Lecture 3

Gidon Rosalki

2025-04-06

1 Finite State Machines and Timing

1.1 Recap

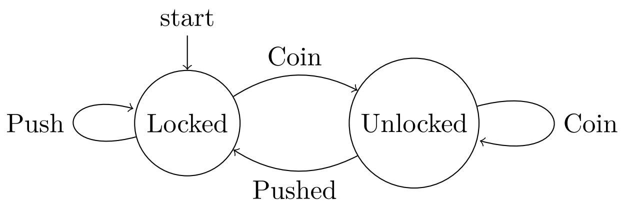
Combinational circuits are circuits where the outputs are determined at any given time by a function of the values and inputs. Sequential circuits are logic circuits that contain a feedback loop. This means that the outputs are determined by the inputs, and the value stored in memory, and the order over time of the inputs matters.
Combinational logic is boolean logic, where we have truth tables, and boolean expressions / boolean functions. In order to move from the table to the expression, we may use Sum of Products / Product of Sums, and the other direction through simple calculation. Additionally boolean expressions may be represented with logical chips, such as NAND, and so on.
We also discussed the gated D-latch, which has 2 inputs, D and G, and 2 outputs Q and Q'. $G = 1 \implies Q = D$, and $G = 0 \implies Q$ does not change. Using 2 D-Latches, we may create a rising edge D Flip Flop, which will **only** change when the clock is turning from 0 to 1. This eliminates the problem of the result value potentially changing many times over the course of the clock having the value 1. We may change the connection of the NOT gate to create a falling edge flip flop.
There are a couple more types of flip flop. The Toggle Flip Flop (T-FF) has the properties: $Q\left(t+1\right) = XOR\left(T, Q\left(t\right)\right)$, and the JK-FF is $Q\left(t+1\right) = JQ'\left(t\right) + kQ'\left(t\right)$.
A register is a set of flip flops which are used to store words. The set of FF will represent a word that can take any value and meaning (for example, a state in a state machine, or to indicate a positioning state (Jerusalem, TLV, Haifa, on the way, not here, etc). The set of FF can represent a binary word, for example 32 FF that hold a 32 bit binary value. All clock inputs are connected to a single clock, and each in line goes to one FF, with each out line coming from one FF. Using all this makes it easy to make a counter, by looping the output into a chip that can add 2 binary numbers, which has one input being the output of the register, and the other as just 1.

1.2 Finite State Machines (FSM)

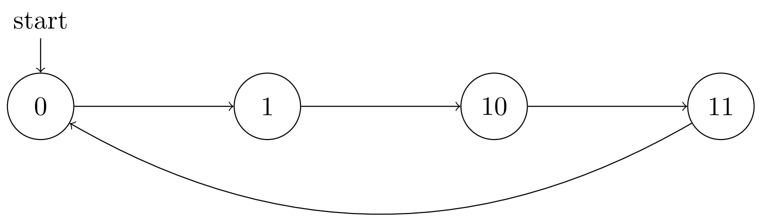
An FSM is a computation model, consisting of:

- A finite set of states - Q
- An initial state - q_0
- A finite set of inputs - Σ
- A finite set of outputs - $F \subseteq Q$
- Transitions: $Q \times \Sigma \to Q$
- Output functions: functions that define the output's values

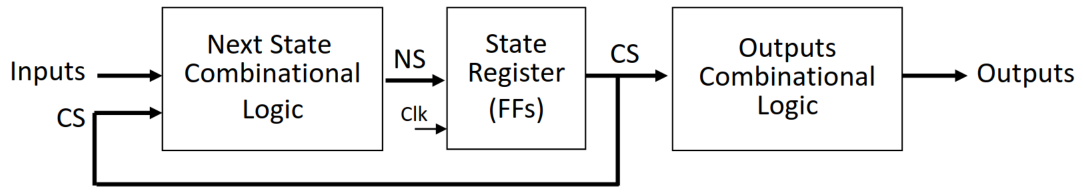
An example is a rotating gate. It can either be locked or unlocked, depending on putting in a coin, and if it got pushed.



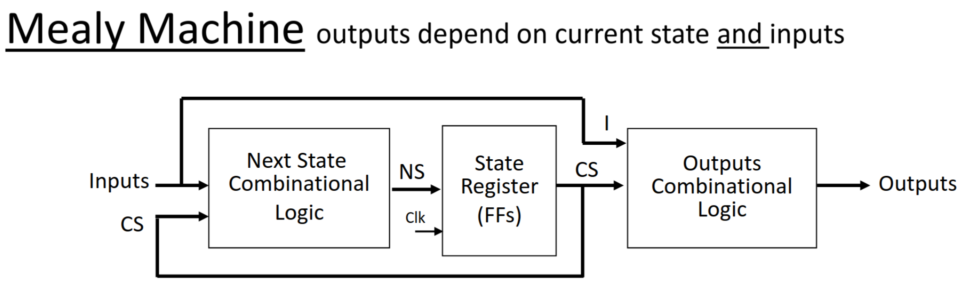
We may represent a binary counter as an FSM



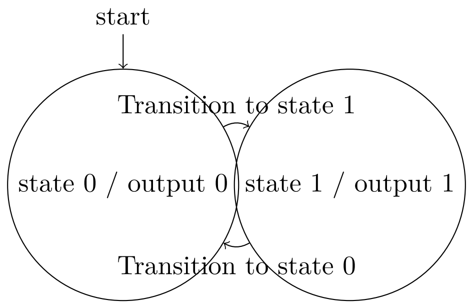
A **Moore Machine** is a machine whose outputs only depend on the current state.



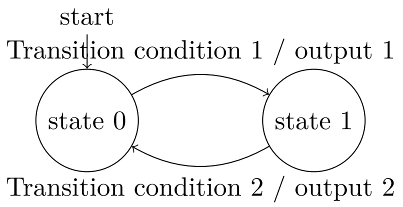
A **Mealy machine**'s outputs depend on the current state *and* the inputs.


Mealy Machine outputs depend on current state and inputs
I
Inputs
CS
Next State Combinational Logic
NS
Clk
State Register (FFs)
CS
Outputs Combinational Logic
Outputs

To make these into diagrams, we use the following conventions: For a Moore machine, since the output state depends solely on the current state, we show the output state inside the states of the FSM. We will separate the state name, and the output, with a slash.



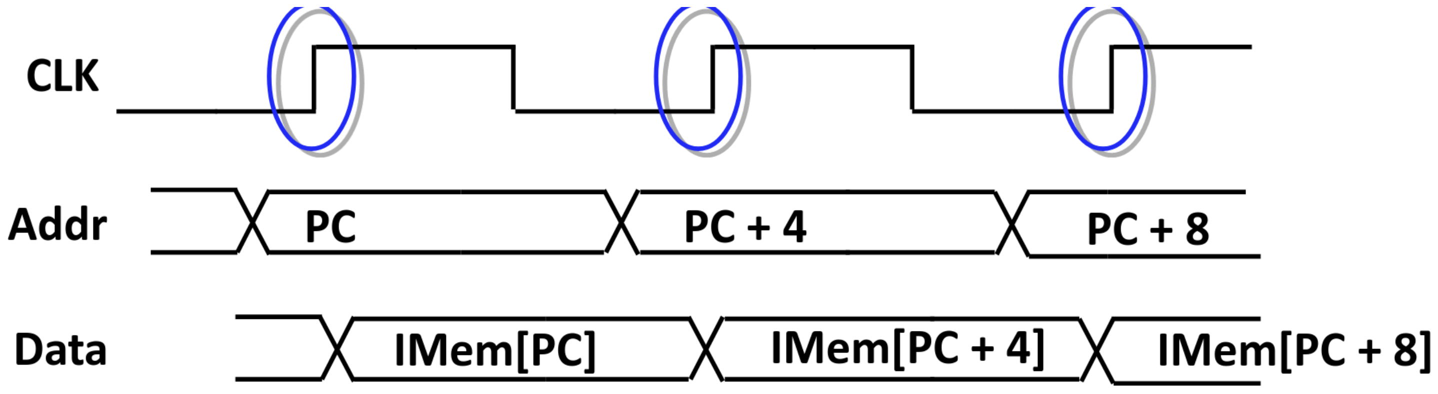
For a Mealy machine, the output depends on both the current state, and the inputs, so it can generate a different output for the same state. The outputs are shown as the transitions, since they depend on the inputs.


start
Transition condition 1 / output 1
state 0
state 1
Transition condition 2 / output 2

Moore and Mealy machines can be functionally equivalent, though Mealy FSMs are a richer description, and may require a smaller number of states. Mealy FSM outputs change as soon as the inputs change, and can thus respond up to 1 clock cycle earlier than an equivalent Moore FSM. Then again, Moore FSMs have no combinational path between inputs and outputs, so timing is easier, and there is less risk of an output glitch. Moore FSMs are more generally used in computer design.

1.2.1 Program counter

The program counter is needed to fetch the next instruction. This requires an FSM that generates the following timing diagram:



We move the PC by 4 every time, since each program is 4 bytes long.

1.3 Timing and frequency

The clock speed (frequency) is one of the main contributors to the performance. In electrical circuits, values are always "generated". The changes are however not immediate, since it takes time to charge / discharge capacitance, which causes delays in circuit response. The output changes due to input changes, with some delay. We may state that electrical circuits always have delays. Data sheets are used to define the voltages for the logic levels. The input voltage lists the minimum/maximum voltage required to be considered as 1 / 0. The out voltage lists the minimum / maximum that is considered a valid 1 / 0.
The propagation delay is the time period over which the changes in inputs propagate to the output. t_{PLH} is the time taken for a propagation from low to high. Similarly t_{PHL} is the time taken for propagation from high to low. In many cases $t_{PLH} = T_{PHL}$, so it is just referred to as t_{pd}. The propagation delay is measured as the 50% level, which is to say the time to reach 50% of the final value. t_{pd} has typical, maximum, and minimum, values, for propagation from input to output. t_{pd-max} is the time after which the change is guaranteed, and t_{pd-min} is the time for which it is guaranteed that the output will not change. All these values can be defined for both gates, or a combinational circuit (many gates). They may differ for different inputs.
Flip flop timing is different. Here we time the propagation of data to Q . This is the time from the relevant clock edge, to the change in Q. This is referred to as t_v, or t_{PCQ}. t_v - output valid time (after clock edge). There are similarly *min* and *max* variants as described above. The requirements on the input are that the time input should be valid relative to the clock to guarantee $Q(t+1) = Q(t)$. t_S is the setup time: before rising edge. t_H is the hold time, after the rising edge.
To find the maximum frequency, we first find t_{cyc}, which is the time from one rising edge of the clock to the next.

$$t_{cyc} \geq t_{v-max} + t_{pd-max} + t_{setup} = t_{path}$$

So then the maximum frequency $F \geq \frac{1}{t_{cyc}}$ To find t_h we find the t_{path_min} for all paths leading to inputs (min time change to reach an FF input) :

$$t_{path\_min} = t_{v\_min(FF1)} + t_{pd\_min\ (comb\ logic)}$$

It must meet the requirement $\min\limits_{path}\{t_{path\_min}\} \geq t_h$. It does not depend on the frequency, and the frequency doesn't depend on t_h, or minimum timing.