Sequential Logic

Sequential VS combinational logic

- Combinational devices: operate on data only;
  provide calculation services (e.g. Nand … ALU)

- Sequential devices: operate on data and a clock signal;
  as such, contain *state* and can ve made to provide storage and synchronization services

- Sequential devices are sometimes called "clocked devices"

- The low-level behavior of clocked / sequential gates is tricky

- The good news:

  - All sequential chips can be based on one low-level sequential gate, called "data flip flop", or DFF

  - The complex clock-dependency details can be encapsulated at the low-level DFF level

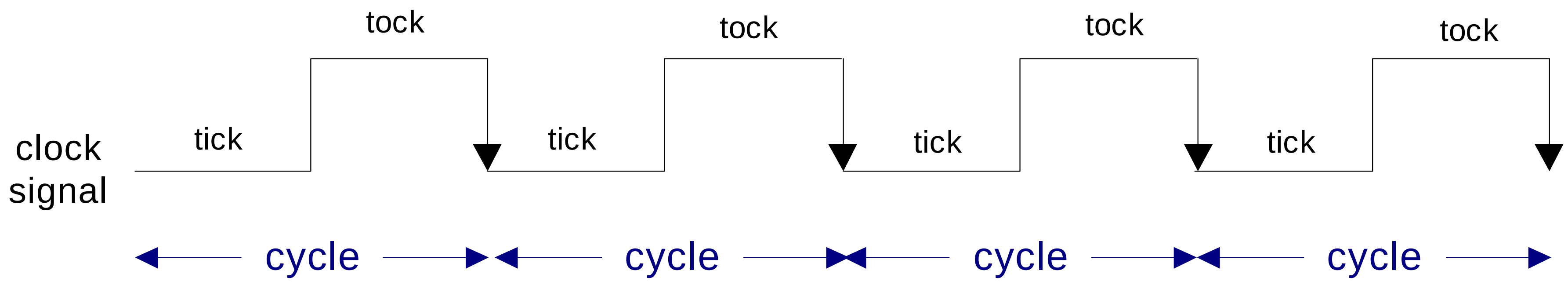
  - All the higher-level sequential chips can be built on top of the DFF using combinational logic only.

Lecture plan

- Clock
- A hierarchy of memory chips:
  - Flip-flop gates
  - Binary cells
  - Registers
  - RAM
- Counters
- Perspective.

The Clock

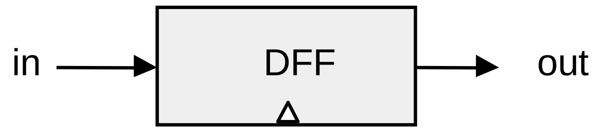
HW simulator demo



- In our jargon, a clock cycle = *tick*-phase (low), followed by a *tock*-phase (high)
- In real hardware, the clock is implemented by an oscillator
- In our hardware simulator, clock cycles can be simulated either
  - Manually, by the user, or
  - "Automatically," by a test script.

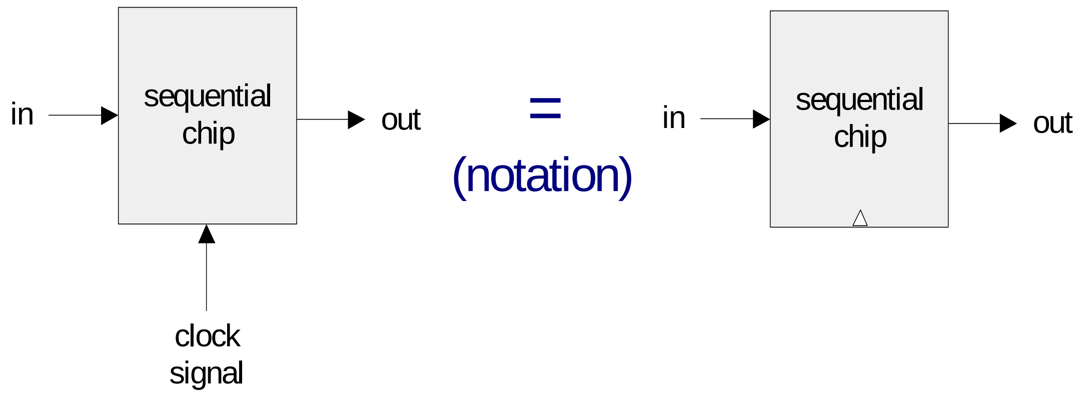
Flip-flop

HW simulator demo



out(t) = in(t-1)

- A fundamental state-keeping device
- For now, let us not worry about the DFF *implementation*
- Memory devices are made from numerous flip-flops, all regulated by the same master clock signal
- Notational convention:

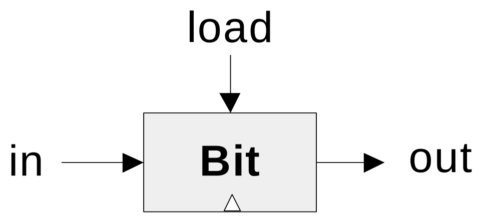


1-bit register (we call it "Bit")

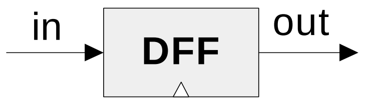
Objective: build a storage unit that can:

(a) Change its state to a given input

(b) Maintain its state over time (until changed)

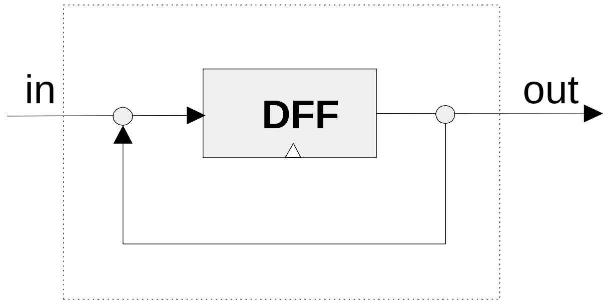


if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)



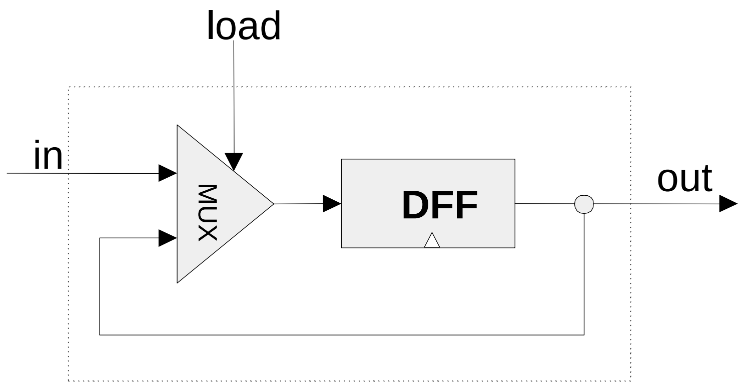
out(t) = in(t-1)

Basic building block



out(t) = out(t-1) ?
out(t) = in(t-1) ?

Won't work



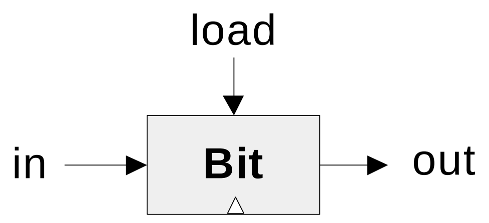
if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

OK

Bit (cont.)

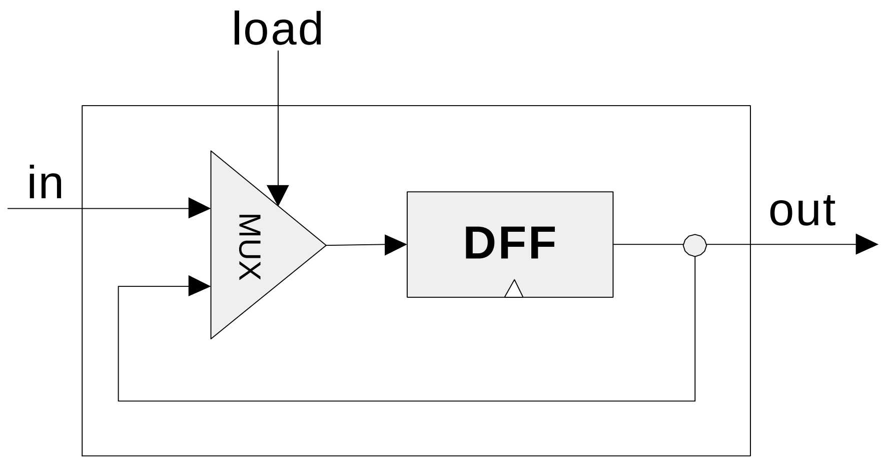
HW simulator demo

Interface



if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

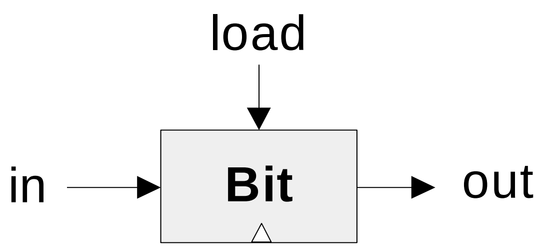
Implementation



- Load bit
- Read logic
- Write logic

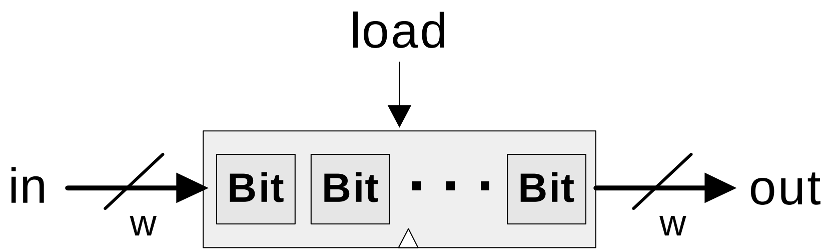
Multi-bit registers

HW simulator demo



if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

1-bit register



if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

w-bit register

- Register's width: a trivial parameter
- Read logic
- Write logic

Aside: Hardware Simulation

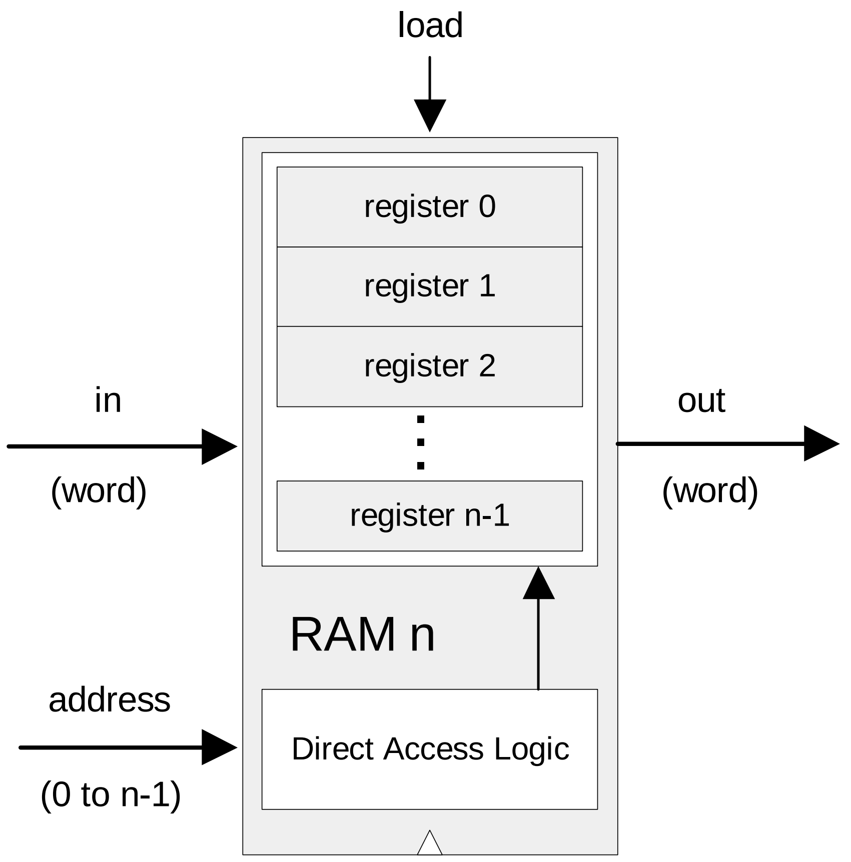
HW simulator demo

Relevant topics from the HW simulator tutorial:

- <u>Clocked chips:</u> When a clocked chip is loaded into the simulator, the clock icon is turned on, and the user can use it to control the clock
- <u>Built-in chips:</u>
  - feature a standard HDL interface yet a Java implementation
  - Provide behavioral simulation services
  - May have GUI effects.

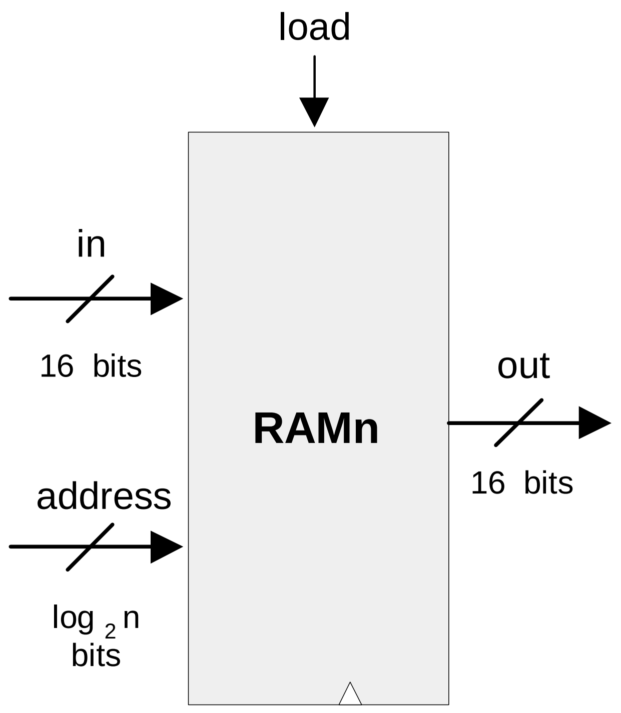
Random Access Memory (RAM)

HW simulator demo



- Read logic
- Write logic.

RAM interface

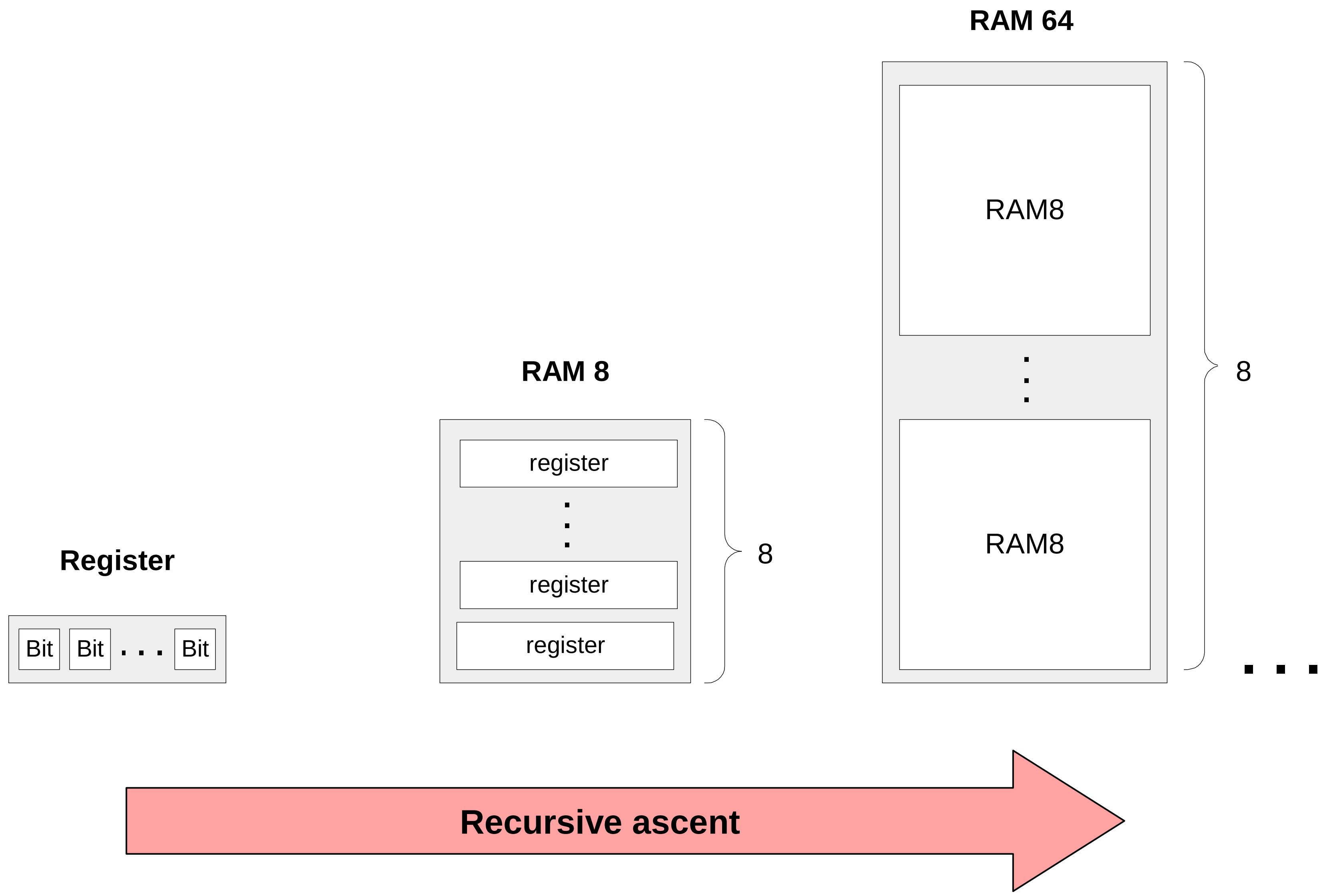


```
Chip name:  RAMn  // n and k are listed below
Inputs:     in[16], address[k], load
Outputs:    out[16]
Function:   out(t)=RAM[address(t)](t)
            If load(t-1) then
               RAM[address(t-1)](t)=in(t-1)
Comment:    "=" is a 16-bit operation.
```

The specific RAM chips needed for the Hack platform are:

Chip name	n	K
RAM8	8	3
RAM64	64	6
RAM512	512	9
RAM4K	4096	12
RAM16K	16384	14

RAM anatomy


Register
Bit
Bit
. . .
Bit
RAM 8
register
register
register
8
RAM 64
RAM8
RAM8
8
. . .
Recursive ascent

Counter

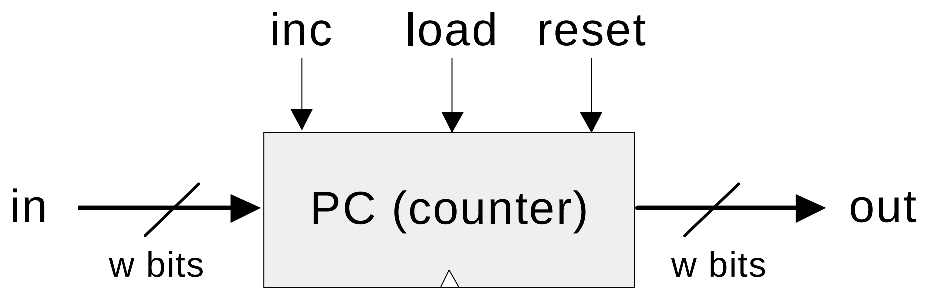
Needed: a storage device that can:

(a) set its state to some base value

(b) increment the state in every clock cycle

(c) maintain its state (stop incrementing) over clock cycles

(d) reset its state

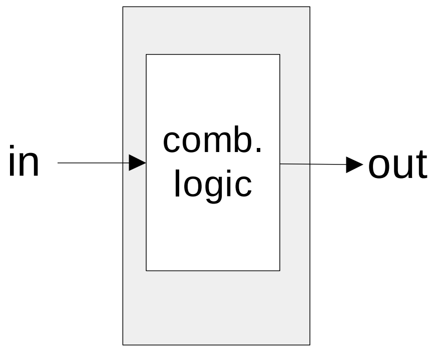


```
If reset(t-1) then out(t)=0
   else if load(t-1) then out(t)=in(t-1)
        else if inc(t-1) then out(t)=out(t-1)+1
             else out(t)=out(t-1)
```

- Typical function: *program counter*
- Implementation: register chip + some combinational logic.

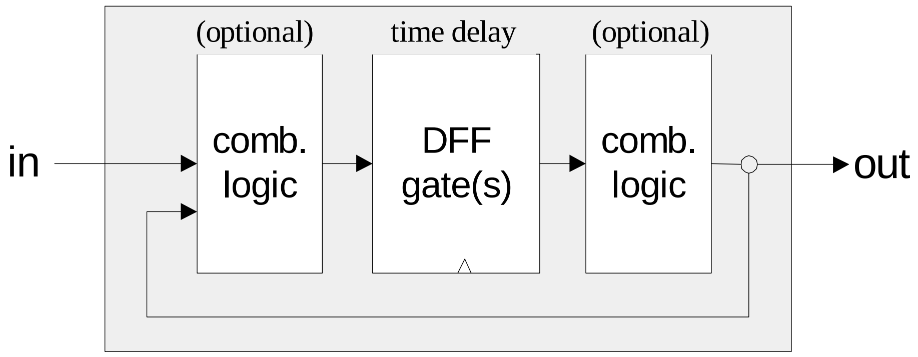
Recap: Sequential VS combinational logic

Combinational chip



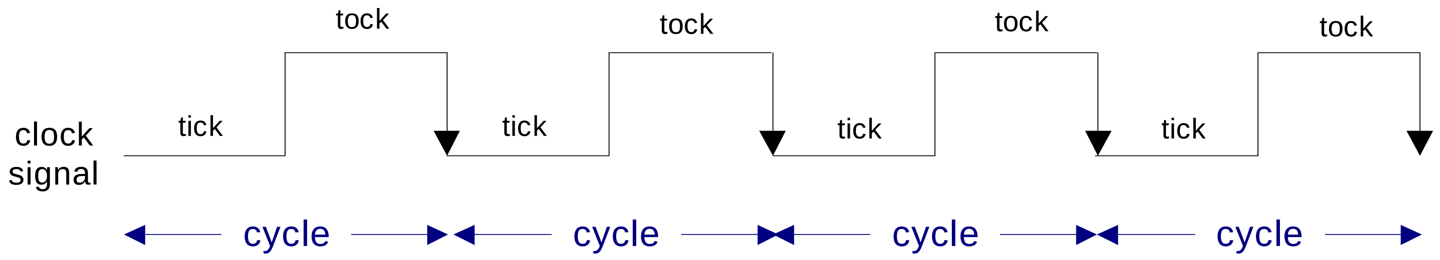
out = *some function of* (in)

Sequential chip

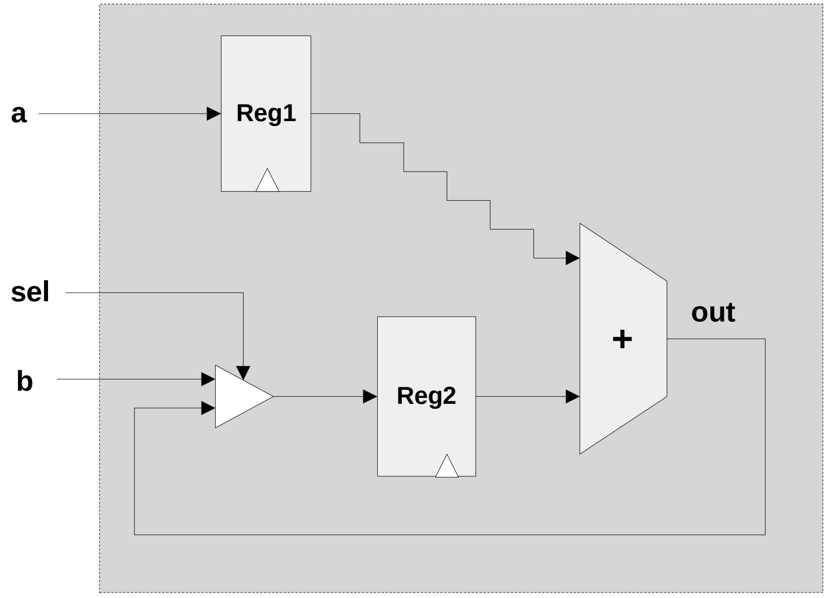


out(t) = *some function of* (in(t-1), out(t-1))

Time matters



- During a tick-tock cycle, the internal states of all the clocked chips are allowed to change, but their outputs are "latched"
- At the beginning of the next cycle, the outputs of all the clocked chips in the architecture commit to the new values.



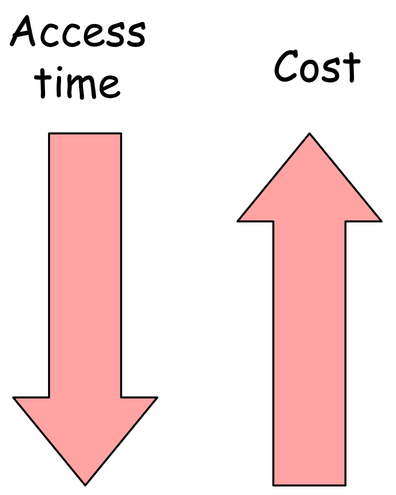
Implications:

- Challenge: propagation delays
- Solution: clock synchronization
- Cycle length and processing speed.

Perspective

- All the memory units described in this lecture are standard
- Typical memory hierarchy
  - SRAM ("static"), typically used for the cache
  - DRAM ("dynamic"), typically used for main memory
  - Disk

  (Elaborate caching / paging algorithms)
- A Flip-flop can be built from Nand gates
- But ... real memory units are highly optimized, using a great variety of storage technologies.


Access time
Cost