An Overview of FLEET

CS-152

Berkeley EECS

FLEET

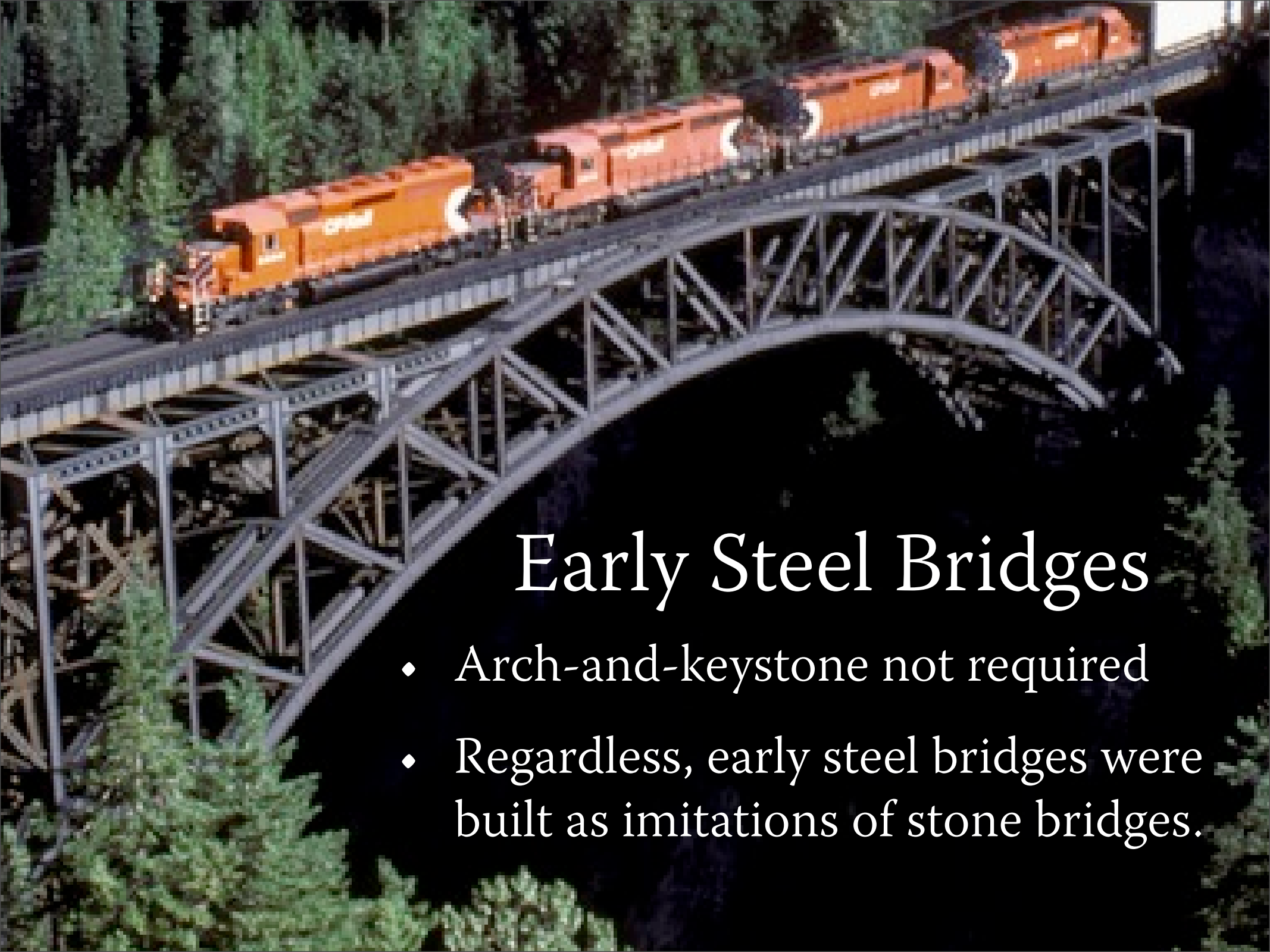
- Brainchild of Ivan Sutherland
  - Fleshed out in collaboration with Berkeley graduate students
- A one-instruction, clockless processor
  - Alternatively: an asynchronous transport-triggered VLIW machine
  - Designed to take advantage of asynchronous circuits
- But first, I'd like to talk about...

Berkeley EECS

- Ancient bridges were made of stone
- Required arch-and-keystone design

Steel

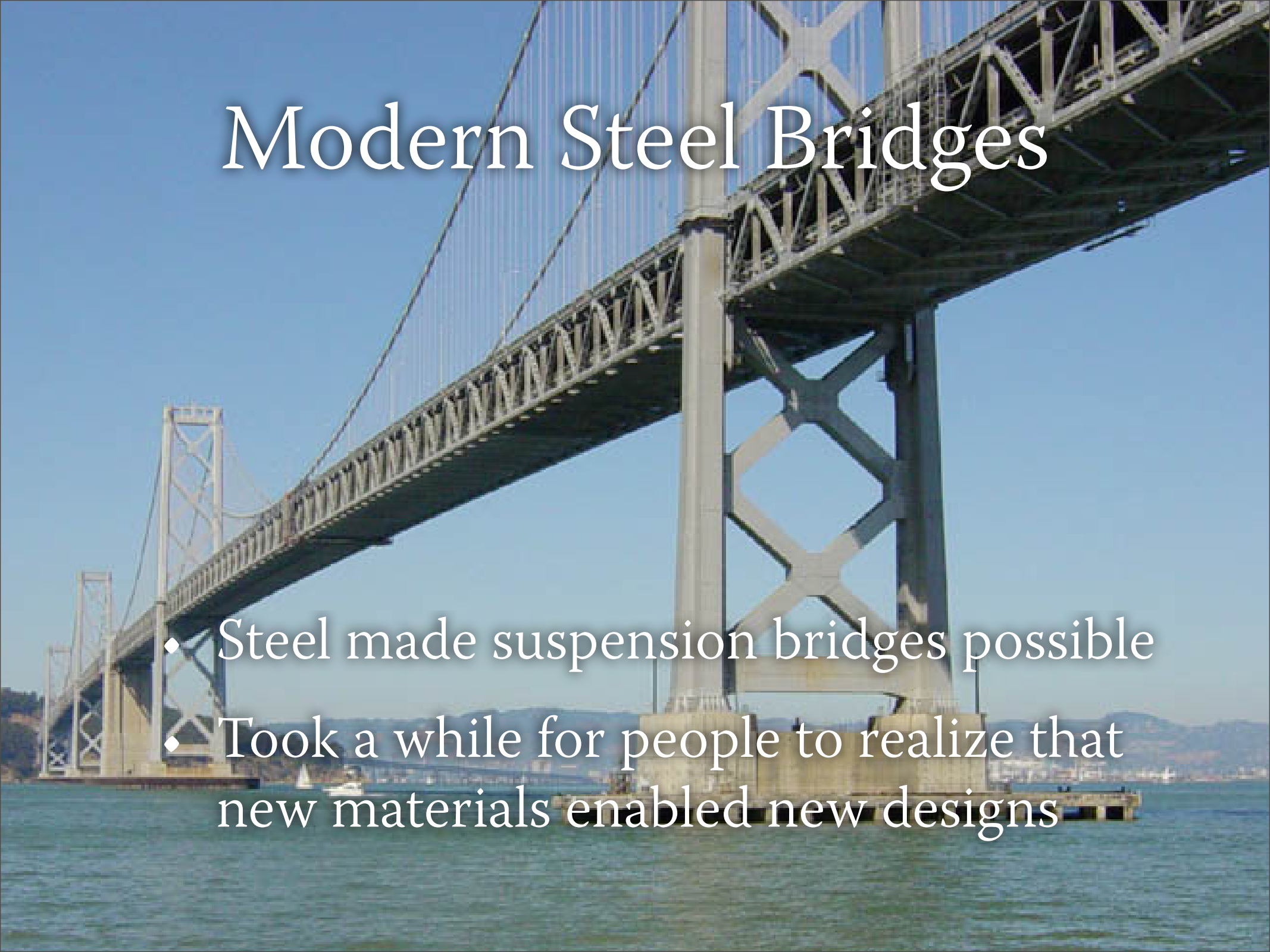
- Much stronger building material than stone



Early Steel Bridges

- Arch-and-keystone not required
- Regardless, early steel bridges were built as imitations of stone bridges.

Modern Steel Bridges



- Steel made suspension bridges possible
- Took a while for people to realize that new materials enabled new designs

What does this have to do with Computer Architecture?

- Synchronous circuits
  - Single-cycle, pipelined, superscalar, Tomasulo, etc
- Asynchronous circuits
  - Early async processors imitated synchronous architectures
  - FLEET is a first draft of what “suspension bridges” might look like

Berkeley EECS

Crash Course on Asynchronous Circuits

Berkeley EECS

KLA

- Kinetic Learning Activity

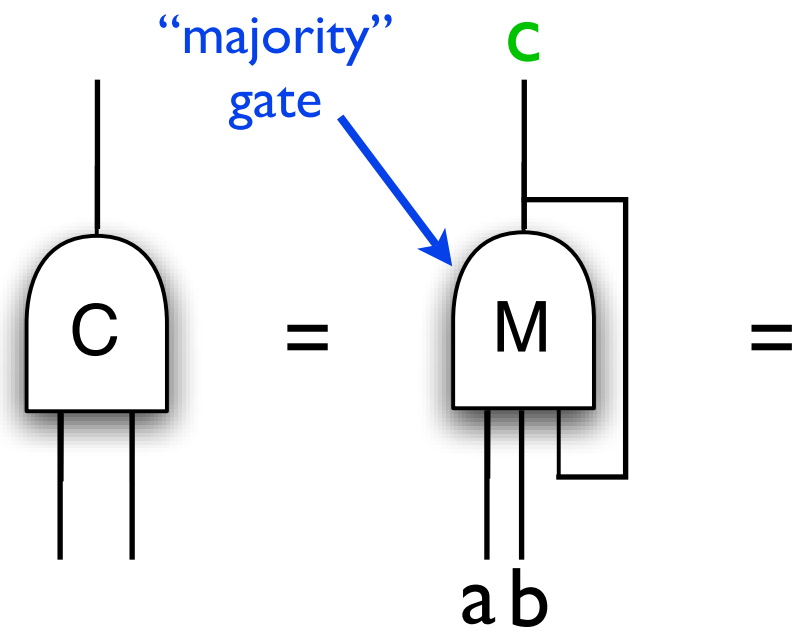
Berkeley EECS

```
IF predecessor != successor
    THEN
        copy predecessor
```

Berkeley EECS

Muller C Element

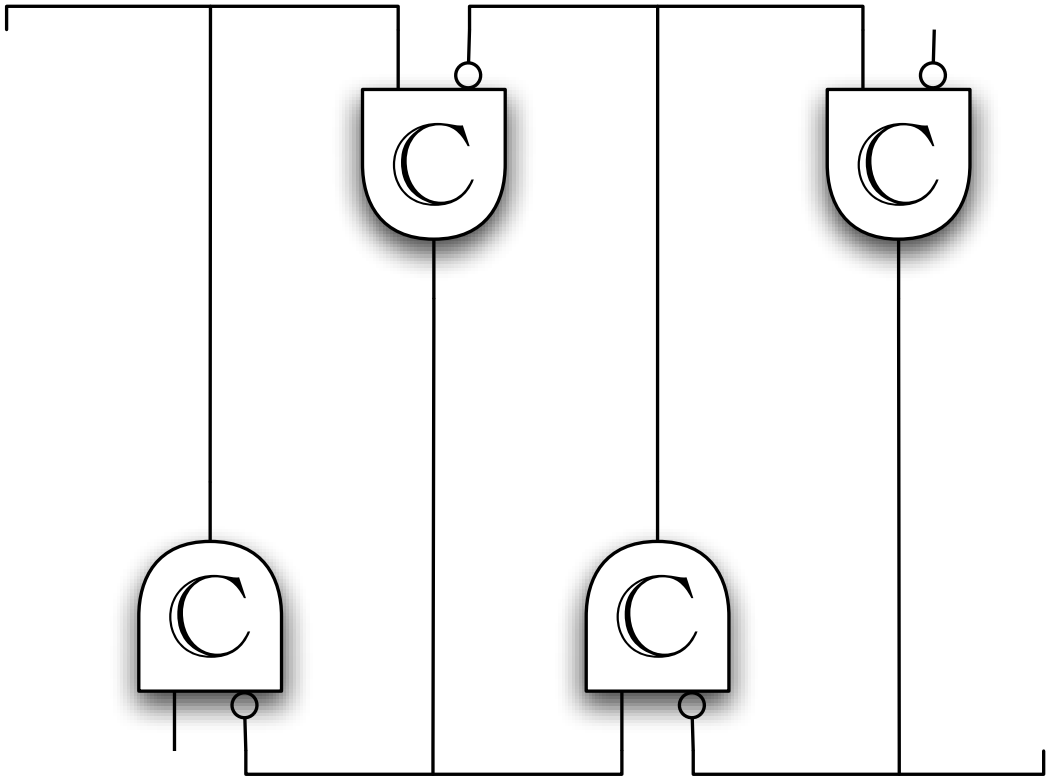
- Majority gate with output looped back
- A two-voter election with "incumbent advantage" in event of a tie



		a	
		0	1
b	0	0	c
	1	c	1

Micropipelines

- Chain of Muller C-Elements
- Each element connects to an inverted input on predecessor, non-inverted on successor

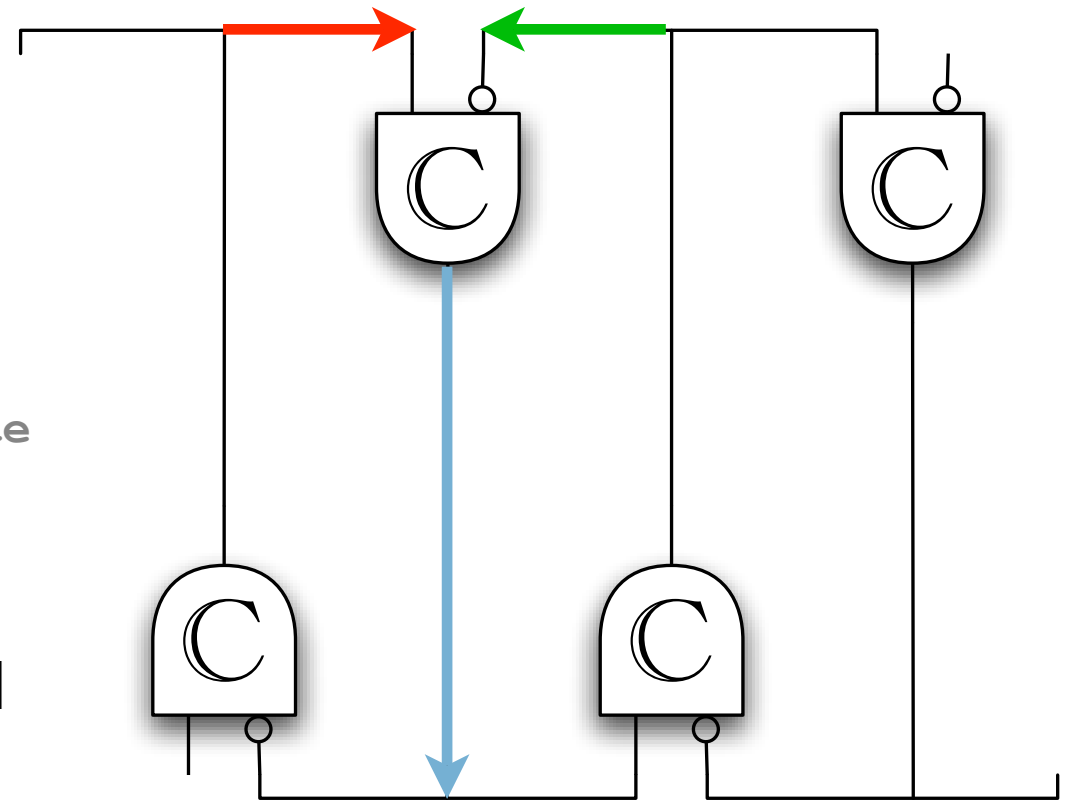


Micropipelines

"each stage of the control...follows a very simple stage state rule:

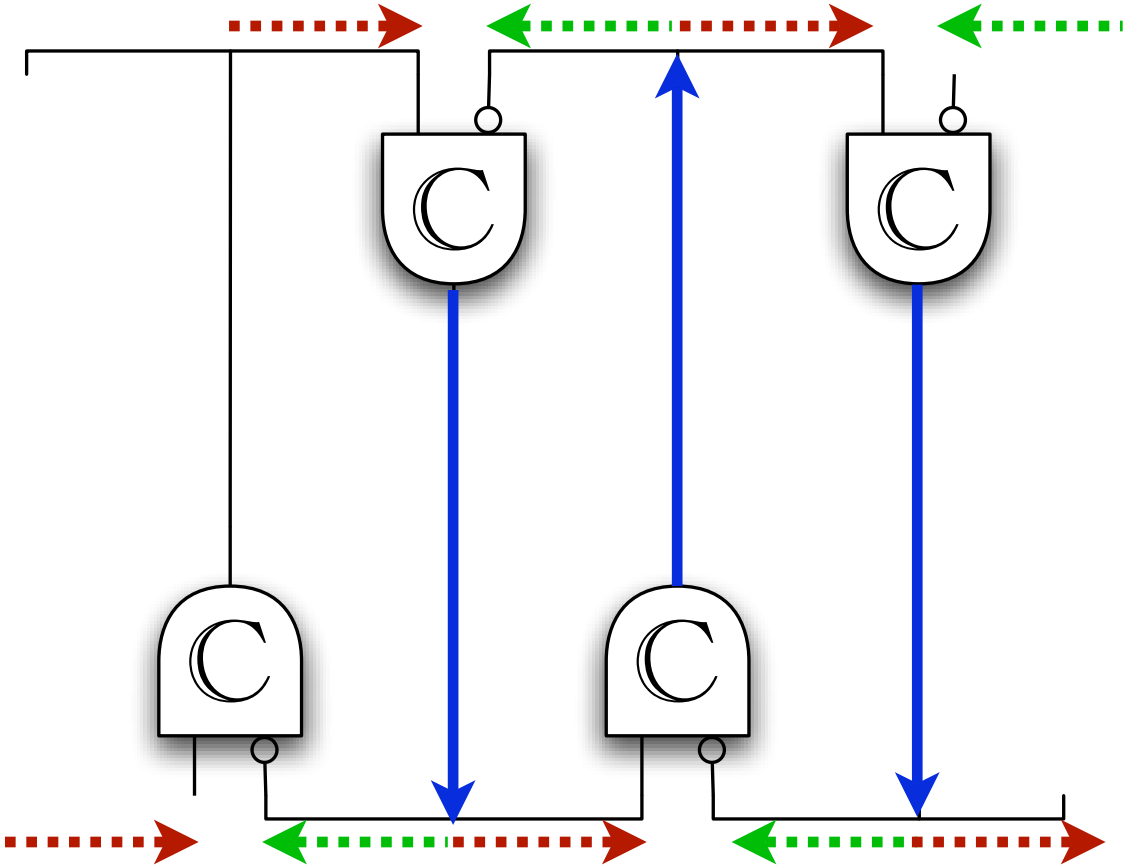
```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```

[IES'89]



Micropipelines

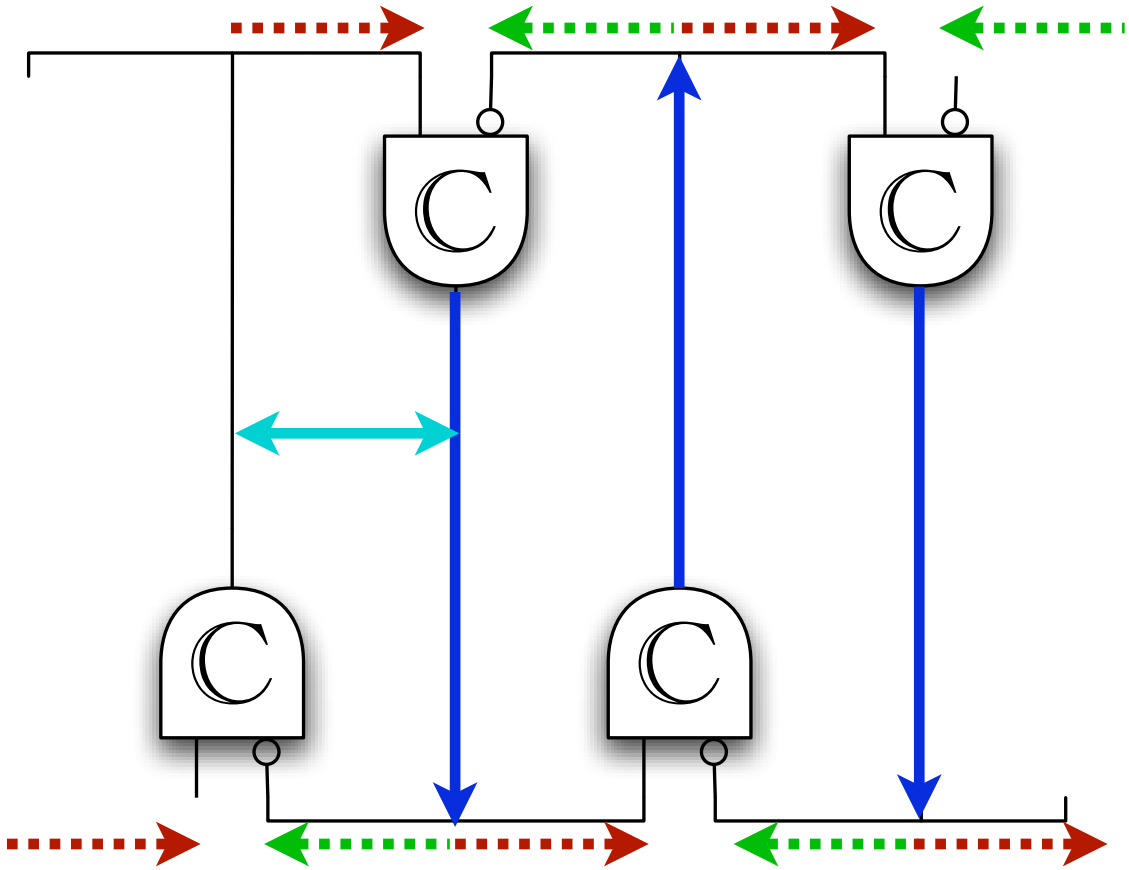
```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```



Micropipelines

```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```

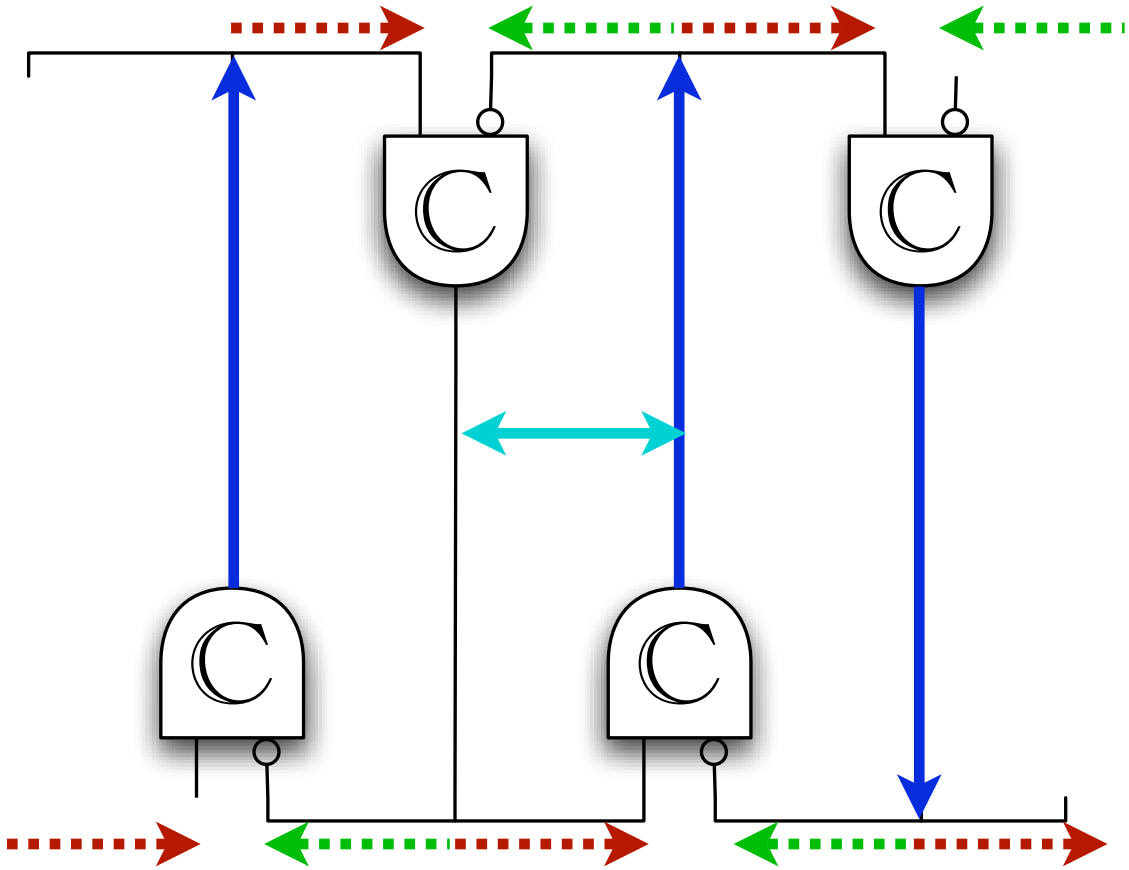
- Any "disagreements" propagate to the right



Micropipelines

```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```

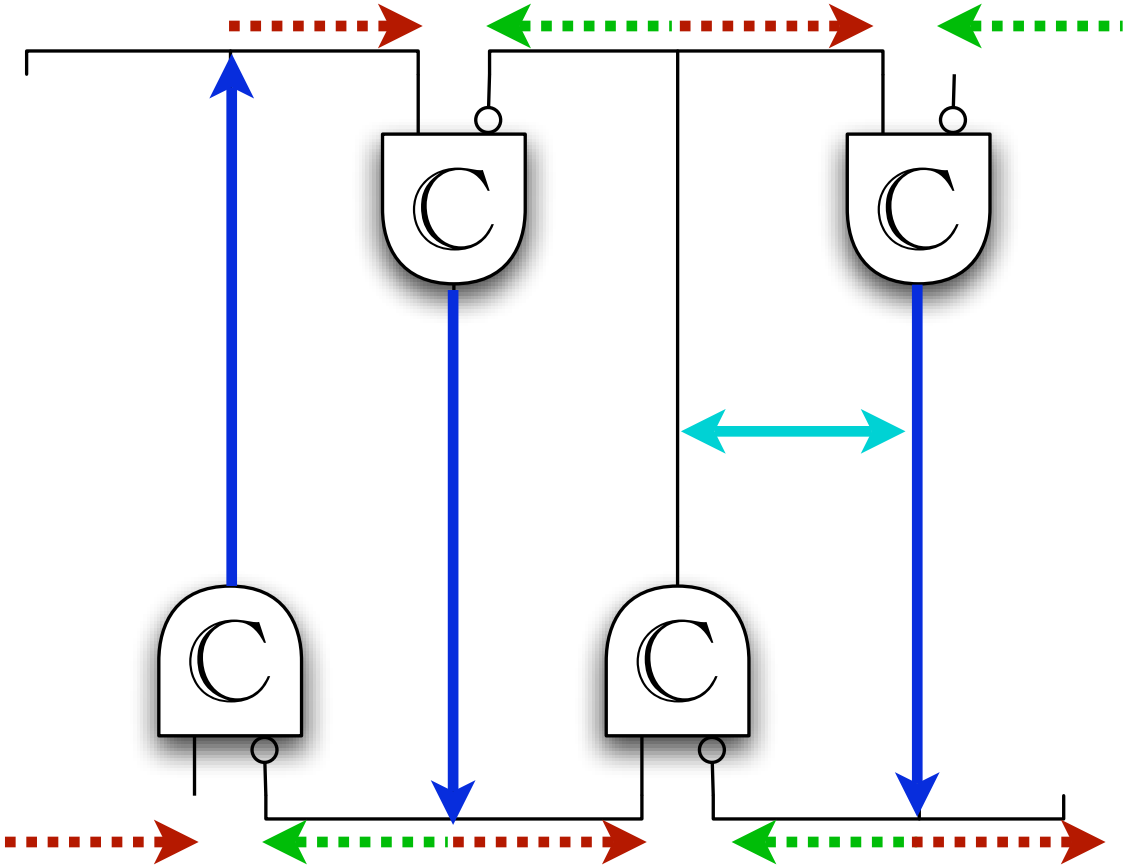
- Any "disagreements" propagate to the right



Micropipelines

```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```

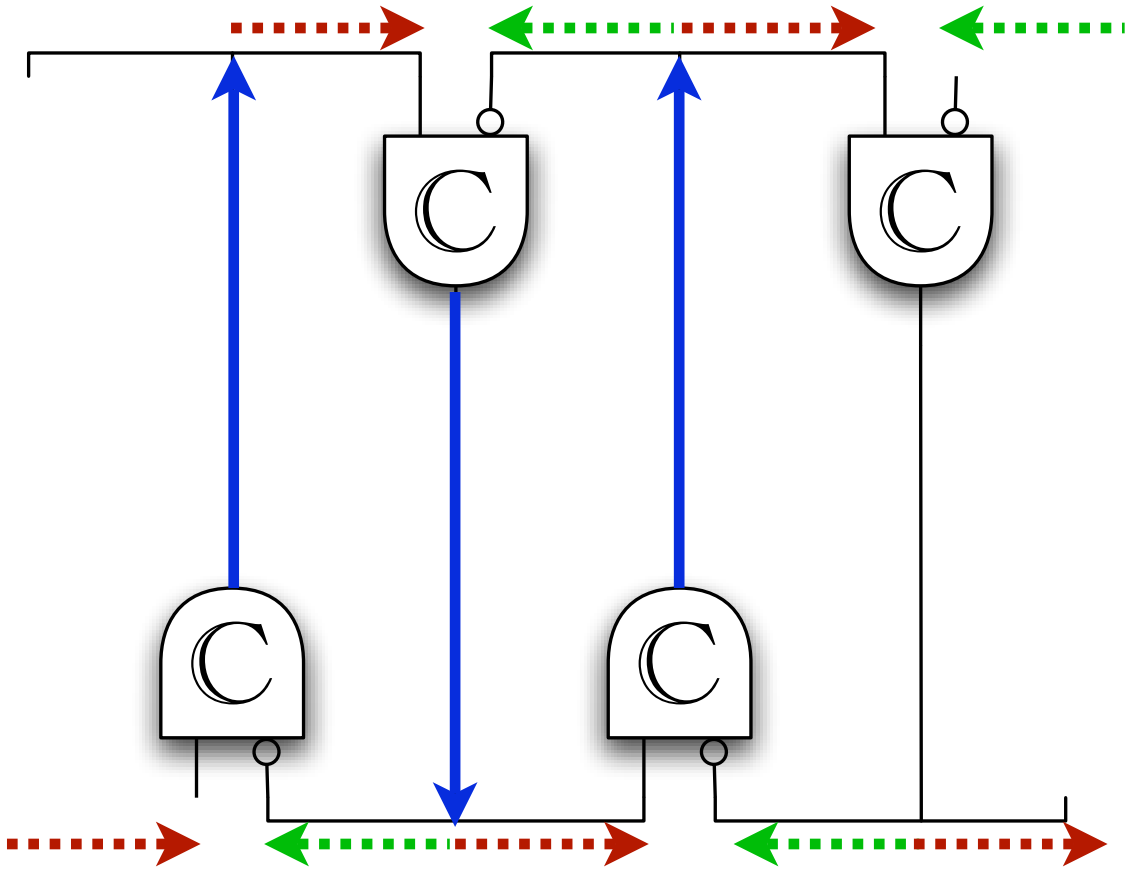
- Any “disagreements” propagate to the right



Micropipelines

```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```

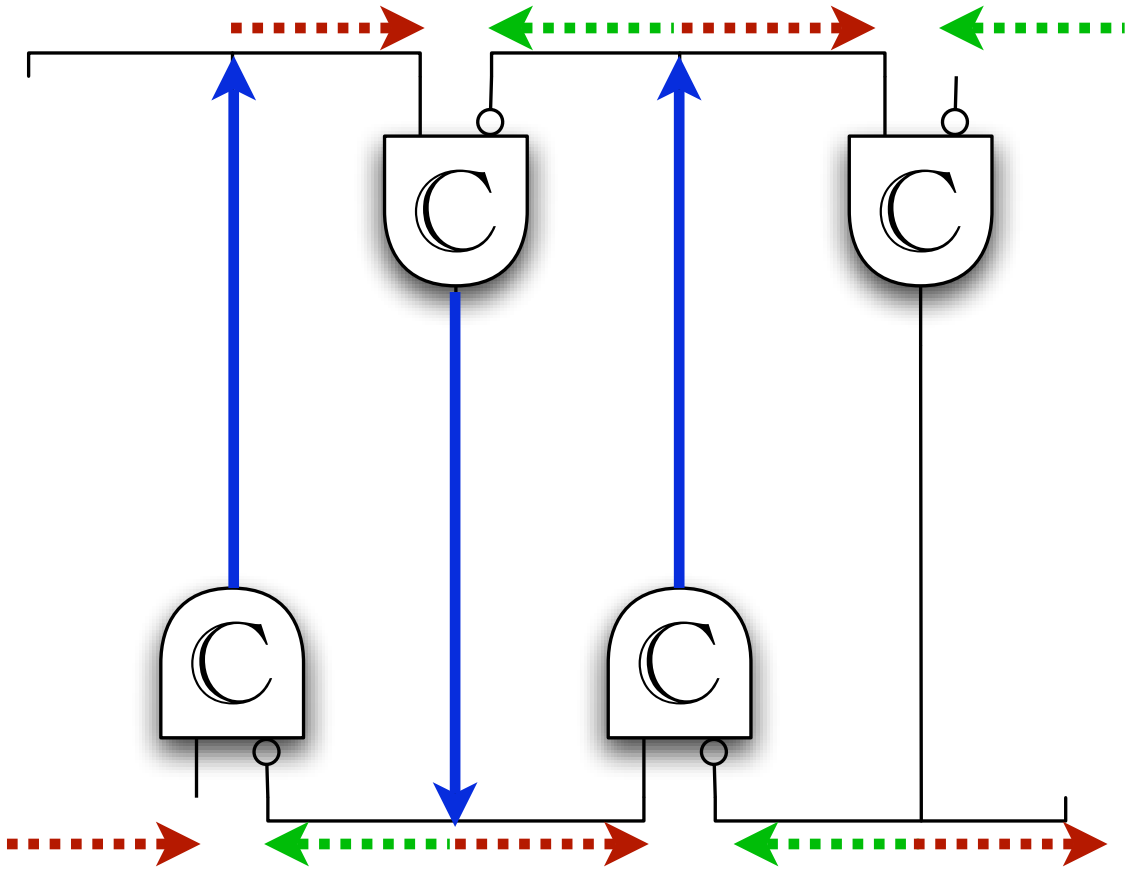
- Any “disagreements” propagate to the right



Micropipelines

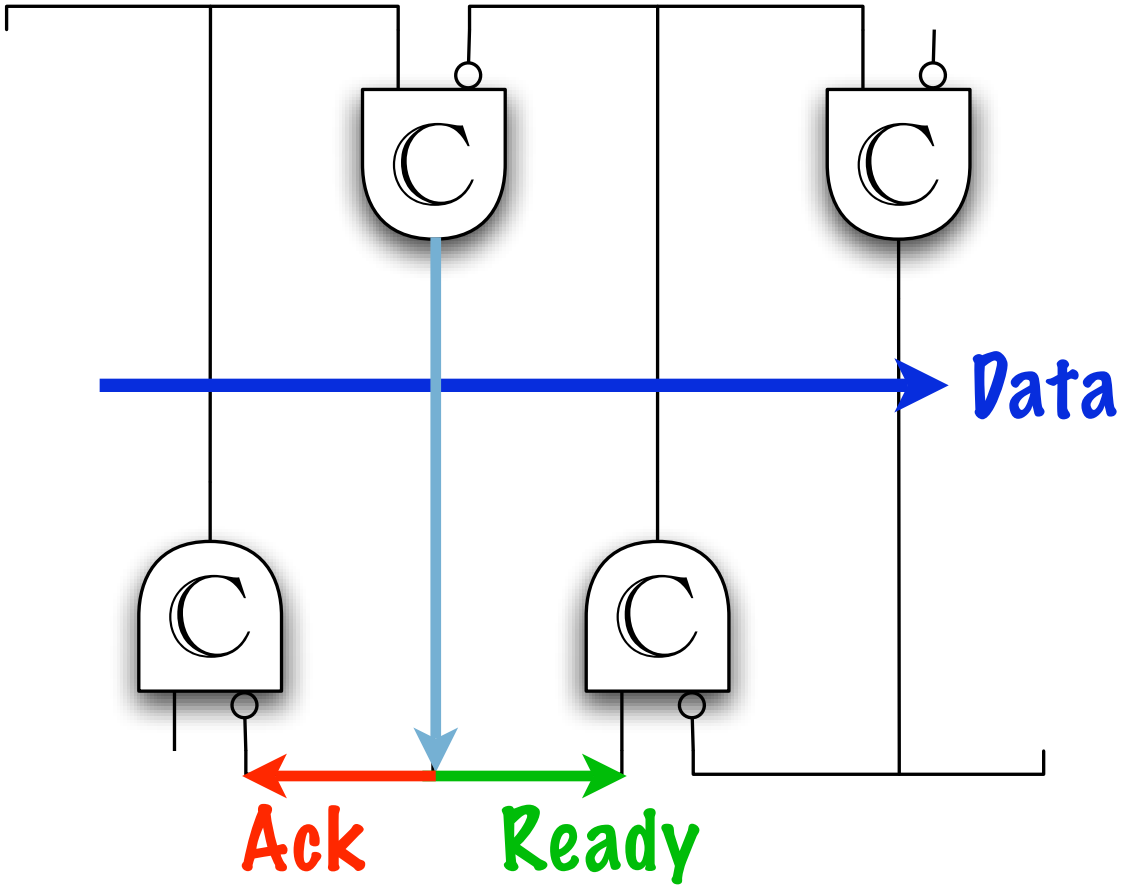
```
IF predecessor != successor
    THEN
      copy predecessor's state
    ELSE
      hold present state
```

- Any "disagreements" propagate to the right
- System is stable when pipeline contains
  - Zero or more agreeing stages
  - Followed by zero or more disagreeing stages



Micropipelines

- Transition signaling
- A transition on a stage's output will:
  - Acknowledge data from the previous segment
  - Signal data ready to next segment



Other Styles Exist

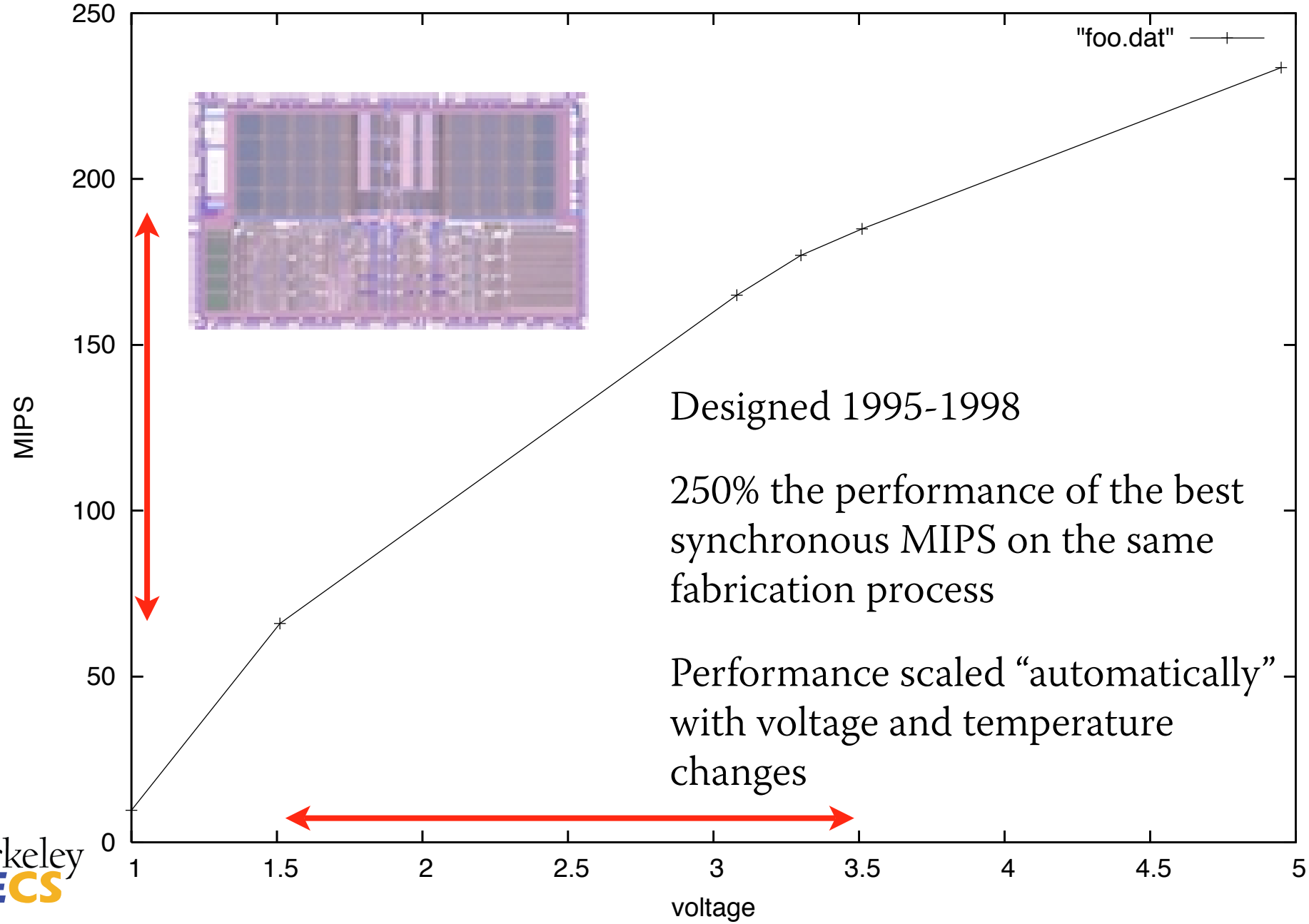
- Too many for this talk

Advantages of Async

- Average (not worst) case timing -- no “timing closure”
- Early completion with some circuit styles
- Separates correctness from performance
- No clock tree
- Low power
- Better modularity
- Better EMI profile
- Extremely robust to variations Process, Temperature, Voltage

Berkeley EECS

Caltech MiniMIPS



Designed 1995-1998

250% the performance of the best synchronous MIPS on the same fabrication process

Performance scaled "automatically" with voltage and temperature changes

Berkeley EECS

Communication Matters

- "Metal rod" model of VLSI wires
- Communication is what matters, so put the programmer in charge of it
- One instruction: MOVE

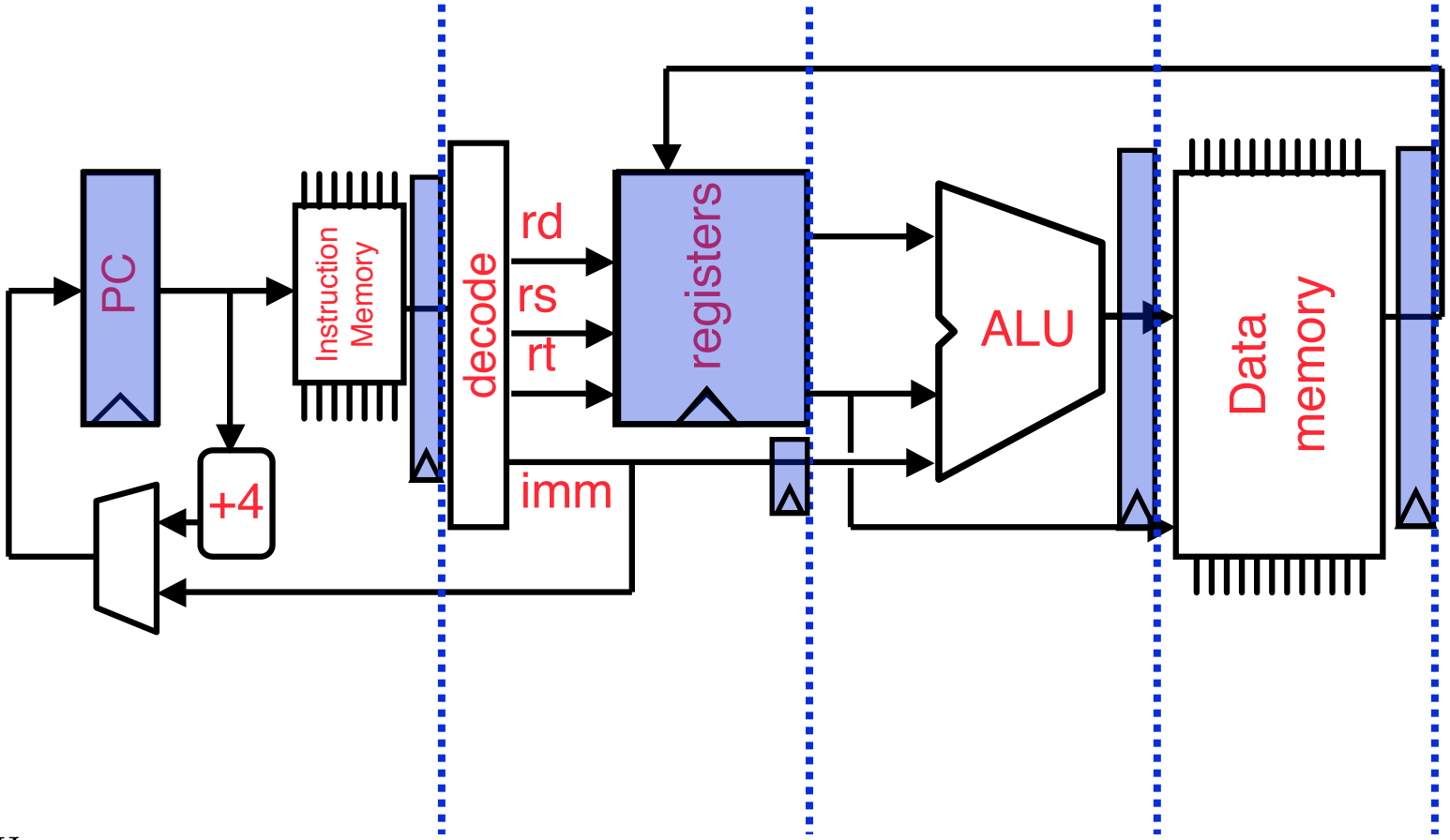
Berkeley EECS

Mike Holenderski's Animation

http://research.cs.berkeley.edu/class/fleet/docs/fleet_animation.swf

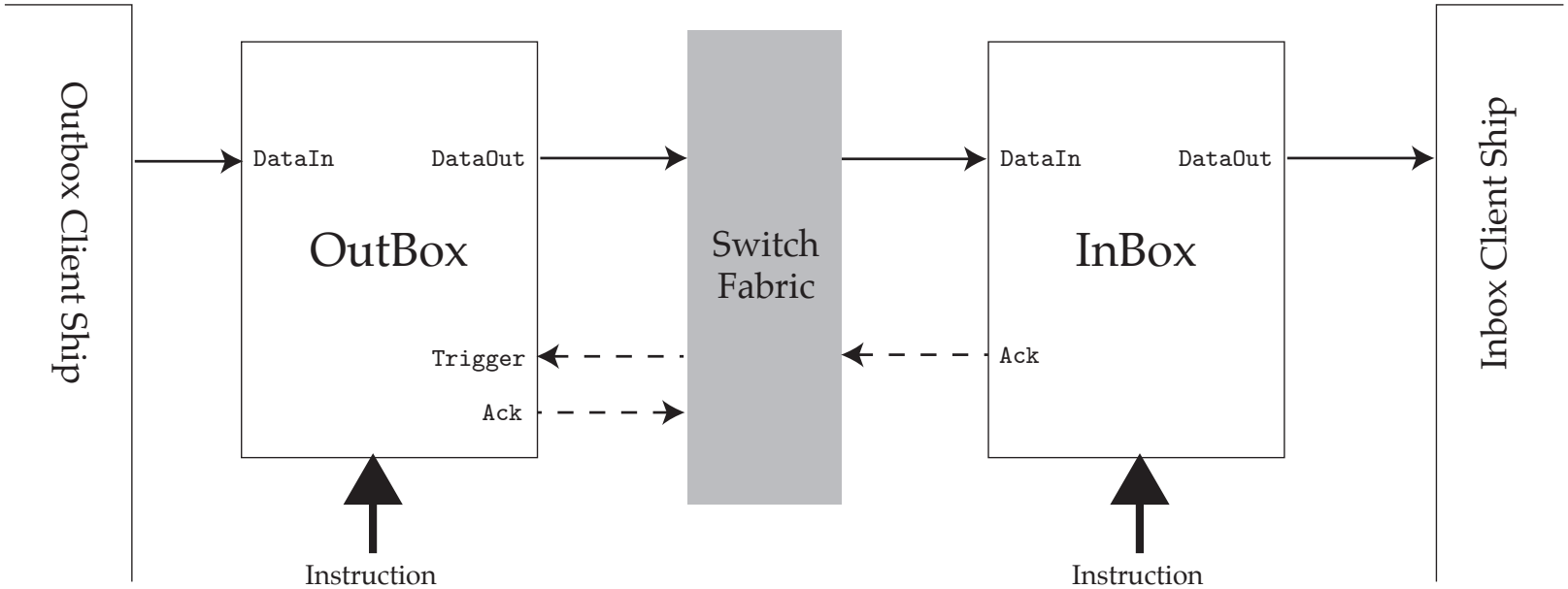
Berkeley EECS

MIPS Pipeline vs FLEET SHIPs



Berkeley EECS

Inboxes and Outboxes



Inboxes and Outboxes

Code "Bags"

- Unordered sets of instructions
- Lets us start fetching the next "block" of code as soon as the current "block" starts executing
  - Unlike sequential-instruction ISAs

Anatomy of a FLEET Instruction

Source Address	DataIn Ignore/Copy/ Take	Triggered	Count	Ack	DataOut	Destination Address
$\log_2$ NUMSOURCES	1.5	1	$\log_2$ MAXMOVE	1	1	$\log_2$ NUMDESTINATIONS

Anatomy of a FLEET Instruction

Berkeley EECS

Instruction Forms

Triggered →

Not Triggered →

	DataIn Ignore	DataIn Copy	DataIn Take
	nop	wait	discard
Ack	nop+ack	wait+ack	discard+ack
DataOut		copy	move *(or accept)*
DataOut +Ack		copy+ack	move+ack *(or accept+ack)*

Berkeley EECS

Synchronous SHIPs

- The fact that the switch fabric is asynchronous is fairly essential to FLEET
- However, SHIPs can be internally synchronous
- Pauseable clocks are an attractive option

Berkeley EECS

More Information

- Website:
  - http://research.cs.berkeley.edu/class/fleet/
- Graduate seminar meets in this room tomorrow, 10am-noon
  - Project presentations are a week from tomorrow

Berkeley EECS