COMS12200 Introduction to Computer Architecture

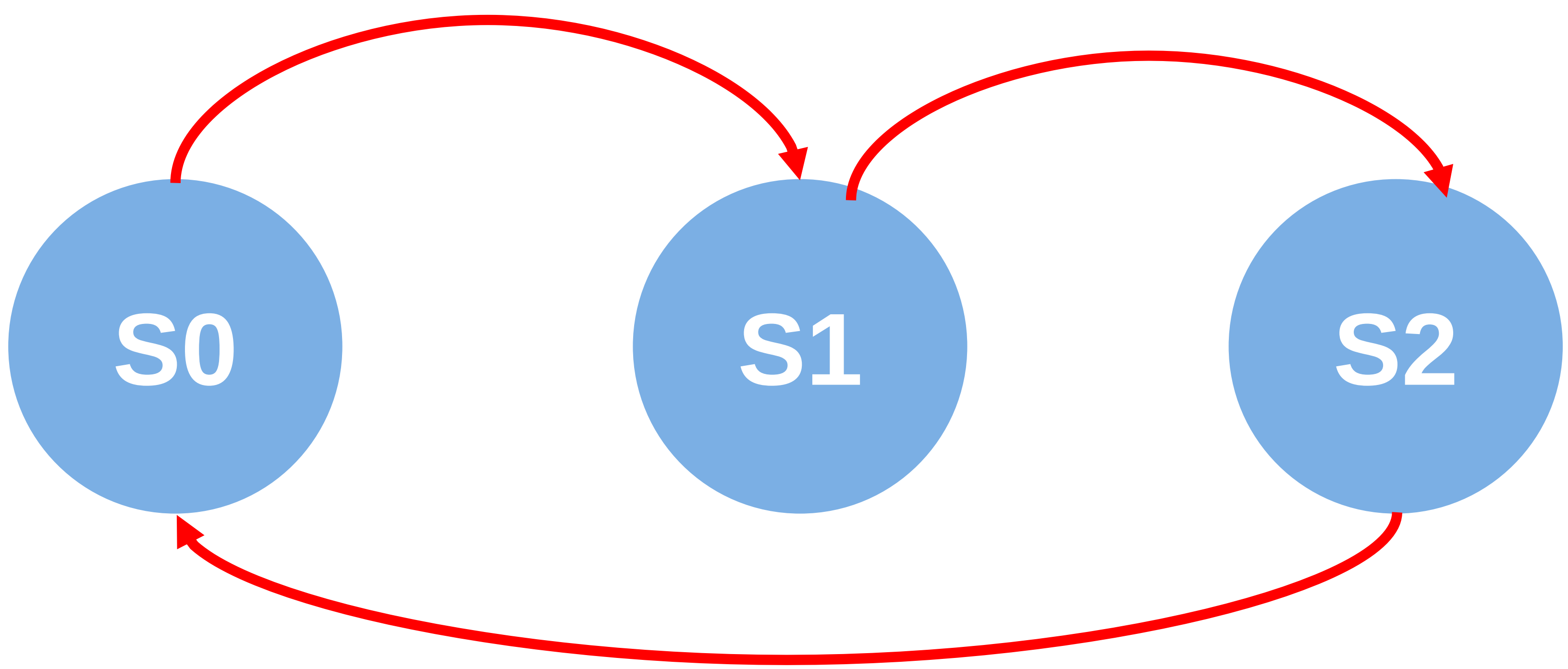
Simon Hollis (simon@cs.bris.ac.uk)

COMS12200 Part 6 – Simon Hollis

STATE MACHINES AND DECODING

State machine recap

- A state machine is one with a finite set of defined states.
- Transitions are made between states.
- Transitions can be gated or ungated.



State machine recap

Each state is uniquely labelled and transitions can also be expressed in a table.

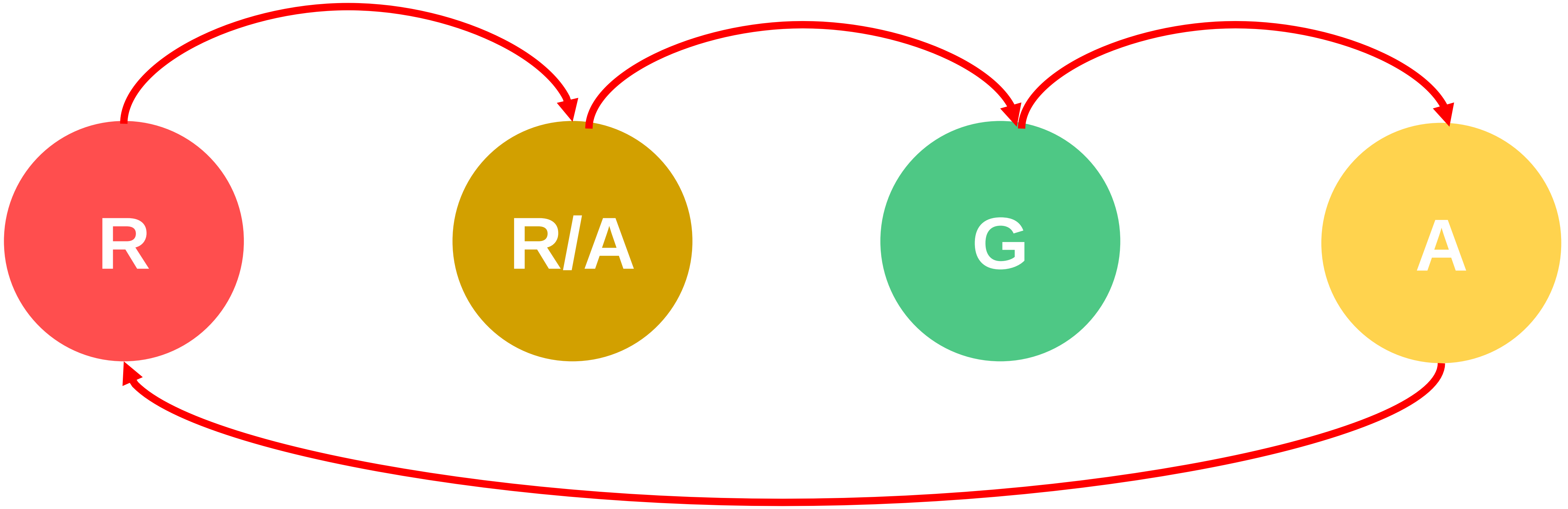
Current state	Next state	Condition
S0	S1	Always
S1	S2	Always
S2	S0	Always

Useful state machines

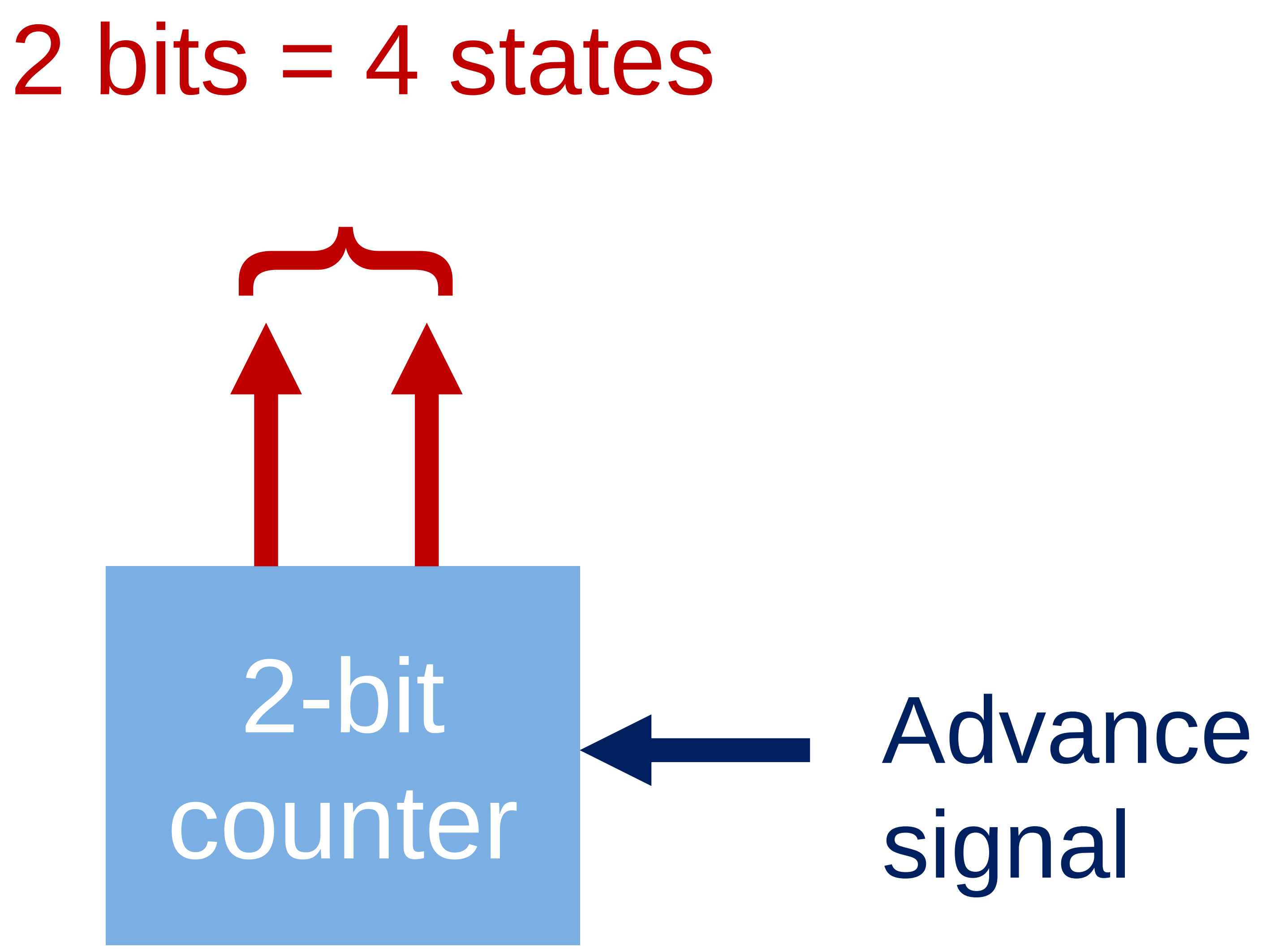
TRAFFIC LIGHTS

Traffic lights

A traffic light is a useful state machine with four states



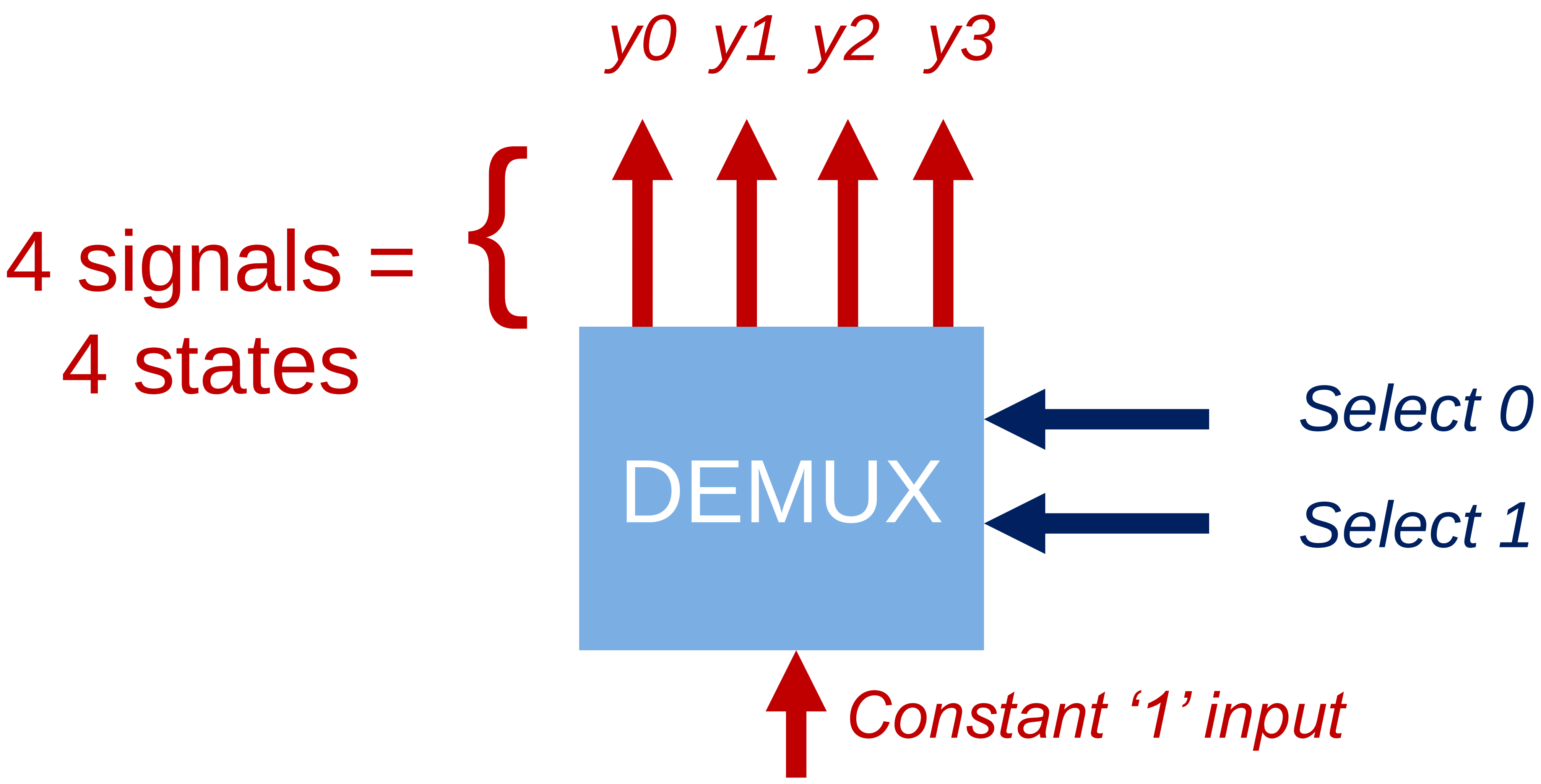
The simplest four state machine



DECODING STATES

A four state decoder

- Q: How can we uniquely decode a state from our simple machine?
- A: A DEMUX solves the problem.



The DEMUX

- How does a DEMUX work?

From states to lights

- How do we convert the decoder outputs to the R, A, G values for a simple traffic light?

From states to lights

- Observe that each of the decoder outputs has produced a minterm
- Each minterm is unique and can be manipulated via boolean logic
- Typically, we will combine minterms via logical gates to obtain a compound output (e.g. Red light from R and R/A minterms)

Dynamic behaviour

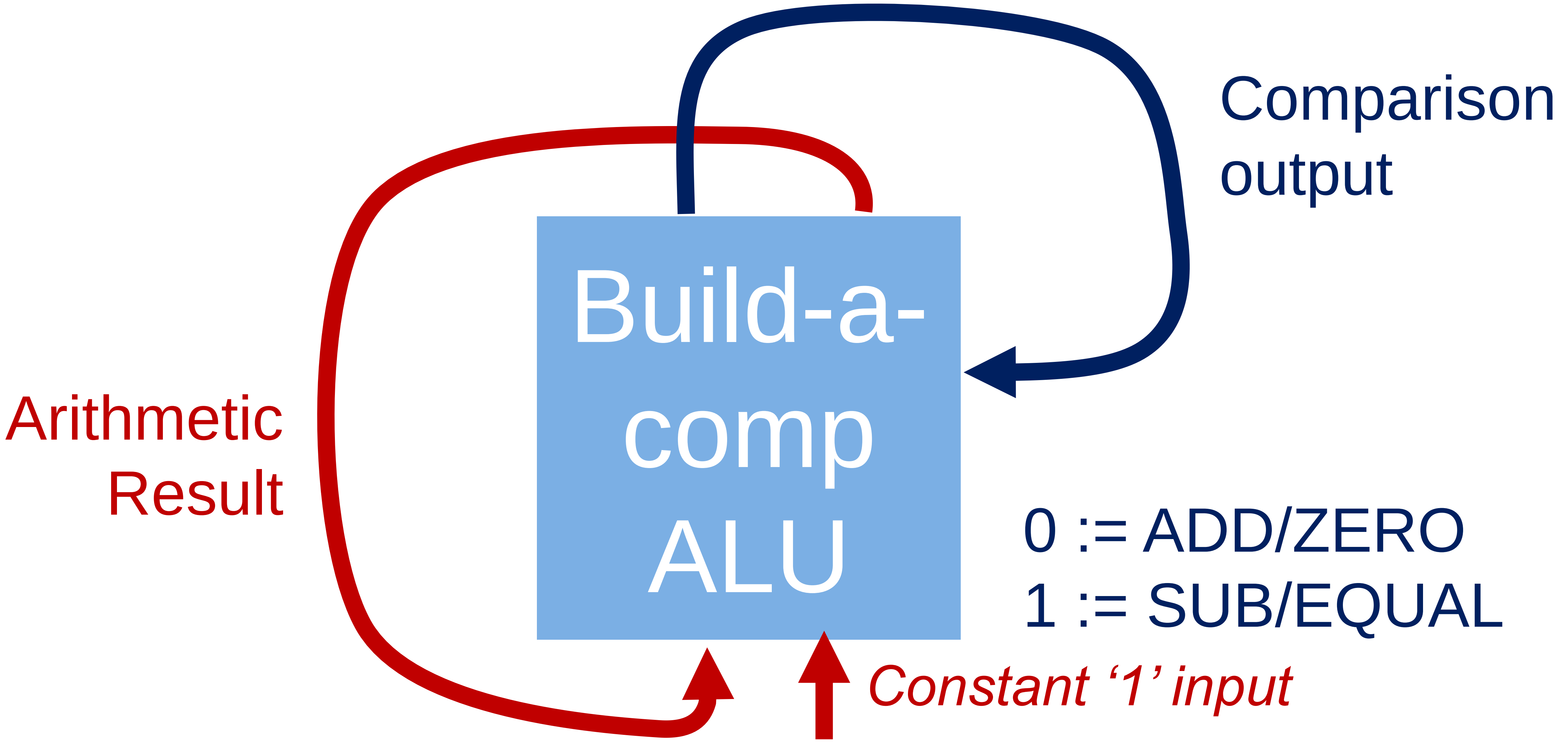
VARIABLE CONTROL SIGNALS

Why vary control?

- In your labs so far, you have made static systems: ones that do the same task again and again.
- To change their behaviour, you have used switches.
- This is great for simple tasks, but we'd really like to be 'hands off' when computing more complex things.

Supporting variable control

Here's a simple example of how we could vary control to do something useful:



The notion of feedback

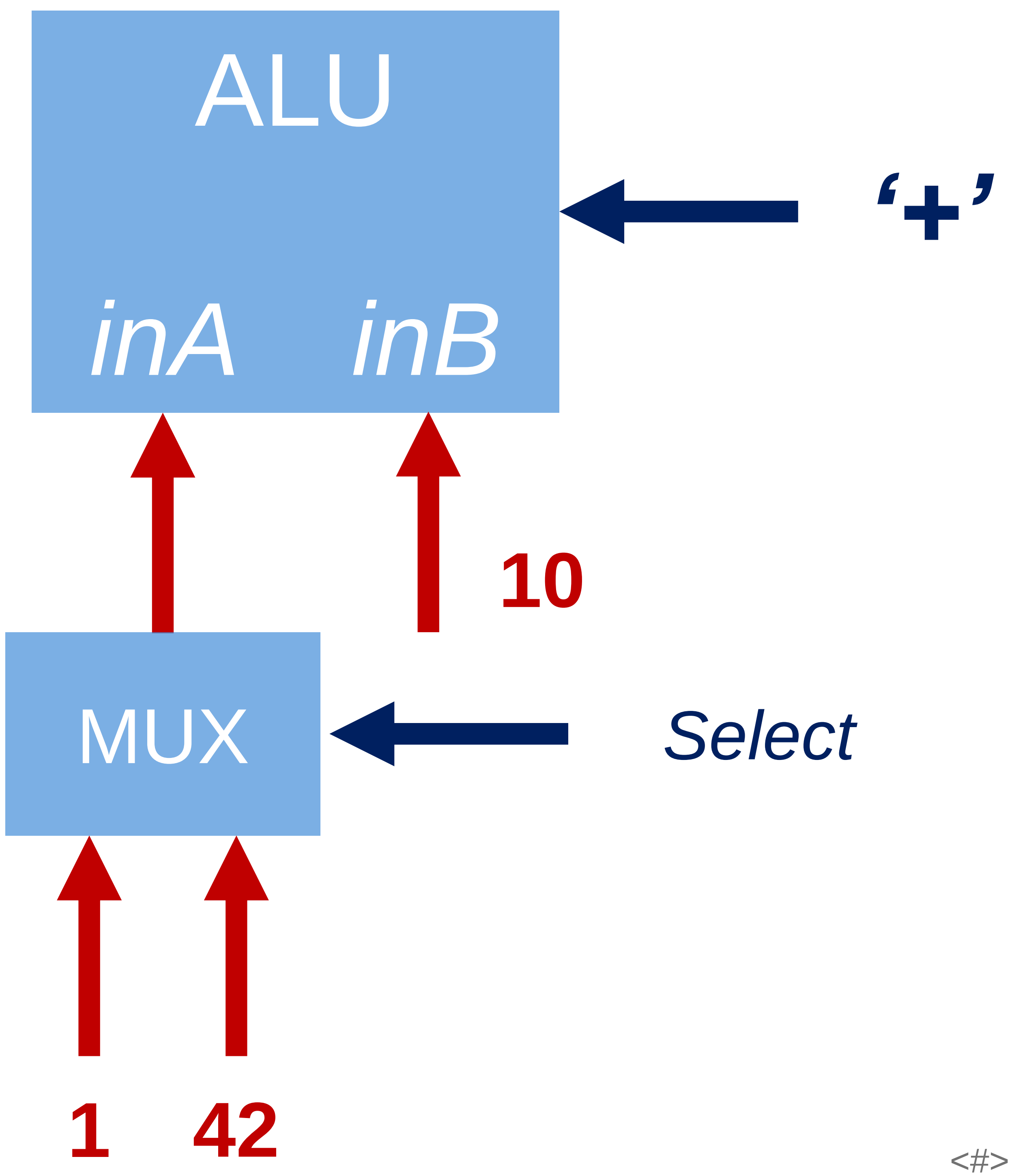
- *Feedback* was the key thing that made the previous example do something interesting.
- Feedback is when a previous output from a system is used to alter its current behaviour:

$$\text{i.e. } S_{n+1} = f(S_n, \text{inputs})$$

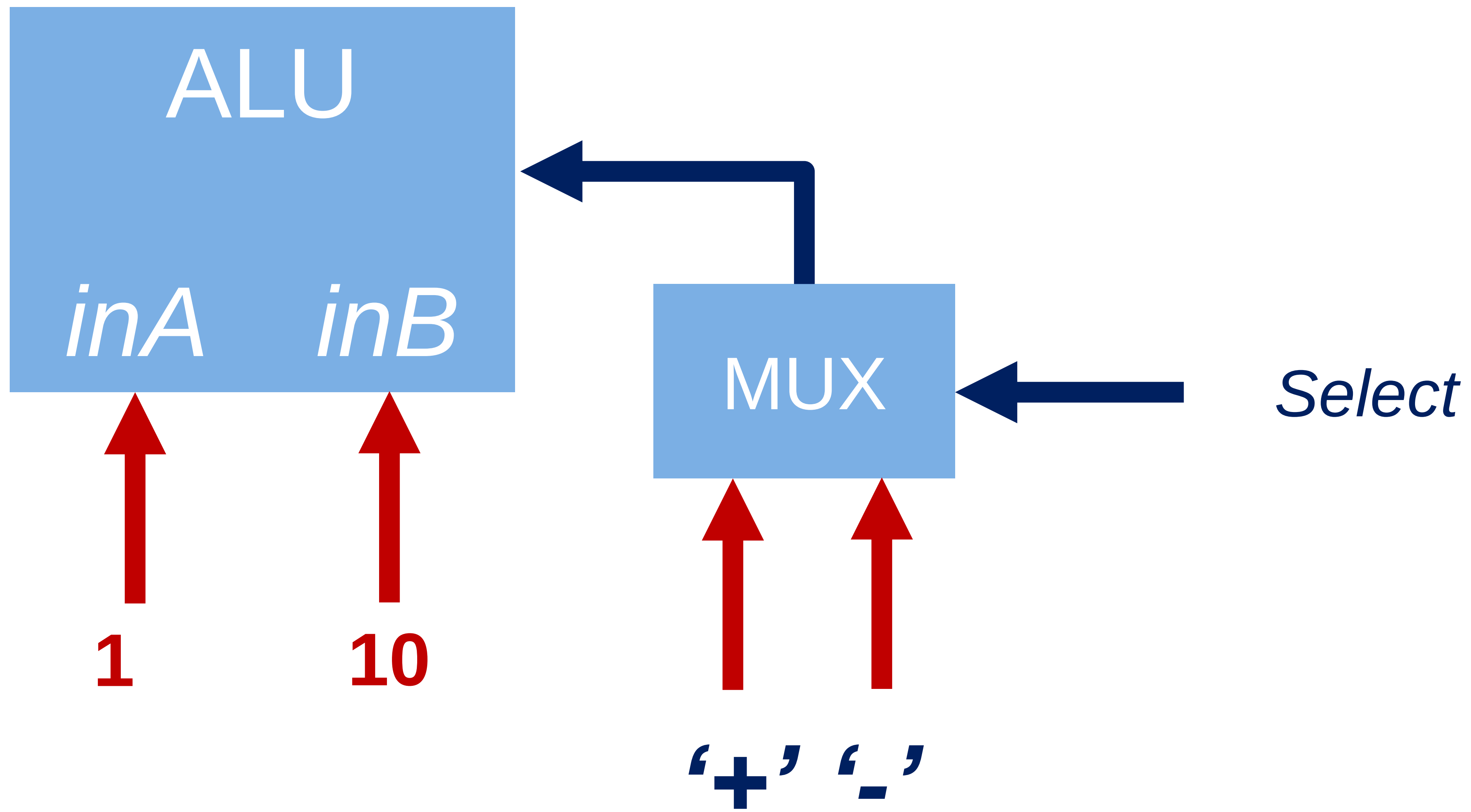
Feedback to change operations

- We can use feedback to alter both the data inputs and/or the function of a unit.
- There are several useful implementation methodologies.
- We’ll now look at one simple one (but with a relatively high implementation cost in real systems).

Changing the data inputs



Changing the control inputs



The MUX

- How does the MUX work?

Selecting the input

- Where do the *select* signals come from?
  - They could come from instructions
    - Instructions will dictate behaviour

  - They could come from feedback
    - Data values will dictate behaviour

Summary

Here are three useful guidelines when creating control flow for state machines:

1. MUXs are useful for selecting inputs
2. DEMUXs are useful for decoding outputs
3. OR gates are useful for combining states and producing feedback signals