Example Using Registers: Temperature Display

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

We consider increasingly better bit storage until we arrived at the robust D flip-flop bit storage.
Finite-State Machines (FSMs) and Controllers

- Want sequential circuit with particular behavior over time
- Example: Laser timer
  - Push button: \textit{x}=1 for 3 clock cycles
  - How? Let’s try three flip-flops
    - \textit{b}=1 gets stored in first D flip-flop
    - Then 2nd flip-flop on next cycle, then 3rd flip-flop on next
    - OR the three flip-flop outputs, so \textit{x} should be 1 for three cycles

Digital Design
Copyright © 2006
Frank Vahid

Need a Better Way to Design Sequential Circuits

- Trial and error is not a good design method
  - Will we be able to "guess" a circuit that works for other desired behavior?
  - How about counting up from 1 to 99? Pulsing an output for 1 cycle every 10 cycles? Detecting the sequence 1 3 5 in binary on a 3-bit input?
  - And, a circuit built by guessing may have undesired behavior
    - Laser timer: What if press button again while \textit{x}=1? \textit{x} then stays one another 3 cycles. Is that what we want?
- 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 to a circuit
    - We need those things for sequence circuit design

Digital Design
Copyright © 2006
Frank Vahid

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 \textit{x} change toggle (0 to 1, or 1 to 0) every clock cycle
    - Two states: "Off" (\textit{x}=0), and "On" (\textit{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)

Digital Design
Copyright © 2006
Frank Vahid

Extending FSM to Three-Cycles High Laser Timer

- Four states
  - Wait in "Off" state while \textit{b} is 0 (\textit{b}’)
  - When \textit{b} is 1 (and rising clock edge), transition to On1
    - \textit{s}=x1
    - On next two clock edges, transition to On2, then On3, which also sets \textit{x}=1
  - So \textit{x}=1 for three cycles after button pressed

Digital Design
Copyright © 2006
Frank Vahid

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
**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

**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)

**FSM Example: Code Detector**

- 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

**FSM Example: Secure Car Key (cont.)**

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

Q: Determine states and r value for given input waveform:

**Improve FSM for Code Detector**

- New transition conditions detect if wrong button pressed, returns to "Wait"
- FSM provides formal, concrete means to accurately define desired behavior

**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 inputs, outputs
      - Known as controller
Design Challenge

- Design Challenge
  - Draw a state diagram for a finite state machine that has an input $X$ and an output $Y$. Whenever $X$ changes from 0 to 1, $Y$ should be 1 for two clock cycles and then return to 0 (even if $X$ is still 1).

Due Next Lecture (as announced in class)
1 point extra credit (Homework)