRAM, uses less power, retains value until power off, insensitive to electrical noise disturbance.

DRAM – holds more data than SRAM, values must be refreshed every 10-100ms, sensitive to disturbance.

### 2. List out the memory high-archy from fastest to slowest.

Registers, SRAM (on-chip), SRAM (off-chip), DRAM (main memory), Local Disks, Web Servers

### 3. What is caching and why do we need it?

Caching – the process of storing data on faster components in order for future requests to be delivered faster. *A smaller, faster storage device that acts as a staging area for a subset of the data.* We need caching because it makes the computer more efficient by storing frequently used data that can be used by the CPU immediately after.

### 4. What is cache associativity and why does it matter?

A cache is divided into groups called sets (tag, index, offset). Helps users understand and specify the location in memory.

### 5. What are different cache eviction strategies, what do they do, and why would one use one over the other?

FIFO, LRU, Random. Helps to decide the optimum algorithm, the most used one today is LRU (least recently used) because there are more cache hits on average. FIFO is easier to program but has low performance overhead.

### 6. What is the difference between write-through and a write-back cache.

Write-through: immediately write data to memory when a write-hit.

Write-back: defer the write to memory as long as possible when a write-hit

### 7. A computer uses 32-bit addressing, 2-way associative cache with a capacity of 32KB. Each cache block contains 16 bytes. Calculate the number of bits needed for the tag, set and block offset.

Address = 32 assoc = 2 CS= 32KB B = 16

Bits for Offset: log2(B) = log(16) = 4 n = 4

Bits for Sets: log2(CS/B\*assoc) = log2(32KB/16\*2) = log2(215 / 25) = log2(210) = 10 n = 10

Bits for Tags: 32 – 4 – 10 = 18 n = 18

### 8. What is Temporal Locality? Spatial Locality?

Temporal locality: recently accessed locations that will likely access again in near future. Code within a loop, same instruction fetched repeatedly. (reuse of data)

Spatial locality: access locations close to ones recently accessed in near future. Data arrays, local variables in stack, Data allocated in chunks like malloc. (use of data close to storage locations)

### 9. Assume we have the following code for multiplying two n by n matrices, a and b, into matrix c:

int i, j, k, sum;

for(i = 0; i < n; i++){

for(j = 0; j < n; i++{

for(k = 0; k < n; k++)

sum += a[i][k] \* b[k][j];

c[i][j] = sum;

}

}

Assume that they are row major order matrices (rows stored sequentially). Also assume we have a block size of 32 bytes, and each element in the matrix is 8 bytes, what is the expected number of misses per inner loop iteration?

Not sure:

a: .25, b:1, c:0

.25+1+0 = 1.25 expected misses per loop

# Digital Logic

### 1. What are the differences between combinational vs sequential circuits?

Combinational: logical gates, behavior is stateless, outputs are function of inputs only

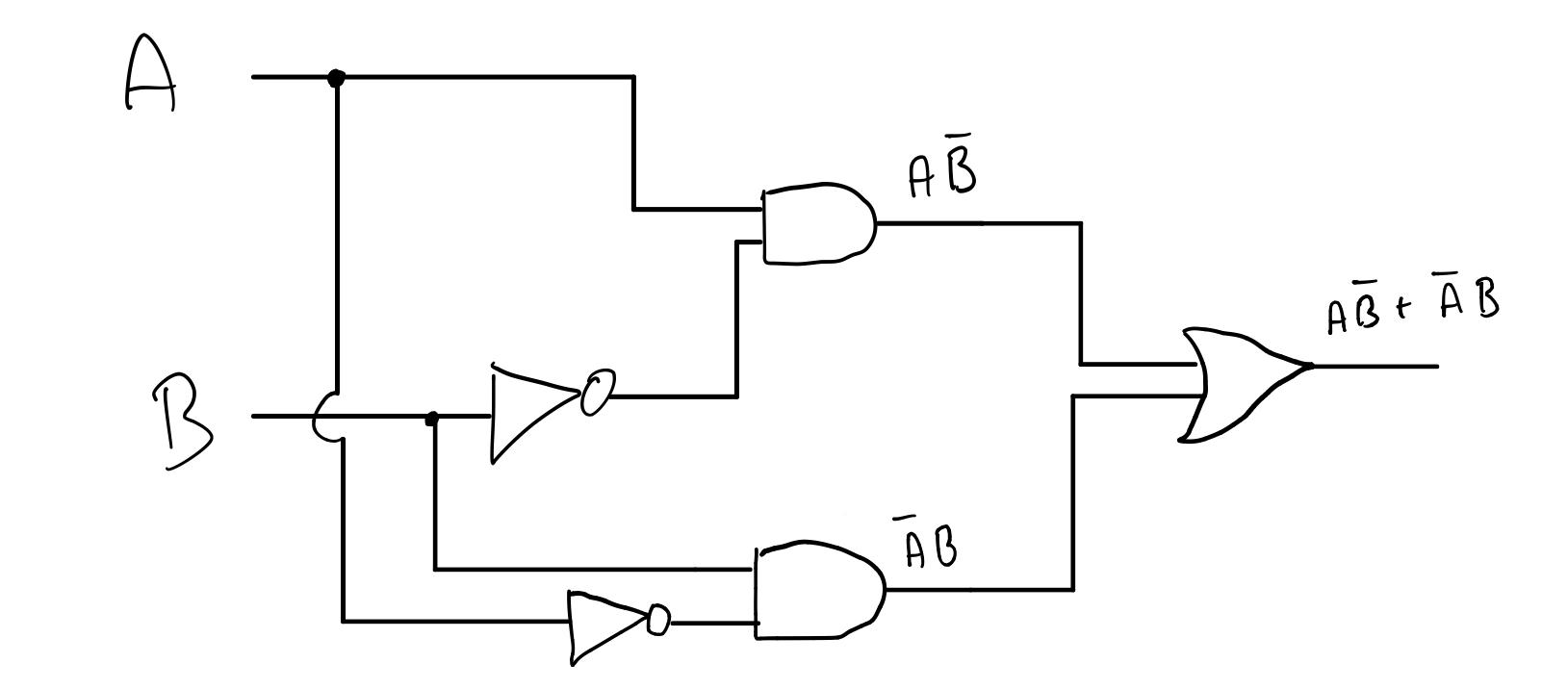
Sequential: loops, outputs depend on both current and prior values, has memory

### 2. Name the basic logic gates.

Or, And, Not, Nand, Nor, Xor

### 3. Show XOR in terms of AND, OR, and NOT gates.

A xor B == ABc + AcB



### 4. What are decoders? multiplexers?

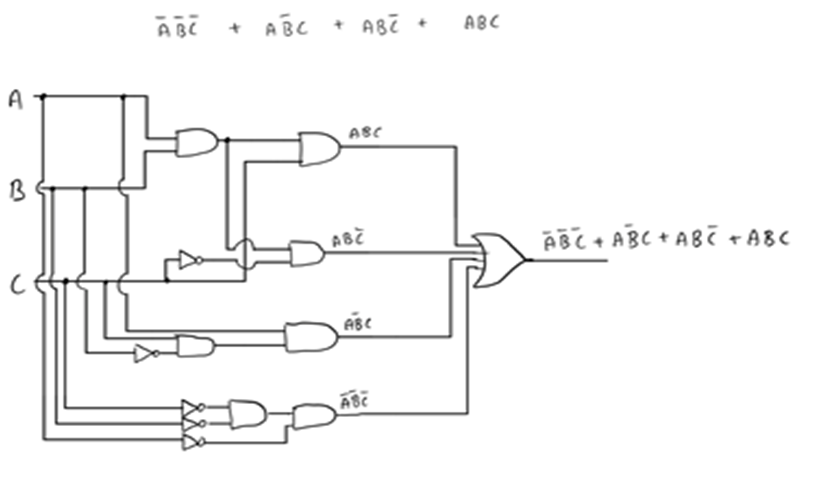
Decoder: converts n inputs to 2n outputs

Multiplexer: converts 2n inputs with n bit selector to 1 output

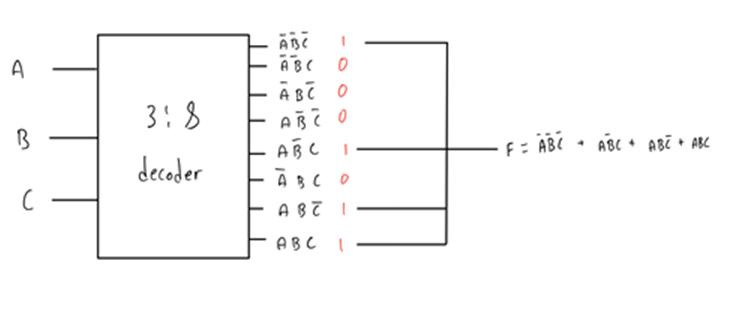
### 5. Assume the following truth table for some arbitrary function f based on inputs A, B, and C:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Output |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

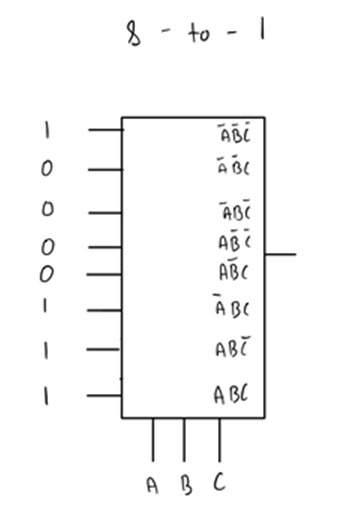
#### (a) Derive the unsimplified Boolean expression for f using minterms and draw out the circuit.



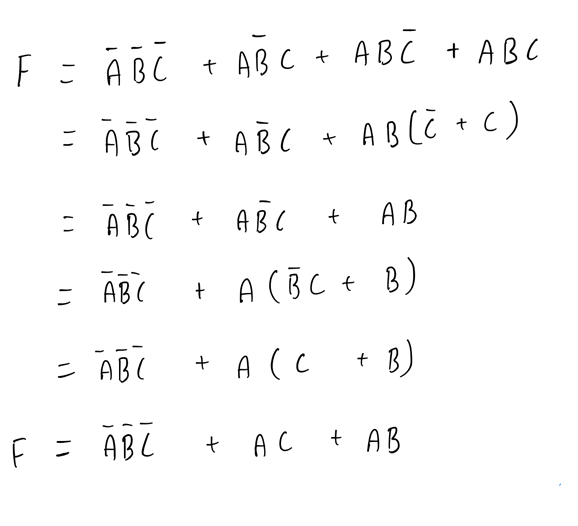
#### (b) Implement f using a 3:8 decoder



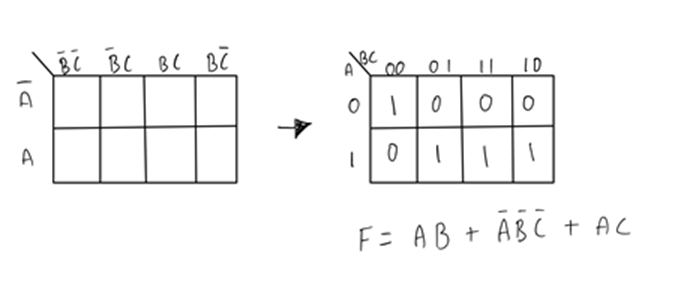
#### (c) Implement f using a 8:1 multiplexer.



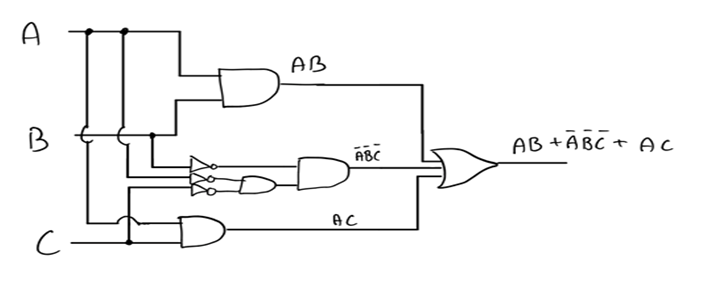
#### (d) Simplify the boolean expression for f using boolean algebra. Show your work.



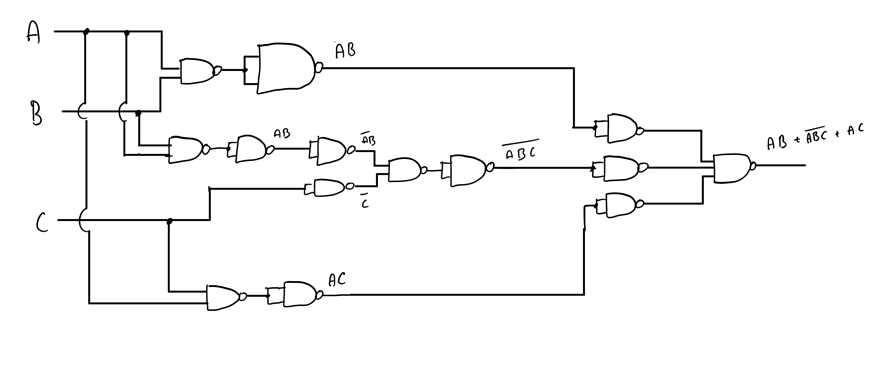
#### (e) Simplify the boolean expression for f using Kmaps. Show your work.



#### (f) Draw the circuit for the simplified boolean expression.



#### (g) Convert the simplified boolean expression to only use NAND gates.



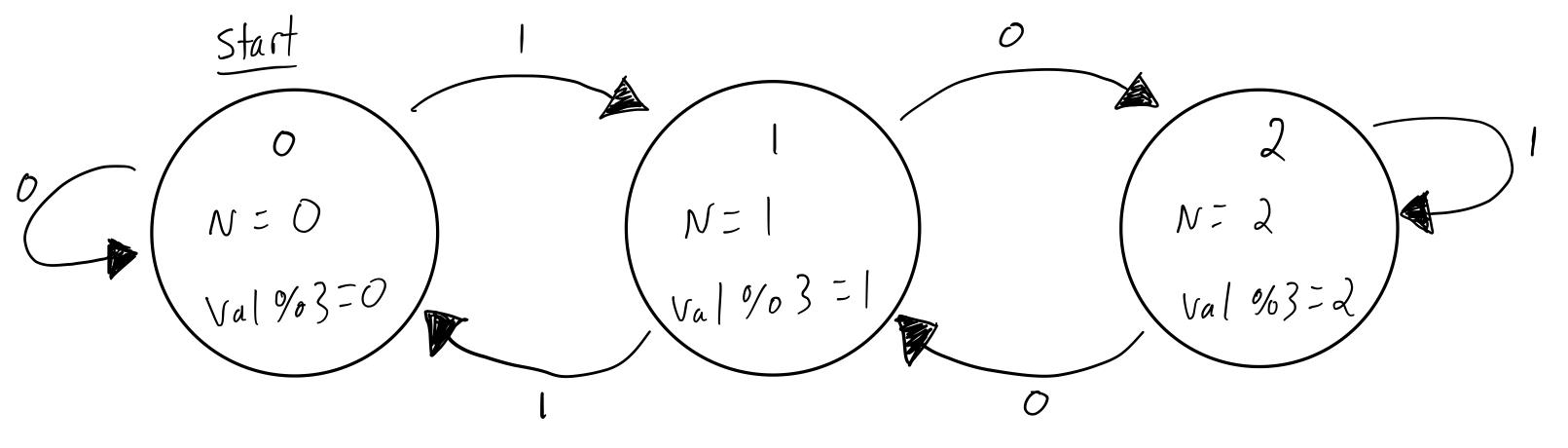
#### **(h)** **Draw the circuit representing the converted boolean expression using only NOT gates and 2-input NAND gates.**

### 6. What is the difference between a latch and a flip- flop? Why is a D flip- flop better than a D latch in achieving consistency?

A latch has the ability for an enable and is asynchronous, the outputs can change as soon as the inputs do. A flip-flop is edge-triggered and only changes state when a control signal goes from high to low or low to high. Flip-flops are better because they are only triggered at the edge of the clock pulse while latches are level triggered and are prone for more glitches.

### 7. Construct a "divisible by 3" FSM that accepts a binary number entered 1 bit at a time, MSB first, and indicate with a light if the number entered so far is divisible by 3.

#### (a) Draw the state transition diagram.



#### (b) Encode the states.

Let:

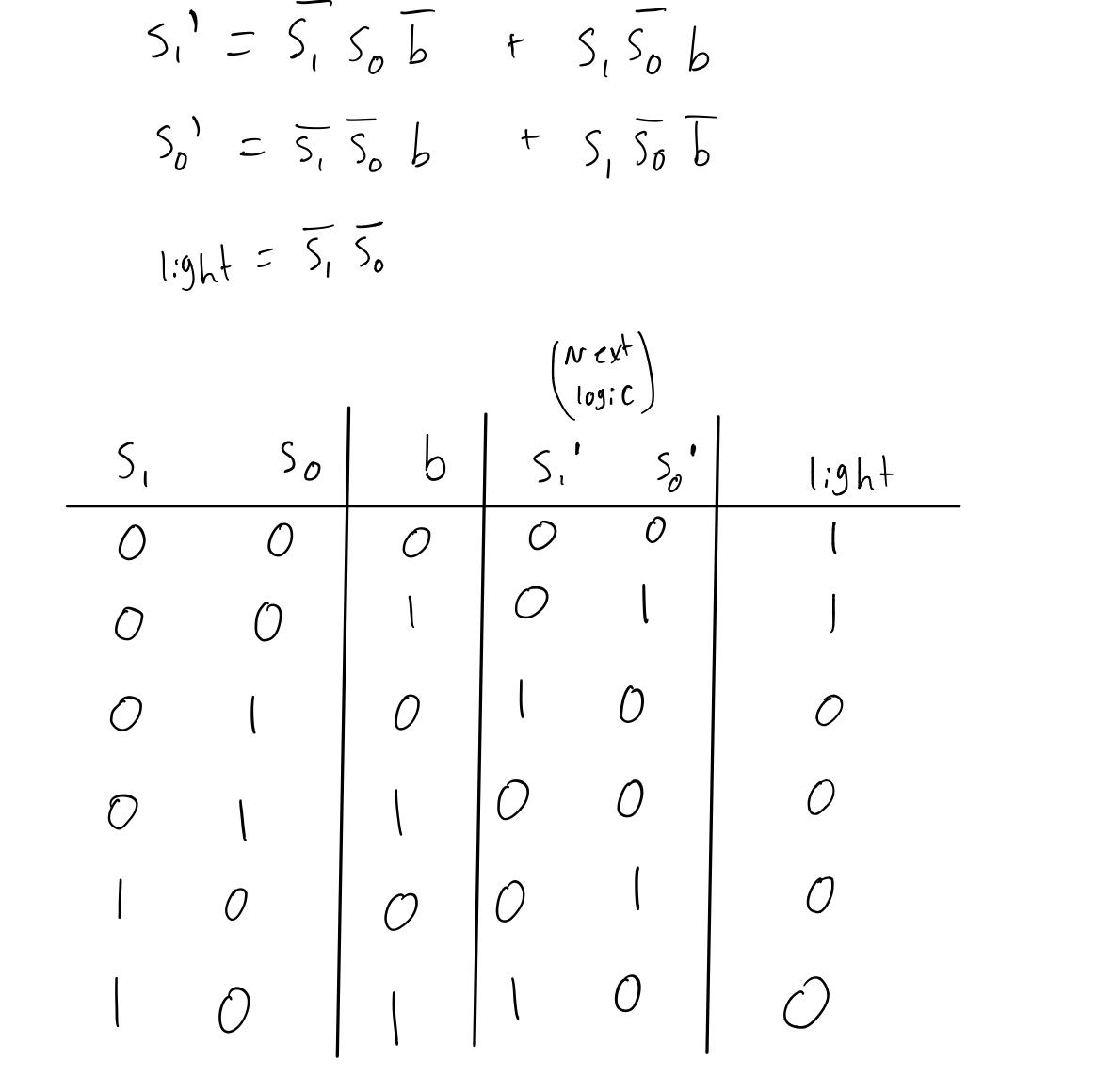
0 = 00

1 = 01

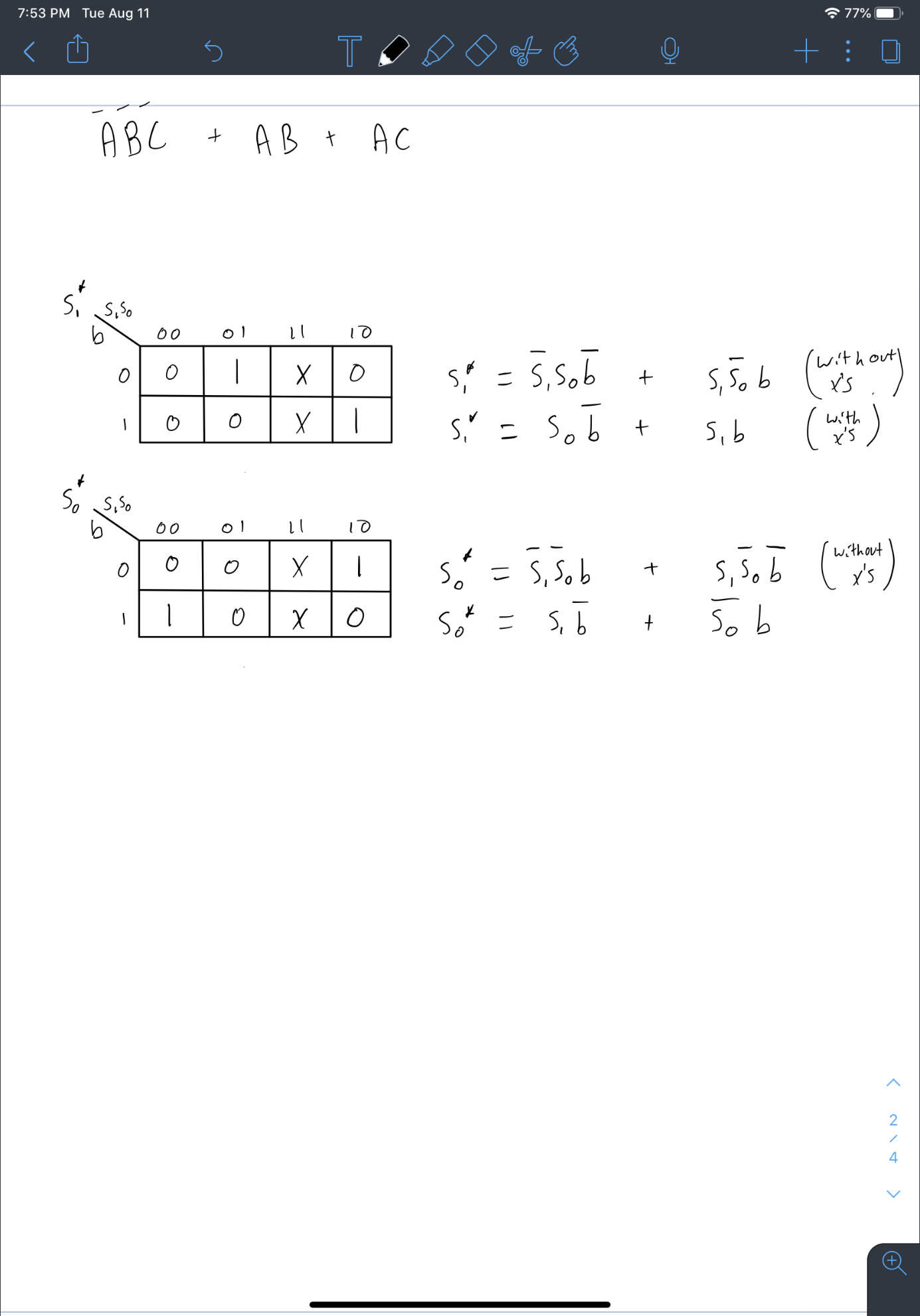
2 = 10

#### (c) Draw the state transition table with the encoded states

Keep in min first 2 rows are 0, next 2 are 1, last 2 are 2



#### (d) Derive the boolean expressions for the next encoded state

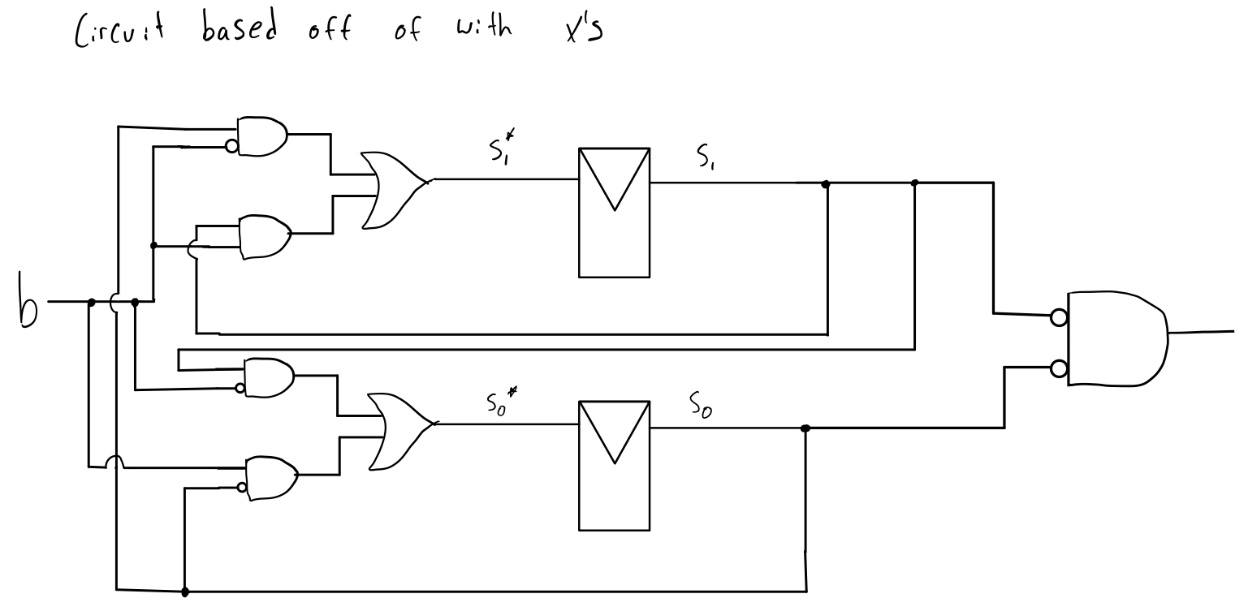


#### (e) Derive the boolean expression for the light signifying the output light based on the state transi-tion table.

\_ \_

Light = s0 s1

#### (f) **Draw out the circuit using two D flip-flop and basic logic gates.**



What is a ripple carry adder?

A digital circuit that produces the arithmetic sum of two binary numbers. Can be constructed with full adders connected in cascaded (check bookmark)

What is a carry lookahead adder?

Solves the carry delay problem from ripple carry adder computing the sum.