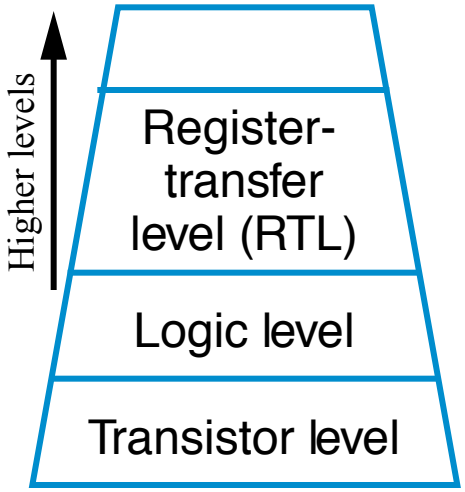
Lecture 21:
RTL Design

CSE 140: Components and Design Techniques for Digital Systems

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego

RTL Design

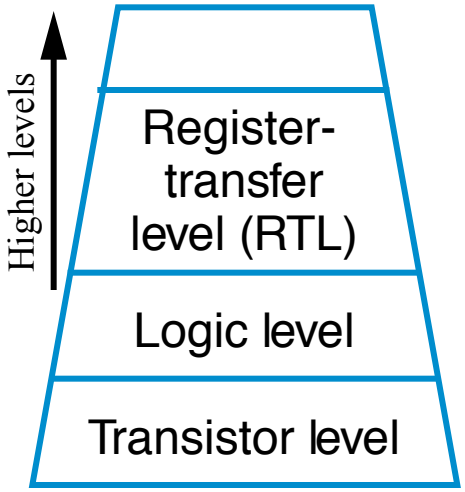
- Combinational design (Logic Level)
  - Capture Comb. behavior: Equations, truth tables
  - Convert to circuit: AND + OR + NOT → Comb. Logic
- Controller Design (Logic Level)
  - Capture sequential behavior: FSMs
  - Convert to circuit: Register + Comb. logic → Controller
- Processor Design (Register Transfer Level)
- Standard Modules (used in Processor design)
  - Adders, shifters, counters, decoder, muxs,…



Levels of digital design abstraction

RTL Design

- Processor Design (Register Transfer Level)
  - Capture behavior: High-level state machine or pseudocode
  - Convert to circuit: Controller + Datapath → Processor
  - Known as "RTL" (register-transfer level) design

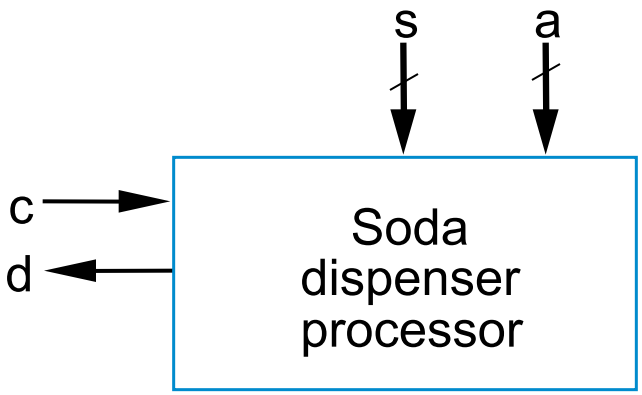


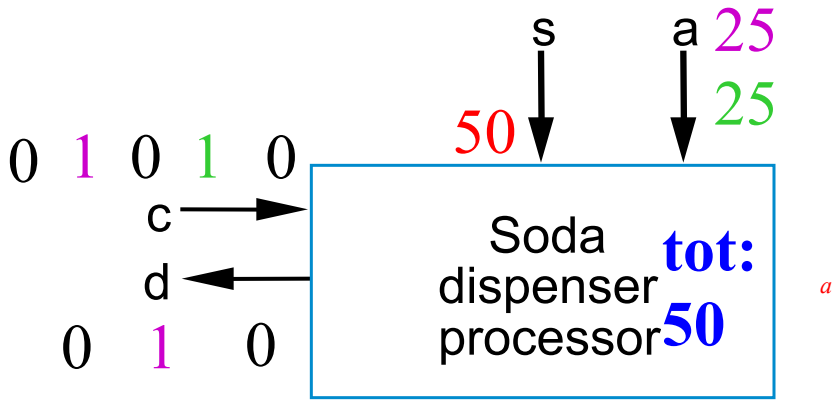
Levels of digital design abstraction

Processors:
- Programmable (microprocessor)
- Custom

High-Level State Machines (HLSMs)

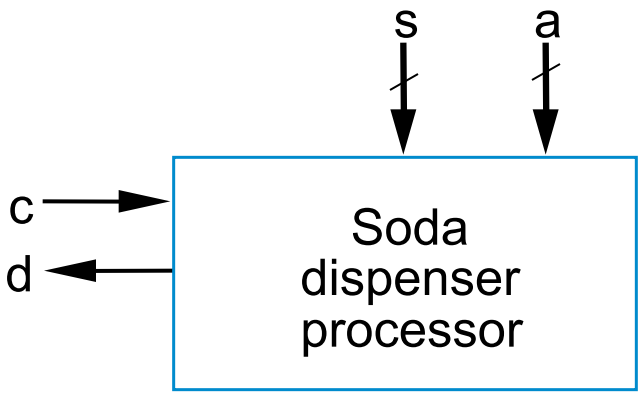
- Some behaviors too complex for equations, truth tables, or FSMs
- Ex: Soda dispenser
  - *c*: bit input, 1 when coin deposited
  - *a*: 8-bit input having value of deposited coin
  - *s*: 8-bit input having cost of a soda
  - *d*: bit output, processor sets to 1 when total value of deposited coins equals or exceeds cost of a soda

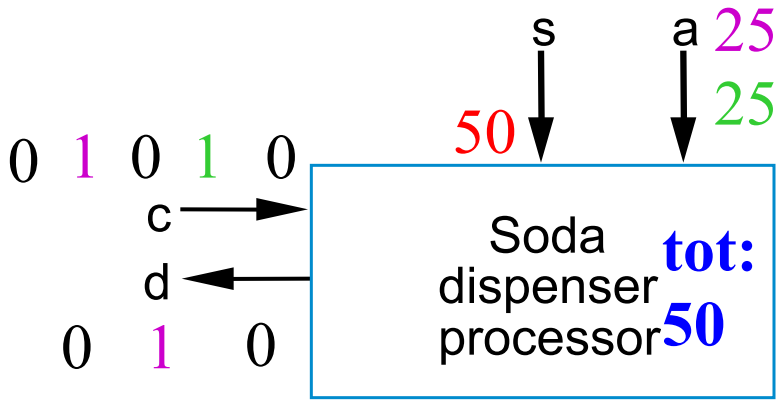




High-Level State Machines (HLSMs)

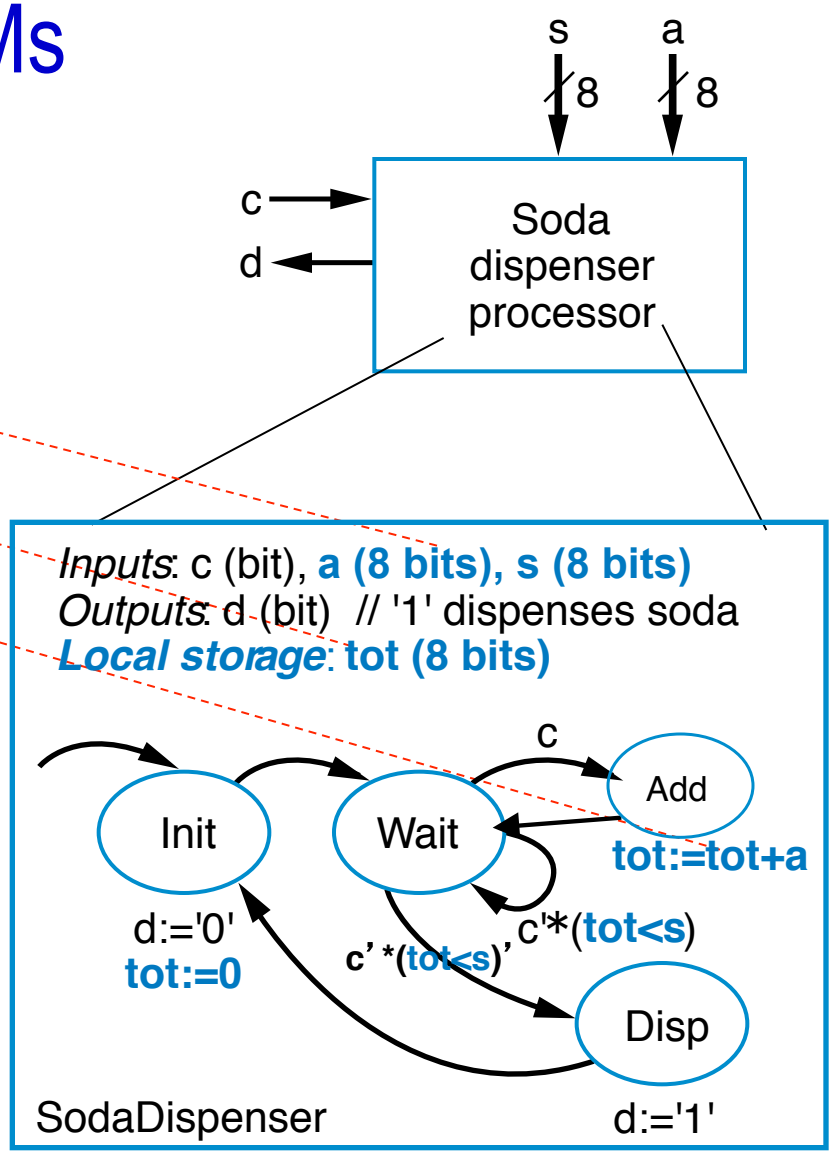
- Which of the following makes the FSM design of this problem difficult?
  - A. 8-bit input/output
  - B. Tracking the current total
  - C. All of the above





HLSMs

- High-level state machine (HLSM) extends FSM with:
  - Multi-bit input/output
  - Local storage
  - Arithmetic operations
- Conventions
  - Numbers:
    - Single-bit: '0' (single quotes)
    - Integer: 0 (no quotes)
    - Multi-bit: “0000” (double quotes)
  - == for equal, := for assignment
  - Multi-bit outputs *must* be registered via local storage
  - // precedes a comment



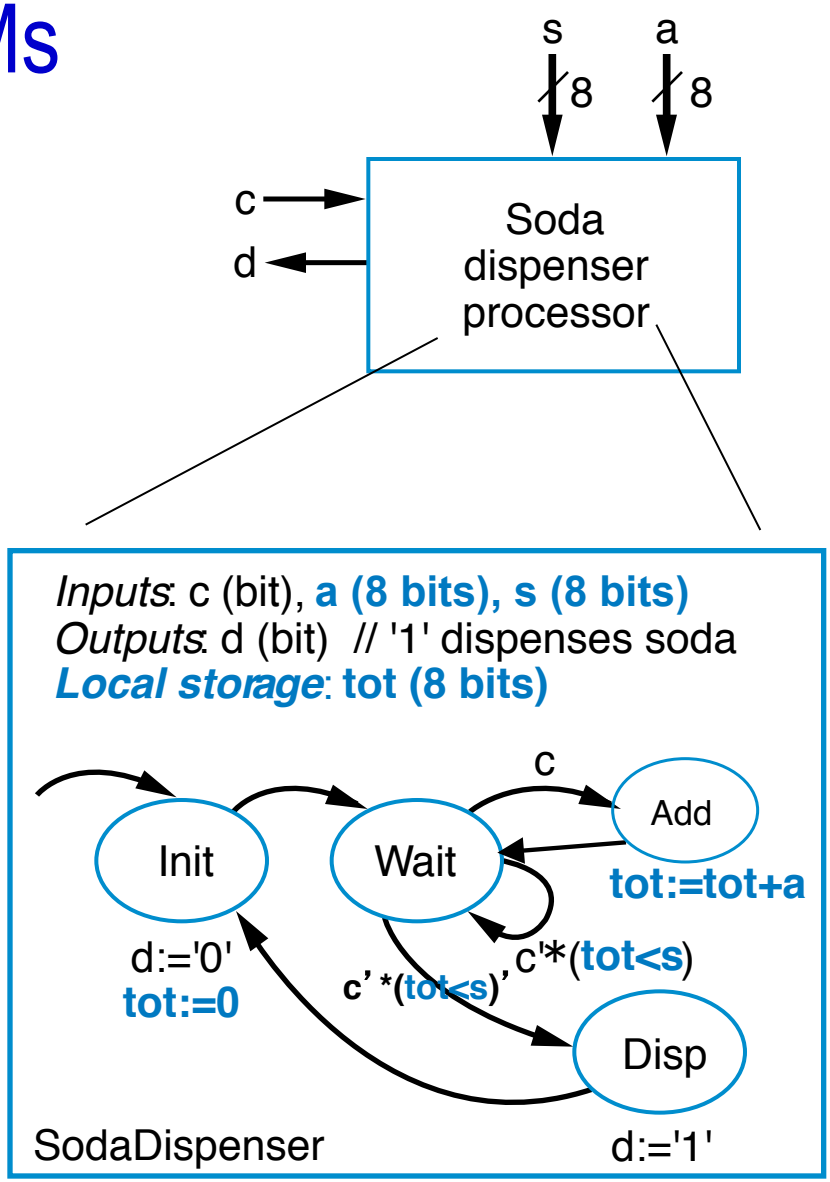
HLSMs

Q: How does the HLSM differ from the FSM for this problem?

A. The HLSM requires storing data, but the FSM doesn't

B. The FSM required storing the state but the HLSM doesn't

C. Implementing the HLSM requires registers, registers are not required to implement the FSM

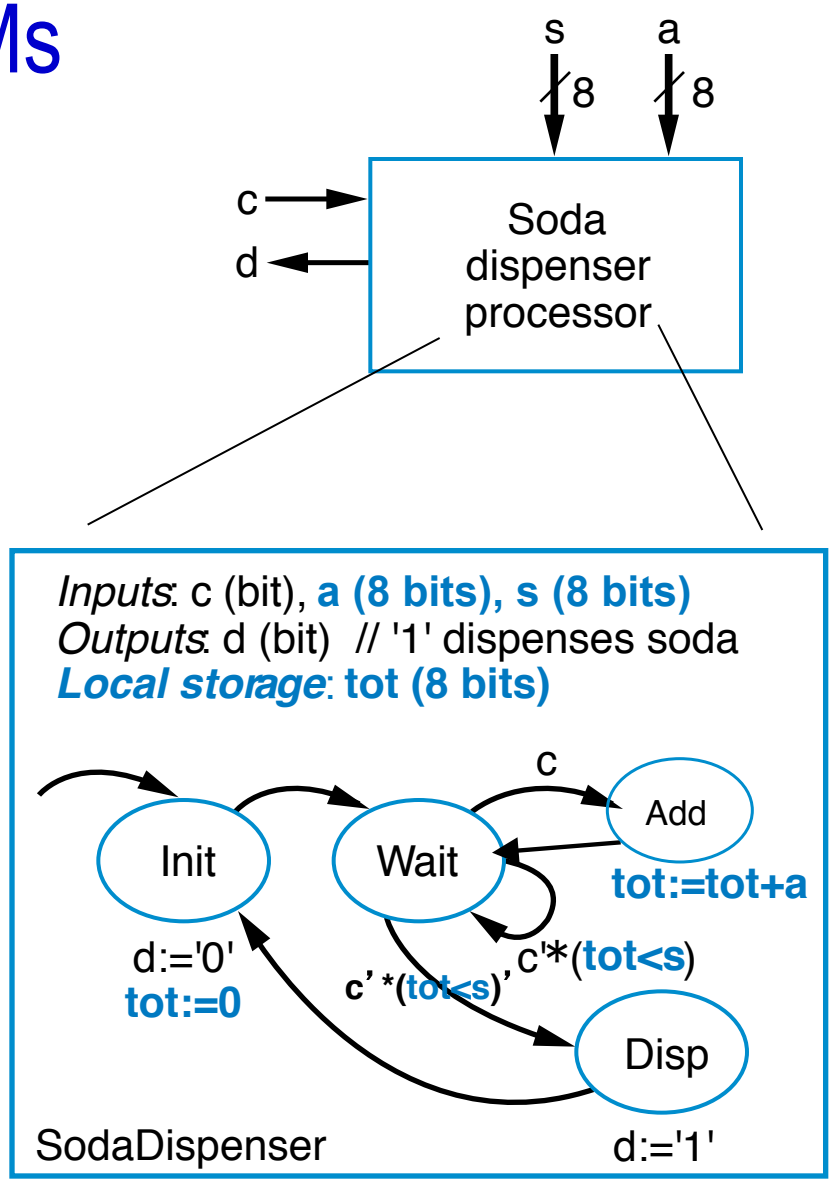


HLSMs

Q: Which of the following is common between HLSMs and FSMs?

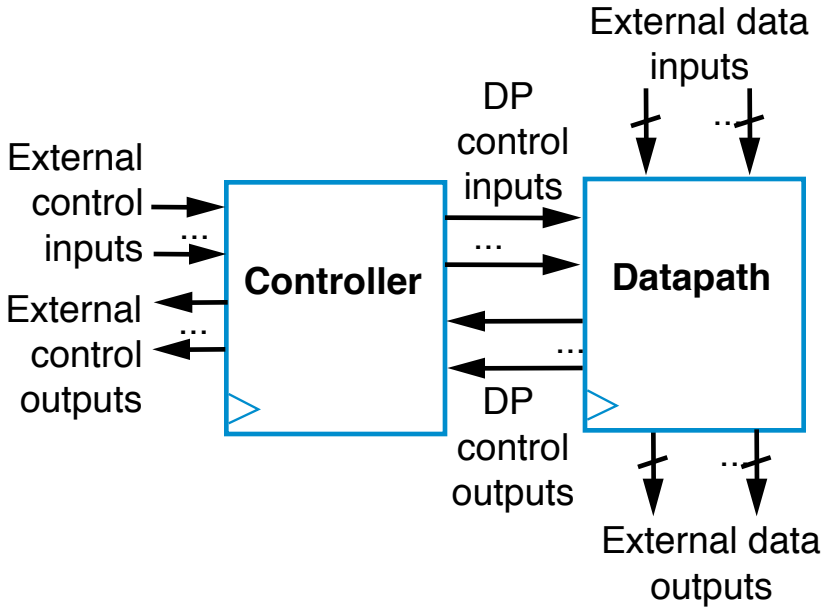
A. Transitions happen at the rising edge of the clock

B. They both have external data and control inputs and outputs



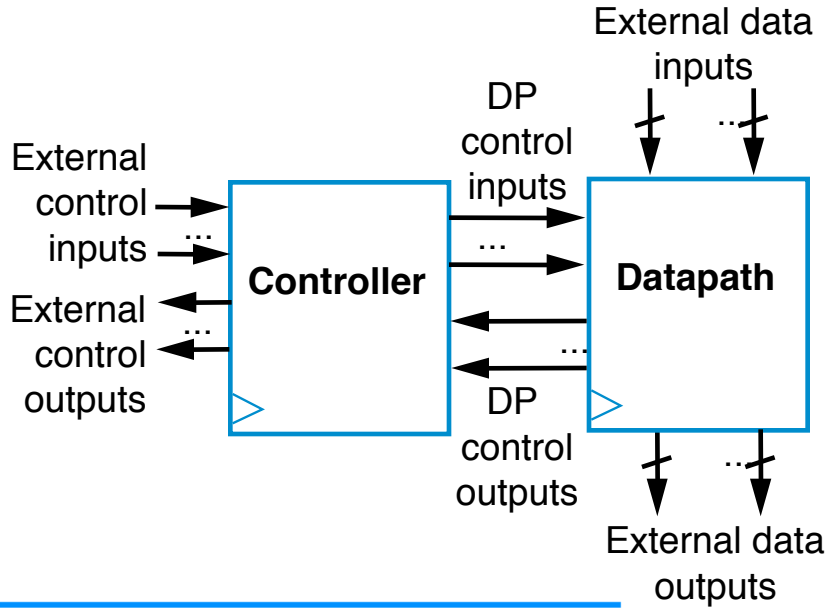
RTL Design Process

- Capture behavior
- Convert to circuit
  - High-level architecture (datapath and controlpath)
  - Datapath capable of HLSM's data operations
  - Controller to control datapath



RTL Design Process

- Capture behavior
- Convert to circuit
  - High-level architecture (datapath and controlpath)
  - Datapath capable of HLSM's data operations
  - Controller to control datapath

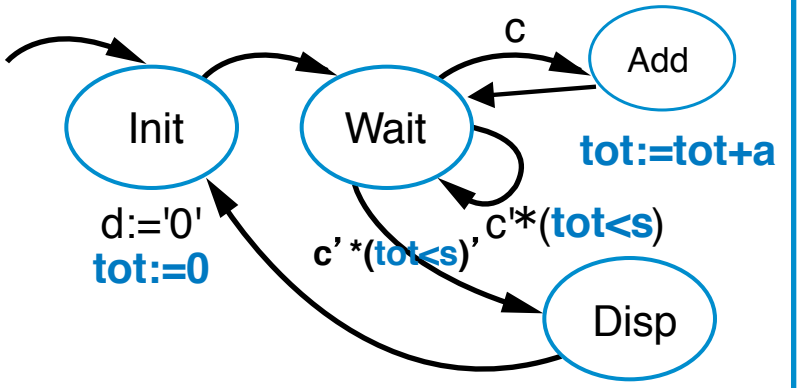


		Step	Description
Step 1: Capture behavior		*Capture a high-level state machine*	Describe the system's desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is "high-level" because the transition conditions and the state actions are more than just Boolean operations on single-bit inputs and outputs.
Step 2: Convert to circuit	**2A**	*Create a datapath*	Create a datapath to carry out the data operations of the high-level state machine.
	2B	*Connect the datapath to a controller*	Connect the datapath to a controller block. Connect external control inputs and outputs to the controller block.
	2C	*Derive the controller's FSM*	Convert the high-level state machine to a finite-state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath.

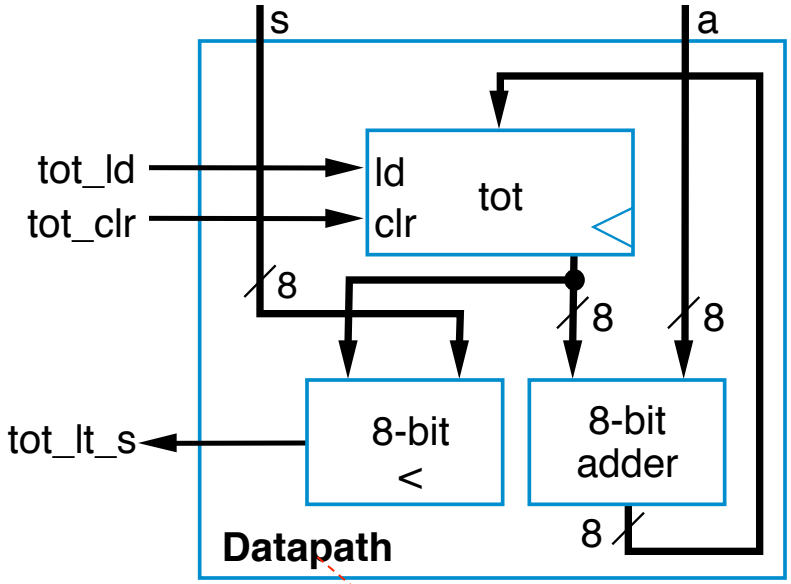
Design datapath (2A)

Step 1

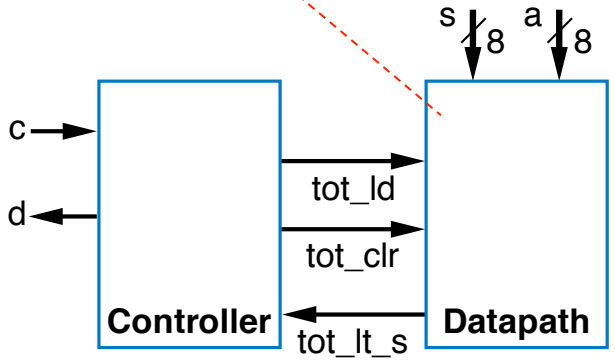
Inputs: c (bit), **a (8 bits), s (8 bits)**
Outputs: d (bit) // '1' dispenses soda
***Local storage*: tot (8 bits)**



What do we need for DP?

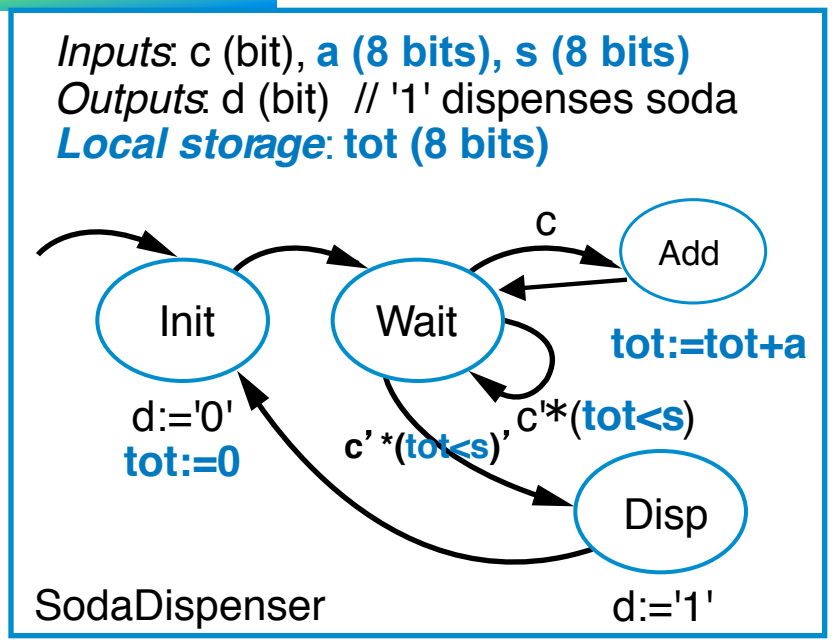


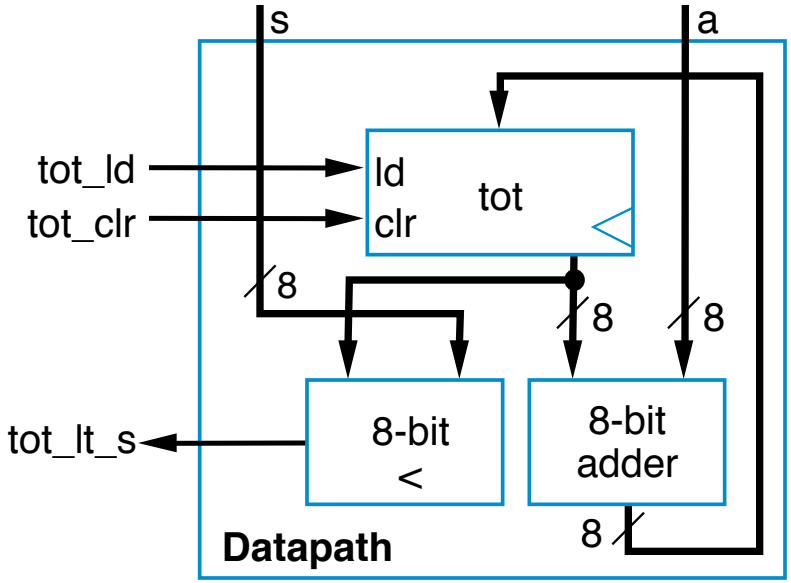
Step 2A



Step 2B

Design the datapth (2A)





Step 2A

Q: According to the current design, under which of the following conditions does the register output 'tot' change?

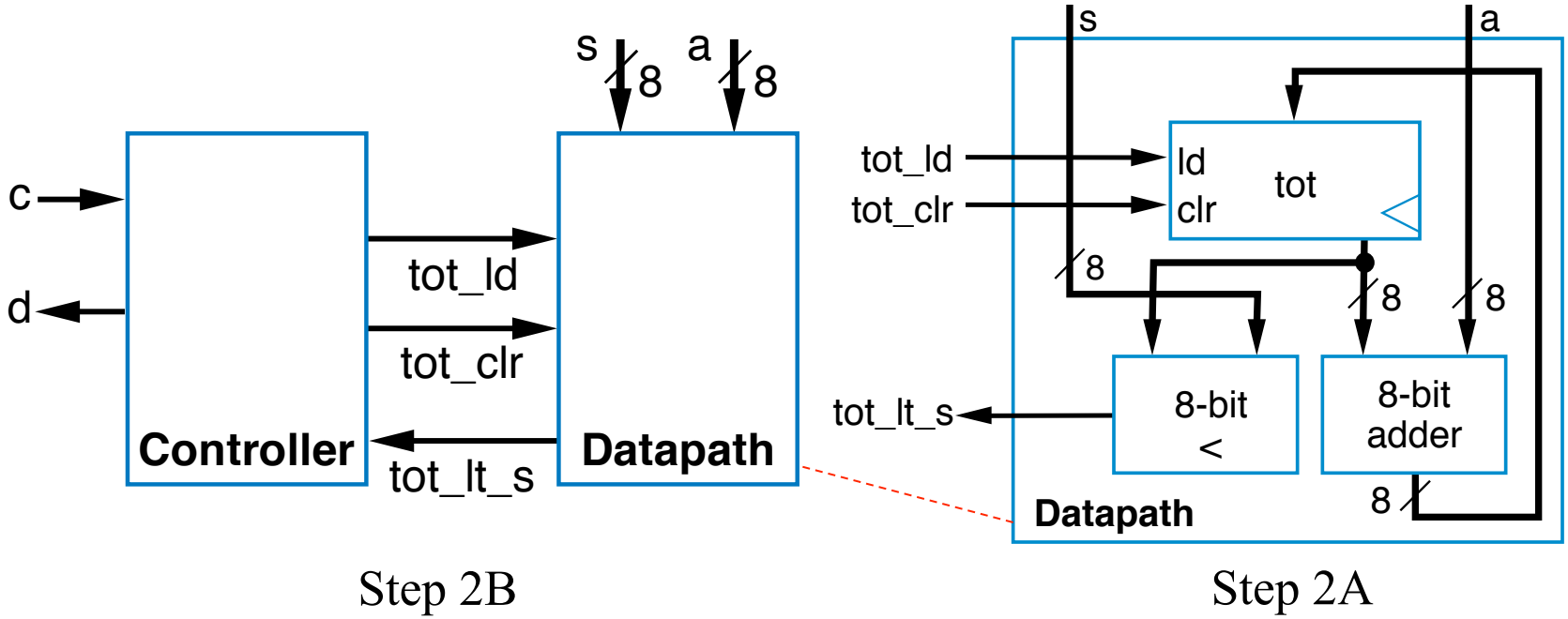
A. Whenever the value of the coin inserted ('a') changes

B. Whenever the cost of the soda ('s') changes

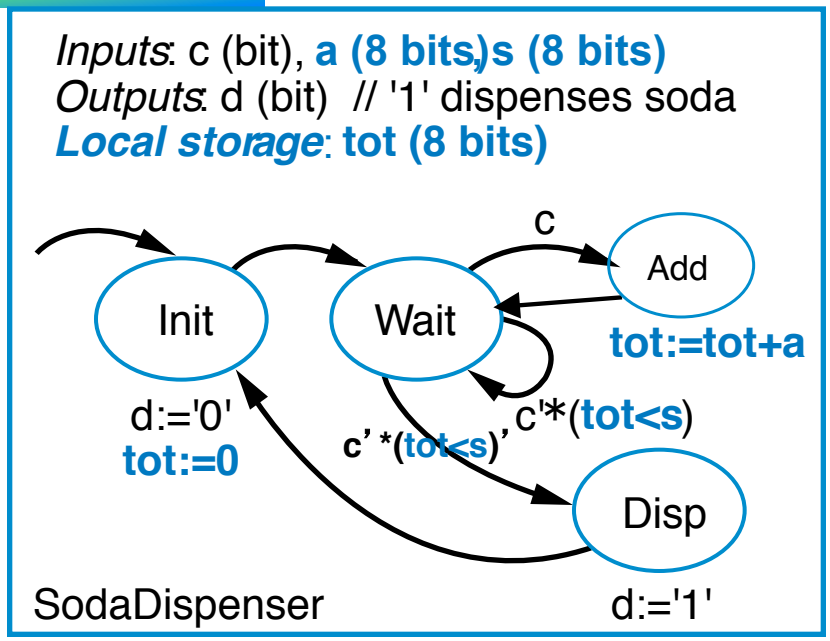
C. When the signal tot_ld becomes high

D. When the signal tot_clr becomes high

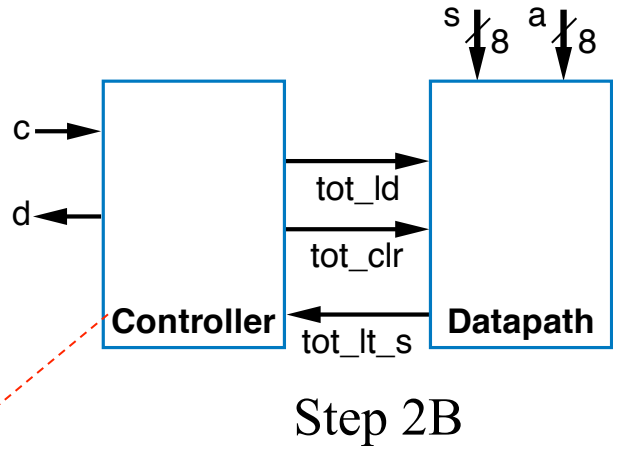
Connect data path to controller(2B)



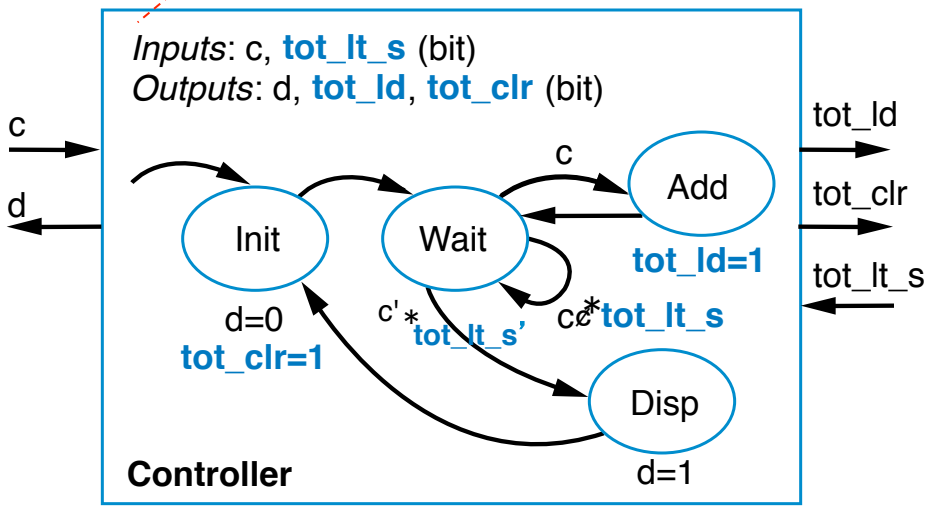
Design control path FSM



Step 1



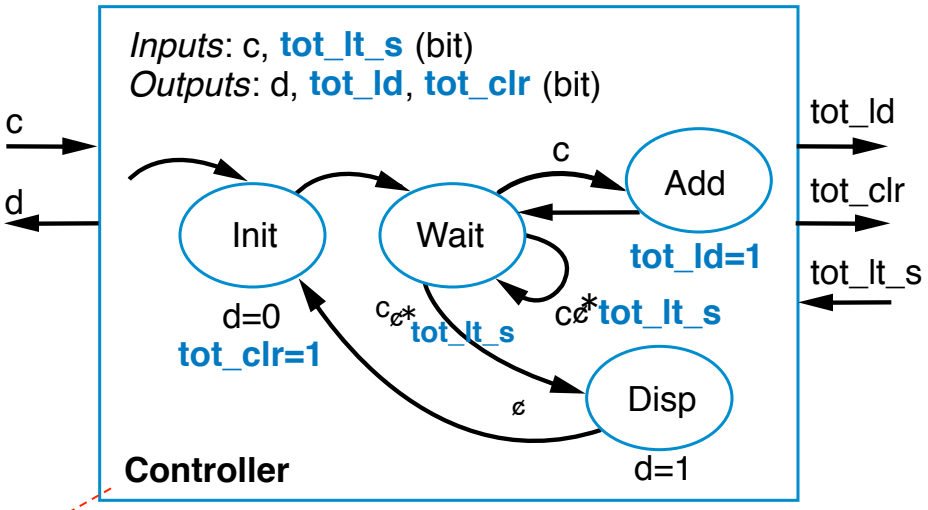
Step 2B



Step 2C

Implement controller FSM

	s1	s0	c	tot_lt_s	n1	n0	d	tot_ld	tot_clr
Init	0	0	0	0	0	1	0	0	1
	0	0	0	1	0	1	0	0	1
	0	0	1	0	0	1	0	0	1
	0	0	1	1	0	1	0	0	1
Wait	0	1	0	0	1	1	0	0	0
	0	1	0	1	0	1	0	0	0
	0	1	1	0	1	0	0	0	0
	0	1	1	1	1	0	0	0	0
Add	1	0	0	0	0	1	0	1	0
			• • •				• • •		
Disp	1	1	0	0	0	0	1	0	0
			• • •				• • •		



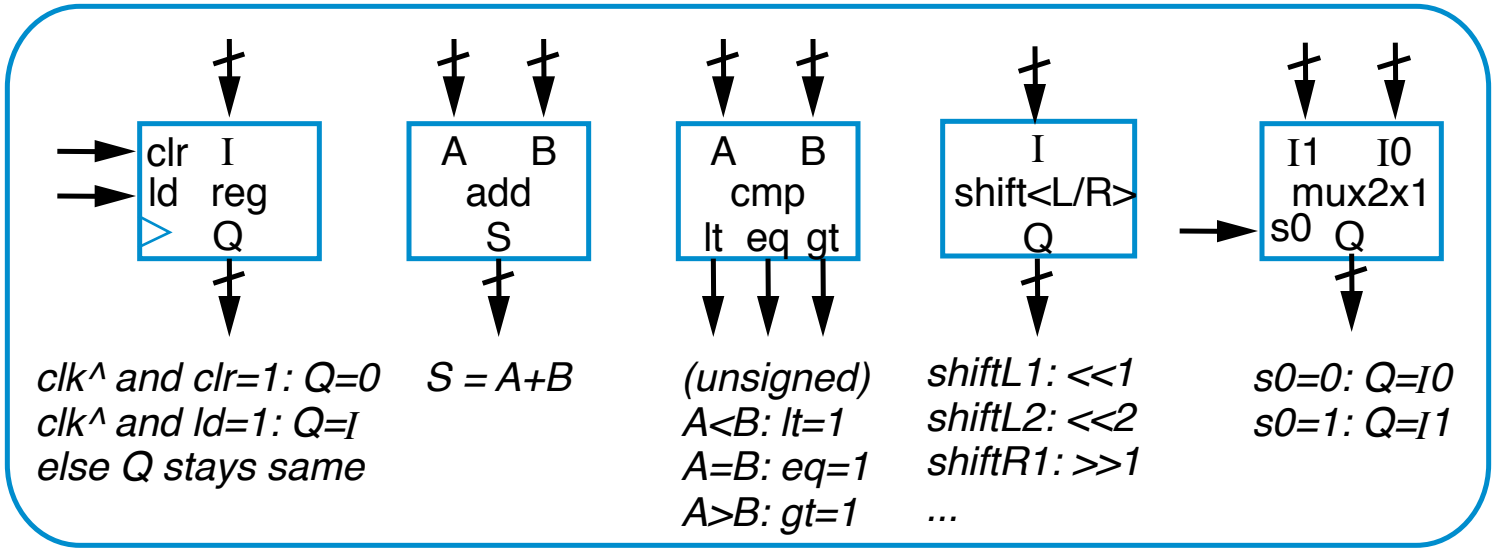
Step 2C

Use controller (FSM) design process from previous lectures to complete the design

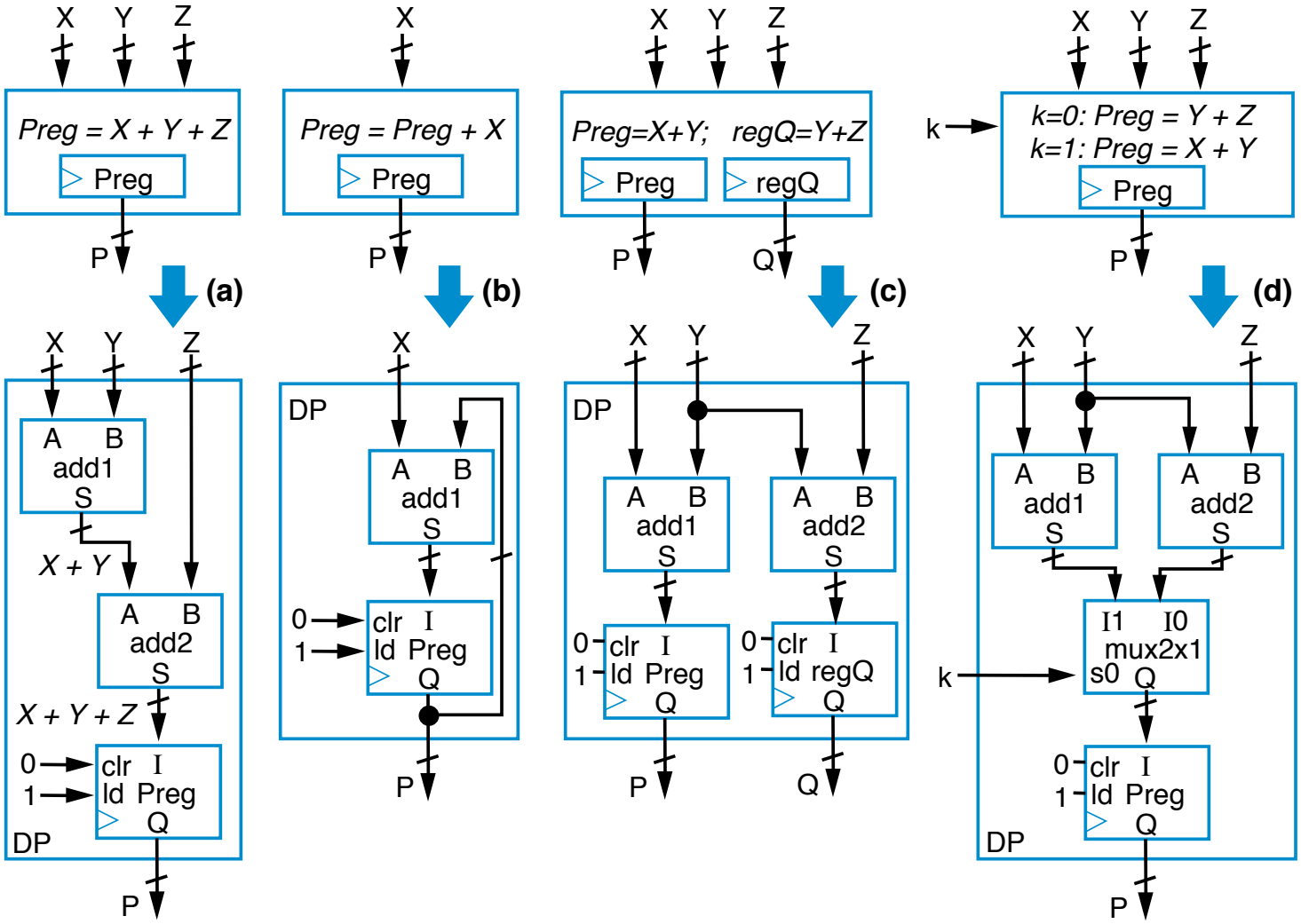
a

RTL Design Process—Step 2A: Create a datapath

- Need component library from which to choose

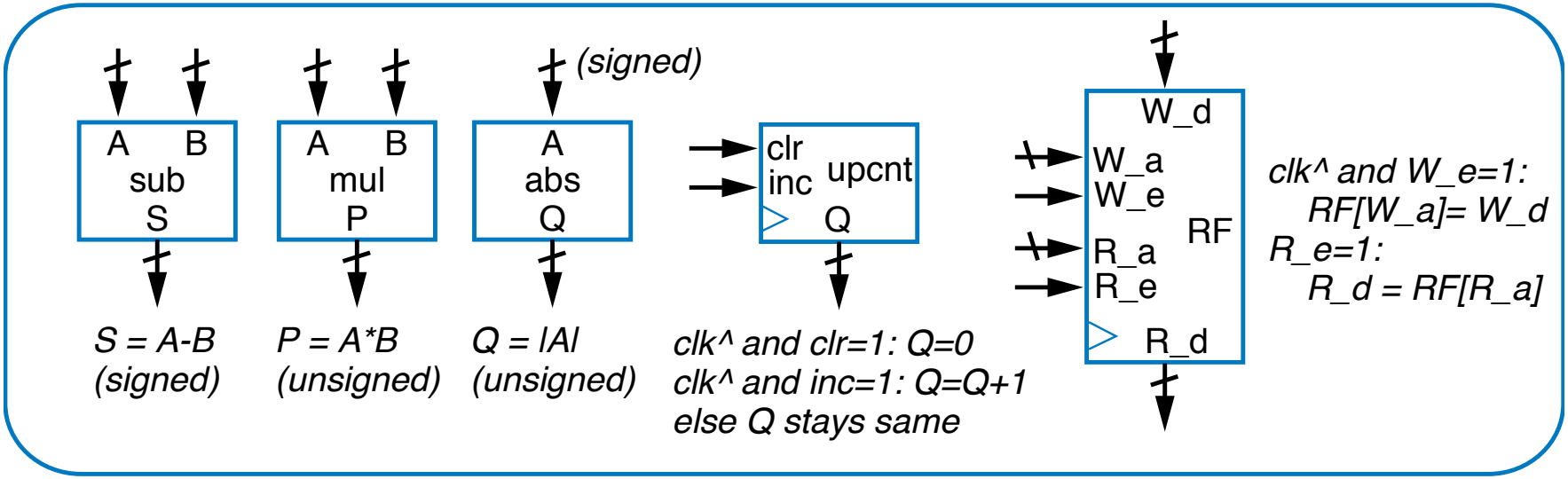


Step 2A: Create a Datapath—Simple Examples



More RTL Design

- Additional datapath components



Summary

- Modern digital design involves creating processor-level components
- High-level state machines
- RTL design process
  - 1. Capture behavior: Use HLSM
  - 2. Convert to circuit
    - A. Create datapath B. Connect DP to controller C. Derive controller FSM