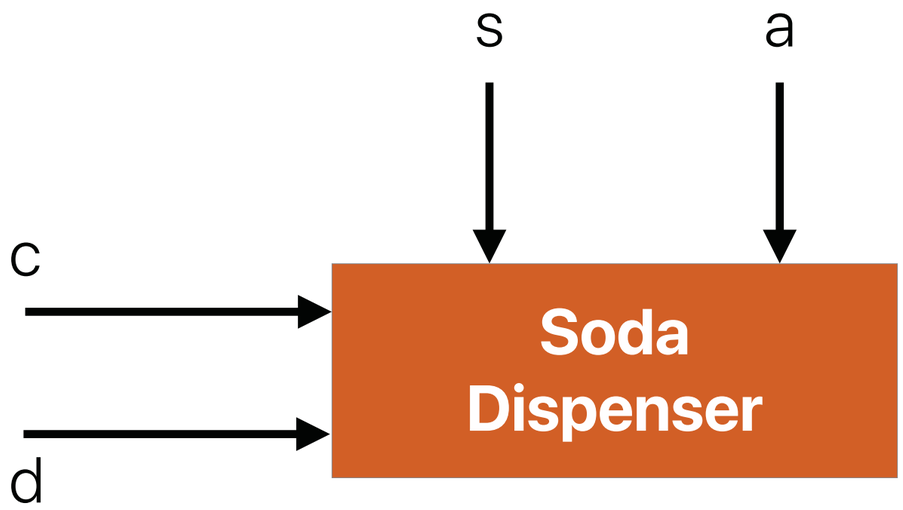
High-level State Machines & RTL Design

Prof. Usagi

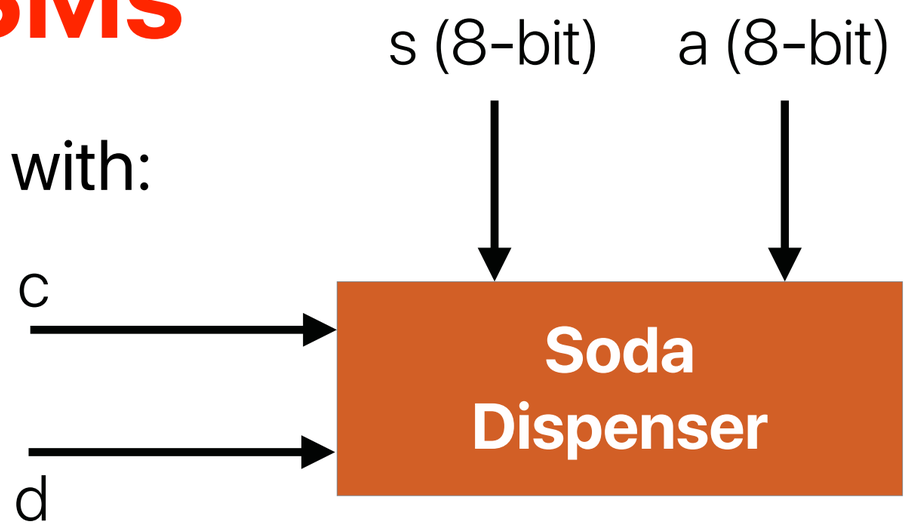
High-Level State Machine

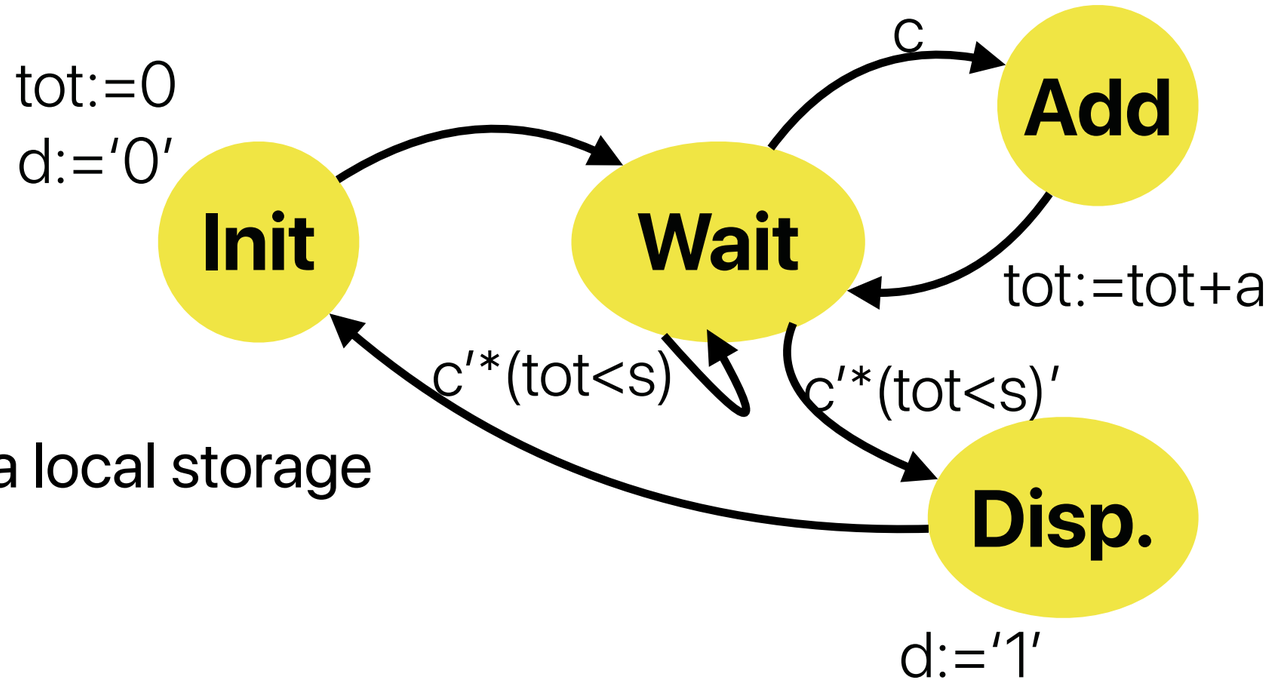
- Some behaviors may be too complex to describe by using classical FSMs
- Soda dispenser
  - c: bit input, 1 when coin deposited
  - a: 8-bit input: value of the deposited coin
  - s: 8-bit input: cost of a soda
  - d: bit output, processor sets it to 1 when total value of deposited coins equals or exceeds cost of a soda



Benefits of HLSMs

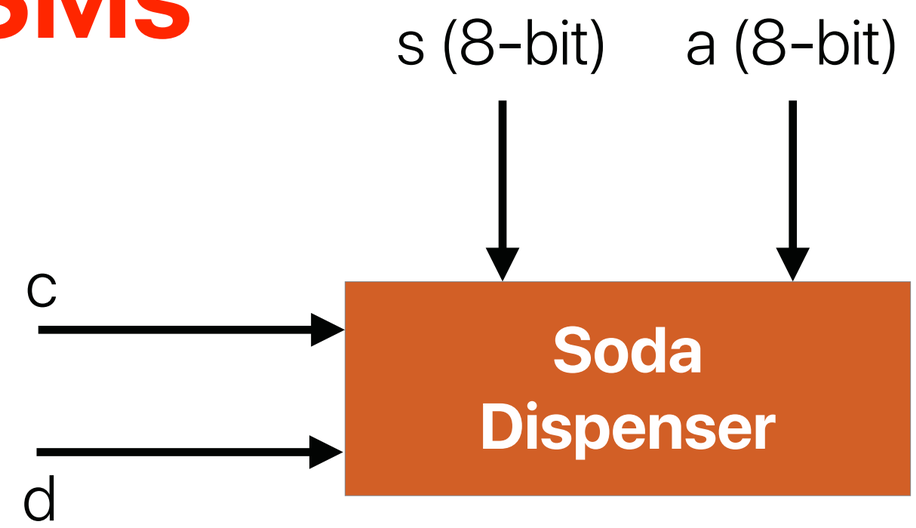
- High-level state machine (HLSM) extends FSM with:
  - Multi-bit input/output
  - Local storage
  - Arithmetic operations
- Conventions
  - Numbers:
    - Single-bit: '0' (single quotes)
    - Integer: 0 (no quotes)
    - Multi-bit: “0000” (double quotes)
  - == for comparison equal
  - Multi-bit outputs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

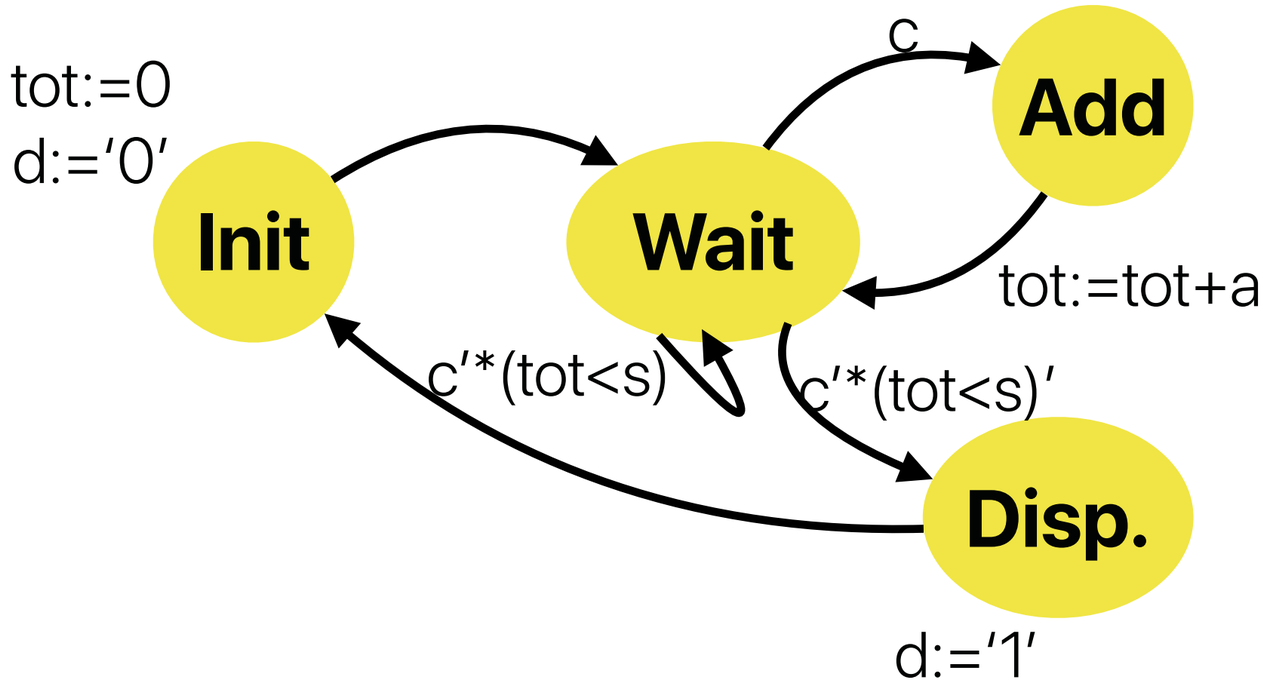




Benefits of HLSMs

- High-level state machine (HLSM) extends FSM with:
  - Multi-bit input/output
  - Local storage
  - Arithmetic operations
- Conventions
  - Each transition is implicitly ANDed with a rising edge of the clock
  - Any bit output not explicitly assigned a value in a state is implicitly assigned to 0. This convention does not apply for multibit outputs
  - Every HLSM multibit output is registered





RTL(Register Transfer Level) Design

RTL Design Process

- Step 1: 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 input and outputs
  - Recommendations:
    - Always list all inputs, outputs and local registers on top of your HLSM diagram
    - Clearly specify the size in bits of each of them
    - On states: update the value of registers, update of outputs
    - On transitions: express conditions in terms of the HLSM inputs or state of the internal values and arithmetic operations between them.

RTL Design Process

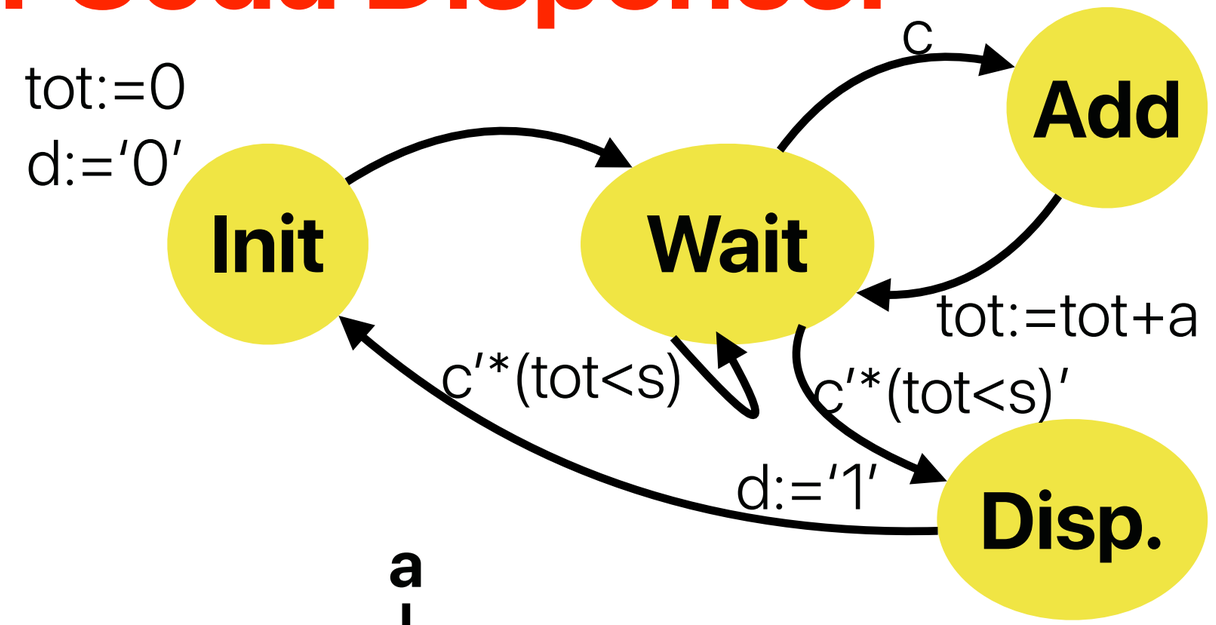
- Step 2: Convert it to a circuit
  - Create a datapath
    - Create a datapath to carry out the data operations of the high level state machine
    - Elements of your datapaths can be registers, adders, comparators, multipliers, dividers, etc.
  - Connect the datapath to a controller
    - Connect the datapath to a controller block.
    - Connect the external control inputs and outputs to the controller block.
    - Clearly label all control signals that are exchanged between the datapath and the controller
  - 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
- Final Step Implement the FSM as a state register and logic

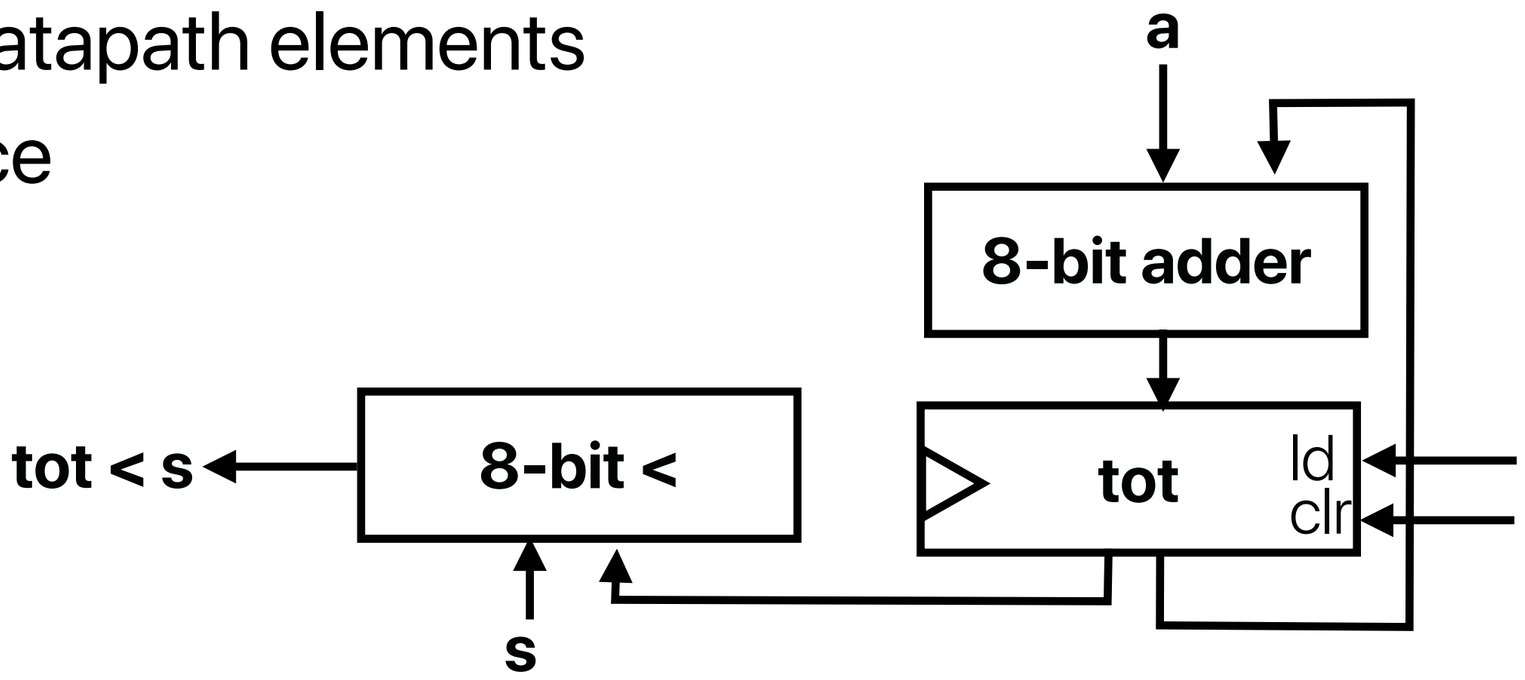
RTL Design Summary

- Capture the behavior with HLSM
- Convertit to a circuit
  - High-level architecture (datapath and control path)
  - Datapath capable of HLSM's data operations
  - Design controller to control the datapath

Create Datapath for Soda Dispenser

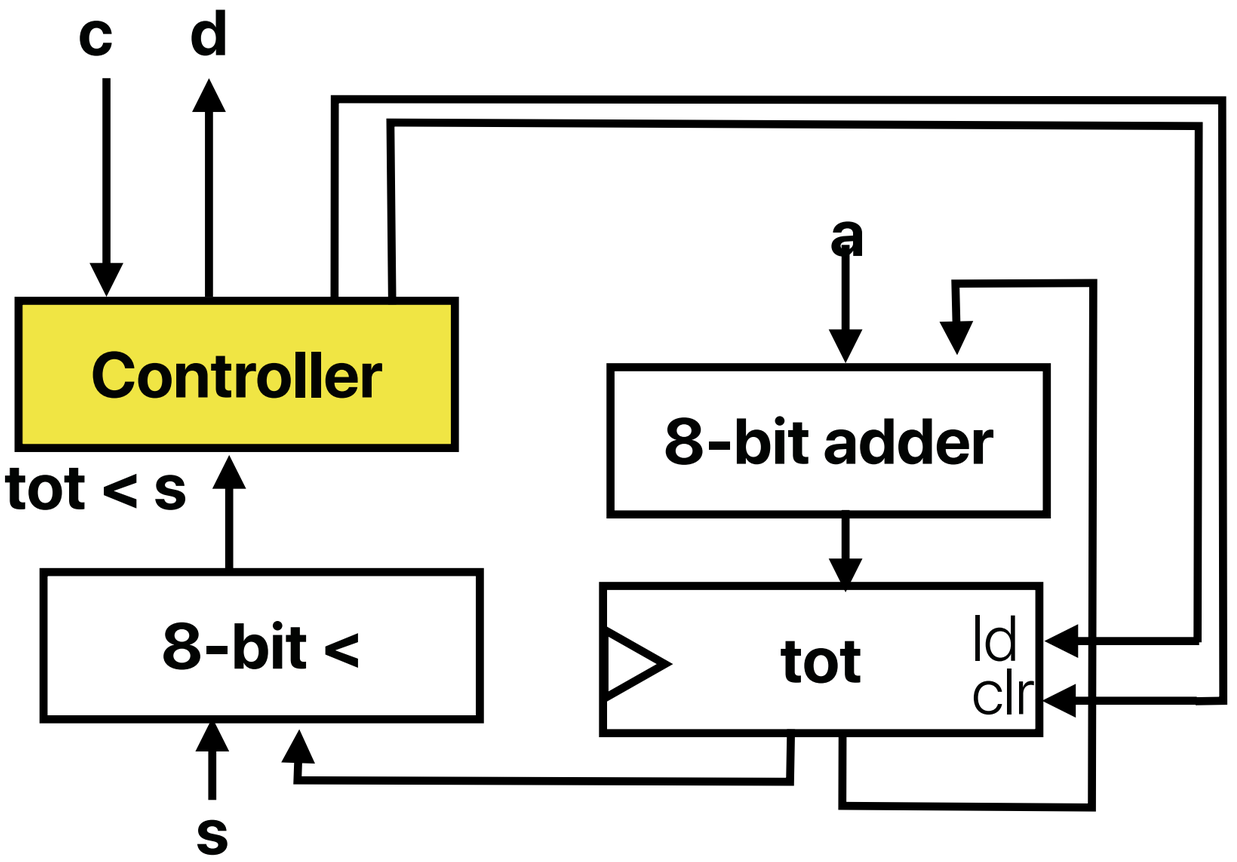
- Register: tot
- Comparator: to compare tot and s
- Adder: to update tot = tot + a
- Connect datapath elements
- I/O interface





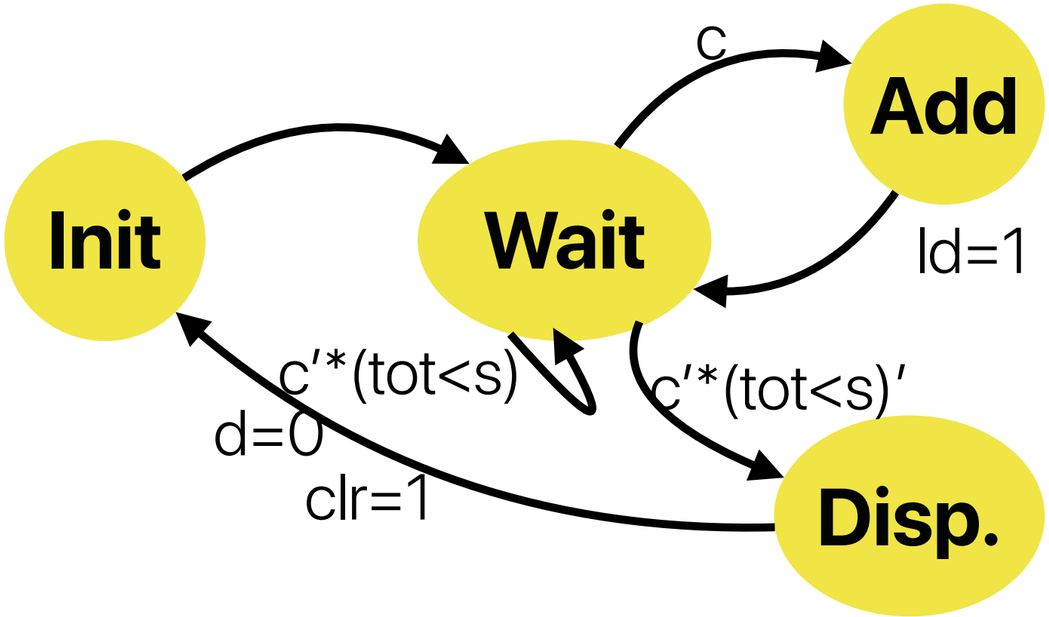
Connect Datapath to a Controller

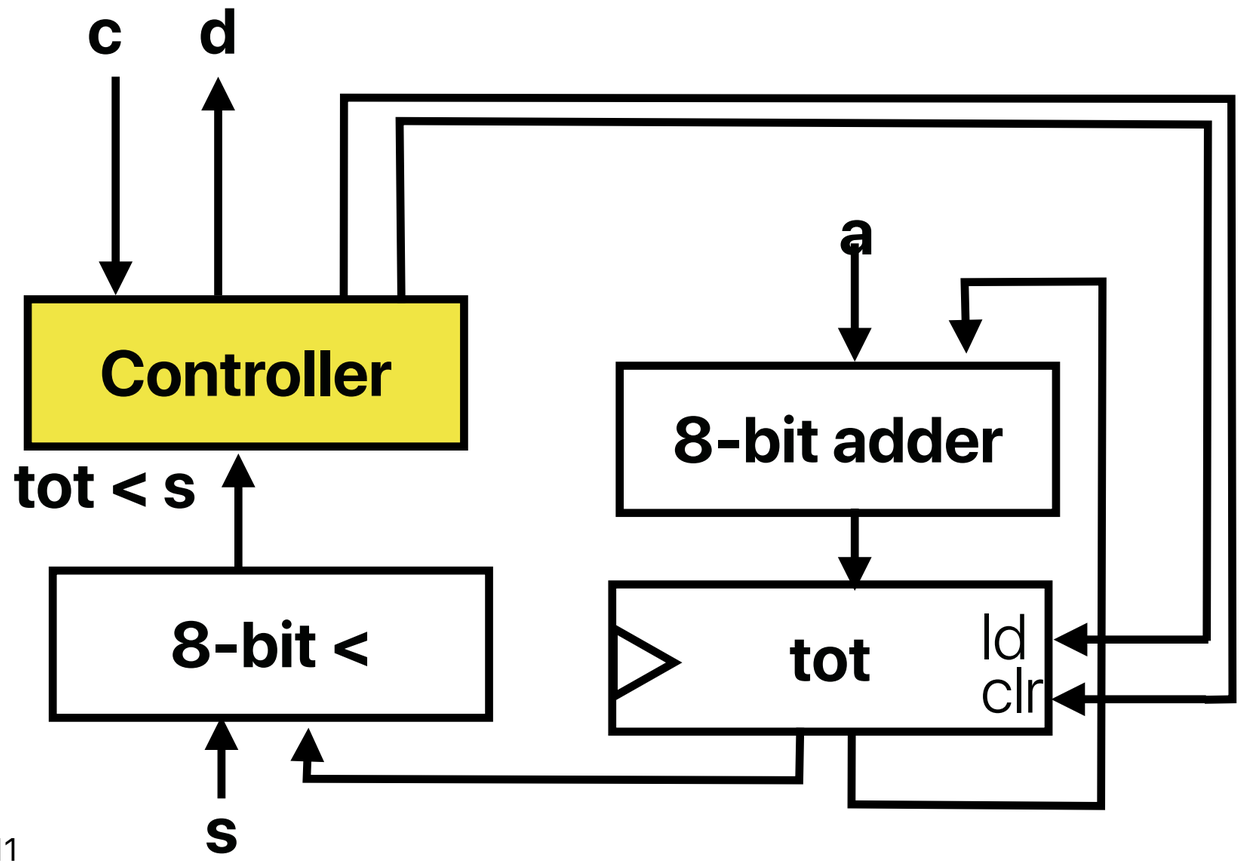
- Controller's inputs
  - External input c (coin detected)
  - Input from datapath comparator's output, which we named tot<s
- Controller's outputs
  - External output d (dispense soda)
  - Outputs to datapathto load and clear the tot register



Derive the Controller's FSM

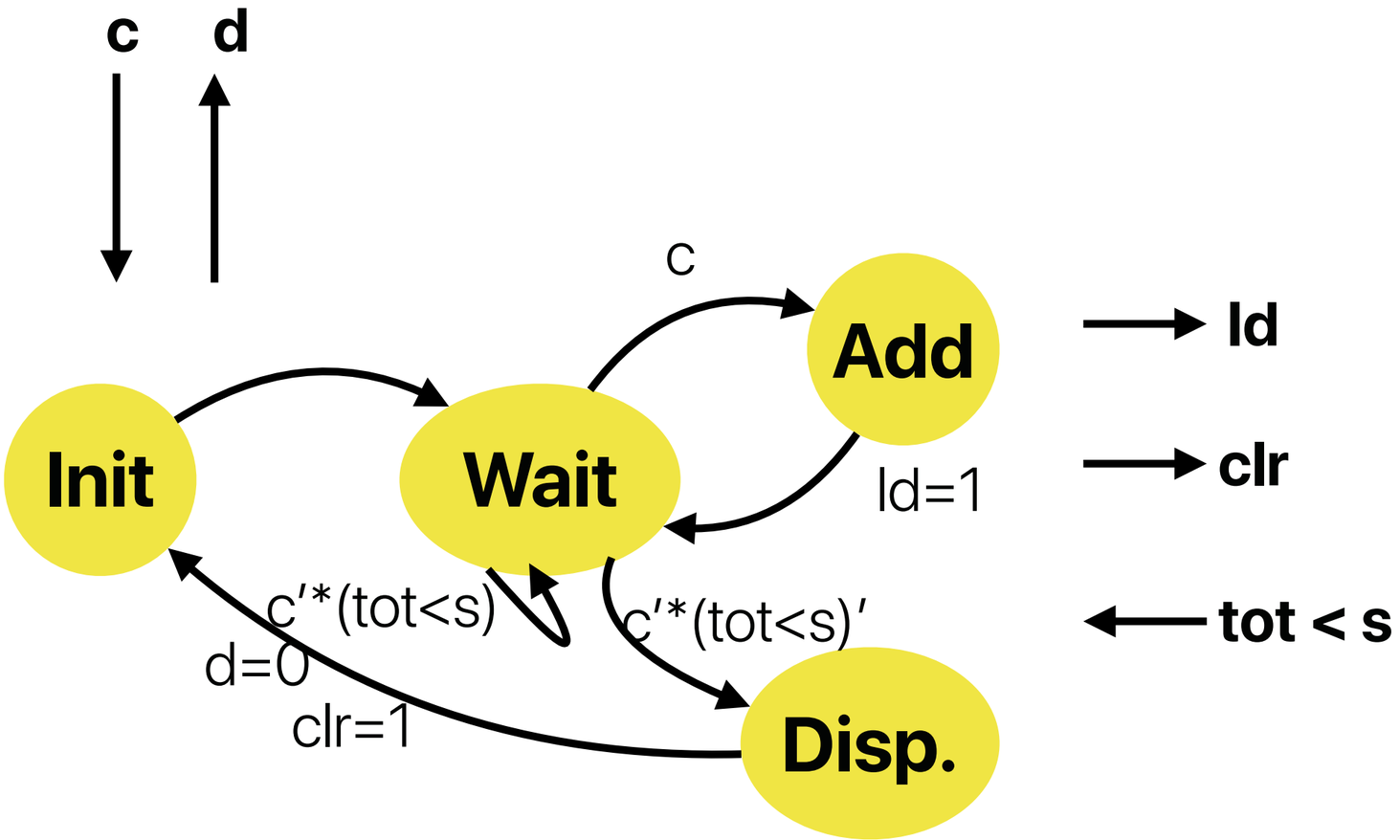
- FSM has the same states and arcs as HLSM
- Replace all references to the data elements in the HLSM with appropriate control signals & values





Final Step: Implement the controller FSM

- Implement the FSM as a state register and logic



Control path of a microprocessor

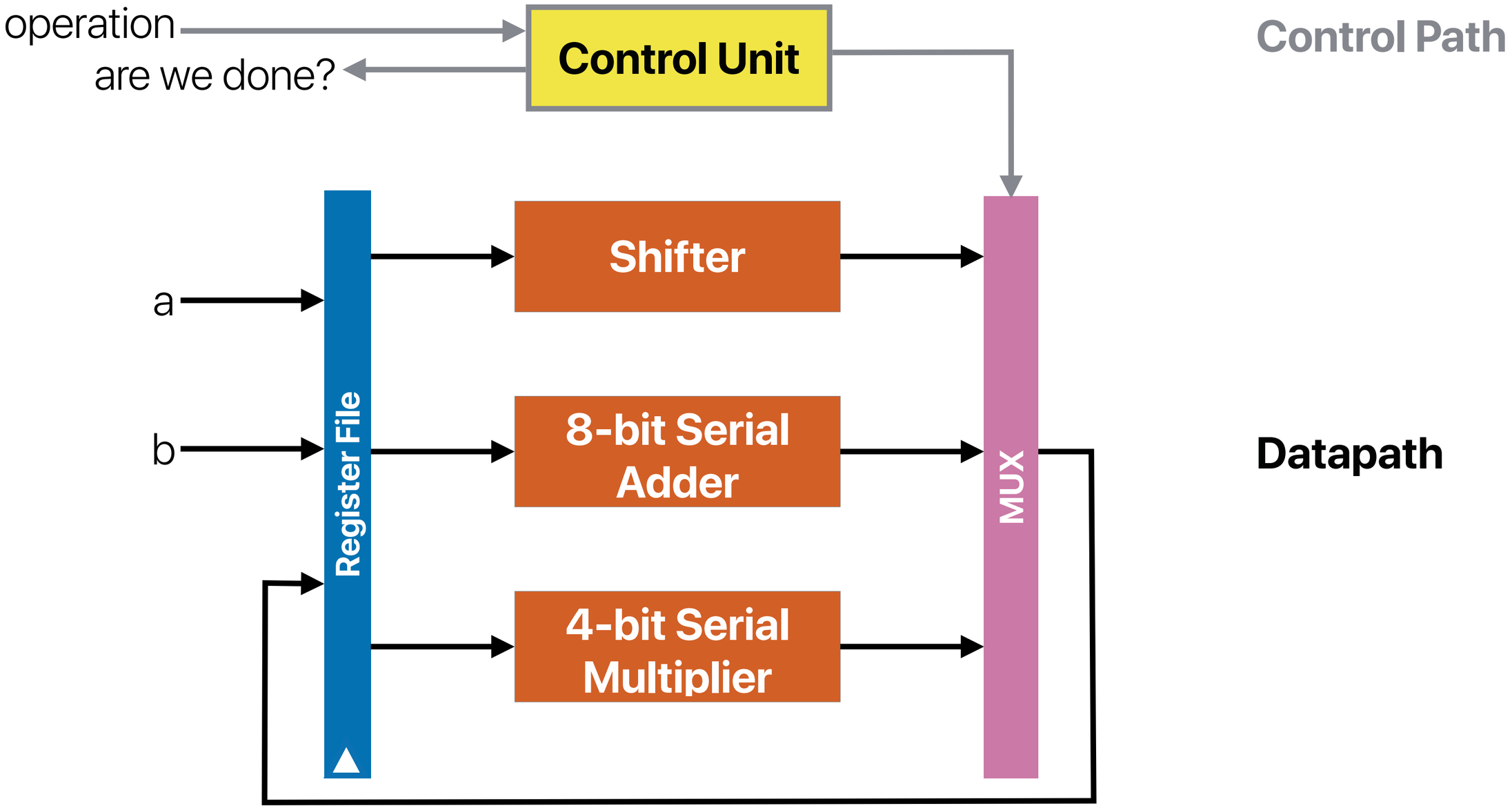
Let’s put all things together!

- We have learned all datapath components for an ALU!
  - Register
  - Shifter
  - Adders
  - Multiplier
- Processor has only one clock generator
  - Each datapath component has a different latency

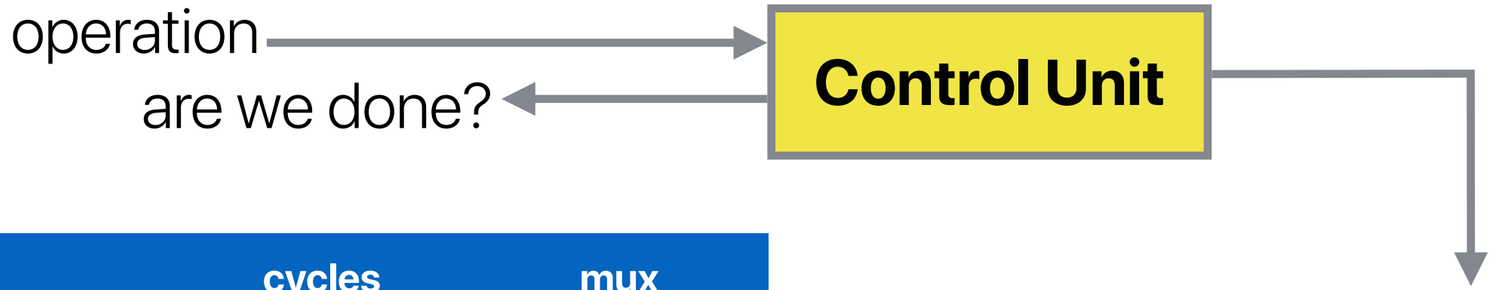
Let’s put all things together!

- We have learned all datapath components for an ALU!
  - Register
  - Shifter
  - Adders
  - Multiplier
- Processor has only one clock generator
  - Each datapath component has a different latency
  - We have make some of the above “serial”

The HLSM for the processor's control



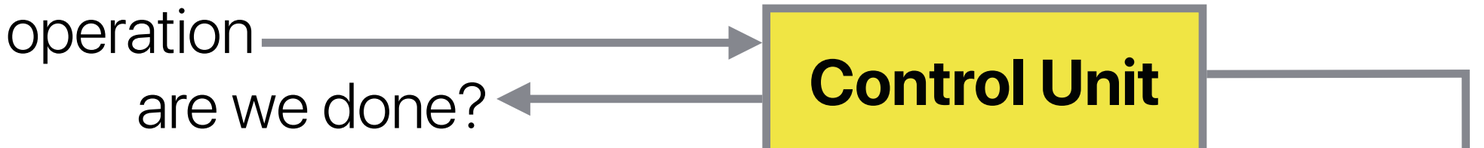
The HLSM for the processor's control



Control Path

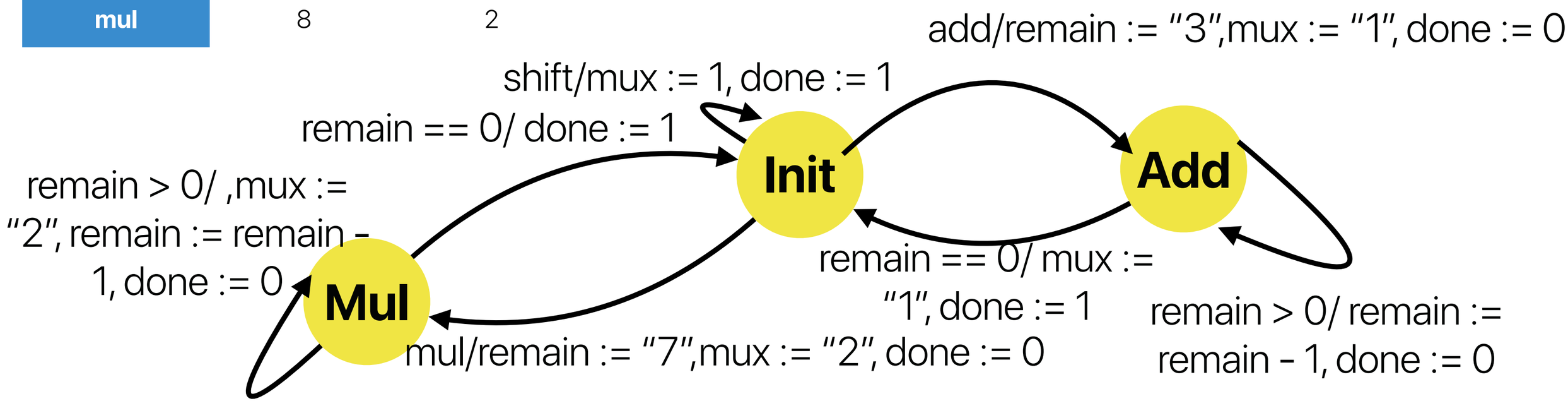
	cycles	mux
shift	1	0
add	4	1
mul	8	2

The HLSM for the processor's control



Control Path

	cycles	mux
shift	1	0
add	4	1
mul	8	2



Electrical
Computer Science
Engineering

120A

つづく