B38DB: Digital Design and Programming
Sequential Logic Design – Finite Sate Machines

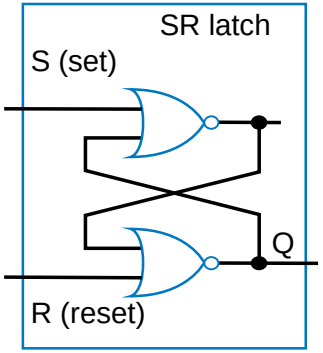
Mustafa Suphi Erden

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

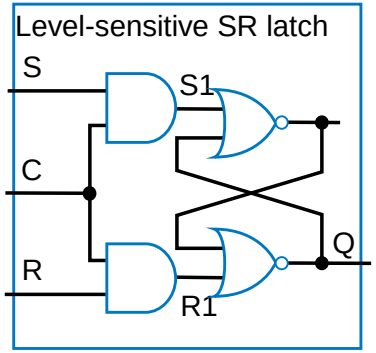
Reminder: Bit Storage Summary

We designed a new building block, a **flip-flop**, that stores one bit.

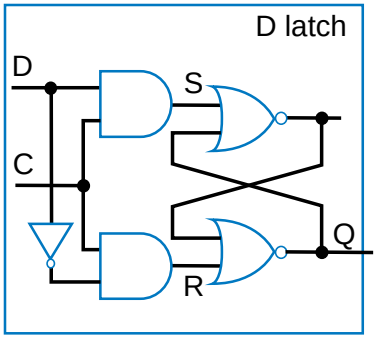
What will we do with this?



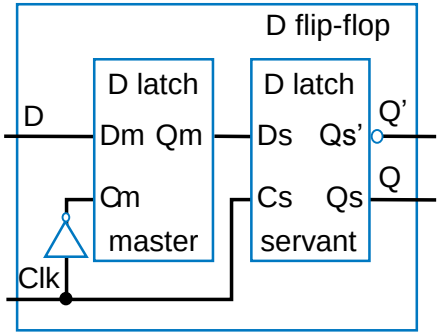
Feature: S=1 sets Q to 1, R=1 resets Q to 0. Problem: SR=11 yield undefined Q.



Feature: S and R only have effect when C=1. We can design outside circuit so SR=11 never happens when C=1. Problem: avoiding SR=11 can be a burden.



Feature: SR can't be 11 if D is stable before and while C=1, and will be 11 for only a brief glitch even if D changes while C=1. Problem: C=1 too long propagates new values through too many latches: too short may not enable a store.



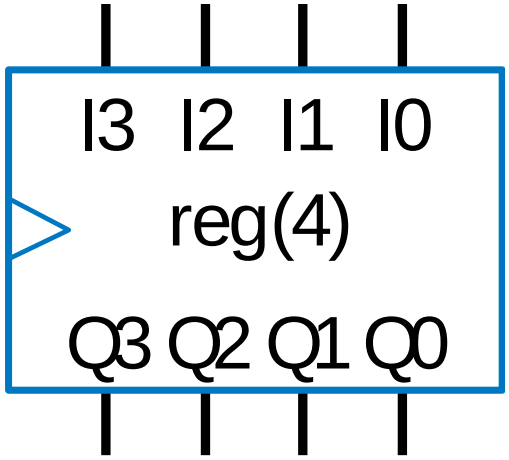
Feature: Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle. Tradeoff: uses more gates internally than D latch, and requires more external gates than SR – but gate count is less of an issue today.

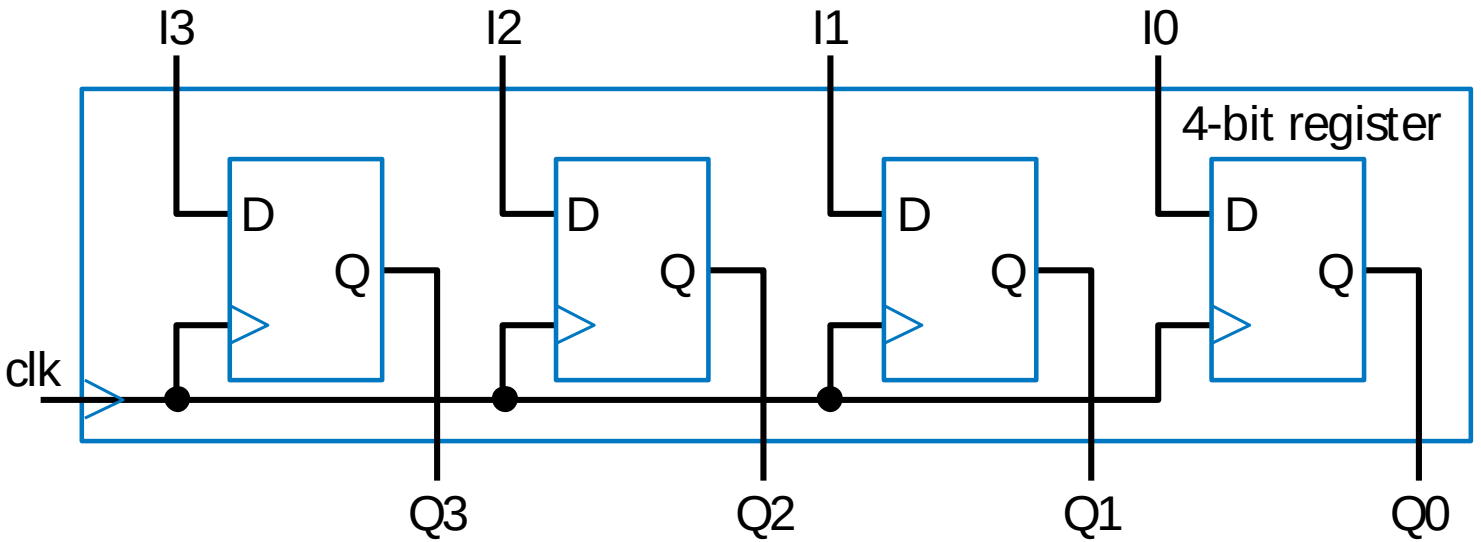
Introduction

- We will:
  - Combine flip-flops to build a multi-bit storage **register**
  - Describe the sequential behavior using a **finite state machine (FSM)**
  - Convert a FSM to a **controller** – a sequential circuit having a register and combinational logic

Register

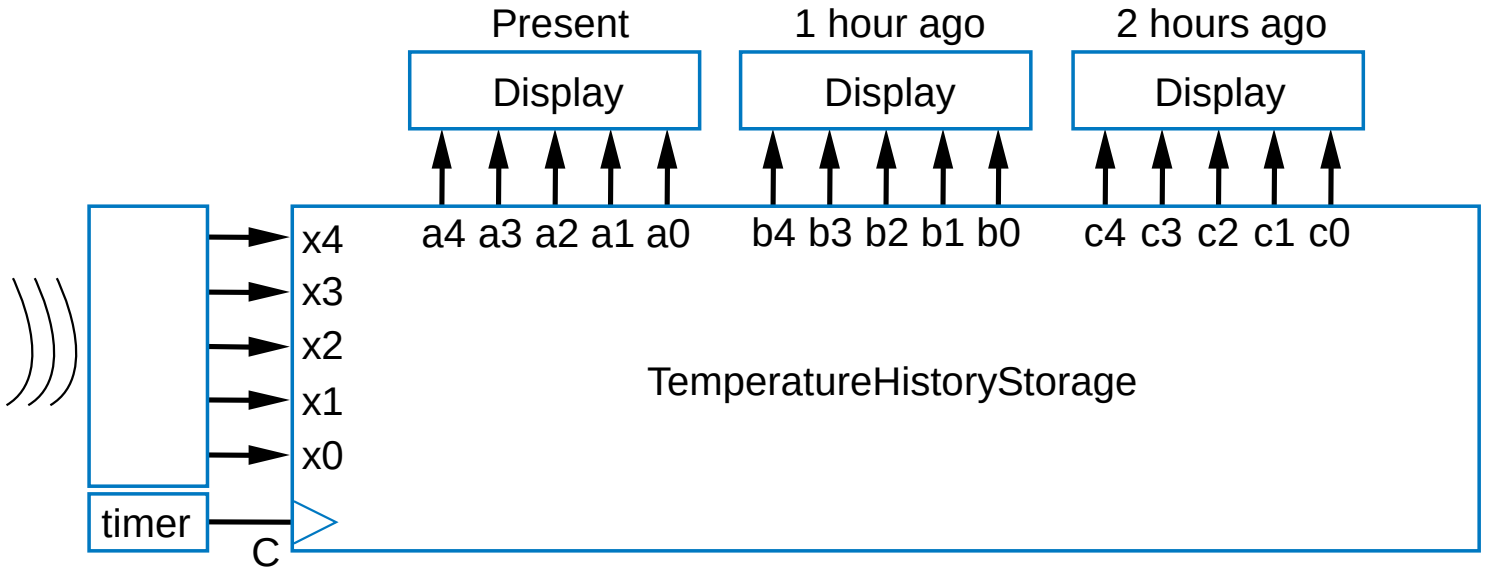
- Typically, we store multi-bit items
  - e.g., storing a 4-bit binary number
- ***Register***: multiple flip-flops sharing a clock signal





Example Using Registers: Temperature Display (1/2)

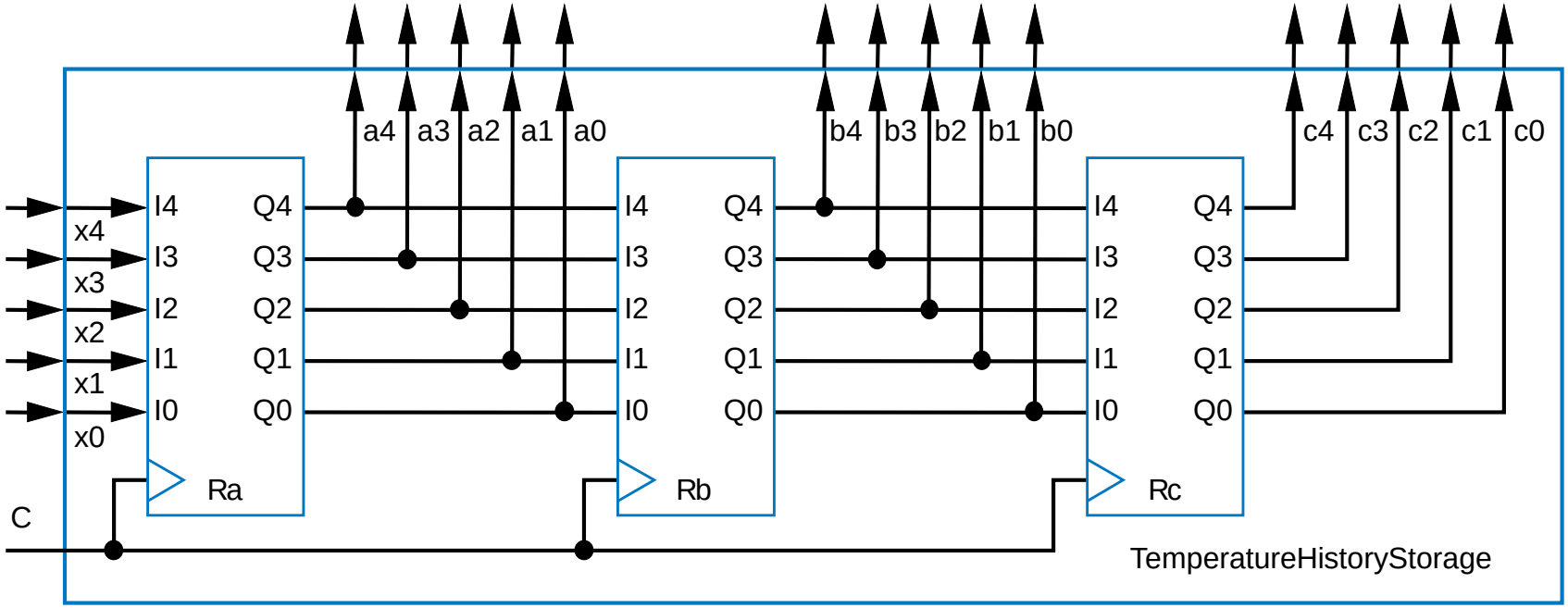
- Temperature history display
  - Sensor outputs temperature as 5-bit binary number
  - Timer pulses C every hour
  - Record temperature on each pulse, display the last three recorded values



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

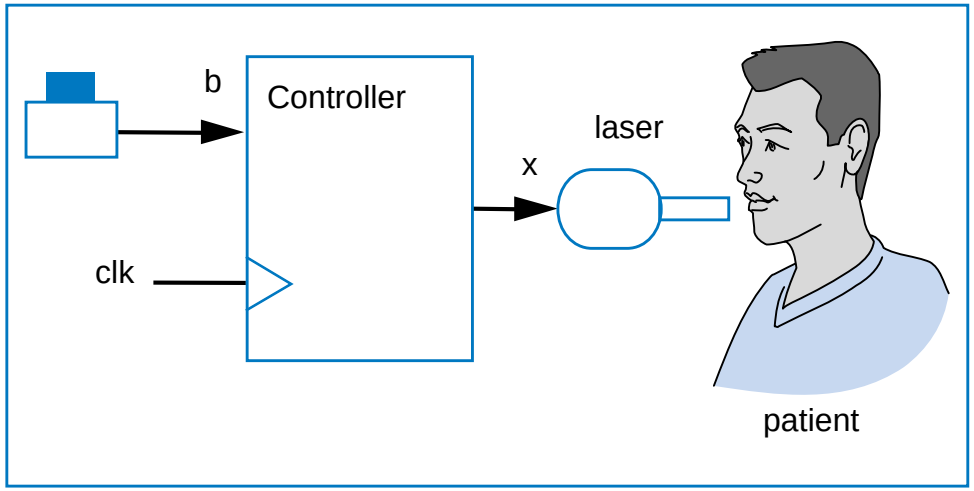
Example Using Registers: Temperature Display (2/2)

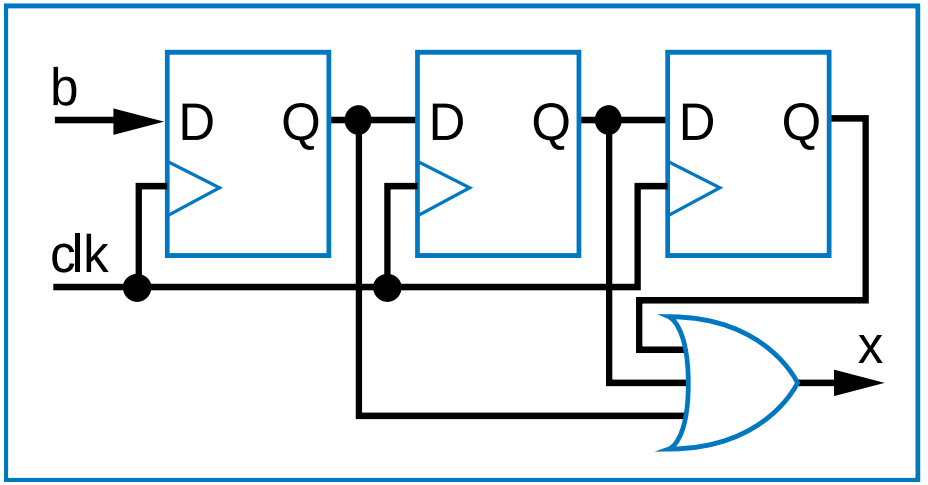
- Use three 5-bit registers



Finite-State Machines (FSMs) and Controllers

- We want to have sequential circuits with particular behavior over time
- Example: Laser timer
  - Push button: x=1 for 3 clock cycles





How to Design Sequential Circuits

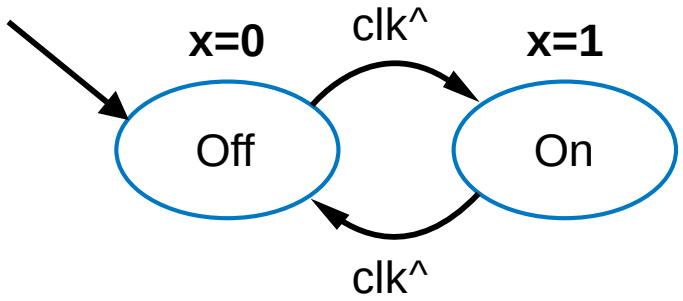
- **Combinational circuit** design process had two important things:
  1. A formal way to describe desired circuit behavior
     - Boolean equation, or truth table
  2. A well-defined process to convert that behavior into a circuit
     - Using Karnough maps

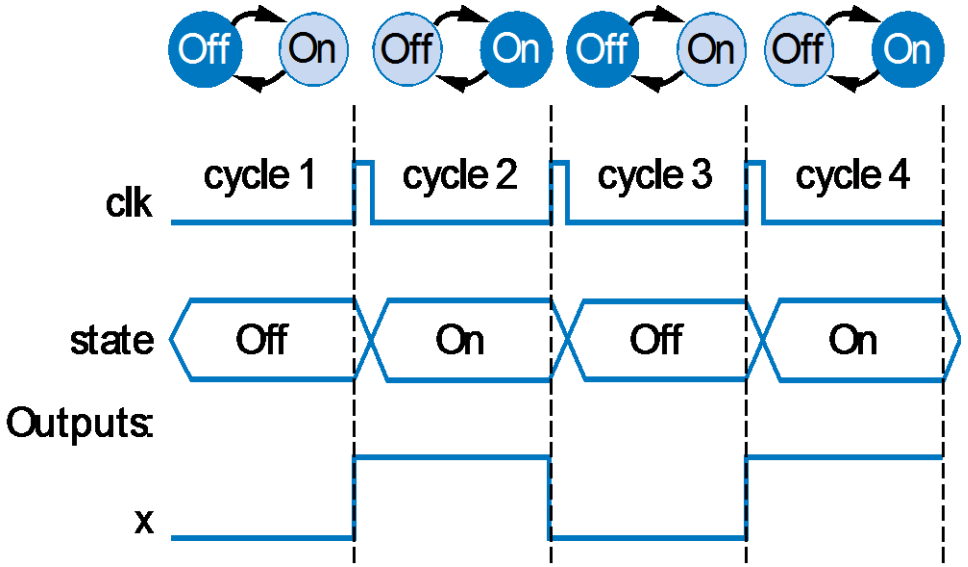
- We need similar methods for **sequential circuit design**:
  1. How to describe desired behavior?
     - **Finite State Machines**
  2. How to convert that behavior into a circuit?
     - **Controller**

FSM: Describing Behavior of a Sequential Circuit

- Finite-State Machine (FSM)
  - Define **states** and **transitions** among the states
    - Example:
    - Make x change its toggle (0 to 1 and 1 to 0) every clock cycle
    - Two states: “Off” (x=0), and “On” (x=1)
    - Arrow with no starting state points to the initial state (when the circuit first starts)

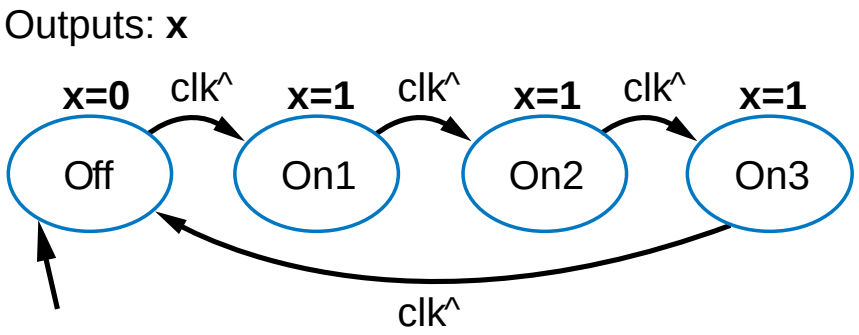
Output: **x**

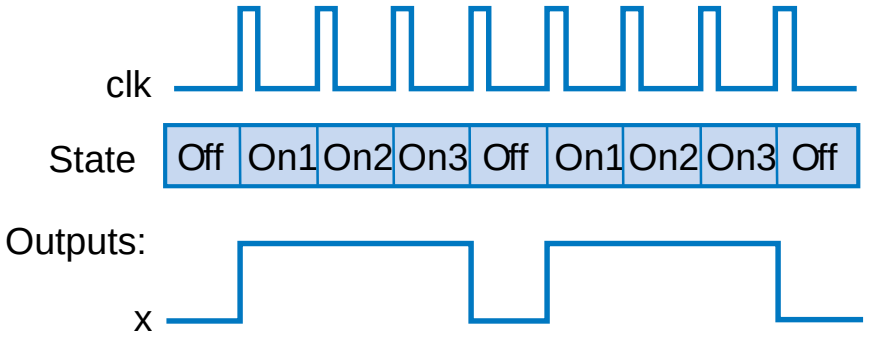




FSM Example: 0,1,1,1, repeating

- We want to generate the 0, 1, 1, 1, 0, 1, 1, 1, ... sequence
  - Each value is for one clock cycle
- Describe it as a FSM
  - Four states
  - Transition on rising clock edge to the next state

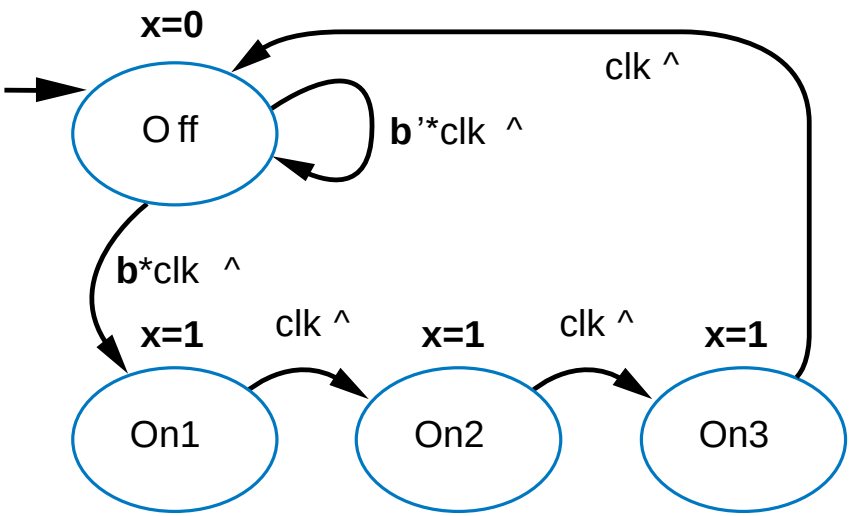


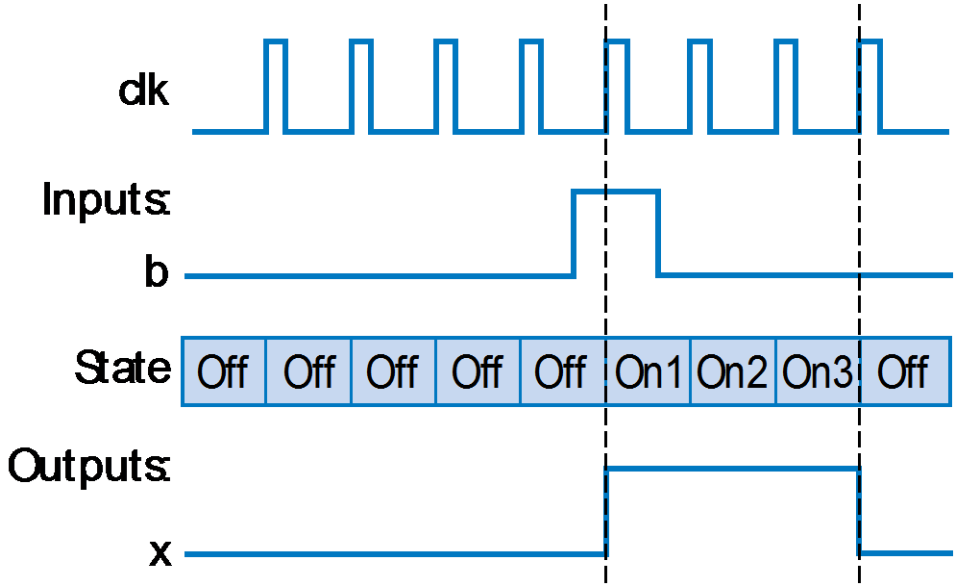


FSM for Three-Cycles High Laser Timer

- Four states
- Wait in the "Off" state while b is 0 (b')
- When b is 1 (and rising clock edge), transition to On1
  - Sets x=1
  - On the next two clock edges, transition to On2, then to On3, which also set x=1
- So x=1 for three cycles after the button is pressed

Input: b; **Output: x**

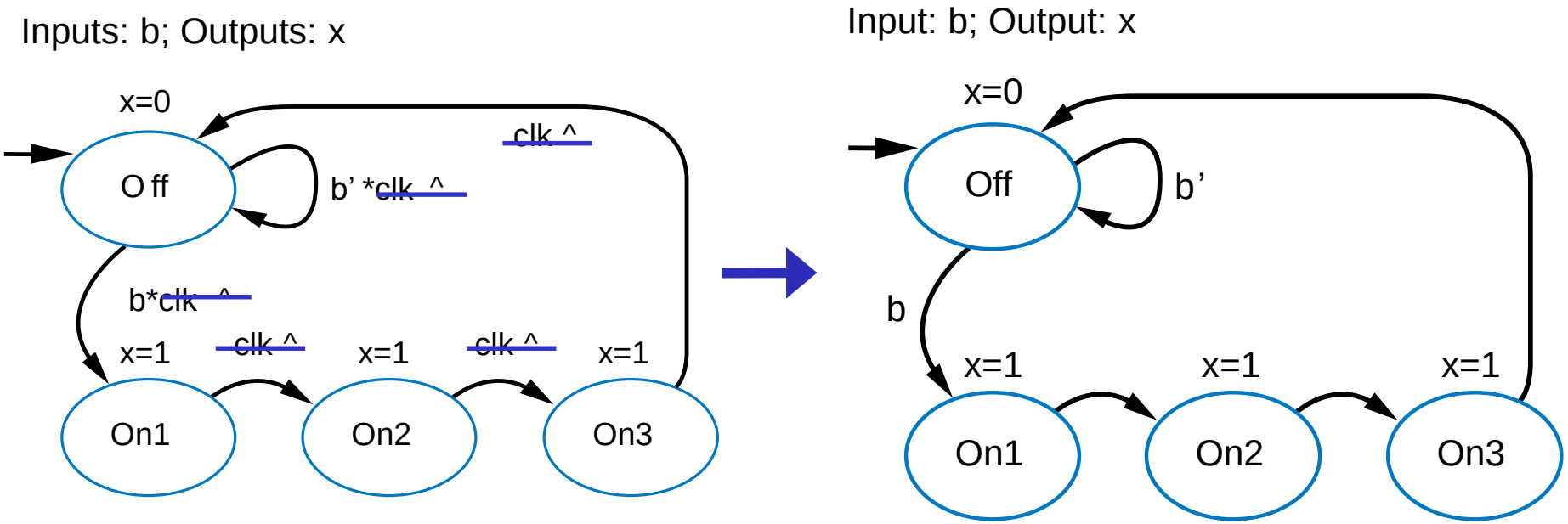




FSM Simplification: Rising Clock Edges Implicit

- Assume every edge has a rising clock, even if not shown
  - In this course: Only consider ***synchronous*** FSMs:

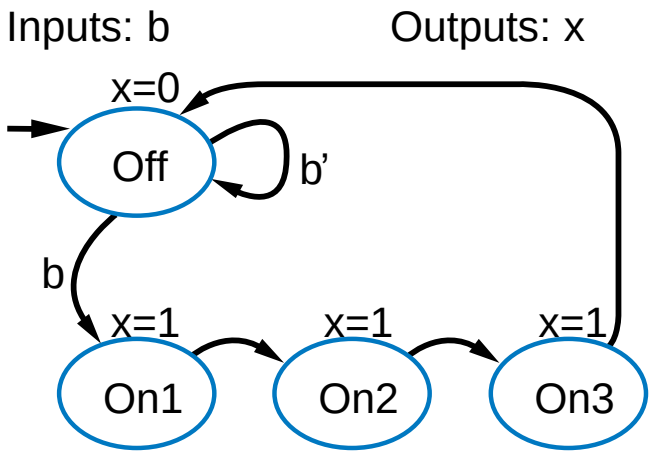
rising edge on *every* transition



Note: *the transitions with no associated condition show direct transitions to the next state on the next clock cycle*

FSM Definition

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

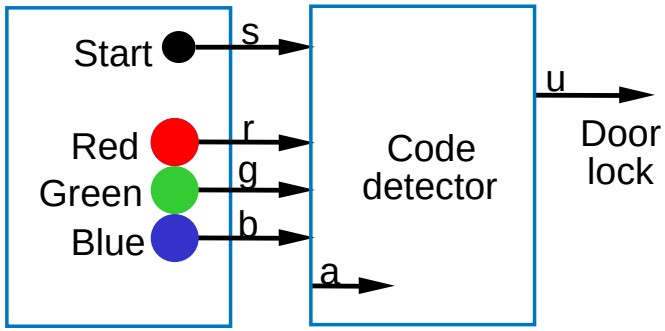


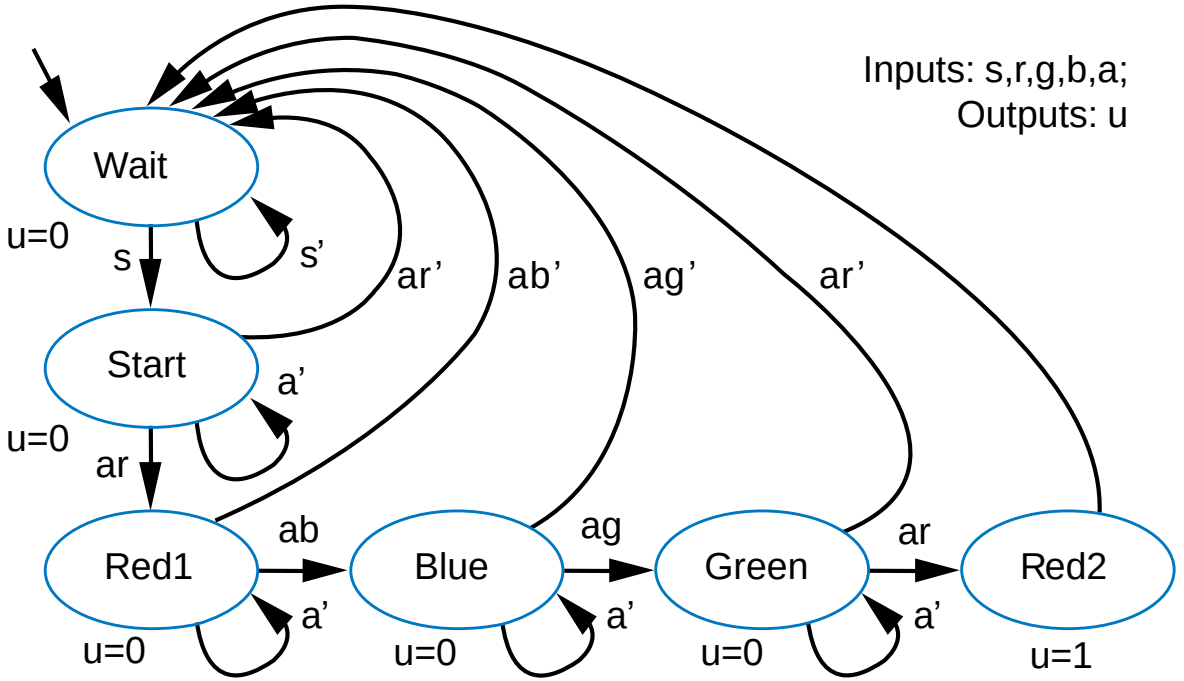
We often draw FSM graphically, known as a ***state diagram***

We can also use a table (state table), or textual languages

FSM Example: Code Detector

- Unlock the door (u=1) only when the buttons are pressed in correct order:
  - **start**, **red**, **blue**, **green**, **red**
- Input from each button: *s*, *r*, *g*, *b*
  - Also, input *a* indicates that some colored button is pressed

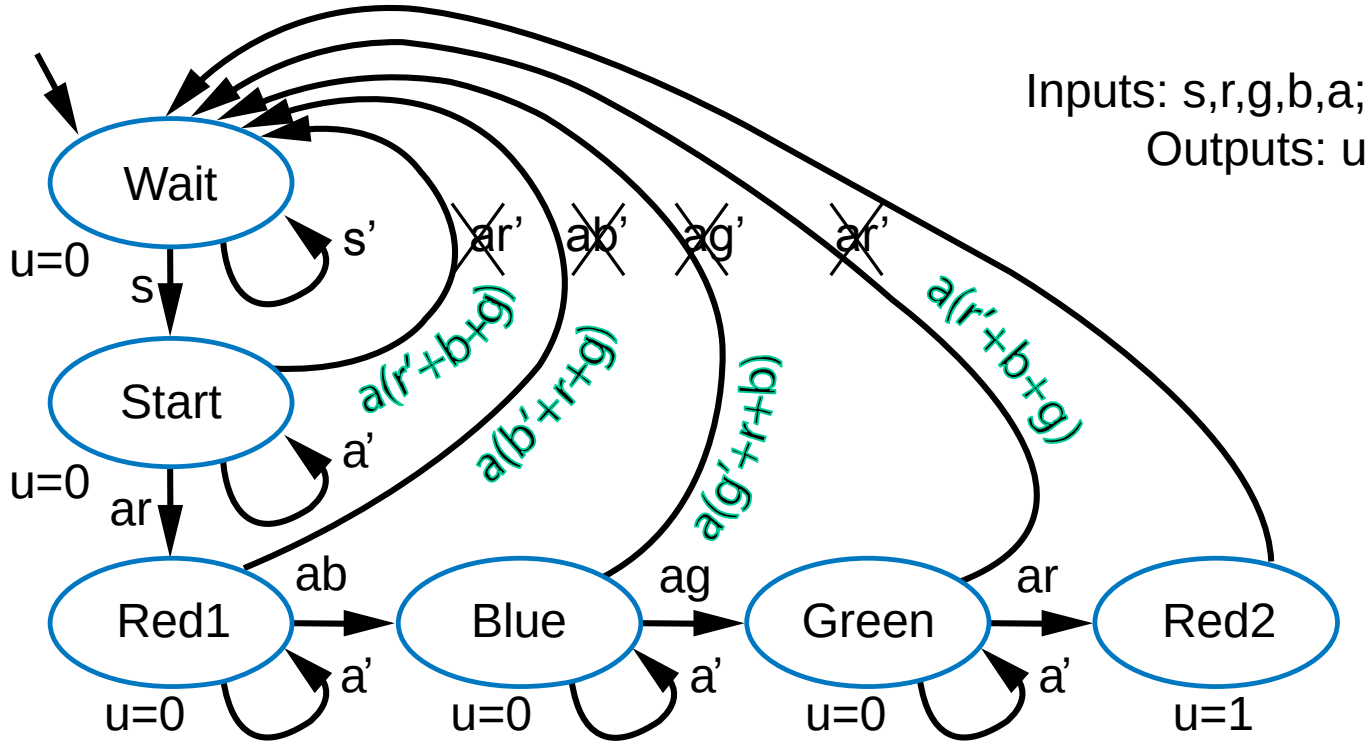




Q: Can you trick this FSM to open the door, without knowing the code?

A: Yes, hold all buttons simultaneously

Improve FSM for Code Detector



- New transition conditions detect when wrong button is pressed, and returns to “Wait”
- Still something is wrong with this FSM, but we will discover it next week.