# Sequential Circuits

---

[I've missed 3–4 lectures at the beginning here]

# Traffic Junction

Consider a junction where traffic is either East-West or North-South.

If there is more traffic East-West, we want our circuit to output 0, and if there is more traffic North-South we want our circuit to output 1.

## First Refinement

If we made this change immediately, then when the traffic is similar, the traffic lights will change very rapidly.

To solve this, say we have a delay unit, so that the output will only change at larger intervals. We have a black box that we connect a clock to, and the output will only be able to change on the clock's time grid. Between clock ticks, the output is kept constant.

## Second Refinement

Now, if one direction has a lot more traffic than another, traffic traveling in the other direction will never get through.

To fix this, we don't allow one direction to be open for too many clock ticks. Every third click (e.g.), we ignore the input, and invert the output instead.

We now have two states associated with an output of 1 or of 0: either this is a fresh 1 or 0, or it is the second 1 or 0 in a row.

## Four States

Let 'green' represent an output of 0, and 'red' represent an ouput of 1. The four states are:

- green, new (state 0)
    - input 0 -> state 1
    - input 1 -> state 2
- green, again (state 1)
    - input 0 -> state 1
    - input 1 -> state 1
- red, new (state 2)
    - input 0 -> state 0
    - input 1 -> state 3
- red, again (state 3)
    - input 0 -> state 0
    - input 1 -> state 0

### State Transition Table

```
z' = f(z, x)
```

where `z'` represents the next state, `z` represents the current state, and `x` represents the input.

| **z\x** | **0** | **1** |
|---|---|---|
| **0** | 1 | 2 |
| **1** | 2 | 2 |
| **2** | 0 | 3 |
| **3** | 0 | 0 |

In this graph, x values are along the header row, and z values are along the leftmost column.

# Sequence Spotter

We want the output of our circuit to be 1 if 011 was detected, and 0 otherwise.

I.e. if the current input is 1, the last input was 1, and the input before that was 0 – we output 1. Otherwise, we output 0.

## Moore Machine [I missed the notes on these]

We have a state for each partial sequence, including one state for having seen nothing. This gives us 4 total:

- Seen nothing
    - output: 0
    - input 0 -> Seen 0
    - input 1 -> Seen nothing
- Seen 0
    - output: 0
    - input 0 -> Seen 0
    - input 1 -> Seen 0, 1
- Seen 0, 1
    - output: 0
    - input 0 -> Seen 0
    - input 1 -> Seen 0, 1, 1
- Seen 0, 1, 1
    - output: 0
    - input 0 -> Seen 0
    - input 1 -> Seen nothing

## Mealy Machine

The output depends on the state and the input, but can change in between clock steps, instead of only being allowed to change at clock steps.

Whereas the Moore circuit will not respond to the input changing in between clock steps, the Mealy circuit will.

Note that state transitions still only happen at clock ticks. All we have changed is that the Mealy machine is still listening to the input at times other than state transitions.

For the Mealy machine, we only have three states:

- Seen 0
- [complete this]

# Synchronous Circuit

A synchronous circuit is one where all state transitions happen in one go, as all flip-flops are clocked by the same clock pulse.

# Traffic Light Circuit Design

We need two flip-flops for this. The input to each is a function of the single input X and of the outputs from each flip-flop.

The output Y is a function also. For a Moore machine, Y is a function only of the outputs from the flip-flops, and does not depend on the input X.

# Secondary State Assignment

This is the process of assigning different states to different flip-flop output combinations (e.g. `z = 0` state corresponds to `Q0 = 0`, `Q1 = 0`).

Different assignments lead to circuits of different complexity, and the rules for assigning this well are based on the concept of adjacent codes.

# Flip-flops [get diagrams]

Consider two inverters, where the output of each is connected to the input of the other.

This circuit can have two stable states – the output at one of the gates is 1 and at the other is 0, or the other way around.

## SR Latch

Consider two NOR gates, where the output of each is routed to one of the inputs of the other. Each gate has one input from outside the circuit, and one input fed

from the output of the other gate.

Remember if one input of a NOR gate is 0, then the output is NOT(the other input), and if one input is a 1, the output is always 0.

Here's the table of states, with the external inputs being `S` and `R`, and the outputs from the NOR gates being `Q` and `Q'`:

| Mode | S | R | Q | Q' |
|---|---|---|---|---|
| hold | 0 | 0 | 0 | 1 |
| hold | 0 | 0 | 1 | 0 |
| reset | 0 | 1 | 0 | 1 |
| set | 1 | 0 | 1 | 0 |
| forbidden | 1 | 1 | 0 | 0 |

Note that there are two stable states for the case where `S = 0` and `R = 0`. This is called the hold case because `Q` and `Q'` will hold the values they had in the previous state.

If `S = 0` and `R = 1`, then the R NOR gate will output 0 (`Q = 0`) and the S NOR gate will function as an inversion of the second input, and so will output 1 (`Q = 1`).

If `S = 1` and `R = 0`, the output will be the other way around.

The `S = 0`, `R = 0` state is referred to as forbidden because, if you change from the forbidden state to the hold state, it is uncertain which hold state output you will get. In practice, it will depend on the delays and the physical properties of the circuit.

## Extension

We now connect an AND gate to S and R, where one input of each is fed by a clock pulse. To the other input of one AND gate we connect our external input, and to the other input of the other we connect an inverter fed by our external input.

Now if the clock pulse is 1, the AND gates are transparent, and it is as if the input (and it's inverse) are connected straight to the SR latch we had before.

If the clock pulse is 0, the SR latch is in hold mode, as the output from both ANDs will be 0.

## Primary-Secondary Flip-Flop

(This is sometimes called the master-slave flip-flop)

This setup is analagous to locks on a canal – since the clock pulse fed to one is inverted to be fed to another, the flip-flops can't be transparent at the same time. One opens, then it closes and the other opens.

This is no longer a level-sensitive circuit – instead the output Q is synchronised with the falling edge of the clock pulse (when the clock goes from 1 to 0).

There is still a small problem with this circuit – on the rising edge (if we change CP from 0 to 1), the primary circuit will be enabled very slightly before the secondary circuit is disabled as the CP has to pass through an inverter, which introduces a slight delay.

This is a suitable building-block for Moore- and Mealy-machines because the edge-sensitive nature prevents the occurrence of undesired negative feedback loops.