### Sequential Logic Design

**Describing Behavior of Sequential Circuit: FSM**

- **Finite-State Machine (FSM)**
  - A way to describe desired behavior of sequential circuit
    - Akin to Boolean equations for combinational behavior
  - List states, and transitions among states
    - Example: Make \( x \) change toggle (0 to 1, or 1 to 0) every clock cycle
    - Two states: “Off” (\( x=0 \)), and “On” (\( x=1 \))
    - Transition from Off to On, or On to Off, on rising clock edge
    - Arrow with no starting state points to initial state (when circuit first starts)

#### Outputs: \( x \)

<table>
<thead>
<tr>
<th>x=0</th>
<th>clk^</th>
<th>x=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>clk^</td>
<td>On</td>
</tr>
</tbody>
</table>

#### FSM Example: Three-Cycles High Laser Timer

- **Three Cycle High Laser Timer**
  - Turn on laser for three cycles whenever button is pressed
  - FSM needs four states
    - Wait in *Off* state while \( b = 0 \) (\( b' \))
    - When \( b = 1 \) (and rising clock edge), transition to *On1*
      - Sets \( x = 1 \)
      - On next two clock edges, transition to *On2*, then *On3* which also set \( x = 1 \)
    - So \( x = 1 \) for three cycles after button pressed

#### Inputs: \( b \); Outputs: \( x \)
Sequential Logic Design
FSM Simplification: Rising Clock Edges Implicit

- Showing rising clock on every transition: cluttered
  - Make implicit -- assume every edge has rising clock, even if not shown
  - What if we wanted a transition without a rising edge
    - We don’t consider such asynchronous FSMs -- less common, and advanced topic
    - Only consider synchronous FSMs -- rising edge on every transition

Inputs: b; Outputs: x

Note: Transition with no associated condition thus transitions to next state on next clock cycle

Sequential Logic Design
FSM Definition

- FSM consists of
  - Set of states
    - Ex: (Off, On1, On2, On3)
  - Set of inputs, set of outputs
    - Ex: Inputs: {x}, Outputs: {b}
  - Initial state
    - Ex: “Off”
  - Set of transitions
    - Describes next states
    - Ex: Has 5 transitions
  - Set of actions
    - Sets outputs while in states
    - Ex: x=0, x=1, x=1, and x=1

We often draw FSM graphically, known as state diagram
Can also use table (state table), or textual languages

Sequential Logic Design
FSM Example: Secure Car Key

- Many new car keys include tiny computer chip
  - When car starts, car’s computer (under engine hood) requests identifier from key
  - Key transmits identifier
    - If not, computer shuts off car
- FSM
  - Wait until computer requests ID (a=1)
  - Transmit ID (in this case, 1101)

Sequential Logic Design
FSM Example: Secure Car Key (cont.)

- Nice feature of FSM
  - Can evaluate output behavior for different input sequence
  - Timing diagrams show states and output values for different input waveforms

Inputs: a; Outputs: r
Sequential Logic Design

FSM Example: Code Detector

- What is the state in the FSM in at the indicated time?
  1. K1
  2. K2
  3. K3
  4. K4

- Unlock door (u=1) only when buttons pressed in sequence:
  - start, then red, blue, green, red
- Input from each button: s, r, g, b
  - Also, output a indicates that some colored button pressed
  - FSM
  - Wait for start (s=1) in “Wait”
  - Once started (“Start”)
    - If see red, go to “Red1”
    - Then, if see blue, go to “Blue”
    - Then, if see green, go to “Green”
    - Then, if see red, go to “Red2”
    - In that state, open the door (u=1)
  - Wrong button at any step, return to “Wait”, without opening door

- Can you trick this FSM to open the door, without knowing the code?
  1. Yes
  2. No
  3. Not sure

- New transition conditions detect if wrong button pressed, returns to “Wait”
- FSM provides formal, concrete means to accurately define desired behavior
Sequential Logic Design
Standard Controller Architecture

- How implement FSM as sequential circuit?
  - Use standard architecture
    - State register -- to store the present state
    - Combinational logic -- to compute outputs, and next state
    - For laser timer FSM
      - 2-bit state register, can represent four states
      - Input b, output x
  - Known as controller

Inputs: b; Outputs: x

- 2-bit state register, can represent four states
- Input b, output x

Known as controller

General version

- 2-bit state register
- FSM inputs
- FSM outputs
- Combinational logic
- State register
- m-bit state register

Diagram: FSM diagram with states On1, On2, On3, Off.