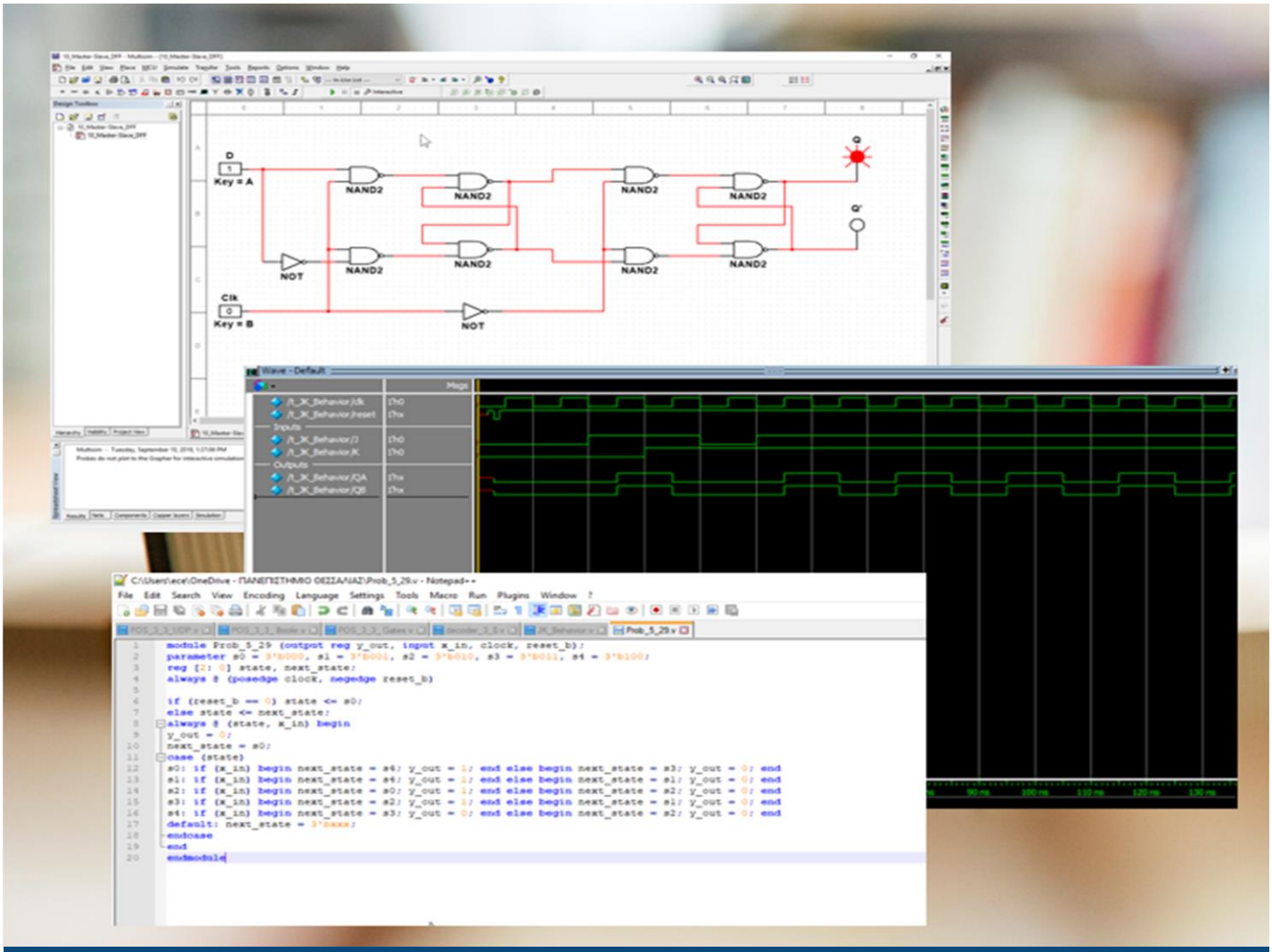
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
χρόνια δημιουργίας
1988 — 2018

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ



ECE119 Ψηφιακή Σχεδίαση

Εργαστηριακές ασκήσεις, Multisim - Verilog

Lab 10: Sequential Circuits - FSM

Καραμπερόπουλος Δημήτρης

Σεπτέμβριος 2023

Lab 10

Required Tools and Technology

Software: NI Multisim 14.0 or newer

- ✓ **Install Multisim:** http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ **View Help:** http://www.ni.com/multisim/technical-resources/

MultisimLive

- ✓ https://www.multisim.com/

Lab 10: Sequential Circuits - FSM

10.1 Theory and Background

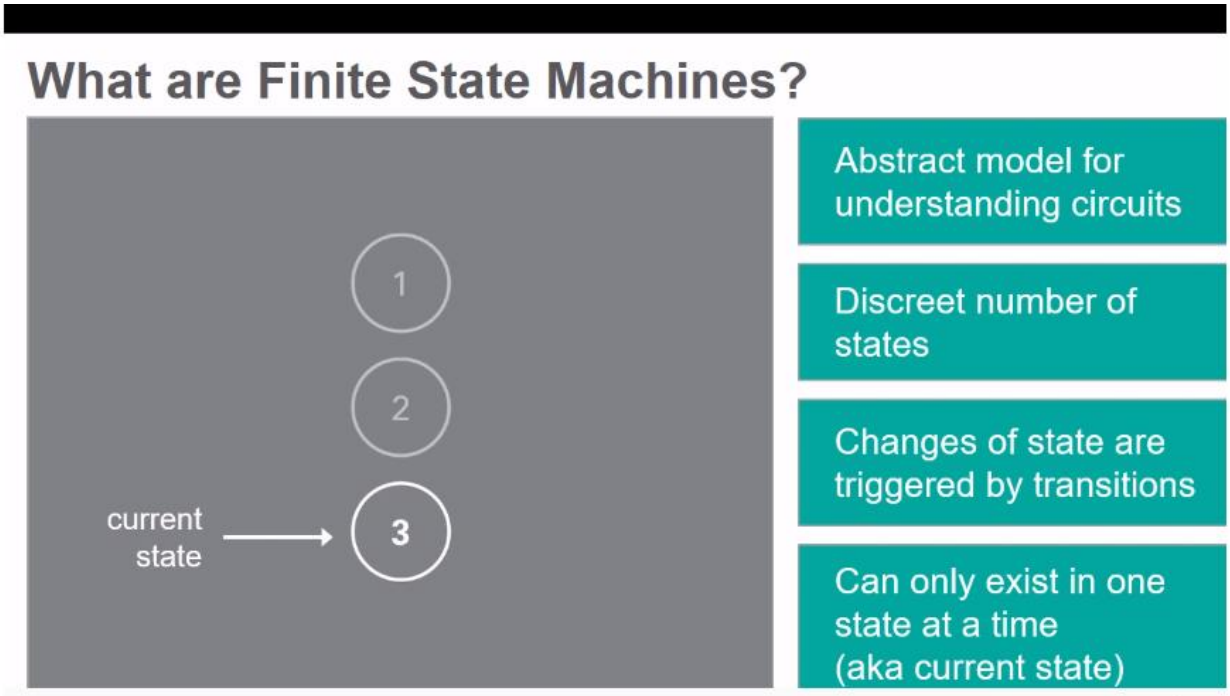


Figure 10-1 Video Screenshot. View the video here: https://youtu.be/I0HBrcE_HOI

Video Summary

- A Finite State Machine is an abstract model for understanding circuits
- There are a discreet number of states that exist in all finite state machines
- Transitions trigger changes of state
- A machine can only exist in one state at any time (the current state)

Finite State Machines

A *finite state machine (FSM)* is an abstract model used to describe the behavior of an object.

- Each circle represents a *state* in the system, while each arrow represents a *transition* to another state.
- Since there are a discrete number of circles, there is only a finite number of states that are possible.
- The machine can only be in one state at any given time. This is known as the *current state***.**
- The arrow labels are *events* that trigger the state transition.
- Finite state machines are used in:
    - Vending machines: dispensing a product when the appropriate amount of money is deposited into it.
    - Elevators: dropping people off to the top floors before coming back down.
    - Traffic lights: changing the color sequence at scheduled times.

The example below shows a simple FSM for a system that unlocks when a correct button sequence is pressed. Consider if the example represented a locked door:

- State 0 (**S0**) represents a state when the lock is enabled
- **S0** moves to **S1** after button '**2**' is pressed.
- **S1** has two options for the next state – **S2** if '**1**' is pressed, otherwise **S0**.
- **S3** represents a state where the lock is disabled and enables once the door is opened.

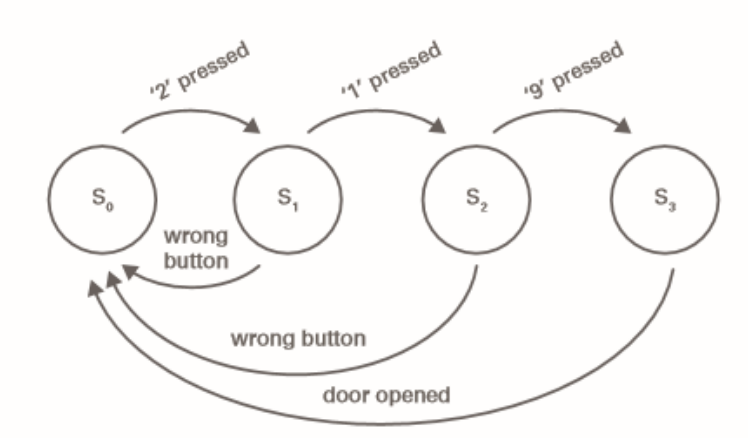


Figure 10-2 Simple Finite State Machine

10.2 Exercise: Finite State Machine

1. Ο ΑΜ του κάθε φοιτητή είναι ένας 5ψήφιος αριθμός. Το πρώτο του ψηφίο είναι πάντα το “0”. Έστω ότι κάποιος έχει το ΑΜ = **03456**.

Θέλουμε να κατασκευάσουμε μία μηχανή πεπερασμένων καταστάσεων η οποία θα έχει **μία είσοδο “in” (1 bit)** και **μία έξοδο “am” (4 bits)**.

Αρχικά στην έξοδο θα έχουμε τιμή “0000”.

Όταν η είσοδος γίνεται “1” στην έξοδο θα εμφανίζονται με τη σειρά οι δυαδικές τιμές των ψηφίων του ΑΜ κυκλικά. Αυτό θα συμβαίνει σε κάθε θετική ακμή του ρολογιού. Δηλαδή για το παραπάνω ΑΜ θα εμφανίζονται:

$$0 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 0 \rightarrow 3 \rightarrow 4 \ldots$$

Μόλις η είσοδος γίνεται “0”, στην επόμενη θετική ακμή του ρολογιού, η έξοδος θα γίνεται πάλι “0000”.

Εάν η είσοδος ξαναγίνει “1” στην έξοδο θα ξαναβλέπουμε πάλι τα ψηφία του ΑΜ κυκλικά.

Θα πρέπει να υπάρχει και σήμα **reset** το οποίο θα είναι ενεργό στο “0”. Δηλαδή εάν και για όσο το reset είναι “0” στην έξοδο ανεξαρτήτως της εισόδου θα πρέπει να έχουμε “0000”. Όταν το reset είναι στο “1” η FSM θα ενεργοποιείται και θα υλοποιείται η παραπάνω διαδικασία απεικόνισης των ψηφίων του ΑΜ.

Εάν η είσοδος είναι “1” και το reset γίνει “0” θα πρέπει κατευθείαν η έξοδος να γίνεται “0000” ασύγχρονα, δηλαδή εκείνη την στιγμή, χωρίς να περιμένει την θετική ακμή του ρολογιού.

Θα πρέπει να έχει και **μία επιπλέον έξοδο “y” (1 bit)** η οποία για κάθε ψηφίο που εμφανίζεται στην έξοδο θα ελέγχει εάν είναι άρτιο ή περιττό. Εάν είναι άρτιο το “y” θα γίνεται “1”, ενώ εάν είναι περιττό το “y” θα γίνεται “0”. Όταν η τιμή εξόδου είναι “0000” το “y” θα παραμένει στο “0”.

Δημιουργήστε ένα αρχείο **“am.v”**

- Verilog:	Όνομα αρχείου “**am.v**”. Προσθήκη στο zip file με όνομα ”Lab10_ονοματεπώνυμο_AM.zip”

Υλοποιήστε την παραπάνω FSM σε ένα module με το όνομα “**am**”.

2. Δημιουργήστε και ένα module "**tb_am**" το οποίο θα είναι ένα test bench για το παραπάνω module. Γράψτε το στο ίδιο Verilog file.

Θα πρέπει να κάνετε τα εξής:

- Στα 0 nsec: in = 0
- clock = 0
- reset = 1
- Στα 5 nsec: reset = 0
- Στα 10 nsec: reset = 1
- Στα 22 nsec: in = 1
- Στα 82 nsec: in = 0
- Στα 97 nsec: in = 1
- Στα 118 nsec: reset = 0
- Στα 128 nsec: reset = 1
- Στα 148 nsec: finish

- Δημιουργείστε ένα clock, με περίοδο **10 nsec,** με μία δομή always.

Ελέγξτε τις κυματομορφές για την σωστές τιμές εξόδου.

Στις κυματομορφές πρέπει πρώτη (επάνω) να βρίσκεται το clock. Έπειτα να βρίσκονται κατά σειρά: reset, in, am[3:0], y, currentState[2:0], nextState[2:0]

Κάντε ένα screenshot των κυματομορφών σας.
(μπορεί να είναι printscreen του υπολογιστή ή μία φωτογραφία με το κινητό σας)

Picture

- Picture: Όνομα αρχείου "**am**".
Προσθήκη στο zip file με όνομα "Lab10_ονοματεπώνυμο_AM.zip"

Παράδειγμα:
Εάν π.χ. το ΑΜ είναι 03456 τότε οι κυματομορφές θα πρέπει να είναι ως εξής: