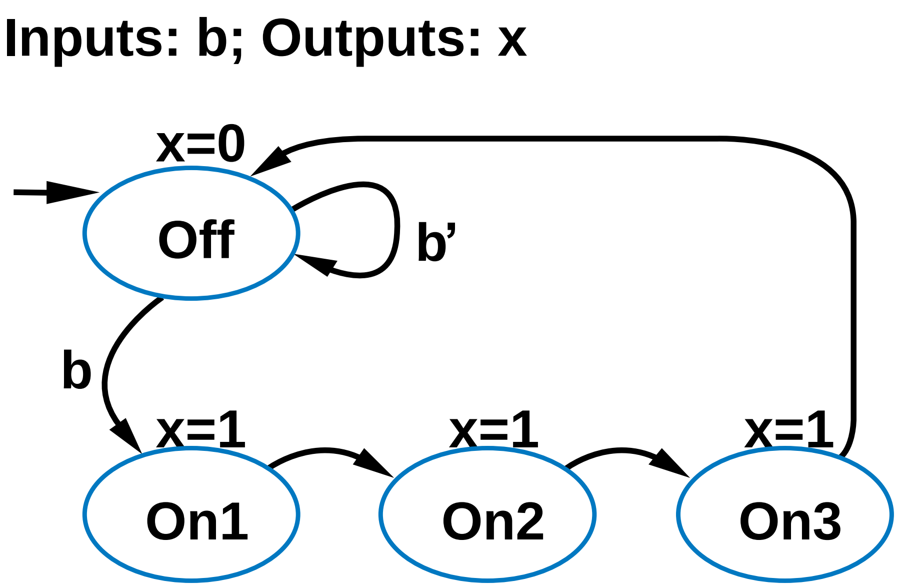
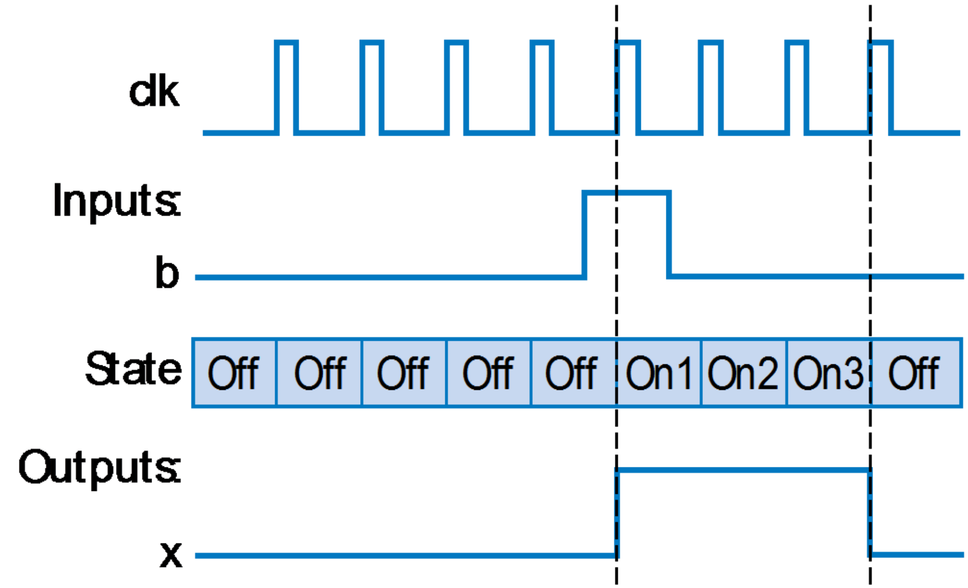
B38DB: Digital Design and Programming
Sequential Logic Design – Controller Design

Mustafa Suphi Erden

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering
Room: EM 2.01
Phone: 0131-4514159
E-mail: m.s.erden@hw.ac.uk

Reminder: FSM for Three-Cycles High Laser Timer

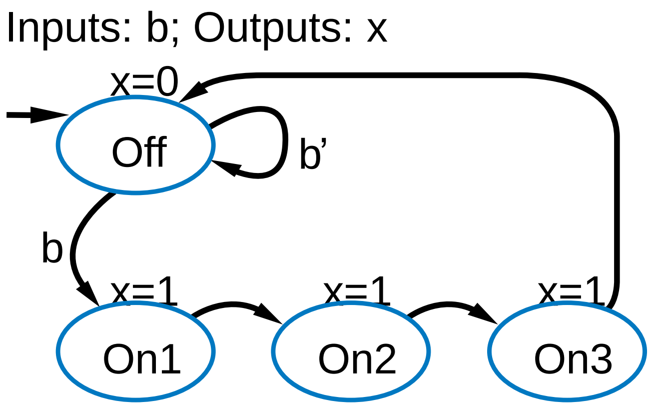


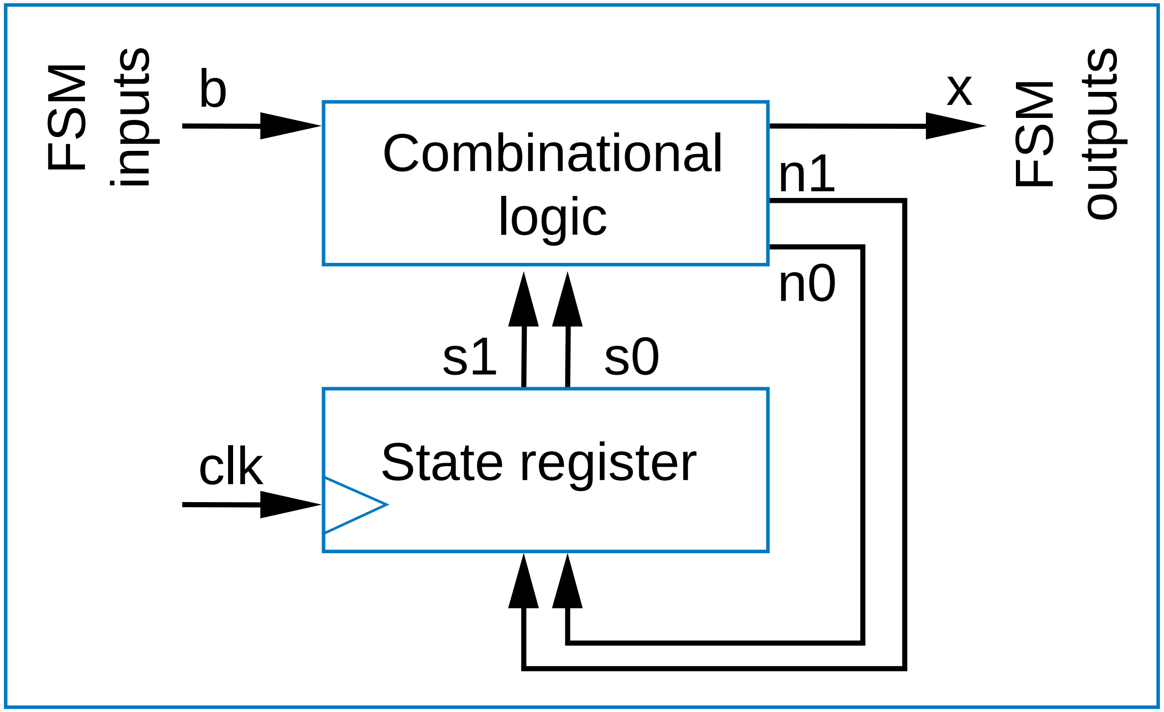


How to Implement this physically?

Standard Controller Architecture

- Convert an FSM to a sequential circuit, known as a ***controller***.
  - Use **standard architecture** (a straightforward design process)
    - State register → <u>stores the present FSM state</u>
    - Combinational logic → <u>computes the outputs & the next state</u> based on the inputs & the current state





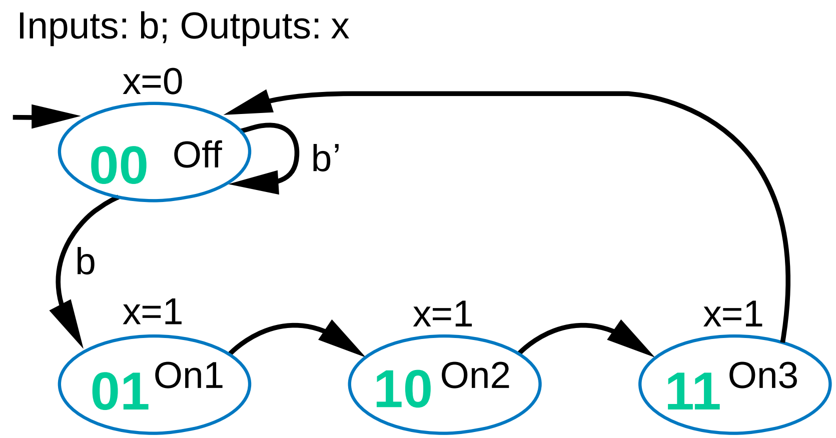
Controller Design

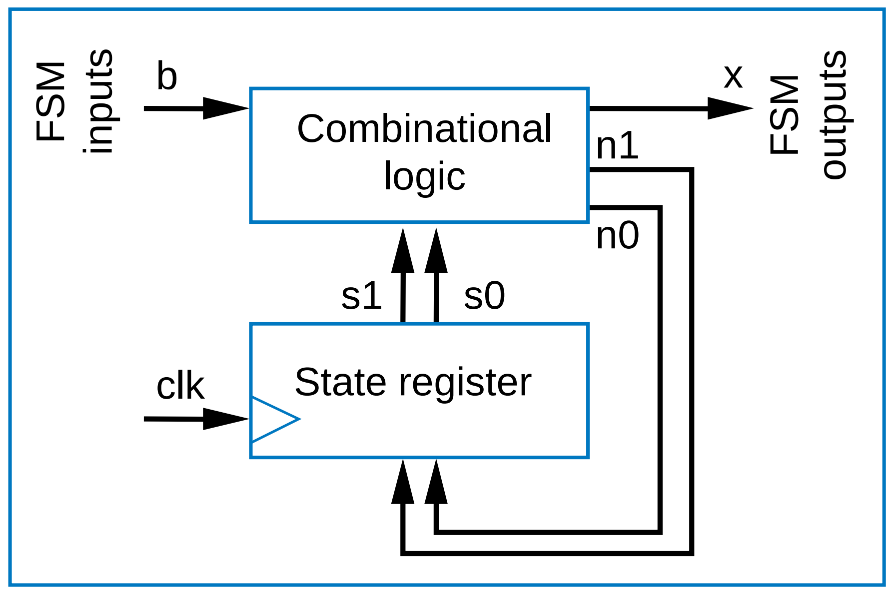
- Five steps controller design process:

	Step	Description
Step 1	*Capture the FSM*	Create an FSM that describes the desired behavior of the controller.
Step 2	*Create the architecture*	Create the standard architecture by using a state register of appropriate width, and combinational logic with inputs being the state register bits and the FSM inputs and outputs being the next state bits and the FSM outputs.
Step 3	*Encode the states*	Assign a unique binary number to each state. Each binary number representing a state is known as an ***encoding.*** Any encoding will do as long as each state has a unique encoding.
Step 4	*Create the state table*	Create a truth table for the combinational logic such that the logic will generate the correct FSM outputs and next state signals. Ordering the inputs with state bits first makes this truth table describe the state behavior, so the table is a state table.
Step 5	*Implement the combinational logic*	Implement the combinational logic using any method.

Controller Design: Laser Timer Example (1/4)

- Step 1: Capture the FSM
  - Already done
- Step 2: Create architecture
  - 2-bit state register (for 4 states)
  - Input b, output x
  - Next state signals n1, n0
- Step 3: Encode the states
  - Any encoding with each state having a unique representation

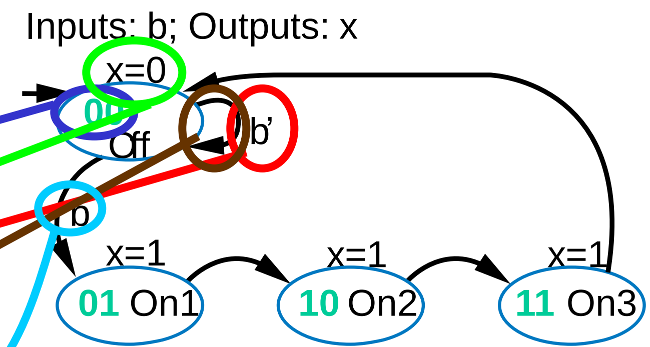


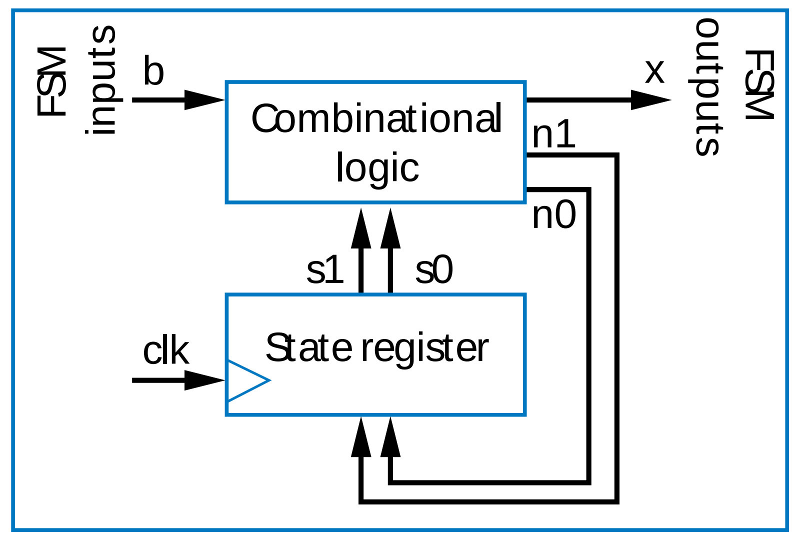


Controller Design: Laser Timer Example (2/4)

- Step 4: Create state table

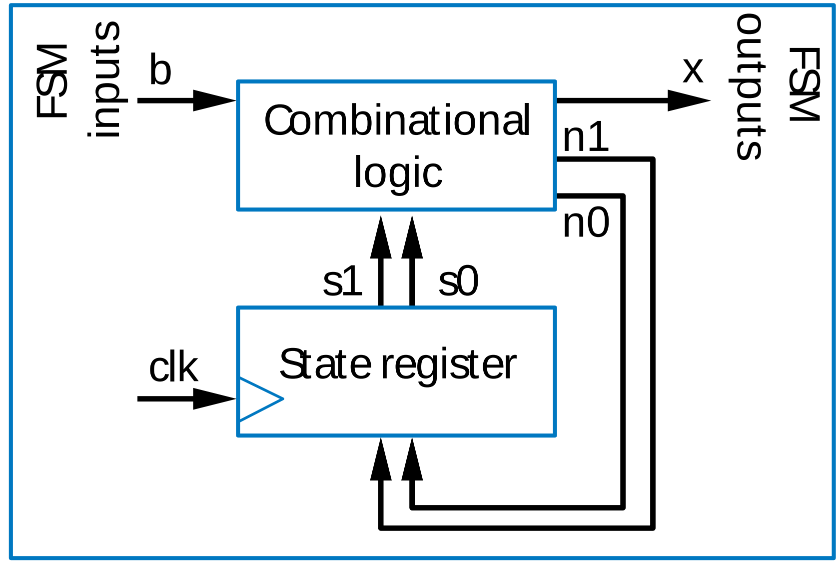
	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0





Controller Design: Laser Timer Example (3/4)

- Step 5: Implement combinational logic



	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

x = s1 + s0 (note from the table that x=1 if s1=1 or s0=1)

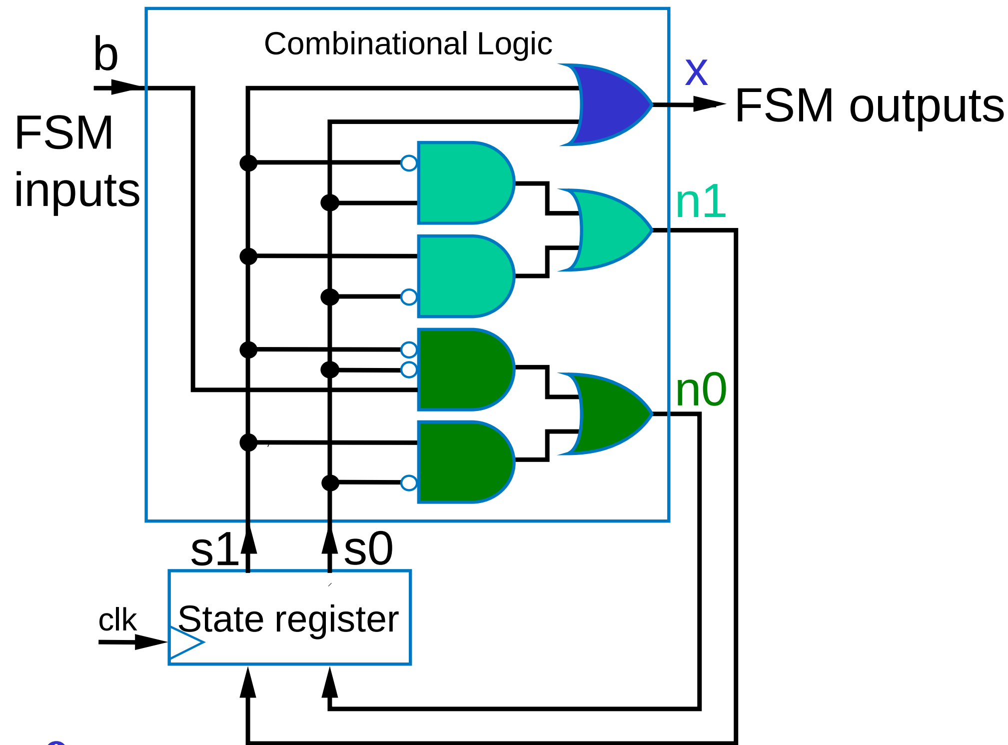
n1 = s1’s0b’ + s1’s0b + s1s0’b’ + s1s0’b
n1 = s1’s0 + s1s0’

n0 = s1’s0’b + s1s0’b’ + s1s0’b
n0 = s1’s0’b + s1s0’

Controller Design: Laser Timer Example (4/4)

- Step 5: Implement combinational logic (cont'd)

	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0



x = s1 + s0

n1 = s1's0 + s1s0'

n0 = s1's0'b + s1s0'

Understanding the Controller's Behavior