Chapter 5

Register Transfer Level (RTL) Design
High Level State Machine (HLSM)
Memory
Chapters 5

High Level Sequential Behavior

- FSM can be used to capture sequential behavior using bit inputs
- HLSM can be used to capture more complex logic involving multi-bit variables

Processors

- 'Processor' is a generic term for a circuit designed using RTL principles
  - Programmable Processor – A generic processor designed to run programs.
  - Custom Processor – Specialized design that implements specific functionality
- Processors can be designed using HLSM

HLSM

- HLSM extends FSMs with features that make it possible to capture more complex behaviors.
  - Multi-bit data inputs and outputs – assume unsigned unless specified
  - Local Storage
  - Arithmetic operations – Add, Multiply, etc.

HLSM Conventions

- HSLMs will follow these conventions
  - All inputs, outputs, and local storage are defined at the top of the HLSM diagram
  - Registered values change on rising clock edges
  - Transition bits are implicitly ANDed with rising clock edge
  - Any output bit not explicitly assigned is 0
  - Any registered value not explicitly assigned holds its value

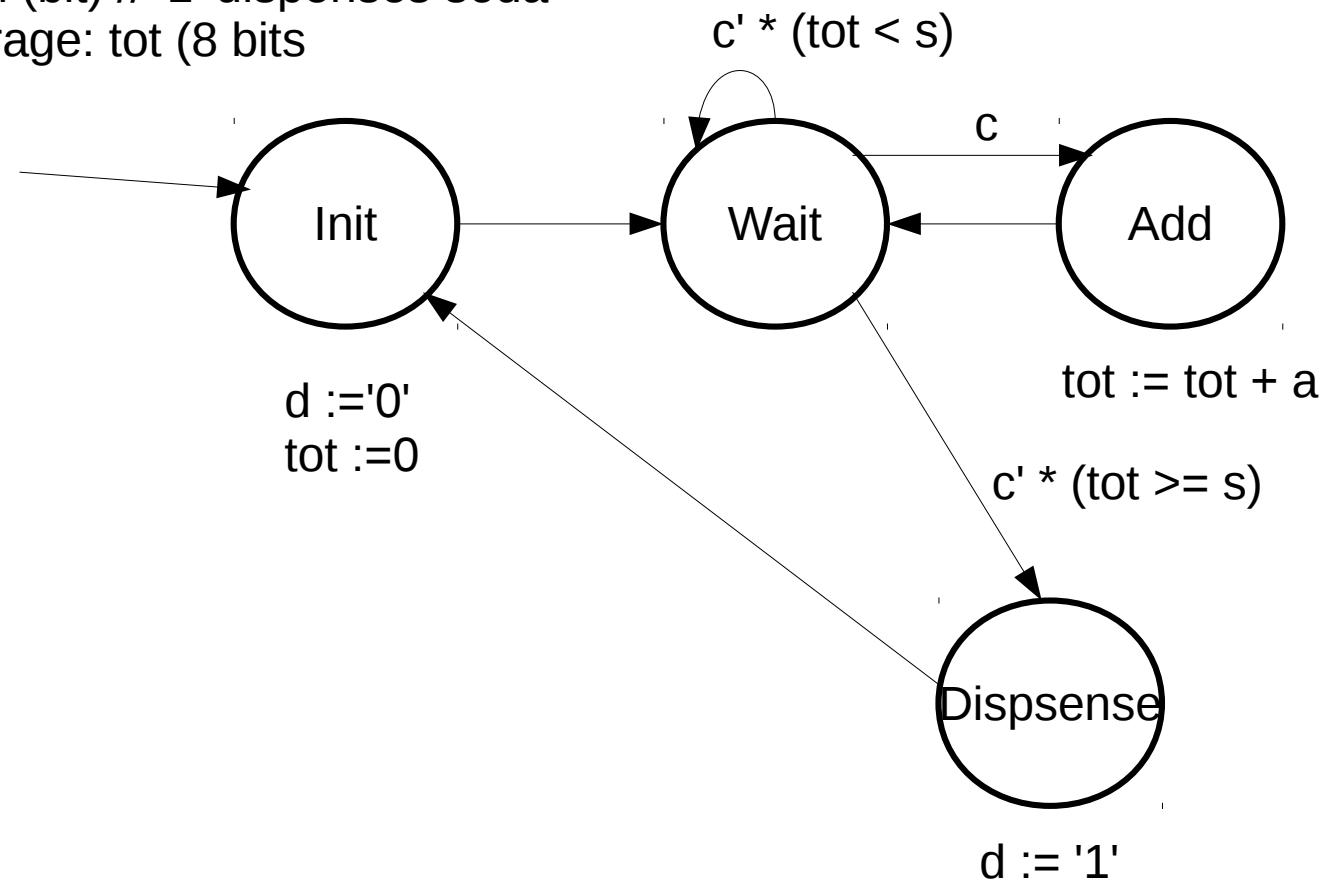
HLSM Conventions (continued)

- Bits are designated by surrounding them with single quotes, integers have no quotes
  - '1' is a bit value, 1 is an integer
  - “:=” assigns a value to a variable, “==” compares two values.
  - “//” defines a comment, just as in C or C++

HLSM Example

- Soda Dispenser Processor

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



HLSM Warnings

- Clocked storage items are not updated in the same clock cycle that their control signals are set. Must wait for rising clock edge
- Note: The register control bits set up in a state prepare the value on the register inputs to be captured on the next clock cycle!

8-bit Up/Down Counter

- Design an HLSM for an 8-bit up/down counter
  - What are the inputs?
  - What are the outputs?
  - What is the local storage (if any)?
  - Draw the HLSM

RTL Design Process

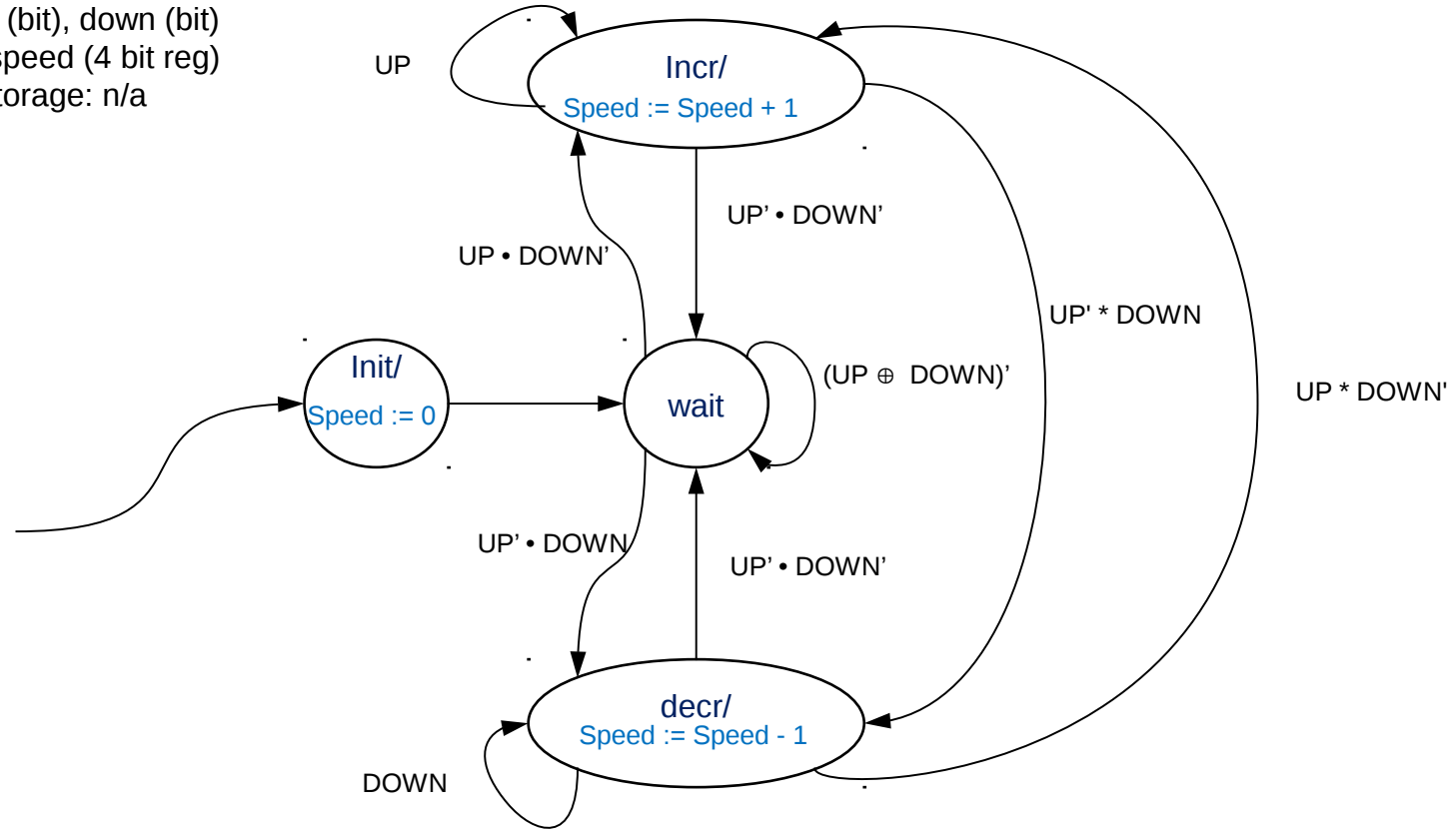
- Capture a HLSM
- Convert to a Circuit
  - Create a datapath circuit
  - Derive the controller's FSM
  - Design controller circuit
  - Connect the datapath to the controller

Treadmill Speed Controller

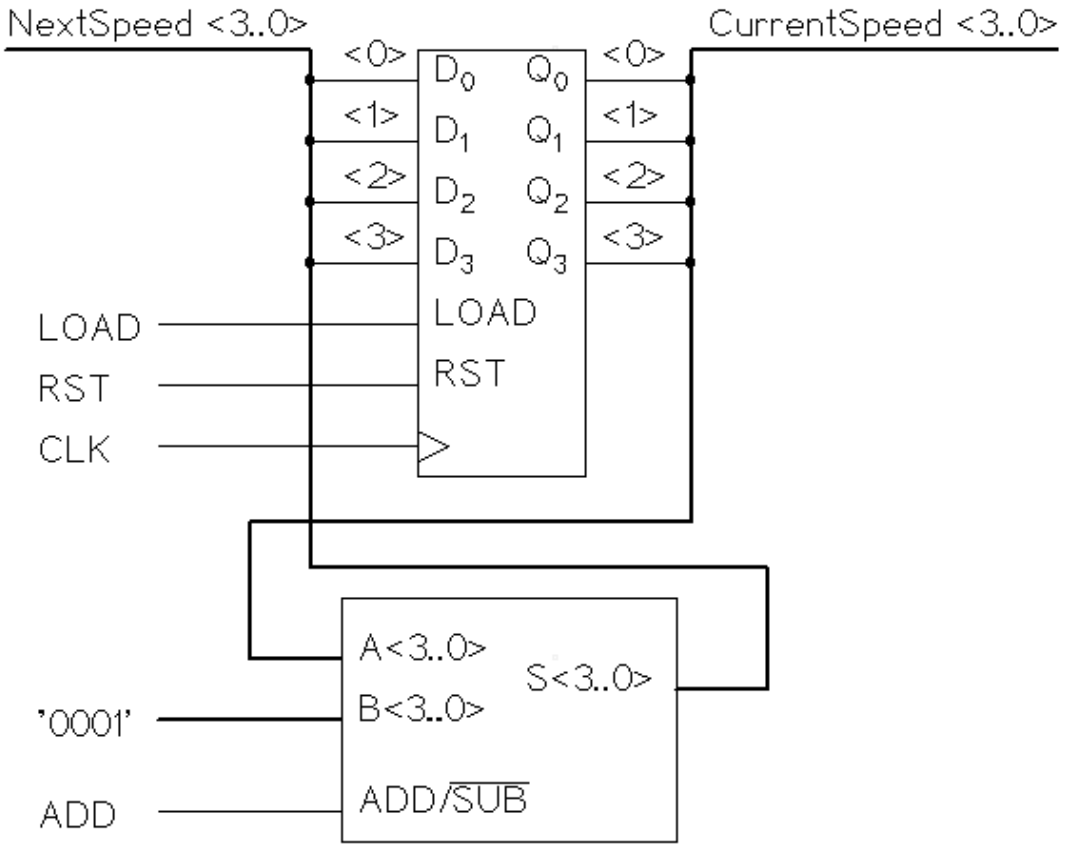
- Design a system to control the speed of the conveyor on a treadmill.
- Speed is a 4-bit value that is controlled by two buttons.
  - Up button increases speed by one
  - Down button decreases speed by one
  - If both are pushed, no change in speed
- Speed must initialize to zero upon start up.

Treadmill HLSM

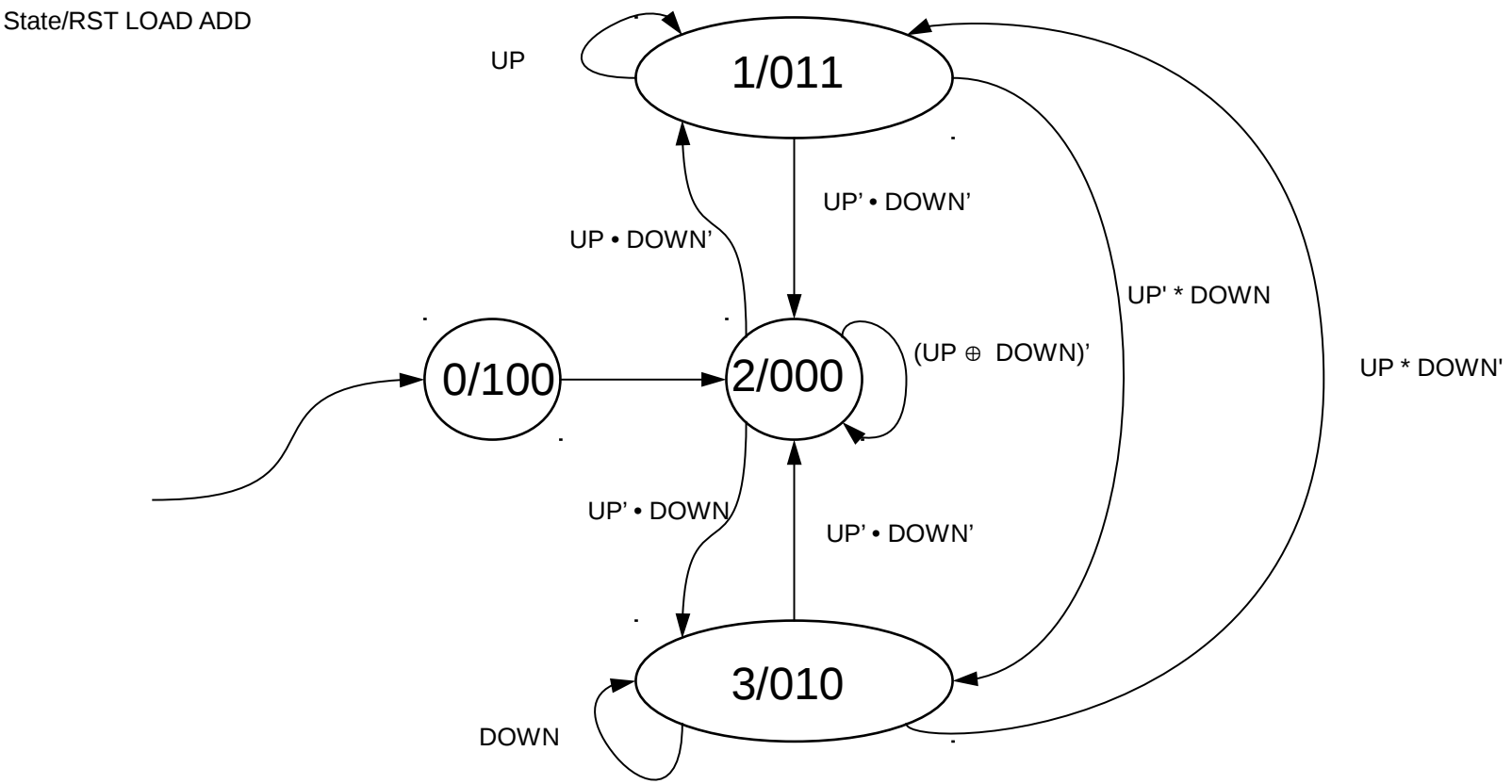
Inputs: up (bit), down (bit)
Outputs: speed (4 bit reg)
Internal Storage: n/a



Treadmill Datapath



Treadmill FSM



Treadmill Circuit

- Finish designing the controller from the Treadmill FSM.
- Connect the controller circuit to the Treadmill Datapath

Types of memory

- Two major types of memory
- Volatile – When power to the device is removed the contents in memory are lost
- Non-Volatile – Contents in memory remain if power is removed

Volatile memory

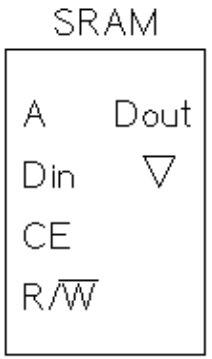
- Usually made form Random Access Memory RAM
- Simple RAM
- First In First Out FIFO.
- Last In First Out LIFO.
- Dual-port RAM – two independent parallel accesses to memory.

Volatile Memory

- RAM has further subdivisions
- Static RAM – Write data at an address location and the memory doesn't change until the next write at same address.
- Dynamic RAM – Same as Static RAM except memory refresh is required. Data in memory decays over a short time and must be refreshed (write the same value again).

Volatile Memory

Static RAM symbol



Static RAM

- The output is usually a three-state output.
- Sometimes the Data in and Data out share the same I/O pins.
- CE controls the output and writing.
- To write – The address and data must be set followed by the CE and R/W' signal set to false.
- To read – The address and CE must be true.

Dynamic RAM

- Similar to Static RAM
- More controls for refresh
- Address is multiplexed between row and column
- Data in and Data out share the same pins.
- Cost and package size is a major advantage.

Non-Volatile Memory

- Read Only Memory – Similar to SRAM but no data input or write signal.
- Programmable ROM – Requires special programming devices.
- Erasable PROM – Has a window to erase with UV light
- Electrically EPROM – Can be programmed in circuit. Slow write speed.

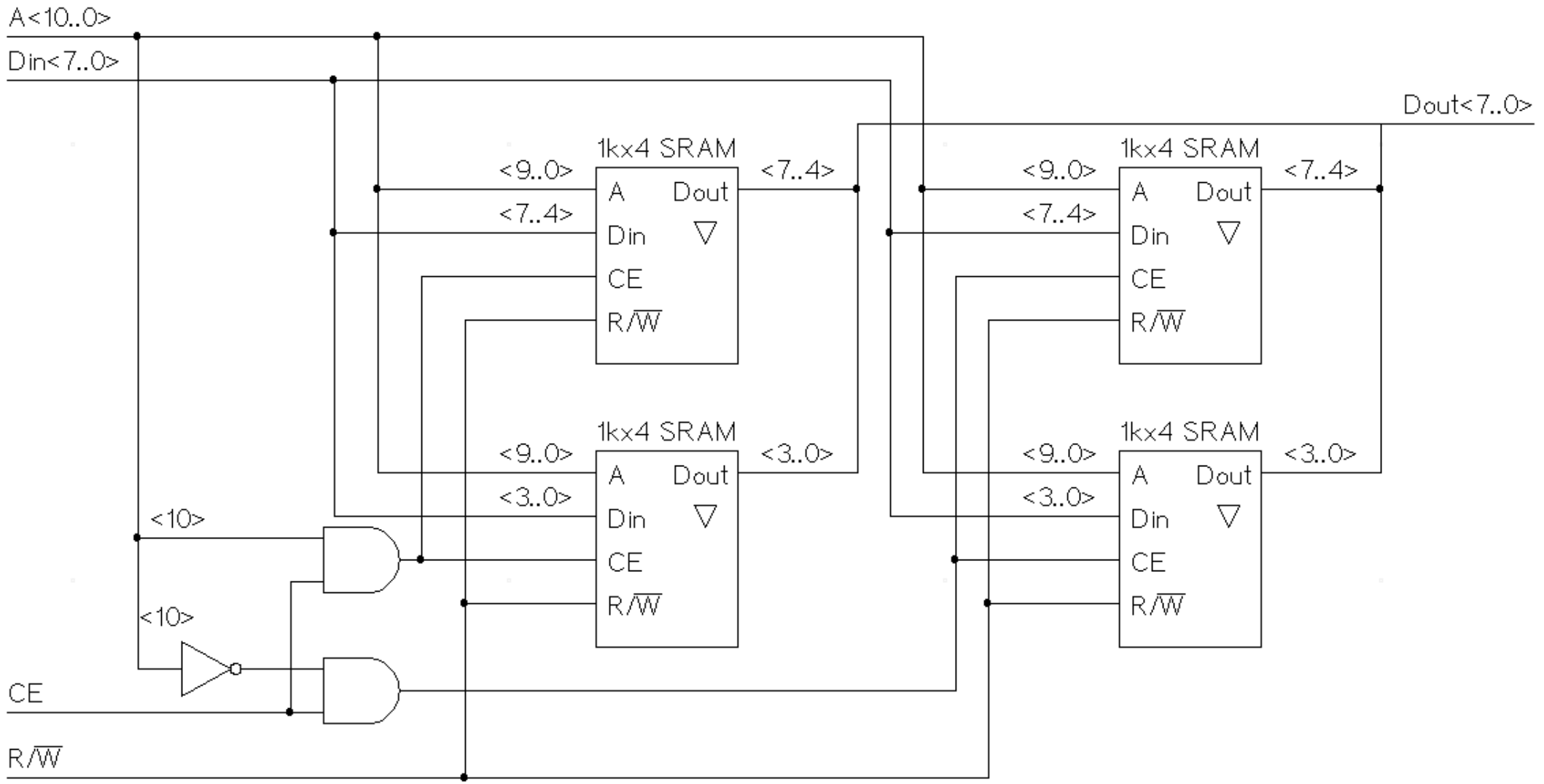
Non-Volatile Memory

- ROM's can be used for implementing logic.
  - Address in, data out.
  - Looks like a huge truth table
  - No logic optimizations
- Programmable Array Logic PAL
- Programmable Logic Array PLA
- Some have a register built-in.

Creating a larger memory

- Memory can be used to create larger memory.
- Increase the number of words
- Increase the number of bits per word

2Kx8 SRAM from 1K x 4 SRAMs



Mixed SRAM and ROM

- Parts of the address range can be ROM while other parts can be SRAM.
- Design a 4kx8 mixed ROM and SRAM memory. The least significant 1kx8 is ROM while the most significant 3kx8 is SRAM. Use 1kx8 ROM and SRAM devices and the minimum additional combinational logic.