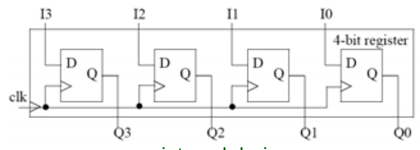
ECE 274 - Digital Logic
Lecture 6

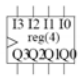
- Lecture 6
  - Basic Register Design
  - Controllers
    - State diagrams
    - Finite State Machines (FSMs)
  - Sequential Logic Design Process

Digital Design
Sequential Logic Design – Controllers: Registers

- Basic 4-bit register:
  - inputs: n-data bits, clock
  - outputs: n-data bits



internal design



block symbol.

Digital Design
Sequential Logic Design – Controllers: Design Example

Circuit Description: Temperature History Storage

Functional Description:

Design a system that records the outside temperature every hours and displays the last three recorded temperatures.

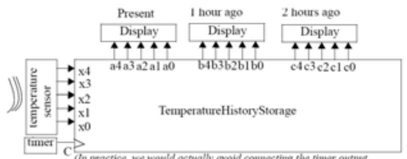
Inputs:

c: clock signal

x4..0: 5-bit temperature reading

Outputs:

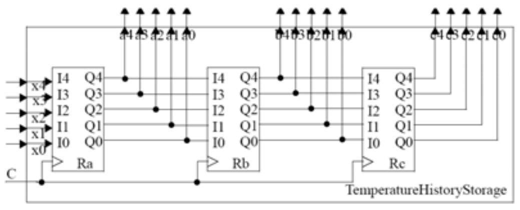
a4..0, b4..0, c4..0: 5-bit temperature readings to be displayed



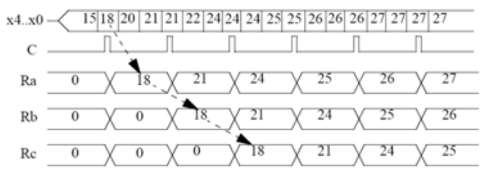
(In practice, we would actually avoid connecting the timer output C to a clock input, instead only connecting an oscillator output to a clock input.)

Digital Design
Sequential Logic Design -- Controllers

Internal design of the TemperatureHistoryStorage component

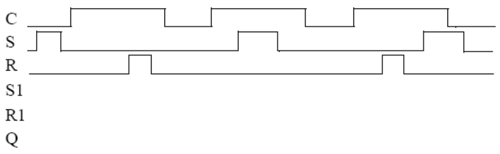


Digital Design
Sequential Logic Design -- Controllers

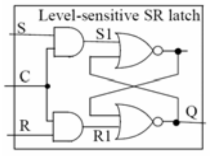


Example of values in the TemperatureHistoryStorage registers. One particular data item, 18, is shown moving through the registers on each clock cycle.

Digital Design
Sequential Logic Design – Controllers: Not really a quiz!!



Trace the behavior of a level-sensitive SR latch for the input pattern above. Complete the timing diagram, assuming the logic gates have a tiny but non-zero delay.



Digital Design
Sequential Logic Design – Controllers: FSM

Circuit Description: Laser Timer System

Functional Description:

Design a system that activates a laser for exactly 30 ns after it receives a button press.

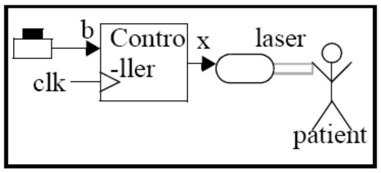
Inputs:

c: 10 ns clock signal

b: button signal

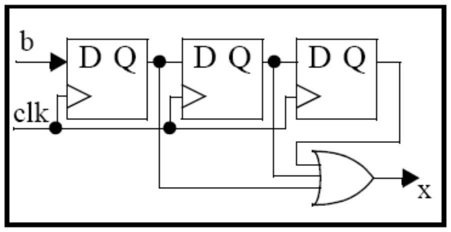
Outputs:

x: laser output



Digital Design
Sequential Logic Design -- Controllers

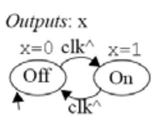
First (bad) attempt to implement the laser surgery system.

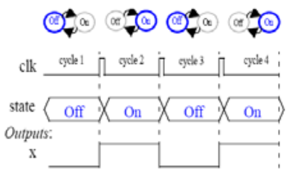


What's so baaaaad about it?

Digital Design
Sequential Logic Design -- Controllers: FSM

A simple state diagram and the timing diagram describing the state diagram's behavior.



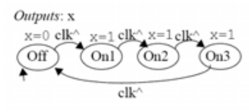


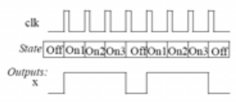
"clk^" represents the rising edge of the clock signal

Digital Design
Sequential Logic Design -- Controllers: FSM

Three-Cycle High System - Finite State Machine

- Set of States; i.e. {Off, On1, On2, On3}
- Set of Inputs/Outputs; i.e. {}/{x}
- Initial State: i.e. {Off}
- Set of Transitions (conditions): (state:input->new state)
  - {Off:!clk^->Off, Off:clk^->On1, On1:clk^->On1...}
- Set of Actions (output values): {Off:x=0, On1:x=1, On2: x=1, On3: x=1}





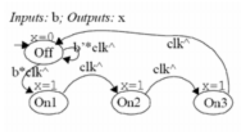
state diagram

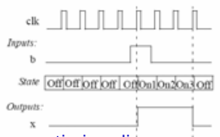
timing diagram

Digital Design
Sequential Logic Design -- Controllers: FSM

Three-Cycle High System - Finite State Machine

- Set of States; i.e. {Off, On1, On2, On3}
- Set of Inputs/Outputs; i.e. {b}/{x}
- Initial State: i.e. {Off}
- Set of Transitions (conditions): (state:input->new state)
  - {Off:b'*!clk^->Off, Off:b*clk^->On1, On1:clk^->On1...}
- Set of Actions (output values): {Off:x=0, On1:x=1, On2: x=1, On3: x=1}





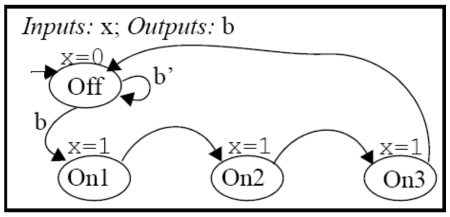
state diagram

timing diagram

Digital Design
Sequential Logic Design -- Controllers: FSM

Simplification in Notation:

- implicit clk^
- every transition is ANDed with a rising clock.



Digital Design
Sequential Logic Design -- Controllers

Why are the heads of keys getting thicker?

The key on the right has a computer chip inside that sends an identifier to the car's computer, thus helping to reduce car thefts.

Digital Design
Sequential Logic Design – Controllers: FSM

Circuit Description: Secure Car Key

Functional Description:

Design a secure car key controller for a key having a code of 1011

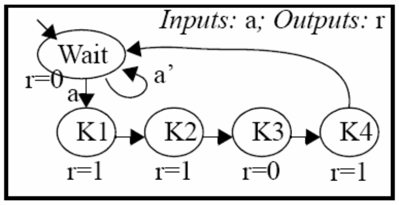
Inputs: clock assumed

a: 1 when the car's computer requests the key ID

Outputs:

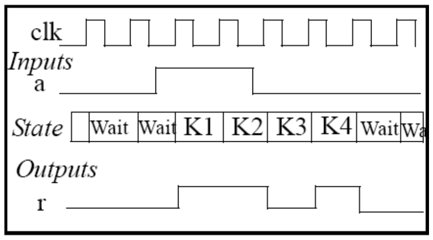
r: individual bits of key code (starting with rightmost bit)

Digital Design
Sequential Logic Design -- Controllers



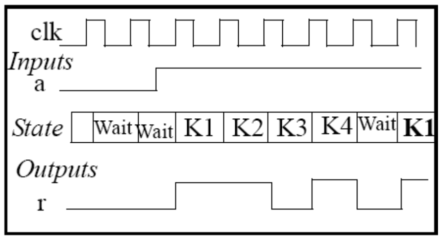
Secure car key FSM.

Digital Design
Sequential Logic Design -- Controllers



Secure car key timing diagram.

Digital Design
Sequential Logic Design -- Controllers



Secure car key timing diagram for a different sequence of values on input a.

Digital Design
Sequential Logic Design – Controllers: FSM

Circuit Description: Code Detector

Functional Description:

Design a system that unlocks a door once is has received the correct sequence of colored buttons

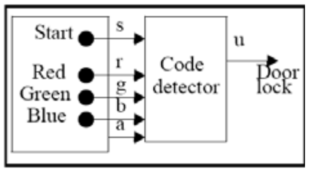
Inputs: clock assumed

buttons: r(red), g(green), b(blue): 1 when button of corresponding color is pressed; 0 otherwise– assume presses synchronized with clock

a(any): 1 if any button(s) have been pressed (while pressed)

Outputs:

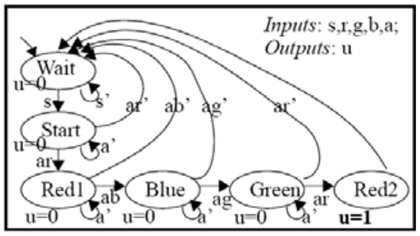
u: signal to unlock door



Digital Design
Sequential Logic Design – Controllers: FSM Design

Build an FSM to Detect Sequence:

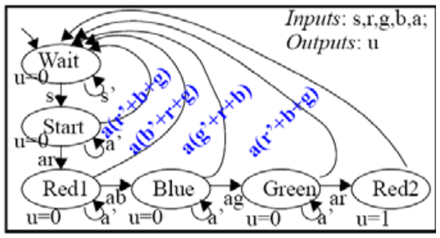
Start->Red->Blue->Green->Red



You can press all three buttons at the same time, and the door will unlock

Digital Design
Sequential Logic Design – Controllers: FSM Design

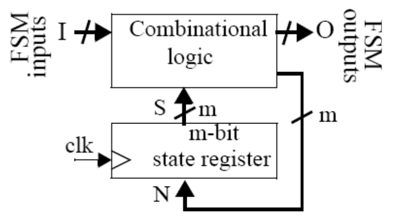
Improved code detector FSM



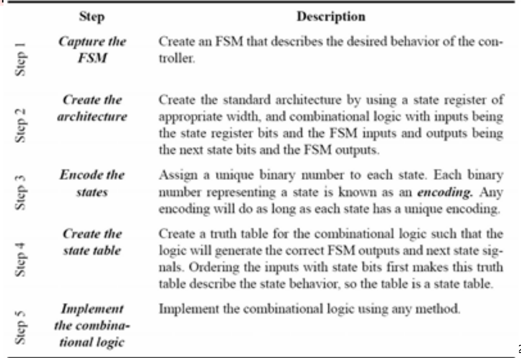
Better, but still flawed: a=r=g=b=1, ab=1, a(b'+r+g)=1

Digital Design
Sequential Logic Design – Controllers: Implementation

Standard controller architecture -- general view



Digital Design
Sequential Logic Design – Controllers: 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.

Digital Design
Sequential Logic Design – Controllers: FSM

Circuit Description: Laser Timer System

Functional Description:

Design a system that activates a laser for exactly 30 ns after it receives a button press.

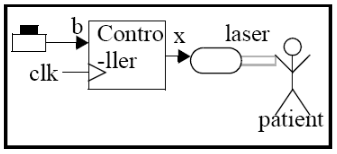
Inputs:

c: 10 ns clock signal

b: button signal

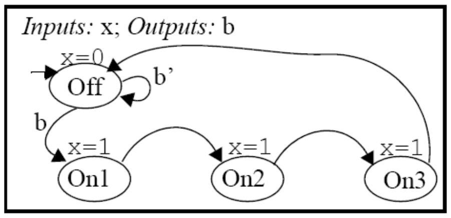
Outputs:

x: laser output



Digital Design
Sequential Logic Design -- Controllers: FSM

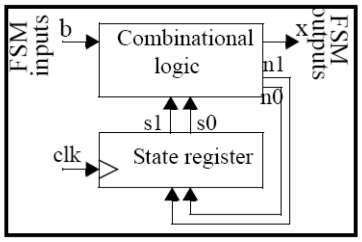
1) Capture the FSM:



Digital Design
Sequential Logic Design – Controllers: Implementation

2) Create the Architecture:

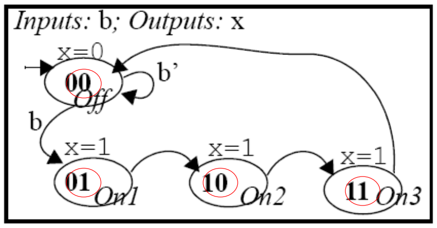
Standard controller architecture for the laser timer.



Digital Design
Sequential Logic Design -- Controllers : Implementation

3) Encode the States:

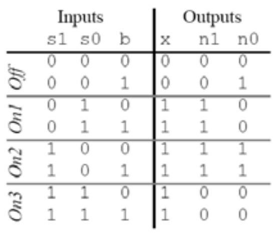
Laser timer state diagram with encoded states.



Digital Design
Sequential Logic Design -- Controllers : Implementation

4) Create the State Table:

State table for laser timer controller

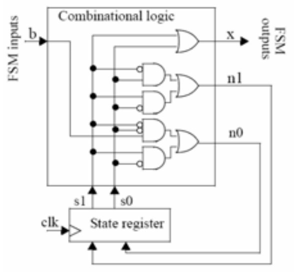


	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

Digital Design
Sequential Logic Design -- Controllers : Implementation

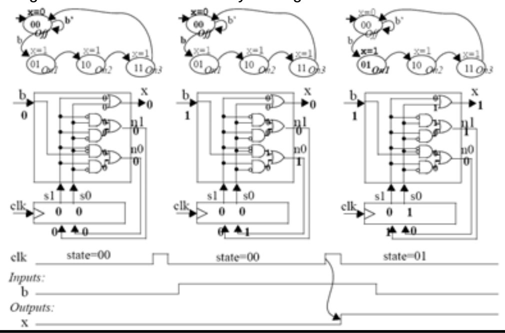
5) Implement Combinational Logic:

Final implementation of the three-cycles-high laser timer controller.

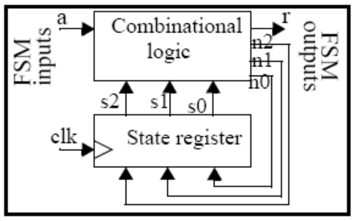


Digital Design
Sequential Logic Design -- Controllers : Implementation

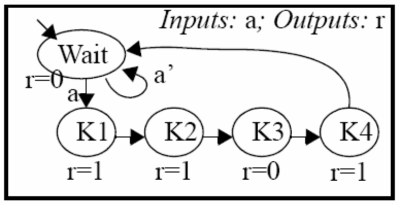
Tracing the behavior of the three-cycles-high laser timer controller.



Digital Design
Sequential Logic Design -- Controllers

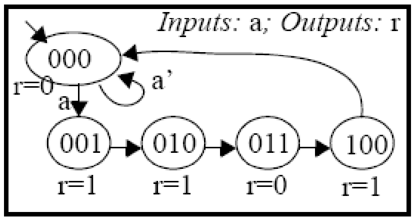


Digital Design
Sequential Logic Design -- Controllers



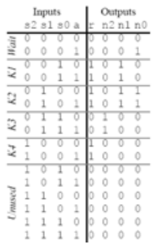
Original secure car key FSM.

Digital Design
Sequential Logic Design -- Controllers



Secure car key FSM with encoded states

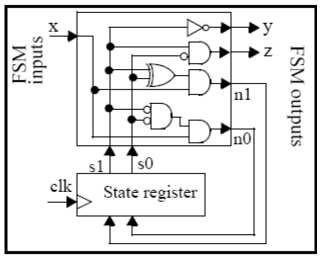
Digital Design
Sequential Logic Design -- Controllers



	Inputs				Outputs			
	s2	s1	s0	a	r	n2	n1	n0
Wait	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	1
K1	0	0	1	0	1	0	1	0
	0	0	1	1	1	0	1	0
K2	0	1	0	0	1	0	1	1
	0	1	0	1	1	0	1	1
K3	0	1	1	0	0	1	0	0
	0	1	1	1	0	1	0	0
K4	1	0	0	0	1	0	0	0
	1	0	0	1	1	0	0	0
Unused	1	0	1	0	0	0	0	0
	1	0	1	1	0	0	0	0
	1	1	0	0	0	0	0	0
	1	1	0	1	0	0	0	0
	1	1	1	0	0	0	0	0
	1	1	1	1	0	0	0	0

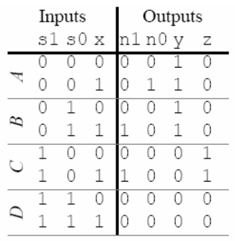
State table for secure car key controller

Digital Design
Sequential Logic Design -- Controllers



An unknown standard controller architecture.

Digital Design
Sequential Logic Design -- Controllers

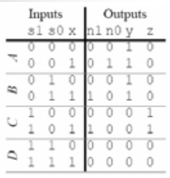


	Inputs			Outputs			
	s1	s0	x	n1	n0	y	z
A	0	0	0	0	0	1	0
	0	0	1	0	1	1	0
B	0	1	0	0	0	1	0
	0	1	1	1	0	1	0
C	1	0	0	0	0	0	1
	1	0	1	1	0	0	1
D	1	1	0	0	0	0	0
	1	1	1	0	0	0	0

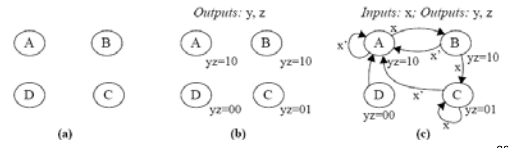
State table for unknown controller.

Digital Design
Sequential Logic Design -- Controllers

- Converting a state table to an FSM diagram
  - (a) Initial FSM
  - (b) FSM with outputs specified
  - (c) FSM with outputs and transitions specified



	Inputs			Outputs			
	s1	s0	x	n1	n0	y	z
A	0	0	0	0	0	1	0
	0	0	1	0	1	1	0
B	0	1	0	0	0	1	0
	0	1	1	1	0	1	0
C	1	0	0	0	0	0	1
	1	0	1	1	0	0	1
D	1	1	0	0	0	0	0
	1	1	1	0	0	0	0


A
B
D
C
(a)
Outputs: y, z
A
B
yz=10
yz=10
D
C
yz=00
yz=01
(b)
Inputs: x; Outputs: y, z
x
x'
A
B
x'
yz=10
yz=10
x
x'
D
C
yz=00
yz=01
x
(c)