Solution (Basics 2. Sequential Logic)

Problem 1

1) Flip-flop: changes on a clock edge

D flip-flop:

D	D_{n-1}	D_n	D_{n+1}
Q	Q_{n-1}	Q_n	Q_{n+1}
CLK ↑	n-1	↑ n	↑ n+1

D_n	Q_{n+1}
0	0
1	1

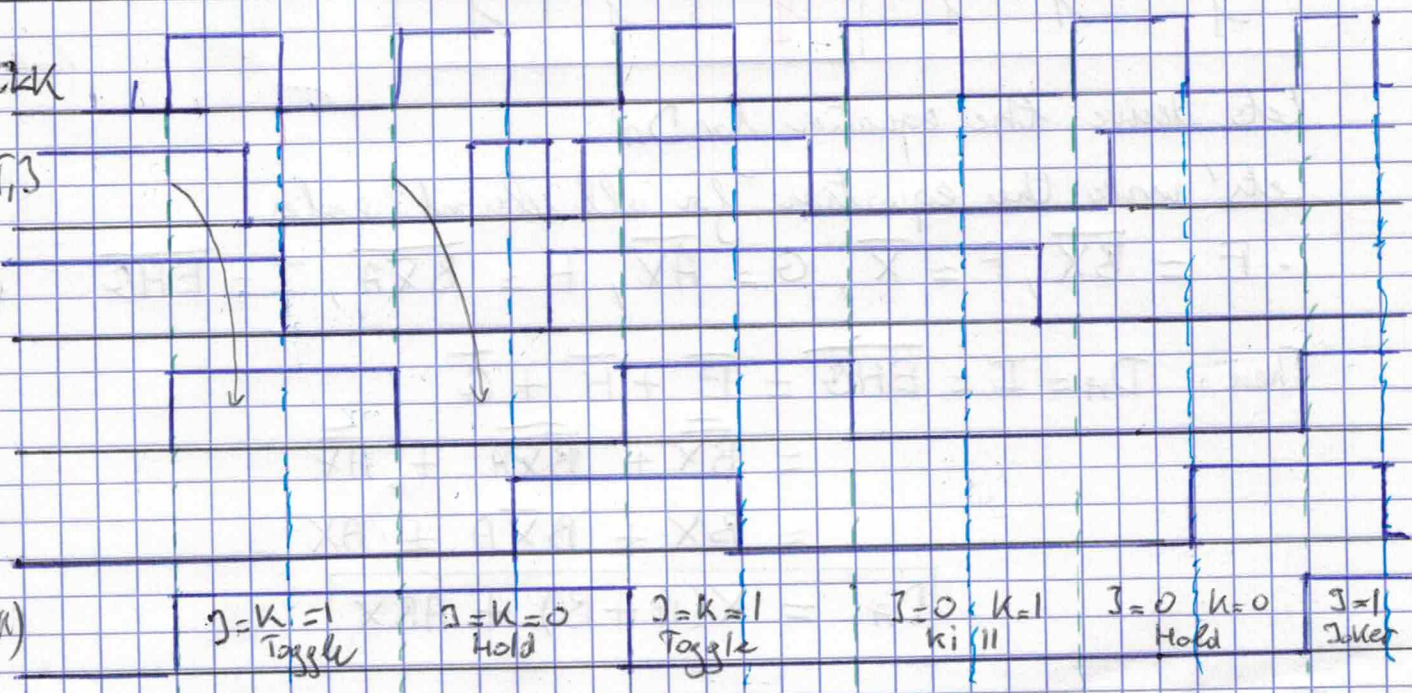
Analogy: think of a flip-flop as a device that changes the input into the output (sink).

However, we should think of it as "what is going to be the next output given the current one?"

Similarly: $T \rightarrow Q_{n+1} = \bar{T} Q_n + T \bar{Q_n}$

$JK \rightarrow Q_{n+1} = J\bar{Q_n} + \bar{K} Q_n$

2)



Exercise 2

Before starting the problem, let's look a bit for the low hanging fruits:

- $Z = B + X$ → ⚠ X is asynchronous!
- "$B_{n+1} = A_n$" (B is linked to the FF_A flip-flop)

We can start the transition table now with these remarks (written in blue)

A	B	X	D_A	D_B	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Lets derive the equation for D_A

Lets' write the equation for all relevant gates

- $E = \overline{BX}$, $F = \overline{X}$, $G = \overline{AX}$, $H = \overline{B\bar{X}A}$, $I = \overline{EHG}$

Then: $D_A = I = \overline{EHG} = \bar{E} + \bar{H} + \bar{G}$

$= \overline{\overline{BX}} + \overline{\overline{B\bar{X}A}} + \overline{\overline{AX}}$

$= BX + B\bar{X}A + AX$

$$\boxed{D_A = X(A+B) + AB\bar{X}}$$

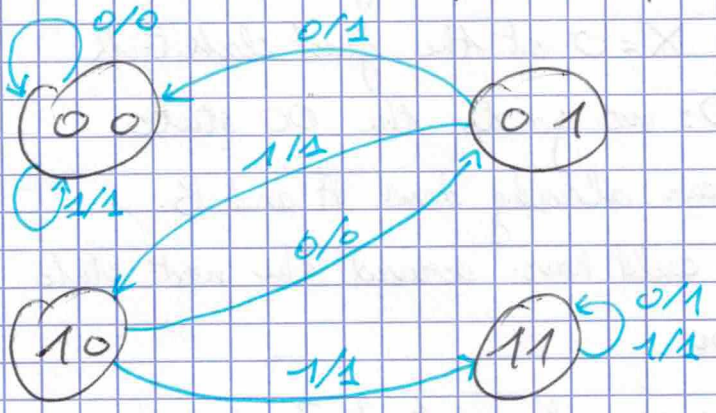
Then we can finalize the state transition table, written in purple.

Problem 2 (cont.)

2) State Diagram

Our state is defined by A and B (current state)

It can be written 00, 01, 10, 11 → 4 states



Let's build the transitions in turquoise

State 00 $A = 0, B = 0$. Only variable, the input X.

- If $X = 0$; $D_A = 0$, $D_B = A = 0$ and $Z = B + X = 0$
- If $X = 1$; $D_A = 0$, $D_B = 0$ and $Z = 1$

Remark: the output depends on the state AND the input.
It's a Mealy Machine!

State 01: $A = 0, B = 1$.

- If $X = 0$, $D_A = 0, D_B = 0$, $Z = 1$
  change of state! $AB \neq D_A D_B$
- If $X = 1$, $D_A = 1$, $D_B = 0$, $Z = 1$

And similarly for the two other states

Sanity check: check that all states have two transitions
(the input X takes $2^1 = 2$ values)

Interpretation

At clock tick n, we are in a state (e.g 01)
Given an input, we can know the next state
AND the current output at the last clock tick

PG 2 (cont.)

3) The "current state" (before the first clock tick) is A=0, B=1

We have to find the "next state"

- We can go to 10 if X=1 at the next first clock tick
- We can go to 00 if X=0 at the first clock tick

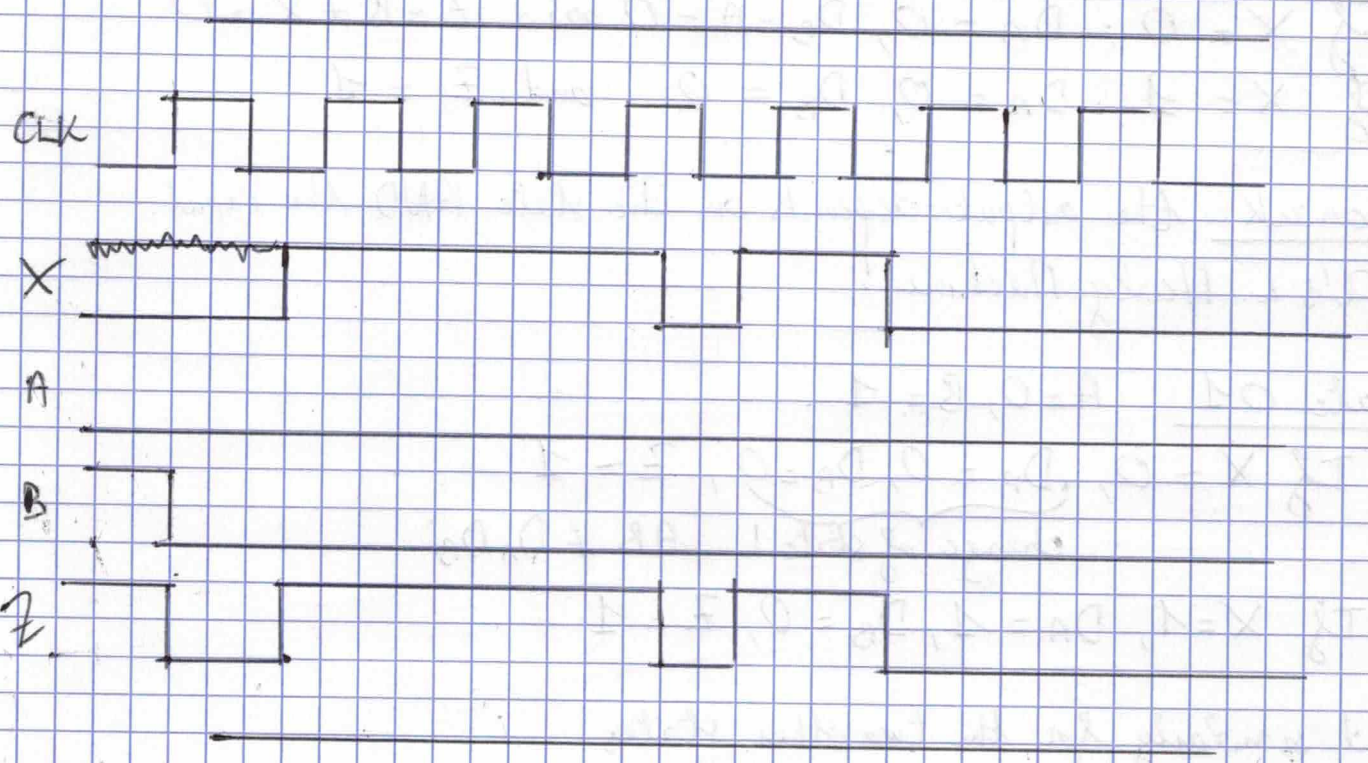
At the clock tick, X=0: we go to the 00 state

00 is a dead end: we can already draw from A and B.

If X were synchronous we could have derived the next state sooner by stating it is 00.

Also, because X is asynchronous, the output Z can change in between the clock ticks.

Matter of fact Z=X after the first clock tick!



Problem 3

Before we begin, let's look at the problem
We have a 3-input full adder, with carry output

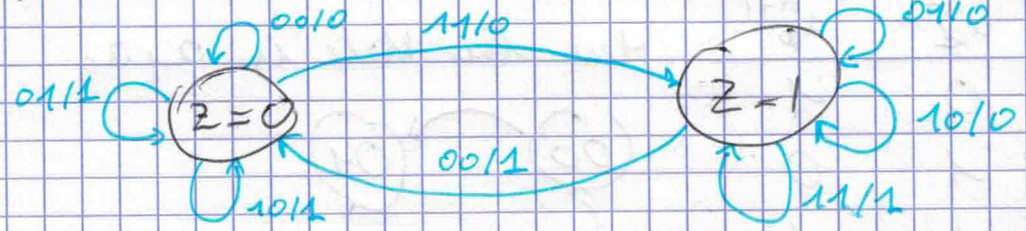
4)

X	Y	Z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

First, the circuit is the sequential: it has
- 2 inputs X and Y
- A "current state" Z
- A "next state" C
- An output S

1) Just fill the truth table. It's just an adder

2) The "current state" is Z → 2^1 states



3) Mealy: S depends on the state Z and the inputs X and Y

4) $T_c \geq t_{pcq} + t_{pd} + t_{setup} = 2 + 4 + 2 = \boxed{8\ ns} \leftrightarrow 125\ MHz$

Problem 4

1) For a JK flip-flop $Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$

Or as a truth table, on a clock edge

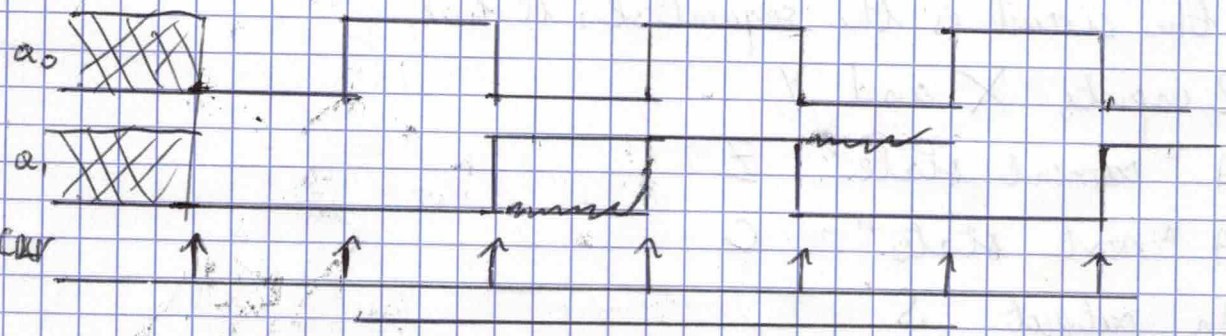
J	K	Q_n	Q_{n+1}	
0	0	X	Q_n	Hold
0	1	X	0	Reset / Kill
1	0	X	1	Set / Solder
1	1	Q_n	$\overline{Q_n}$	Toggle

X = 0 or 1

2) Counter: something that counts (duh)

↳ 2 bits: 0, 1, 2, 3 ↔ 00, 01, 10, 11

Let's start with a "perfect" chronogram ie one that makes the function. Let $a_1 a_0$ be the output of the counter

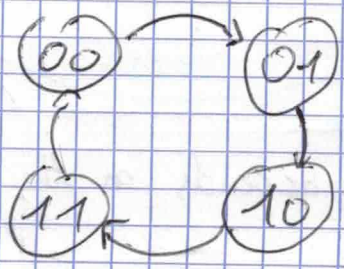


Observations - a_0 toggles at each clock tick

- a_1 toggles when a_0 was 1 right before the clock tick.

a_1^n	a_0^n	a_1^{n+1}	a_0^{n+1}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

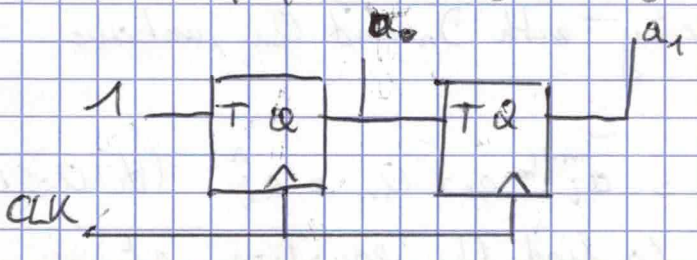
Here the state is $a_1 a_0$



We can assume there is a constant input

Problem 4 (cont.)

We can propose the following schematic



Problem 5

Before we begin, let's look at the problem overall. We need
- To count on 3 bits → 0, 1, 2, ..., 7
- To count up or down: either 0, 1, 2 ..., 7 or 7, 6, 5, ..., 0, 7...
- To have a reset, we would need to force the "next state" to be 0.

Note: The exercise was first written to have a reset in the circuit. As it was not that interesting, we can get rid of it

1)

a_2^n	a_1^n	a_0^n	U	a_2^{n+1}	a_1^{n+1}	a_0^{n+1}
0	0	0	0	1	1	1
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	1
1	1	0	1	1	1	1
1	1	1	0	1	1	0
1	1	1	1	0	0	0

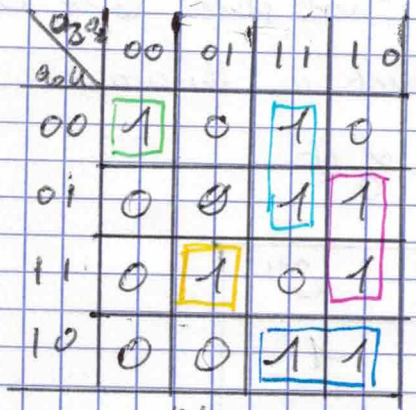
2) The state transition table gives the output directly for D-flip-flops

To use the D flip-flop of writing with D_n and Q_n, we have $Q_{n+1} = D_n$.

In the transition table, $Q_{n+1} = a_i^{n+1}$ and $Q_n = a_i^n$ $(\forall i\ 0 \leq i \leq 2)$

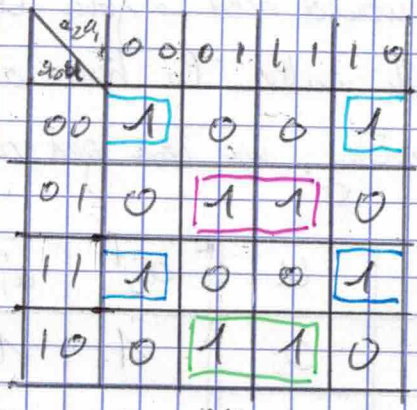
By using K-maps, we will try to find the equation at the inputs of the flip-flops to encode the state transitions.

In our model, we have:
- 1 input: u. 1 if counting up, 0 if counting down
- 3 state variables: a_i



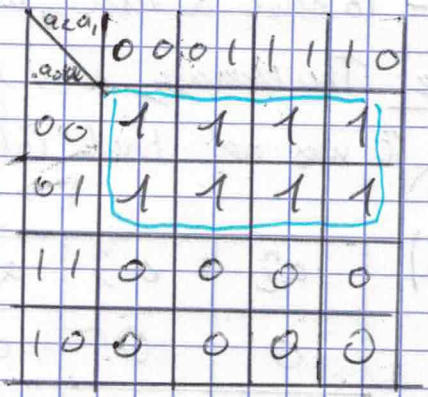
a_2^{n+1}

$a_2 a_1 \overline{a_0}$
$+ a_2 \overline{a_1} u$
$+ a_2 a_0 \overline{u}$
$+ \overline{a_2} \overline{a_1} \overline{a_0} u$
$+ \overline{a_2} a_1 a_0 u$



a_1^{n+1}

$\overline{a_1} \overline{a_0} u$
$+ a_1 \overline{a_0} u$
$+ \overline{a_1} a_0 u$
$+ a_1 a_0 \overline{u}$



a_0^{n+1}

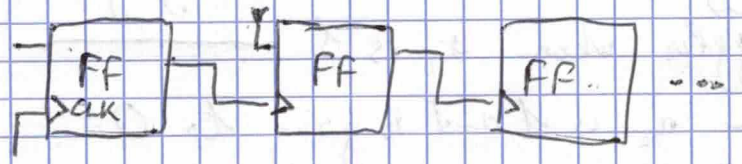
$= \overline{a_0}$

3) Not difficult if you take an empty piece of paper

Problem 6

74-393

As described, the 393 is a ripple counter.
A ripple counter has a structure similar to the following



For a 4-bit counter the state-transition table is:

q_3^n	q_2^n	q_1^n	q_0^n	q_3^{n+1}	q_2^{n+1}	q_1^{n+1}	q_0^{n+1}
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

We can derive the equation for the counter by a method similar to problem 5.

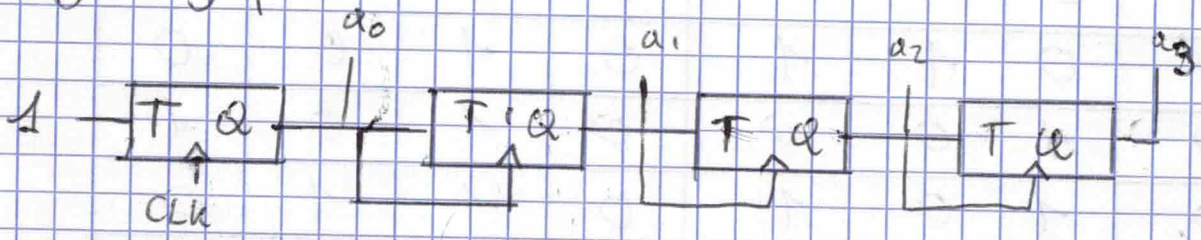
But we can be smart !

- The LSB (a_0) toggles on each clock tick.
- The second bit (a_1) toggles when a_0 is 1 and is going to 0
- The third bit (a_2) toggles when a_1 is 1
- The MSB toggles when a_2 is 1 and is going to 0

Let's analyze this remark

- "From 1 to 0" : falling-edge trigger
- "Toggling" : T flip-flop

The following is hence valid for falling-edge triggered flip-flop



If we wanted the rising-edge triggered version, we would just need to invert CLK

Problem 6 (cont.)

74-163

We can build a synchronous 4-bit counter with a so-called "Look-through" method

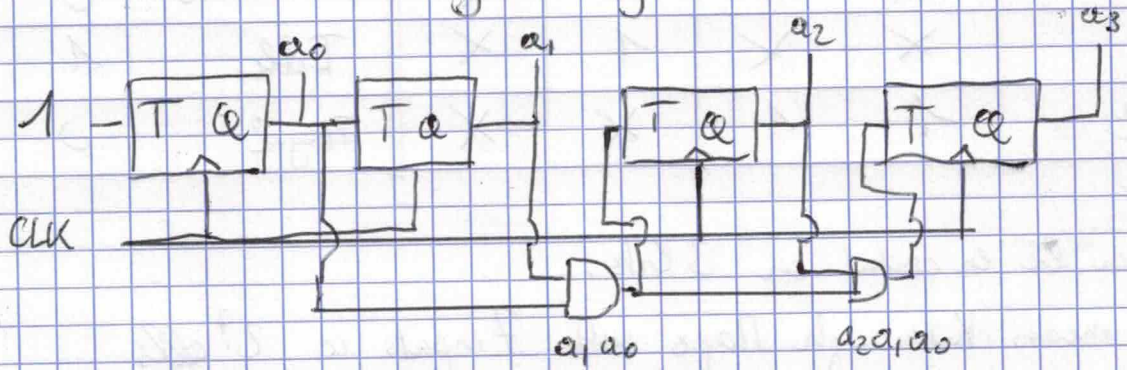
- a_0 toggles on each clock tick
- a_1 should go to 1 or 0 when a_0 is 1 in the current state. In other words it toggles when $a_0 = 1$
- a_2 will toggle when both a_0 and a_1 are 1
- a_3 ——————— a_2, a_1 and a_0 are 1s.

So

$$\begin{cases} a_0^{n+1} = \overline{a_0^n} = \overline{a_0^n} \times 1 + 0 \times a_0^n \\ a_1^{n+1} = \underbrace{a_0^n \overline{a_1^n}}_{\text{Toggling 1}} + \underbrace{\overline{a_0^n} a_1}_{\text{Hold if 0}} \\ a_2^{n+1} = \overline{a_2^n}(a_0 a_1) + a_2^n(\overline{a_0} + \overline{a_1}) \\ a_3^{n+1} = \overline{a_3^n}(a_0 a_1 a_2) + a_3^n(\overline{a_0 a_1 a_2}) \end{cases}$$

So with T flip-flops, of form $Q_{n+1} = \bar{T}\overline{Q_n} + \bar{T}Q_n$

we can derive the following circuit



No need to consider falling or rising edge here as the clock is not part of the values

Problem 4

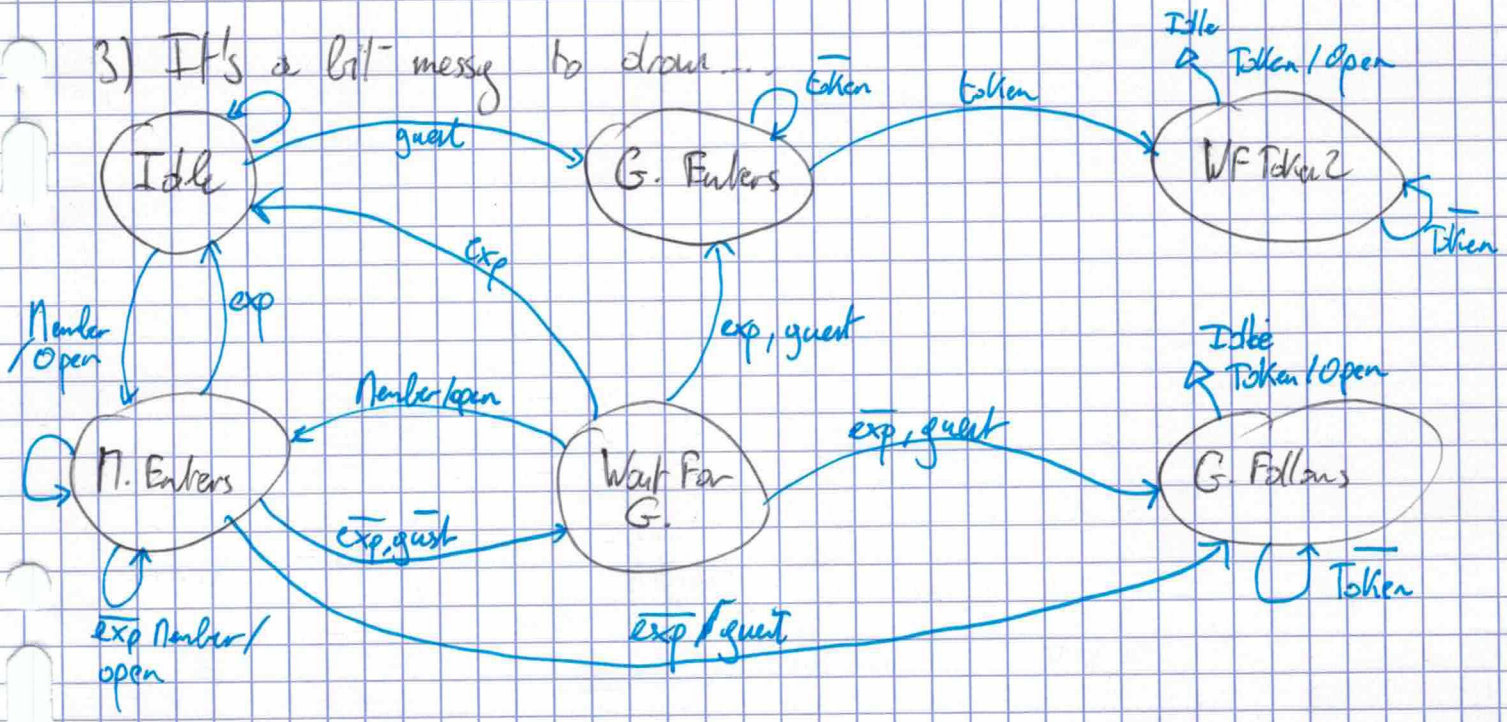
1)

State Name	Member	Guest	Token	Exp	Next State	Open
Idle	0	0	X	X	Idle	0
Idle	0	1	X	X	Guest Enters	0
Idle	1	0	X	X	Member enters	1
Member Enters	0	0	X	0	Wait for Guest	0
"	0	0	X	1	Idle	0
"	0	1	X	0	Guest follows	0
"	0	1	X	1	Guest enters	0
"	1	0	X	X	Member Enters	1
Wait for guest	0	0	X	0	Wait for Guest	0
"	0	0	X	1	Idle	0
"	0	1	X	0	Guest Follows	0
"	0	1	X	1	Guest Enters	0
"	1	0	X	X	Member Enters	1
Guest Follows	X	X	0	X	Guest follows	0
"	X	X	1	X	Idle	0
Guest Enter	X	X	0	X	Guest Enters	0
"	X	X	1	X	Wat For Token 2	0
Wait For Tk 2	X	X	0	X	Wat for Token 2	0
"	X	X	1	X	Idle	1
Illegal	1	1	X	X	Illegal	0

2) State can be encoded on 3 bits.
Which means Karnaugh Maps with 7 inputs ie 2^7 cells
Long and Tedious.
A good challenge though.

Problem 7 (cont.)

3) It's a bit messy to draw...



See the truth table for detailed transitions

In particular, the "closed barrier" output is not written. It's assumed to be there unless written otherwise

4) Output depends on state AND Inputs : it's a Mealy Machine.

END