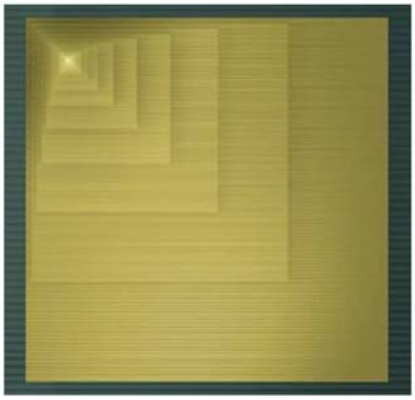


# Digital Design

## Chapter 4b:
## Datapath Components - Sequential

Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
http://www.ddvahid.com

# Introduction

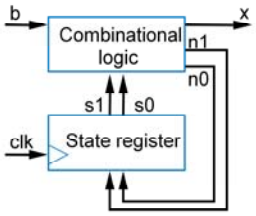
- Chapters 3: Introduced increasingly complex digital building blocks
  - Basic registers, and controllers
- Controllers good for systems with control inputs/outputs
  - ***Control*** input: Single bit (or just a few), representing environment event or state
    - e.g., 1 bit representing button pressed
  - ***Data*** input: Multiple bits collectively representing single entity
    - e.g., 7 bits representing temperature in binary
- Need building blocks for data
  - ***Datapath components***, aka register-transfer-level (RTL) components, store/transform data
    - Put datapath components together to form a ***datapath***
- This chapter introduces numerous datapath components, and simple datapaths
  - Next chapter will combine controllers and datapaths into “processors”

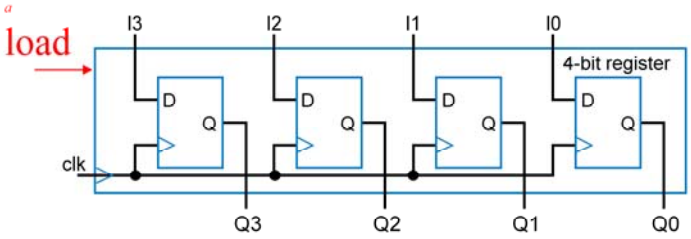


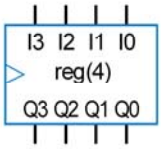
Note: Slides with animation are denoted with a small red "a" near the animated items

# Registers

- Can store data, very common in datapaths
- Basic register of Ch 3: Loaded every cycle
  - Useful for implementing FSM -- stores encoded state
  - For other uses, may want to load only on certain cycles



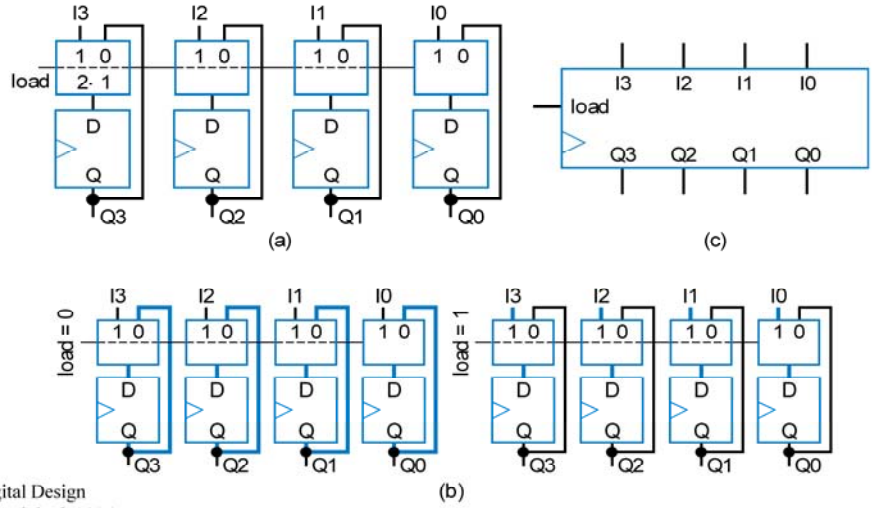




*Basic register loads on every clock cycle*

*How extend to only load on certain cycles?*

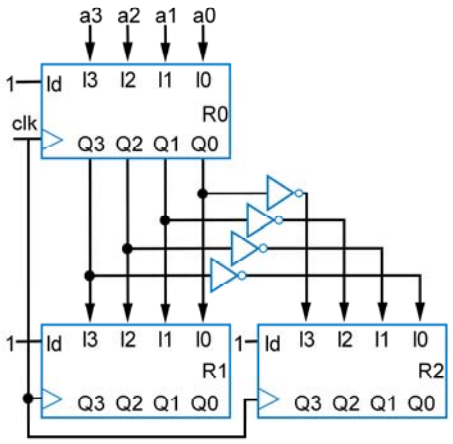
# Register with Parallel Load

- Add 2x1 mux to front of each flip-flop
- Register's load input selects mux input to pass
  - Either existing flip-flop value, or new value to load

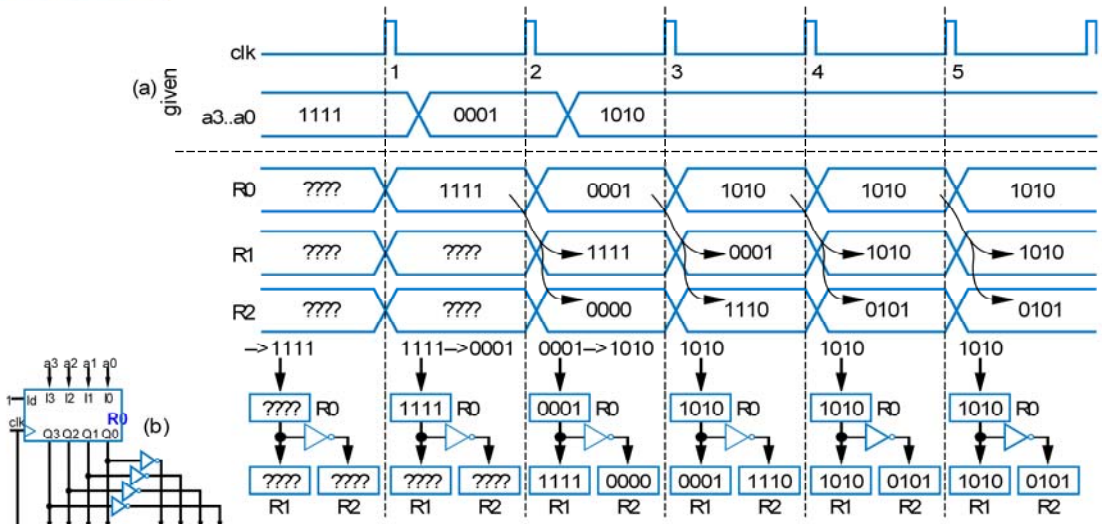


# Basic Example using Registers

- This example will show how registers load simultaneously on clock cycles
  - Notice that all load inputs set to 1 in this example -- just for demonstration purposes

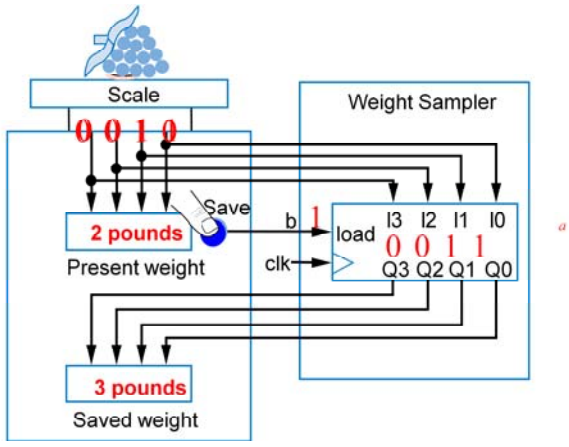



# Basic Example Using Registers (cont.)



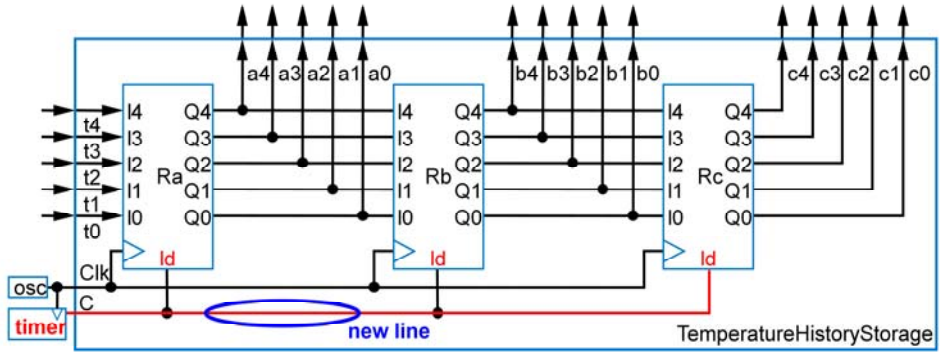
# Register Example using the Load Input: Weight Sampler

- Scale has two displays
  - Present weight
  - Saved weight
  - Useful to compare present item with previous item
- Use register to store weight
  - Pressing button causes present weight to be stored in register
    - Register contents always displayed as "Saved weight," even when new present weight appears

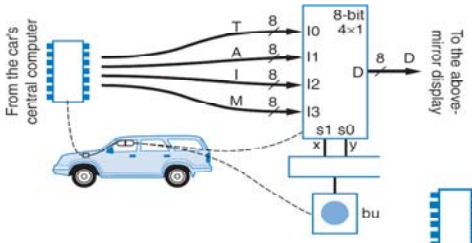


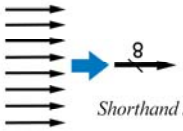
# Register Example: Temperature History Display

- Recall Chpt 3 example
  - Timer pulse every hour
  - Previously used as clock. Better design only connects oscillator to clock inputs -- use registers with load input, connect to timer pulse.

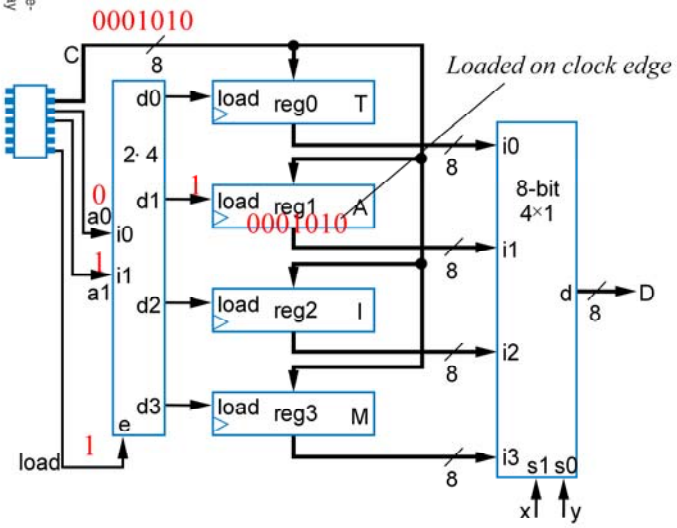


# Register Example: Above-Mirror Display



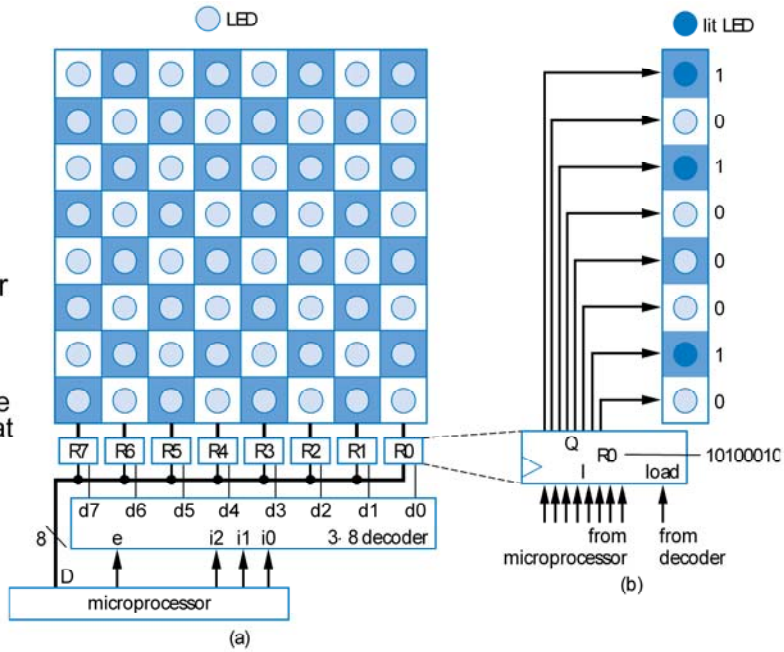




- Ch2 example: Four simultaneous values from car's computer
- To reduce wires: Computer writes only 1 value at a time, loads into one of four registers
  - Was: 8+8+8+8 = 32 wires
  - Now: 8 +2+1 = 11 wires

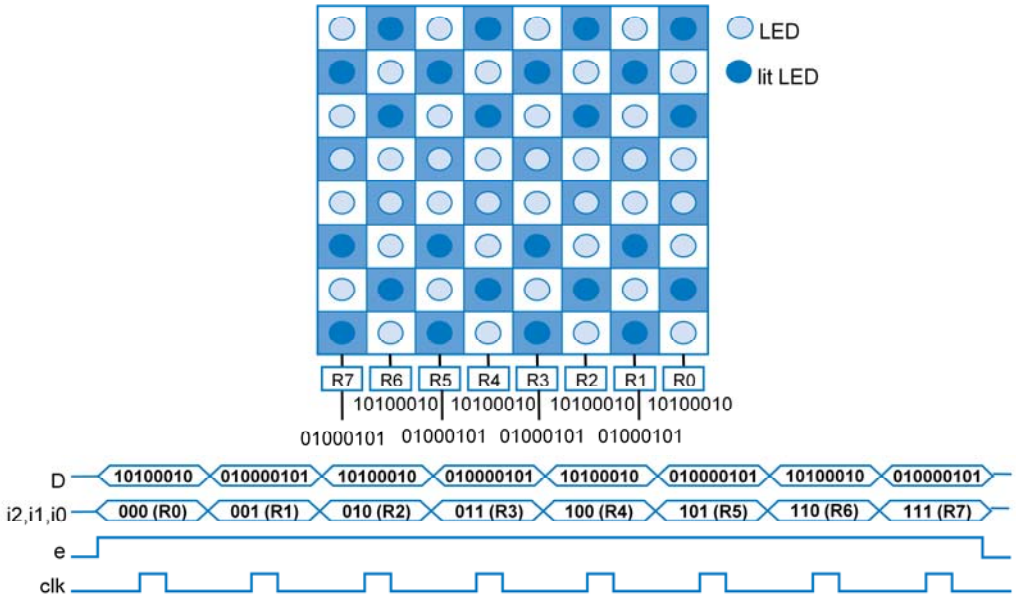



# Register Example: Computerized Checkerboard

- Each register holds values for one column of lights
  - 1 lights light
- Microprocessor loads one register at a time
  - Occurs fast enough that user sees entire board change at once



# Register Example: Computerized Checkerboard

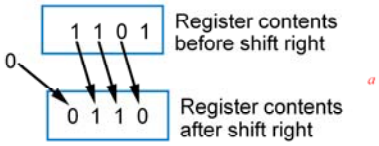






# Shift Register

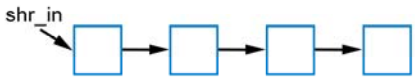
- Shift right
  - Move each bit one position right
  - Shift in 0 to leftmost bit




Q: Do four right shifts on 1001, showing value after each shift

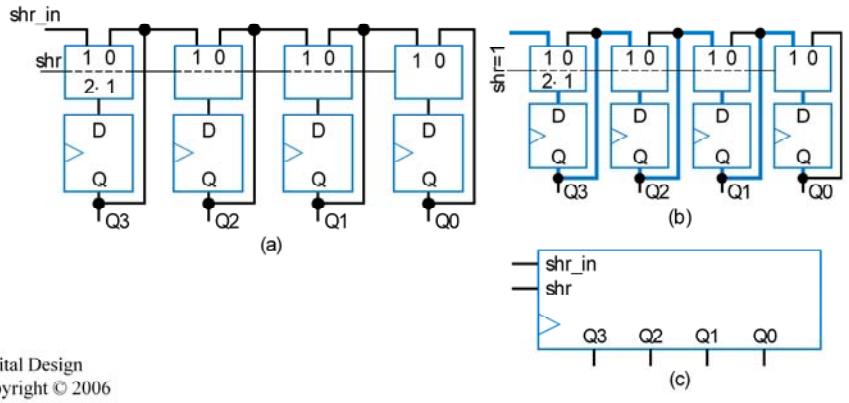
A: 1001 (original)
0100
0010
0001
0000

- Implementation: Connect flip-flop output to next flip-flop's input



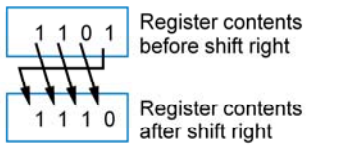

# Shift Register

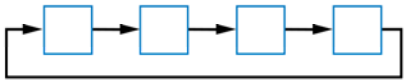
- To allow register to either shift or retain, use 2x1 muxes
  - shr: 0 means retain, 1 shift
  - shr_in: value to shift in
    - May be 0, or 1
- Note: Can easily design shift register that shifts left instead



# Rotate Register

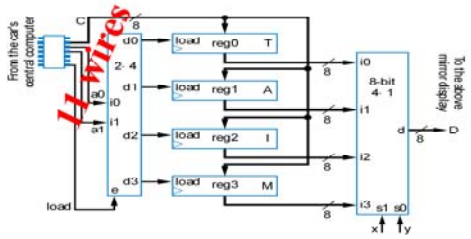
- Rotate right: Like shift right, but leftmost bit comes from rightmost bit
- a.k.a. Barrel Shifter
- Shift registers can be made to shift left as well …

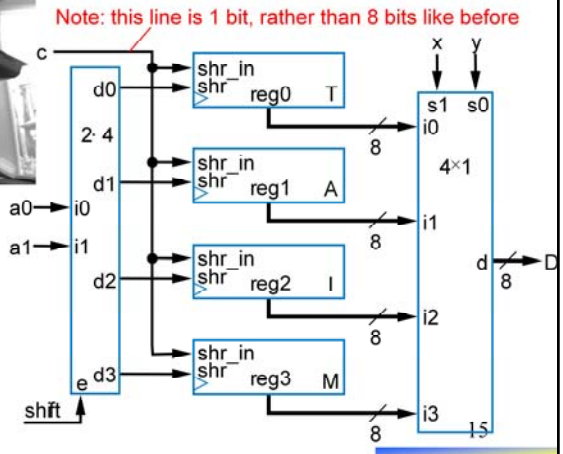


# Shift Register Example: Above-Mirror Display

- Earlier example: 8 +2+1 = 11wires from car's computer to above-mirror display's four registers
  - Better than 32 wires, but 11 still a lot -- want fewer for smaller wire bundles
- Use shift registers
  - Wires: 1+2+1=4
  - Computer sends one value at a time, one bit per clock cycle



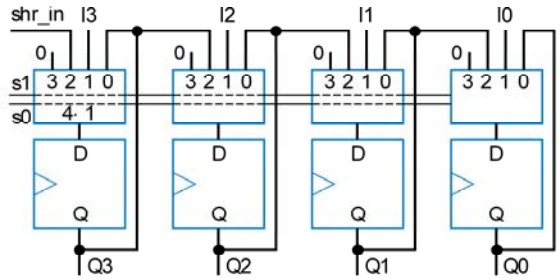



# Multifunction Registers

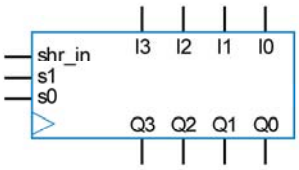
- Many registers have multiple functions
  - Load, shift, clear (load all 0s)
  - And retain present value, of course
- Easily designed using muxes
  - Just connect each mux input to achieve desired function

Functions:

| s1 | s0 | Operation |
|---|---|---|
| 0 | 0 | Maintain present value |
| 0 | 1 | Parallel load |
| 1 | 0 | Shift right |
| 1 | 1 | (unused - let's load 0s) |




(a)




(b)

# Multifunction Registers

| s1 | s0 | Operation |
|---|---|---|
| 0 | 0 | Maintain present value |
| 0 | 1 | Parallel load |
| 1 | 0 | Shift right |
| 1 | 1 | Shift left |