Combinational vs Sequential Logic

Combinational Logic

- Output relies entirely on inputs
- Stateless
- Uses logicla components, like gates, as building blocks
- Examples:
  - ALU
  - Multiplexers
  - Comparator

Sequential Logic

- Output relies oninputs and state
- Stateful
- Can include feedbacks: output can help determine the next state
- Examples
  - Registers
  - Memory
  - FSM

Level vs Edge-Triggered Logic

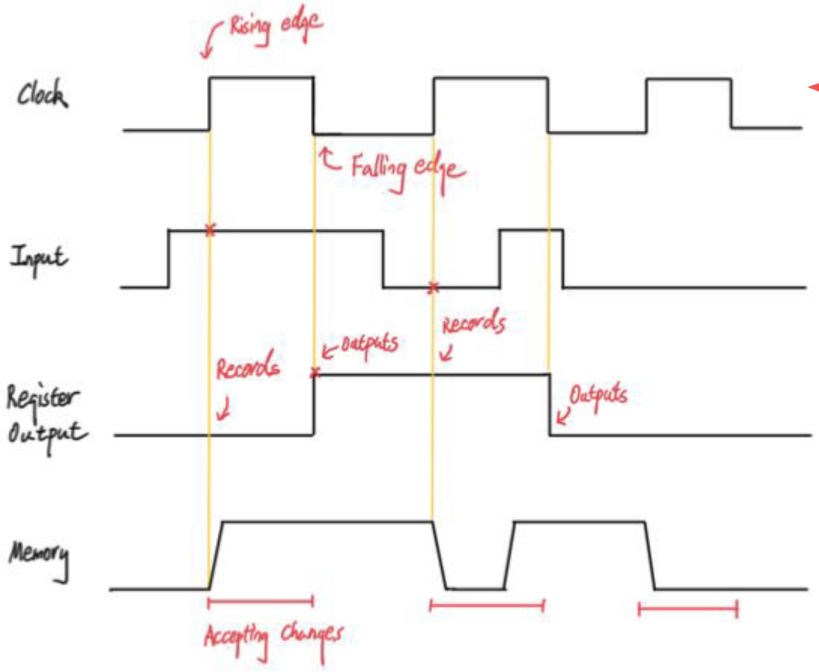
These terms describe when the output or state changes

Level

- State changes at level clock cycles when the clock is high
- Example: memory

Edge-Triggered

- State changes when the clock is falling
- Ex: registers
  - Delayed flip-flop records input at rising edge, propage state to output at falling edge
  - Outputs of registers should be considered as available only at the next clock cycle



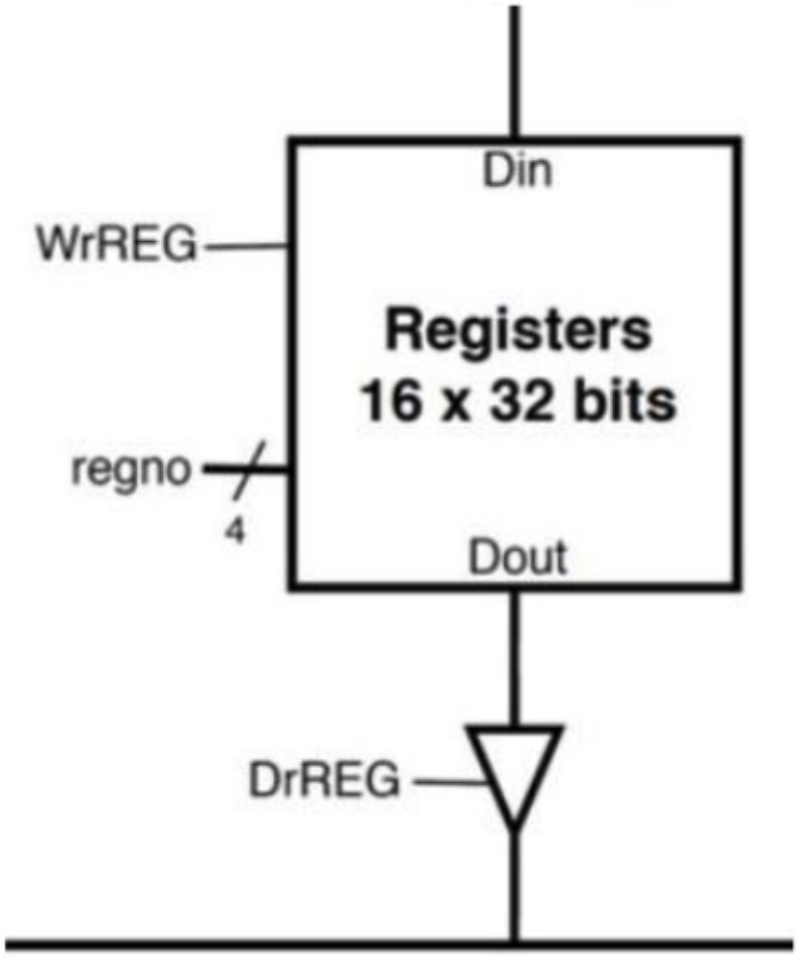
Synchronizes the hardware components
Use the sum of the worst-case delays of components in the longest data path

Datapath Components

Register File

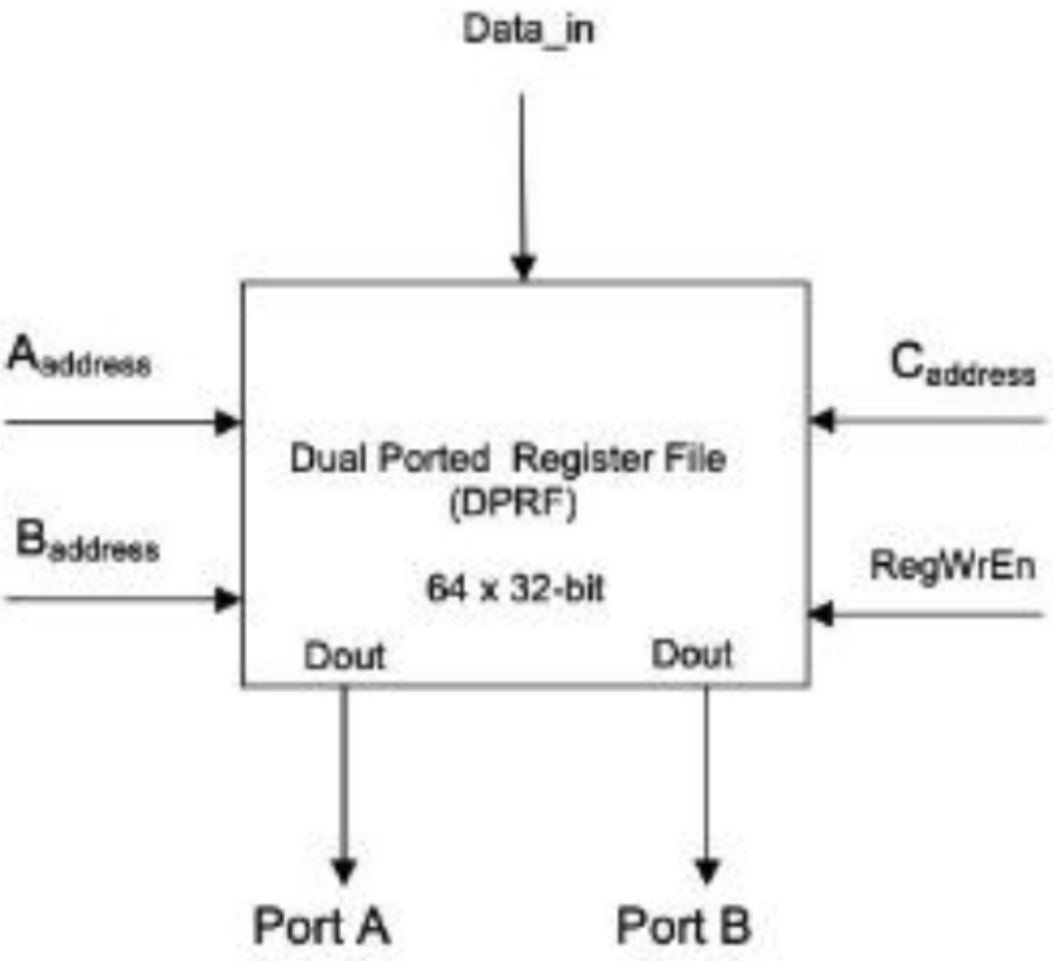
Single Ported/SPRF

- One read/write at a time
- One address line
- LC-2200/LC-902



Dual Ported/DPRF

- Simultaneously read 2 registers and write register
- Two address lines for reading, one for writing
- LC-3



The Bus

- Sharing resources
  - The highway that connects to multiple components
  - Why can't we just wire all the components directly to each other?
- Single-drive, multi-read
- One at a time
  - Only one component can talk or place a value on the bus at a time
  - The number of busses dictates how many data can be shared in one cycle
  - Tri-state buffers:
    - Make sure that there's only one value on the bus at a time
    - Enabling a tri-state buffer puts that value on the bus
  - What happens if multiple components can talk at once?

Single Bus Design

- One bus that covers all

- Components take turns talking, like in a group meeting
- LC-22 and LC-901

Dual Bus Design

- Two sets of bus wires
- Two sets of data can be transferred (e.g. address and data) at one time
- LC-3

FSM

- Divide these macrostates into microstates: basically, how much of each macrostate can we do in one clock cycle?
- Divide these macrostates into microstates: basically, how much of each macrostate can we do in one clock cycle?

3 ROM Control Unit

Main ROM

- Has datapath signals for each microstate
- Each microstate is one row in the main ROM (at a certain index = the state number)

Sequencer ROM

- For each opcode (IR[31:28], tell the start of that instruction EXECUTE stage)

Comparison (CC) ROM

- Tells you the next state based on the results of the comparison output
- "To branch or not to branch"
- Used in branch instruction

Main ROM:

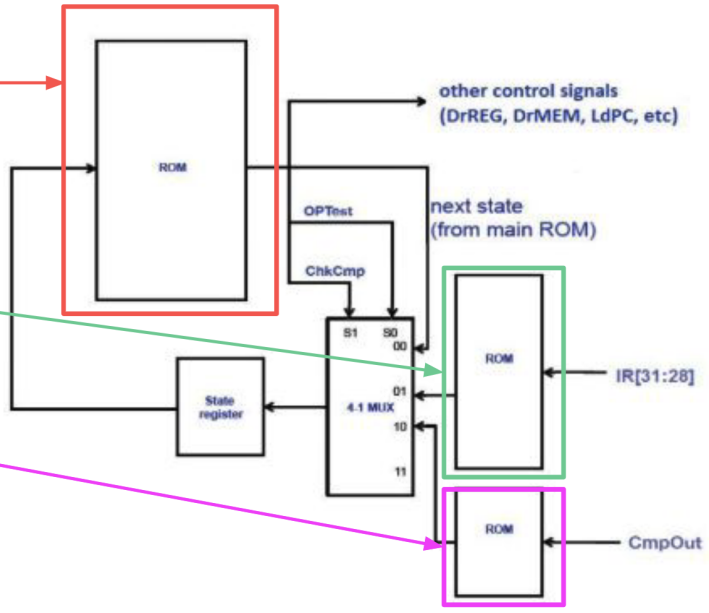
- Has datapath signals for each microstate
- Each microstate is one row in the main ROM (at a certain index = the state number)

Sequencer ROM:

- For each opcode (IR[31:28], tell the start of that instruction EXECUTE stage)

Comparison (CC) ROM:

- Tells you the next state based on the results of the comparison output
- "To branch or not to branch"
- Used in branch instruction



Fetch

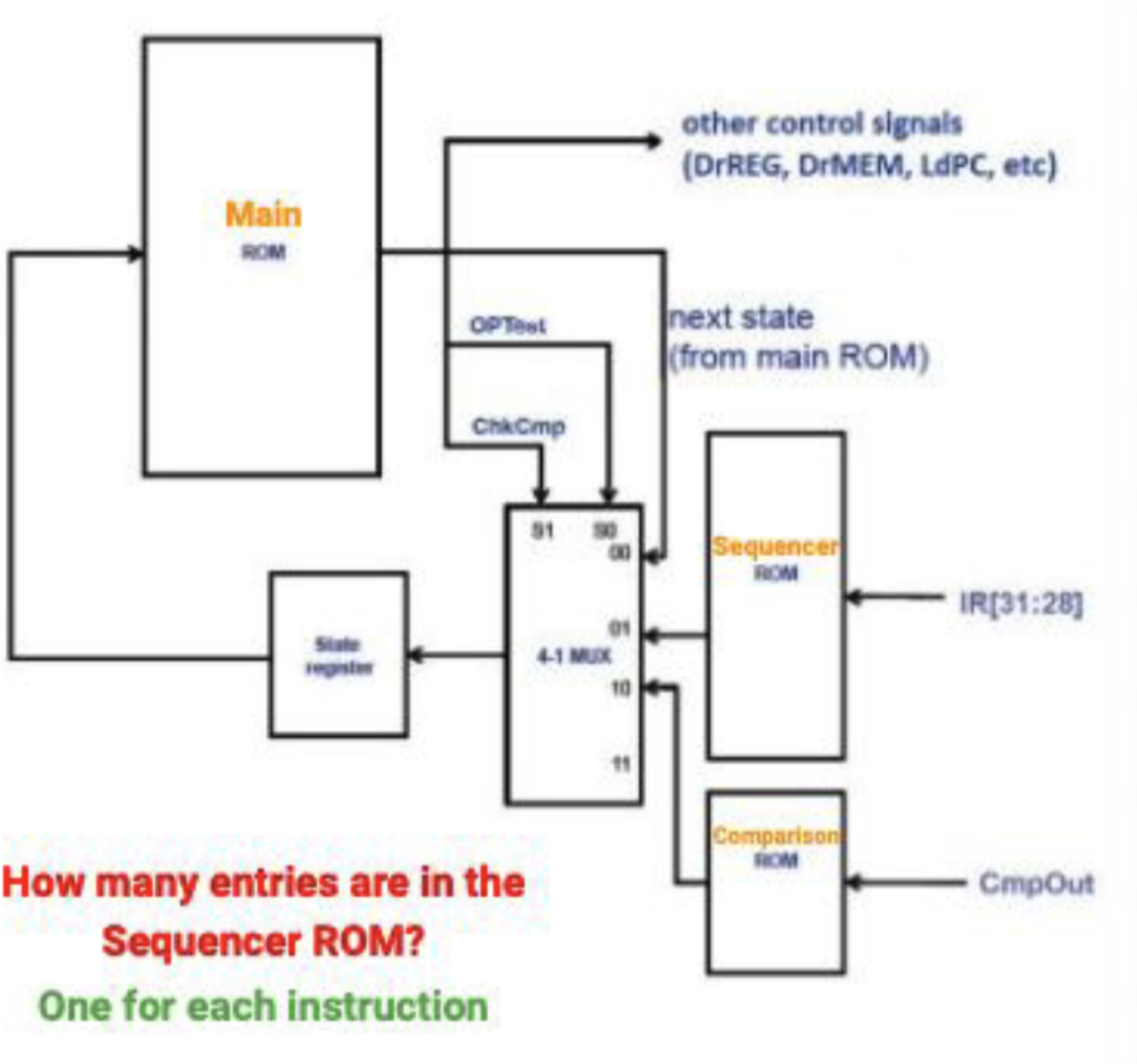
- Gets instruction from memory
- Increment the PC for the next macrostate

Directions

1. Load PC value into MAR
2. Drive mem[MAR] to IR
3. Load PC value into A
4. PC += 1

Decode

- Uses the value in the Sequencer ROM (after FETCH) to get the index in the main ROM for the start of each instruction's macrostate
- Uses OPTest to select the Sequencer ROM output as the next state
- Why?
  - Main ROM can only handle when there is one possible next state
  - After FETCH, there's a bunch of possible states



Conditional Branching and CC ROM

- ChkCmp selects the output from the comparison ROM
- For operations with conditional branching, we either want to
  - Update the PC with PC + offset
  - Do nothing and continue with next instruction
- CmpOut is the result of the comparison
  - 1 = condition met
  - 0 = condition not met