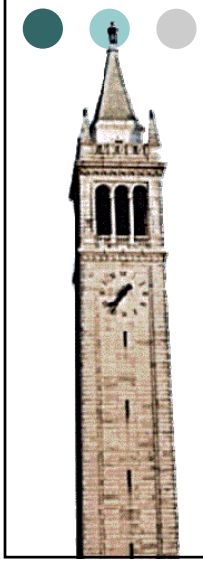


Fundamental Algorithms for System Modeling, Analysis, and Optimization

Jaijeet Roychowdhury, Stavros Tripakis
UC Berkeley
EECS 144/244
Fall 2014

Week 2 - Lecture 1: Discrete Systems

Quiz

- Express the following in your favorite mathematical formalism:
  - You can fool some people sometimes
  - You can fool some of the people all of the time
  - You can fool some people sometimes but you can't fool all the people all the time [Bob Marley]
  - You can fool some of the people all of the time, and all of the people some of the time, but you can not fool all of the people all of the time [Abraham Lincoln]

DISCRETE **SYSTEMS**

3

What is a “system” ?

4

System: definition

- Something that has:
  - State
  - Dynamics: rules that govern the evolution of the state in time

5

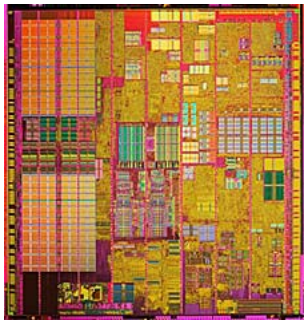
System: definition

- Something that has:
  - **State**
  - Dynamics: rules that govern the evolution of the state in **time**
- It may also have:
  - Inputs: they influence how system evolves
  - Outputs: this is what we observe

6

Example: digital circuits

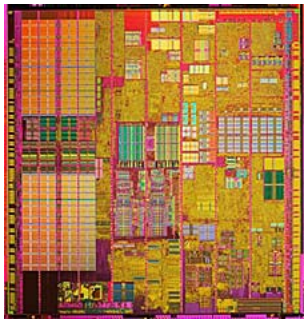
- Digital circuit:
  - State: ???
  - Dynamics: ???



7

Example: digital circuits

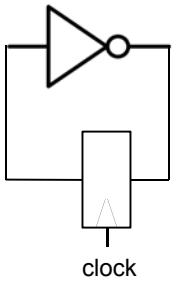
- Digital circuit:
  - State: value of every register, memory element
  - Dynamics:
    - Defined by the combinational part (logical gates)
    - Time: discrete, or “logical” (ticks of the clock)



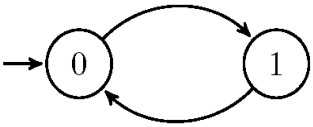
8

Example: digital circuits

- **Systems** vs. **models**



System
(the "real" circuit)



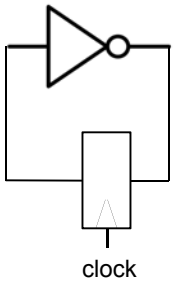
Model
(a finite-state machine)

To reason about systems (analyze, make predictions, prove things, ...), we need mathematical models

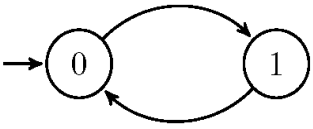
9

Example: digital circuits

- **Systems** vs. **models**



System
(the "real" circuit)




```
node Circuit ()
returns (Output: bool);
let
  Output = false -> not pre Output;
tel
```

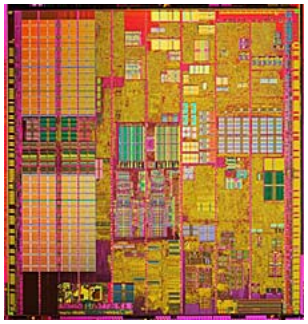
Different models
(finite-state machines)

Different representations (languages, syntaxes) of the same underlying mathematical model

10

Example: digital circuits

- Digital circuit as a system:
  - State: value of every register, memory element
  - Dynamics:
    - Defined by the combinational part (logical gates)
    - Time: discrete, or "logical" (ticks of the clock)
  - **Or**:
  - State: all currents and voltages at all transistors at a given time t
  - Dynamics: physics of electronic circuits (differential algebraic equations)



Different levels of **abstraction**

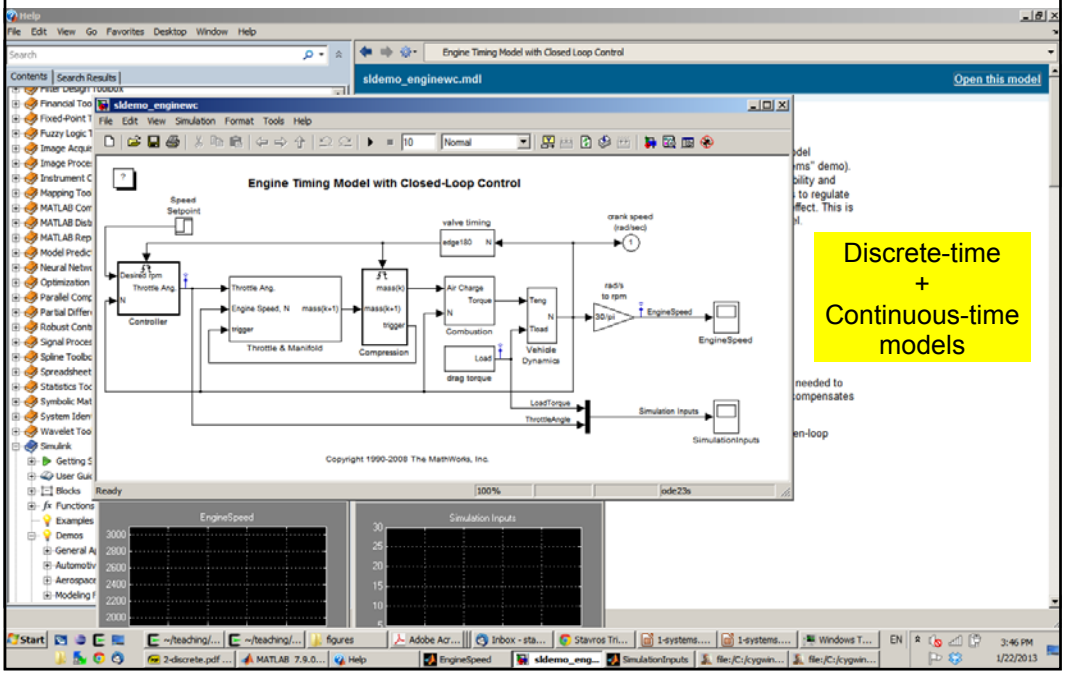
11

Multi-paradigm modeling

- Different representations (languages, syntaxes) of the same underlying formalism.
- Different modeling formalisms often needed to describe the same system, e.g., at different levels of abstraction.
- Different modeling formalisms often needed to describe different parts of the system (subsystems).

12

Example: plant + controller



Classes of systems/models considered in this course

- Discrete: state machines, …
- Continuous: differential equations, …
- Timed: discrete-event, timed automata, …
- Dataflow: process networks, SDF, …
- Probabilistic: Markov chains, …

Back to:

DISCRETE SYSTEMS

15

Discrete systems

Automata, state machines, transition systems, …

- States
- Transitions: discrete moves from one state to the next

- "logical" time = order of transitions
- As opposed to quantitative, "real-time" models such as differential equations or timed automata (we will see those later).

16

Finite State Machines

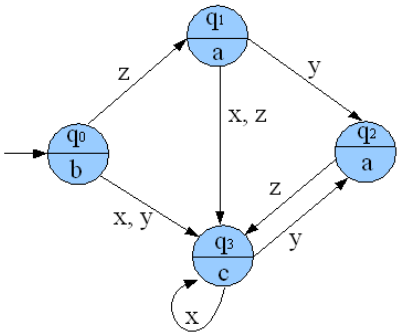
Moore machines

States: {q0, q1, q2, q3}

Initial state: q0

Input symbols: {x,y,z}

Output symbols: {a,b,c}



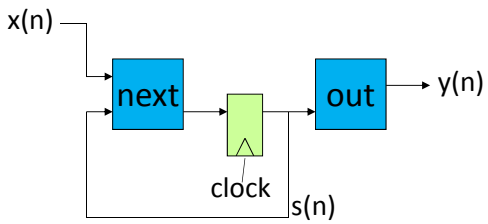
Output function:

$$out : States \rightarrow Outputs$$

Transition function:

$$next: States \times Inputs \rightarrow States$$

Moore machine: a circuit view



19

Mealy machines

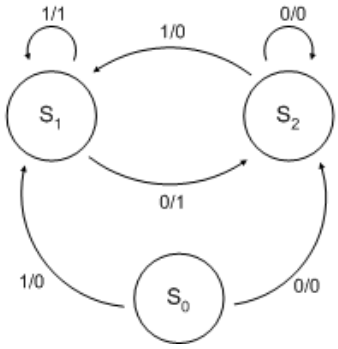
States: {S0, S1, S2}

Initial state: S0

Input symbols: {0,1}

Output symbols: {0,1}

Output function:



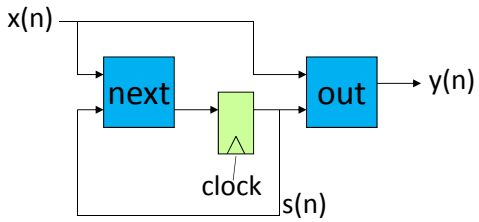
$$out : States \times Inputs \rightarrow Outputs$$

Transition function:

$$next: States \times Inputs \rightarrow States$$

20

Mealy machine: a circuit view



21

Finite State Machines – Formal Definition

An FSM is a tuple

$$(I, O, S, s_0, \delta, \lambda)$$

- I: set of inputs
- O: set of outputs
- S: set of states
- $s_0 \in S$: initial state
- $\delta : S \times I \to S$: transition function
- λ: output function
  - If the FSM is of type **Moore**:

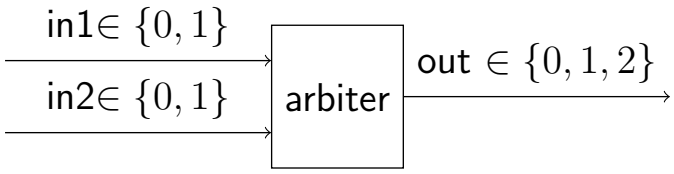
$$\lambda : S \to O$$

  - If the FSM is of type **Mealy**:

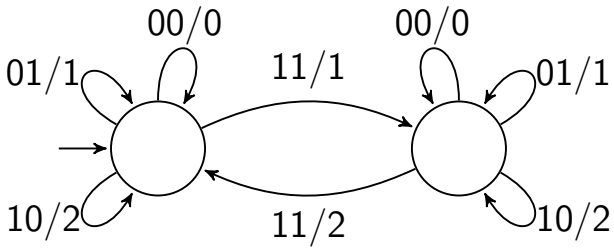
$$\lambda : S \times I \to O$$

Example: Mealy Machine

structure:

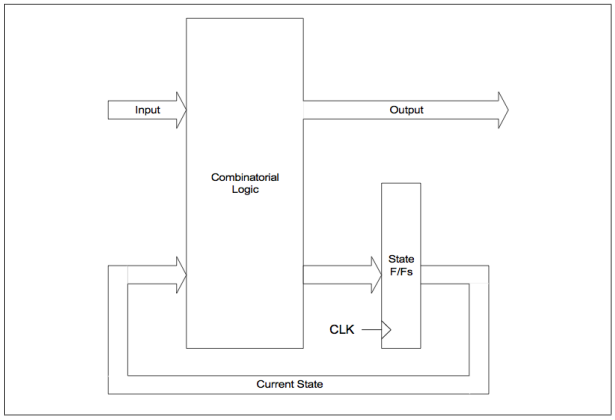


behavior:



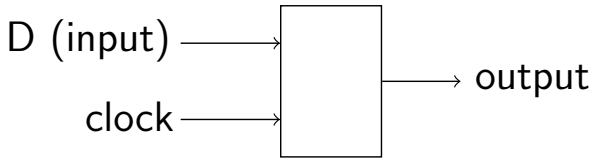
CIRCUITS

Synchronous Circuits – Generic structural view:



- Combinational logic part: a network of logical gates (AND, OR, NOT, XOR, ...).
- Memory/state of the circuit: some type of digital memory element (e.g., D-type flip-flop).
- Synchronous: clock arriving conceptually synchronously (simultaneously) at all flip-flops.
- Circuit: a network of connected gates and flip-flops ("netlist").

Memory element: D flip-flop



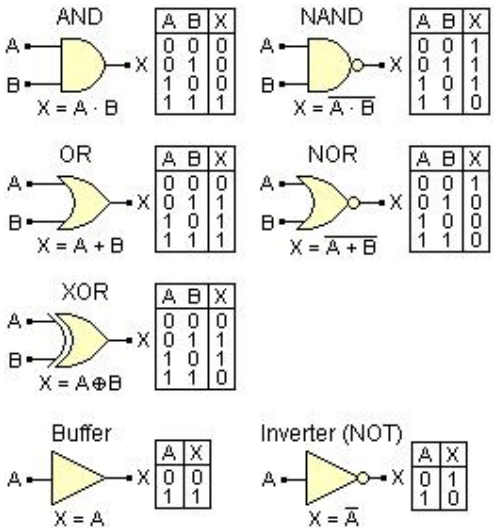
Behavior (simplified[1]):

- Clock input defines a set of times $t_1, t_2, t_3, ...$ (e.g., up-edges of a periodic pulse).
- The value of output remains constant during the interval $[t_k, t_{k+1})$ and equal to the value of the input D at t_k.
- "Door-opening" metaphor.
- Memory elements often have more inputs (e.g., resets to initialize state).

[1] More accurate description of timing behavior in timing analysis lecture.

Is the D flip-flop a state machine?

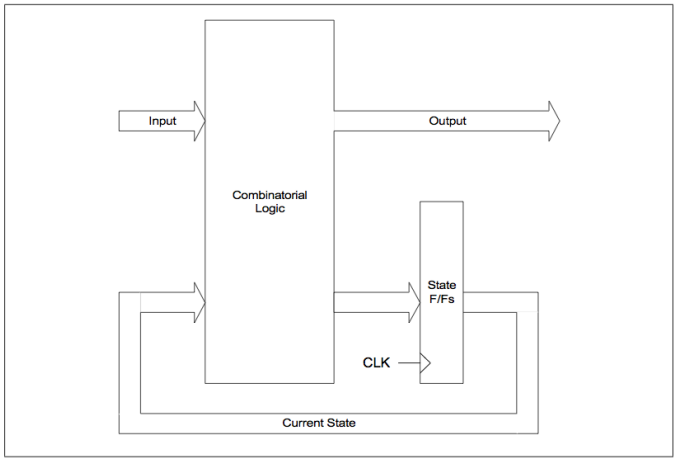
Combinational logic gates



Are logic gates state machines?

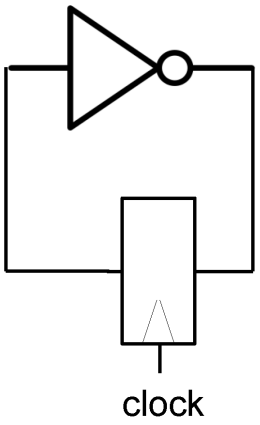
Digital Circuits: Networks of Flip-Flops and Logic Gates

For now, we consider **acyclic** circuits: they **can** have feedback, but any feedback loops are "broken" by flip-flops:



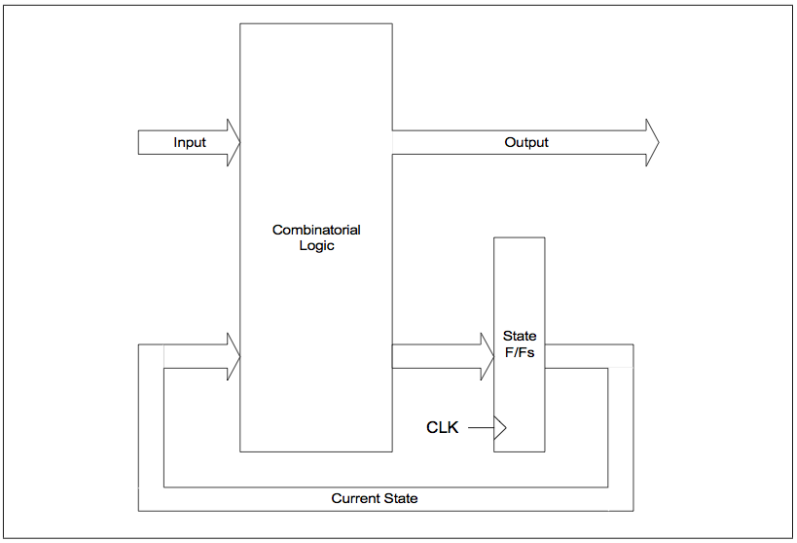
Are the dynamics of such circuits well-defined? How?

From Circuits to State Machines



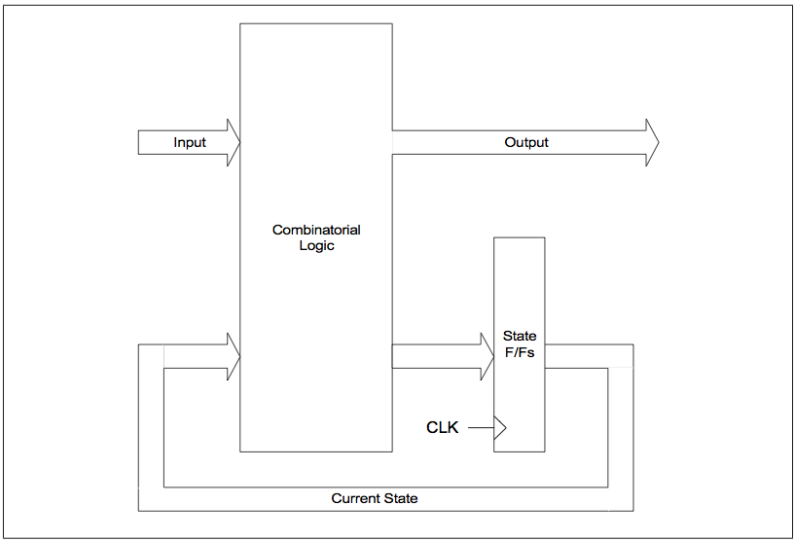
Is this a state machine?

From Circuits to State Machines

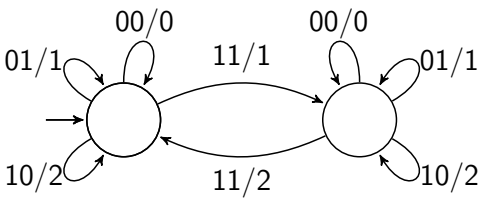


Is this a state machine? Is it a Mealy or Moore machine?
How are $(I, O, S, s_0, \delta, \lambda)$ defined?
What would a Moore Machine look like?

State Machines and Synchronous Circuits

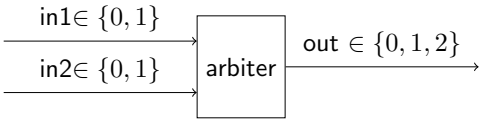


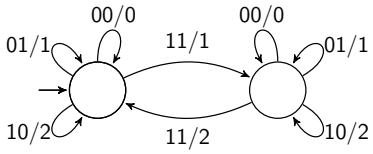
Is this a good drawing?



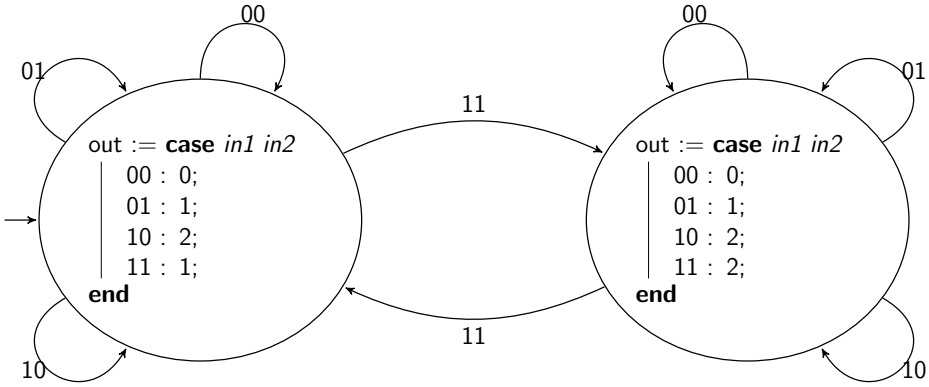
Drawing Mealy Machines Correctly

Traditional drawing mixes transition and output functions, although these are independent (this matters in the case of circuits, for instance, where outputs might change multiple times before stabilizing – c.f. discussion on circuits that follows):



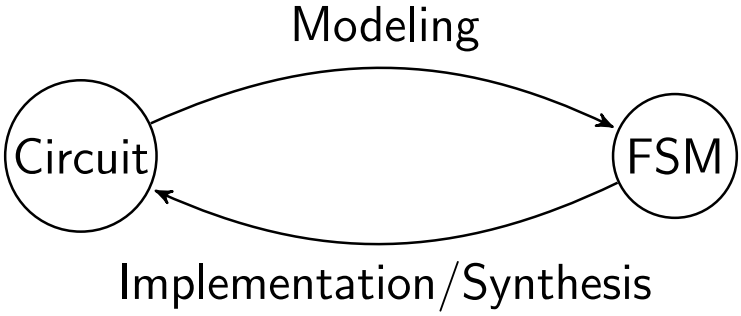


Better drawing:



Modeling and Implementation/Synthesis

What we have done / what we will do next:



From FSMs to Circuits

"Brute-force" implementation:

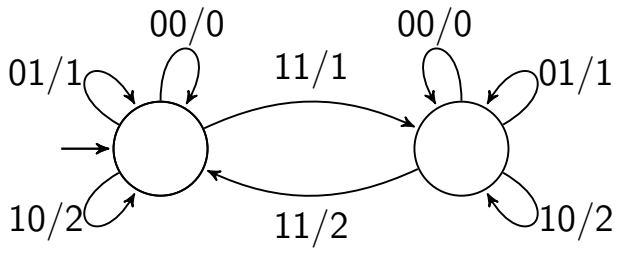
- $\log n$ flip-flops, where $n = |S|$ = number of states of the FSM.
- $\log k$ input wires, where $k = |I|$ = number of input symbols.
- $\log m$ output wires, where $m = |O|$ = number of output symbols.
- Multiplexers to implement transition and output functions.

More efficient implementations: the **logic synthesis** problem.
Several subproblems:

- State encoding (or *state assignment*)
- Logic minimization
- ...

From FSMs to Circuits

Let's implement this FSM (on whiteboard):



From FSMs to Circuits

Several combinatorial optimization problems.
E.g., *state assignment* (state encoding): how to encode the states of a given FSM as boolean vectors. Which of the many possible encodings to choose?
Example (taken from [Kohavi, 1978]):

Table 12.1 Machine M_1

	NS		z	
PS	$x=0$	$x=1$	$x=0$	$x=1$
A	A	D	0	1
B	A	C	0	0
C	C	B	0	0
D	C	A	0	1

		Y_1Y_2		z	
	y_1y_2	$x=0$	$x=1$	$x=0$	$x=1$
A	00	00	10	0	1
B	01	00	11	0	0
C	11	11	01	0	0
D	10	11	00	0	1

(a) Assignment α

		Y_1Y_2		z	
	y_1y_2	$x=0$	$x=1$	$x=0$	$x=1$
A	00	00	11	0	1
B	01	00	10	0	0
C	10	10	01	0	0
D	11	10	00	0	1

(b) Assignment β

From FSMs to Circuits

The two state encodings result in two very different circuits:

Fig. 12.1 First realization of M_1.

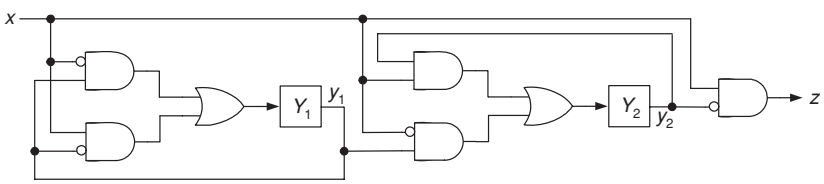
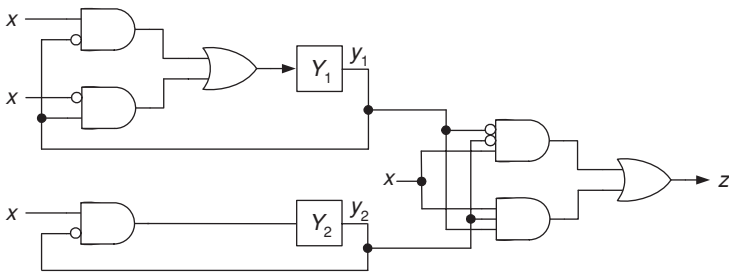


Fig. 12.2 Second realization of M_1.



Figures taken from [Kohavi, 1978].

An elegant notation for (not necessarily finite) state machines: Lustre

A program in the synchronous language Lustre [Halbwachs et al., 1991]:

```
node Edge (X : bool) returns (E : bool);
let
  E = false -> X and not pre X ;
tel
```

Can you guess its meaning?

$$\begin{aligned} E_0 &= false \\ E_{k+1} &= X_{k+1} \wedge \neg X_k \end{aligned}$$

Quiz: write a counter in Lustre.

Bibliography


Hachtel, G. D. and Somenzi, F. (1996).
Logic Synthesis and Verification Algorithms.
Kluwer.


Halbwachs, N., Caspi, P., Raymond, P., and Pilaud, D. (1991).
The synchronous dataflow programming language Lustre.
Proceedings of the IEEE, 79(9):1305–1320.


Hopcroft, J. E. and Ullman, J. D. (1990).
Introduction To Automata Theory, Languages, And Computation.
Addison-Wesley.


Kohavi, Z. (1978).
Switching and finite automata theory, 2nd ed.
McGraw-Hill.