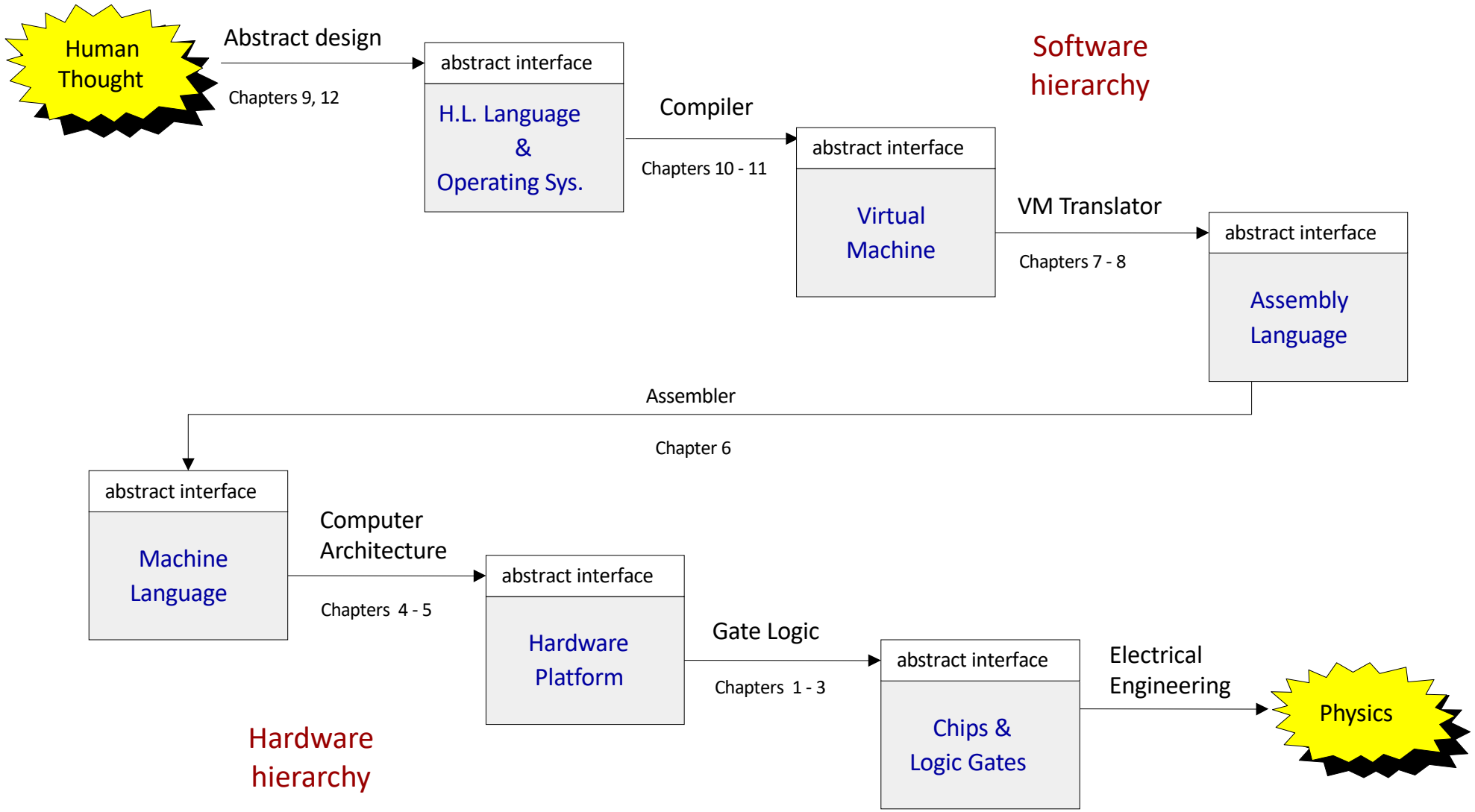
School of Computer Science

COMP SCI 2000 Computer Systems
Lecture 3

adelaide.edu.au

seek LIGHT

The whole system



(Abstraction–implementation paradigm)

Review - combinational logic

- Our adder design is very basic: no parallelism
- You can construct many gates from NAND – this is just one example of how gates are built up.
- By understanding arithmetic, we can combine gates to add two numbers, then use full-adders to add larger numbers.
- Combinational logic operates on data only;
  provide calculation services (e.g. Nand … ALU)
- Workshop 2 looks at how to build an ALU

Sequential logic

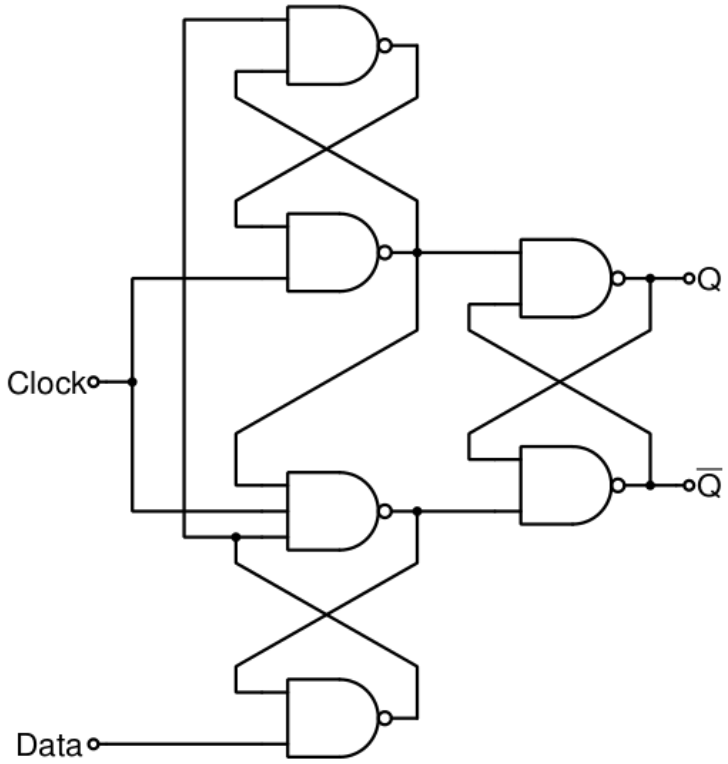
- This operates on data and a clock signal;
  as such, can be made to be *state*-aware and provide storage and synchronization services
  - What does *state-aware* mean?
- Sequential devices are sometimes called “clocked devices”
- The low-level behavior of clocked / sequential gates is tricky

Good news!

- All sequential chips can be based on one low-level sequential gate, called “data flip flop”, or DFF (DFFs can be made from NAND gates).
- The clock-dependency details can be encapsulated at the low-level DFF level
- Higher-level sequential chips can be built on top of DFF gates using combinational logic only
- You’re probably getting used to the idea that, if you understand some of the low-level stuff, you can build more and more complex machines.

Inside a DFF

- Just so you can see it can be done with NANDs
- We won't worry about the details inside a DFF in this course!

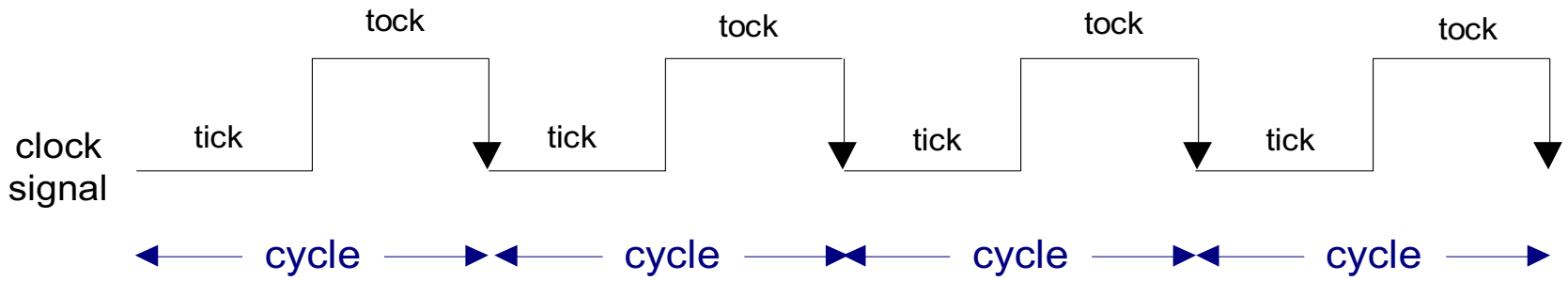


What we're doing now

- This lecture and next we are going to talk about:
  - Hardware clocks
  - Memory chip hierarchy
    - Flip-flop gates
    - Binary cells
    - Registers
    - Random Access Memory
  - Counters
  - Putting it all together

What's a clock in this context?

- We use clocks to measure time.
- In this case, the clock *signal* is used to allow us to reflect what happens as the state of hardware changes over time.
- The hardware clock signal *oscillates* in low/high tick/tock phase.

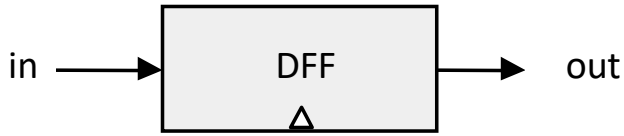


Clocks

- The speed of the clock is going to affect the speed of everything in the computer as a clock cycle is the smallest unit of time in which anything can change.
- In actual hardware, it's an oscillator.
- In the simulator, the user can feed the clock signal in manually or we can use a test script.

State

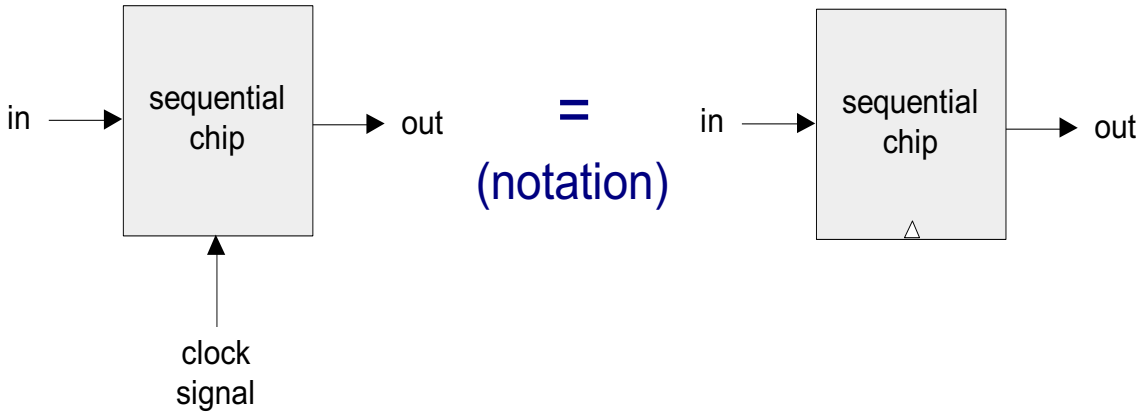
- The state of an element is the condition it is in at a given time.
- A binary digit (bit) can be 0 or 1 but the value of that bit, at a certain time, is its state.
- What does this look like as a concept?
- *out* will give us a 0 or 1 depending on what is in there.



out(t) = in(t-1)

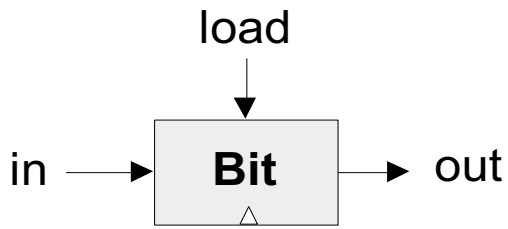
Using the clock

- Memory is made up of lots of these cells, all running on the same master clock signal.
- (We're not worrying about the implementation at the moment.)
- When do we change the contents? The *in* signal will always be set to something!



Let's build a bit! (1-bit register)

- We want to be able to:
  - Change the state of the bit
  - Retain that state until we want to change it.

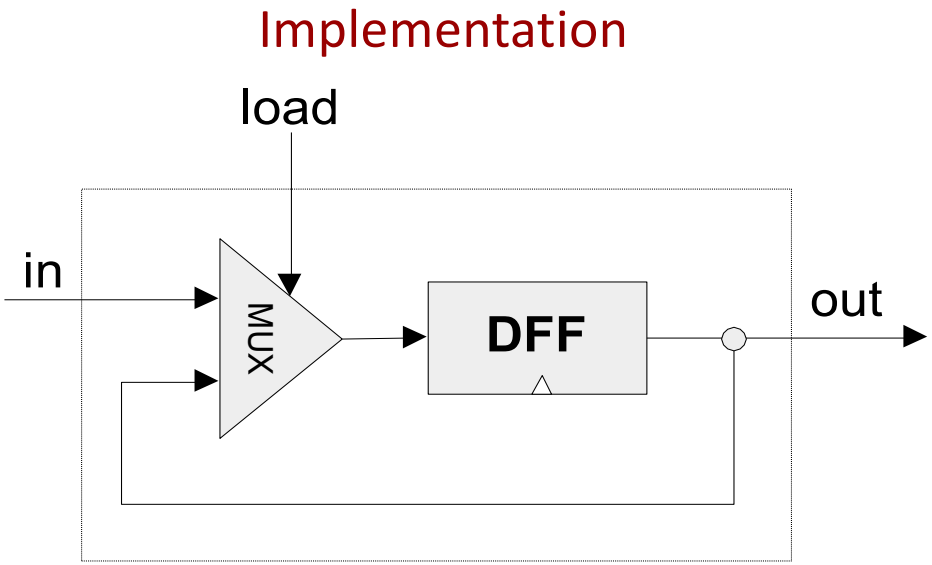


if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

- Now we can tell the bit when to change. Can we build this from the DFF and gates we already know?

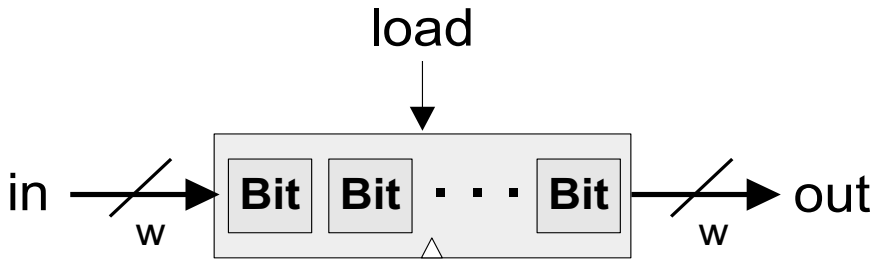
Bit register

- Notice how we use the load signal.
- Now we have read logic and write logic.



Bigger registers

- We know we can work with more than one bit.

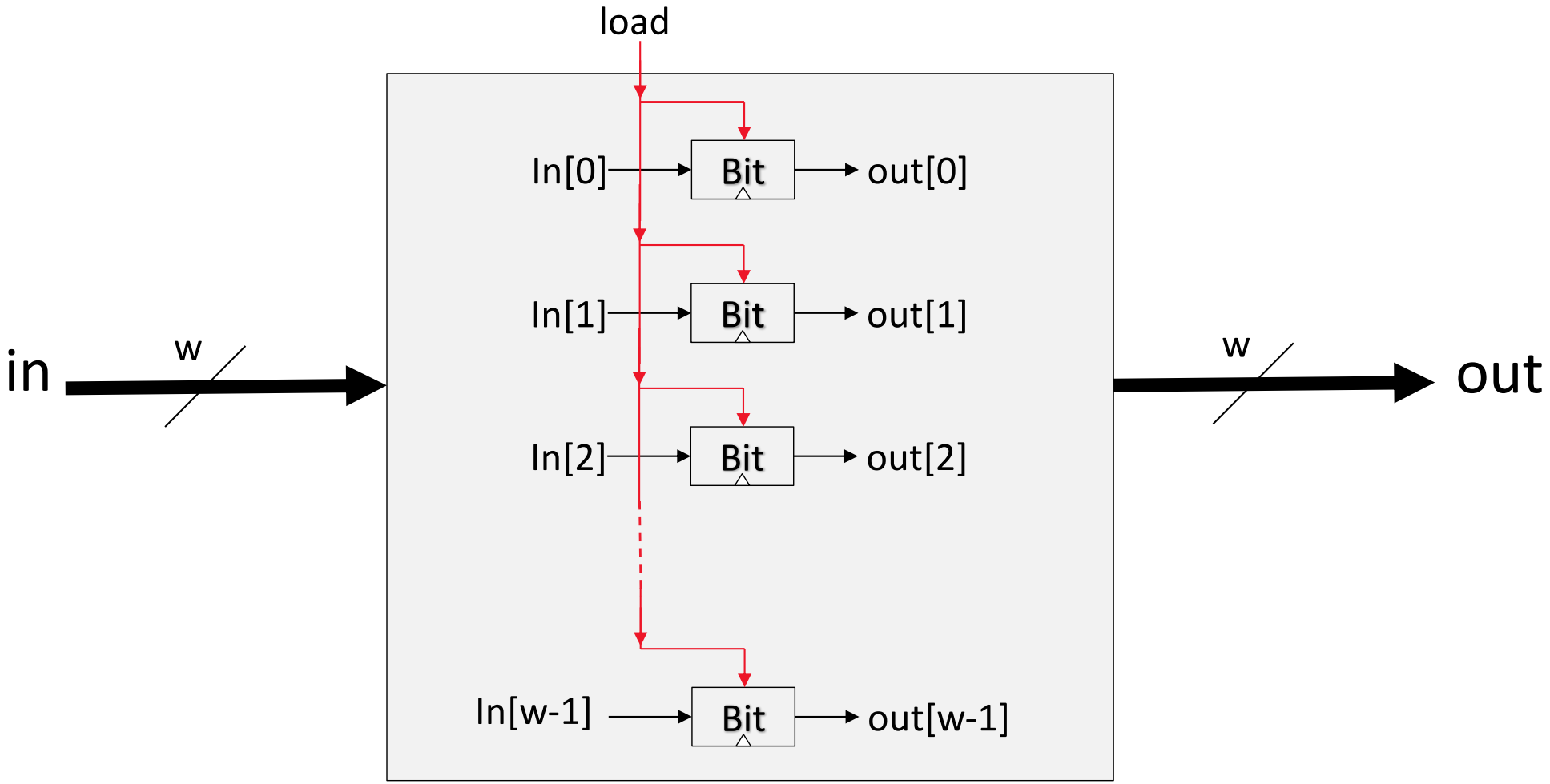


if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

w-bit register

Bigger registers

- We know we can work with more than one bit.



Simulator

- Clocked chips: When a clocked chip is loaded into the simulator, the clock icon is enabled, allowing clock control
- Built-in chips:
  - feature a standard HDL interface yet a Java implementation
  - Provide behavioral simulation services
  - May feature GUI effects (at the simulator level only).