EP1200 Introduction to Computing Systems Engineering

Sequential Logic

Perspective on Boolean logic

Each Boolean function has a canonical representation

- expressed in terms of And, Not, Or
- And, Not, Or can be expressed in terms of Nand alone

Universal building blocks

- mass production
- unique topology

Perspective on Boolean arithmetic

Combinational logic: time for output to be trusted?

The adder design is very basic: no parallelism

- It pays to optimize adders

The ALU is also very basic without multiplication and division

- What about more advanced math operations?
- A typical hardware/software tradeoff

Sequential vs. combinational logic

Combinational devices

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

Problem

- When is the output signal stable and useful?

Sequential devices

- operate on data and a clock signal
- storage and synchronization
- called “clocked devices”

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

- All sequential chips can be based on one low-level sequential gate
  - the “data flip flop”, or DFF
- The clock-dependency encapsulated at the low-level DFF level
- Higher-level sequential chips can be built on top of DFF gates using combinational logic only.

Lecture plan

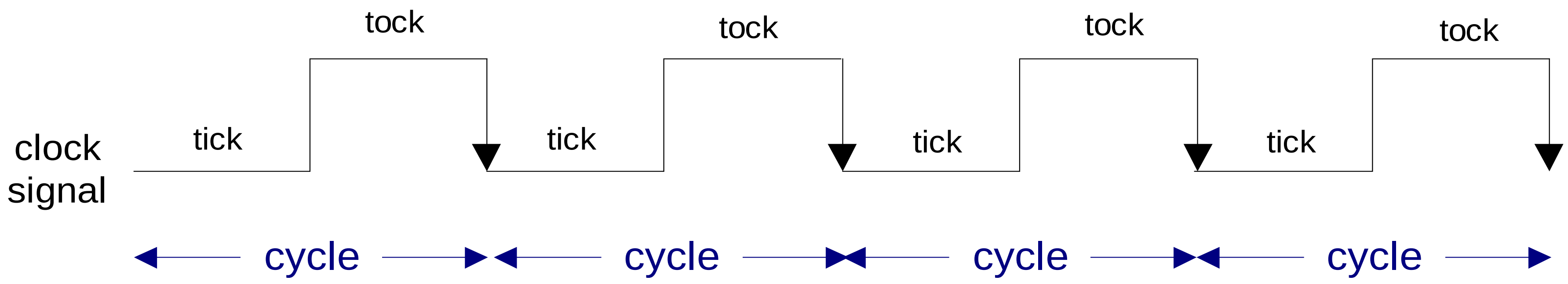
Clock

A hierarchy of memory chips

- Flip-flop gates
- Binary cells
- Registers
- RAM

Counters

The clock



A clock cycle

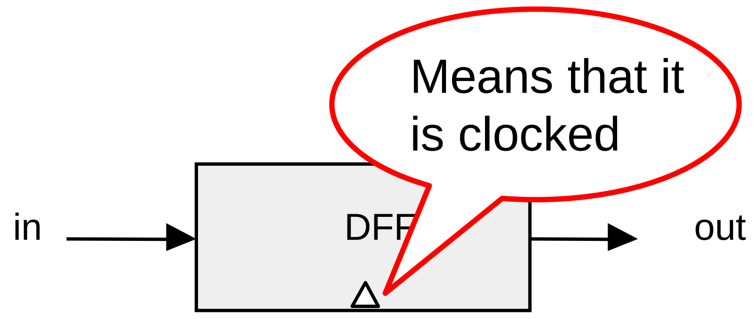
- *tick*-phase (low) and a
- *tock*-phase (high)

In real hardware, the clock is implemented by an oscillator

In the hardware simulator, clock cycles generated

- manually by the user
- automatically by a test script

Data flip-flop



out(t) = in(t-1)

A fundamental state-keeping device

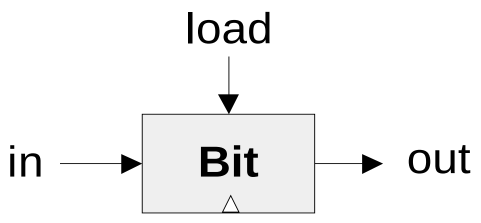
Not concerned with DFF *implementation*

- given, similarly to NAND for combinational logic

Memory devices

- many flip-flops
- regulated by the same master clock signal

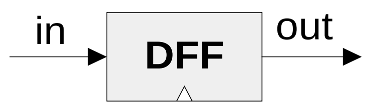
A one-bit register



A storage unit that can

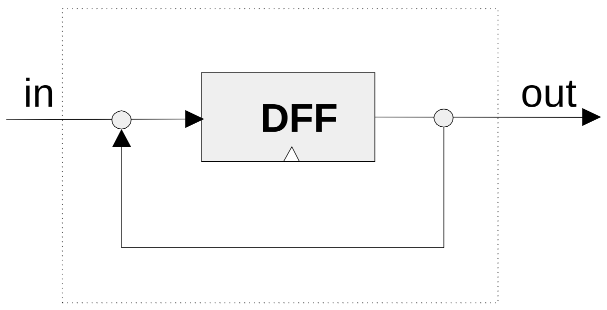
- change its state to a given input
- maintain its state over time (until changed)

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



out(t) = in(t-1)

Basic building block

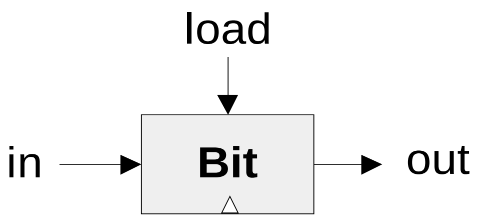


out(t) = out(t-1) ?
out(t) = in(t-1) ?

Won't work

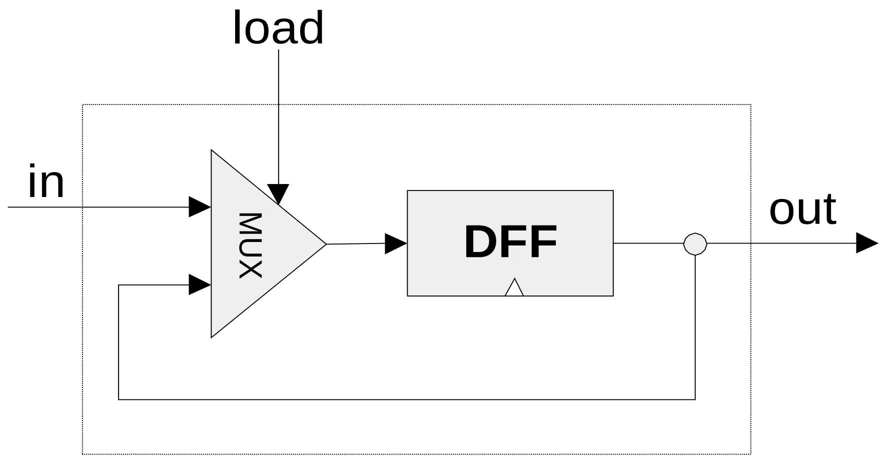
A one-bit register

Interface

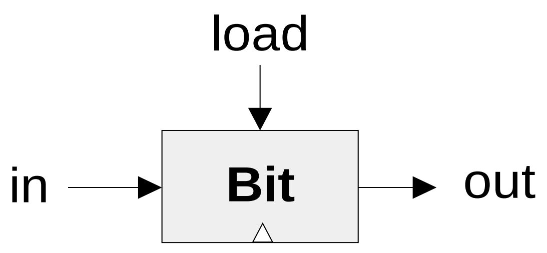


if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

Implementation

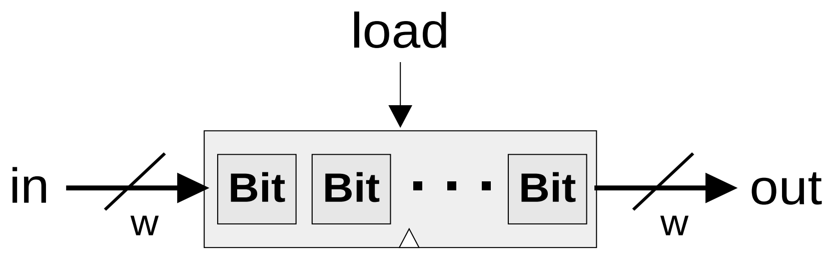


Multi-bit register



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

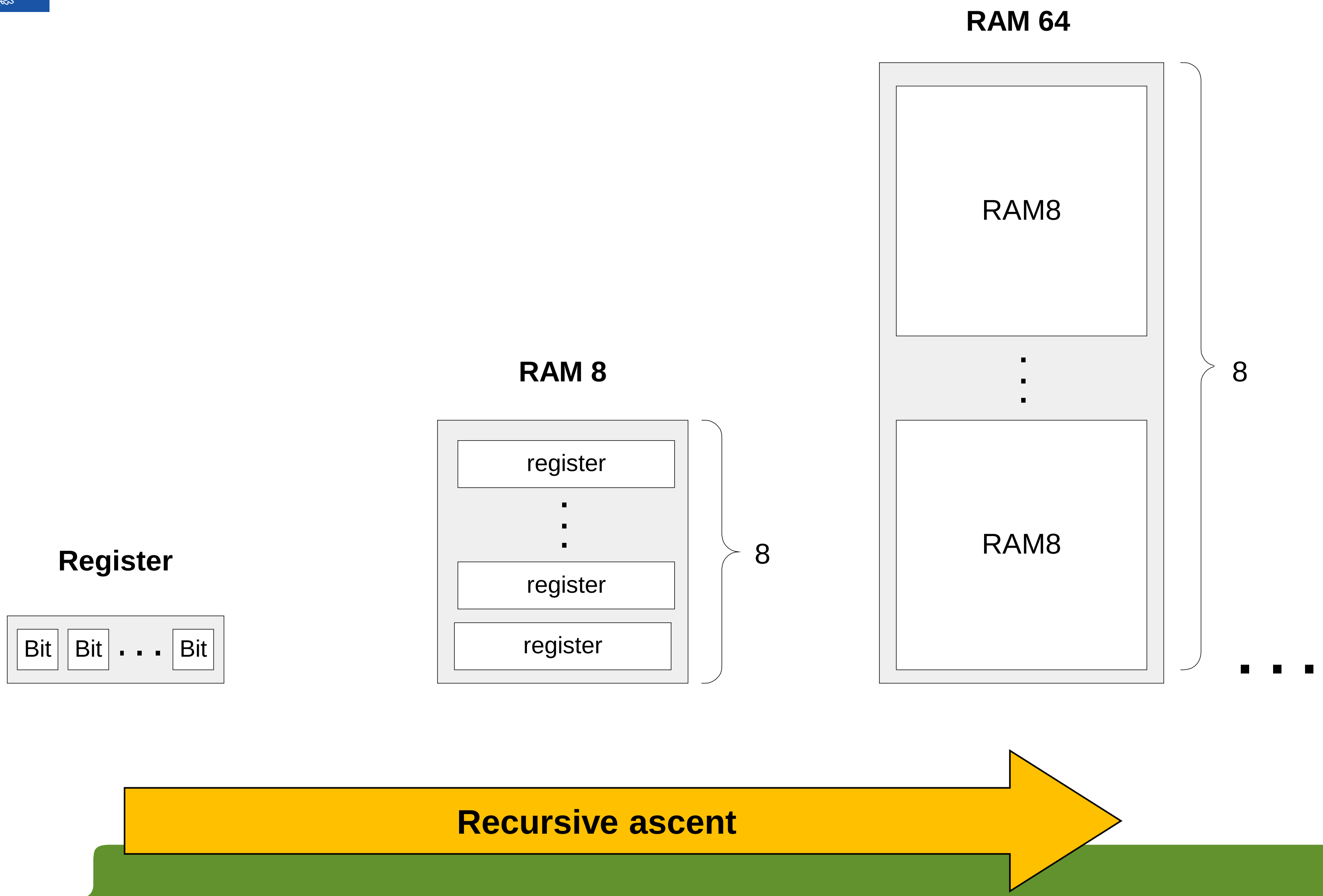
1-bit register



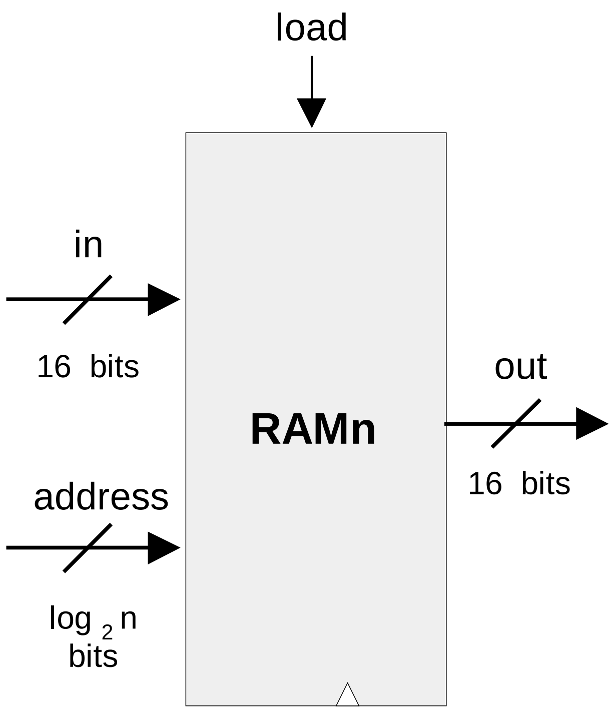
if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

w-bit register

RAM hierarchy



RAM interface

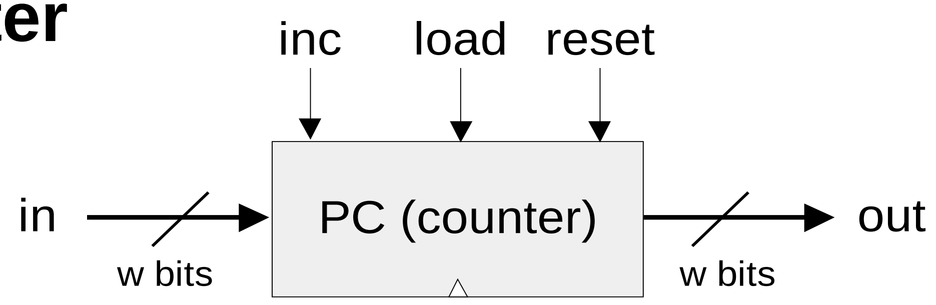


```
Chip name:  RAMn   // n and k are listed below
Inputs:     in[16], address[k], load
Outputs:    out[16]
Function:   out(t)=RAM[address(t)](t)
            If load(t-1) then
               RAM[address(t-1)](t)=in(t-1)
Comment:    "=" is a 16-bit operation.
```

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

Program Counter



```
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)
```

A storage device that can

- set its state to a base value (load)
- increment the state in every clock cycle
- maintain its state (stop incrementing) over clock cycles
- reset its state

Typical function: *program counter*

Implementation: register chip + combinational logic

For next class

1. Read
   - Chapter 3 on sequential logic
   - Appendix A, pages 289-296
2. Do project 2 from course web
   - In folders
     - projects/03/a and projects/03/b (large RAM)
   - Hints
     - Fan out, use “out=a, out=b, …”
     - For PC, several control signals may be set at the same time
       - follow the if clauses in the definition of the chip
3. Hand in Project 3 by March 28 at 8.00