CS184a: Computer Architecture (Structures and Organization)

Day4: October 4, 2000

Memories, ALUs, and Virtualization

Last Time

- Arithmetic: addition, subtraction
- Reuse:
  - pipelining
  - bit-serial (vectorization)
  - shared datapath elements
- FSMDs
- Area/Time Tradeoffs
- Latency and Throughput

Today

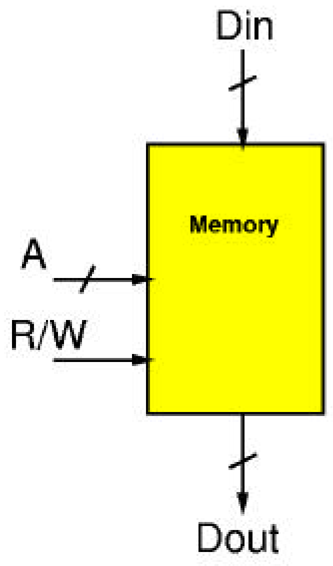
- Memory
  - features
  - design
  - technology
  - impact on computability
- ALUs
- Virtualization

Memory

- What's a memory?
- What's special about a memory?

Memory Function

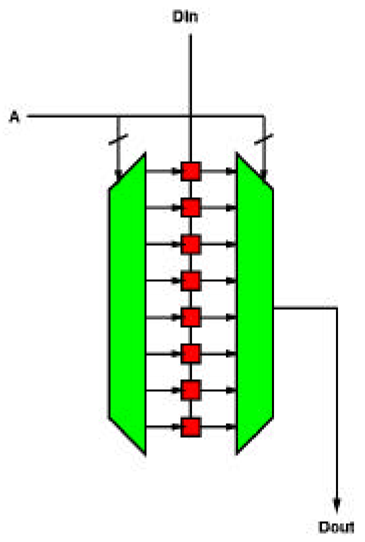
- Typical:
  - Data Input Bus
  - Data Output Bus
  - Address
    - (location or name)
  - read/write control



Memory

- Block for storing data for later retrieval
- State element
- What’s different between a memory and a collection of registers like we’ve been discussing?

Collection of Registers

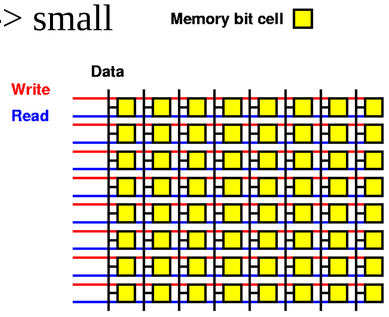


Memory Uniqueness

- Cost
- Compact state element
- Packs data very tightly
- At the expense of sequentializing access
- Example of Area-Time tradeoff
  - and a key enabler

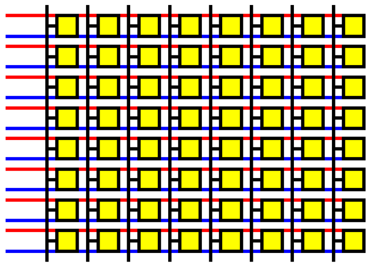
Memory Organization

- Key idea: sharing
  - factor out common components among state elements
  - can have big, elements if amortize costs
  - state element unique -> small



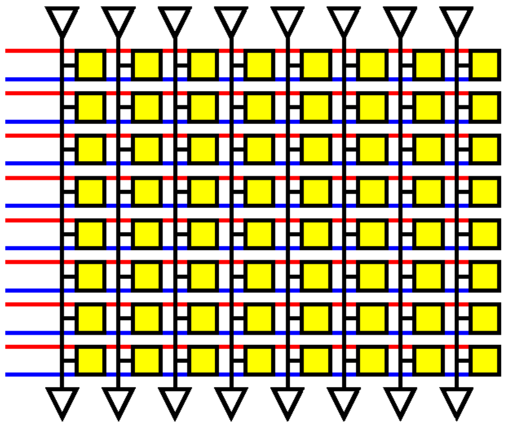
Memory Organization

- Share: Interconnect
  - Input bus
  - Output bus
  - Control routing
- VERY topology/wire cost aware design
- Note local, abuttment wiring



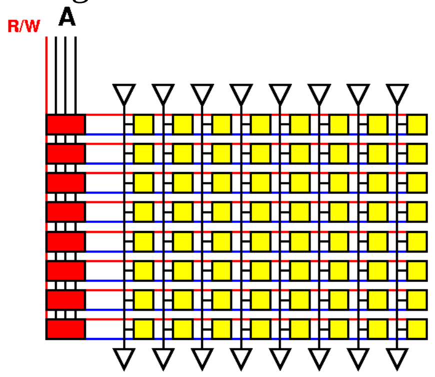
Share Interconnect

- Input Sharing
  - wiring
  - drivers
- Output Sharing
  - wiring
  - sensing
  - driving

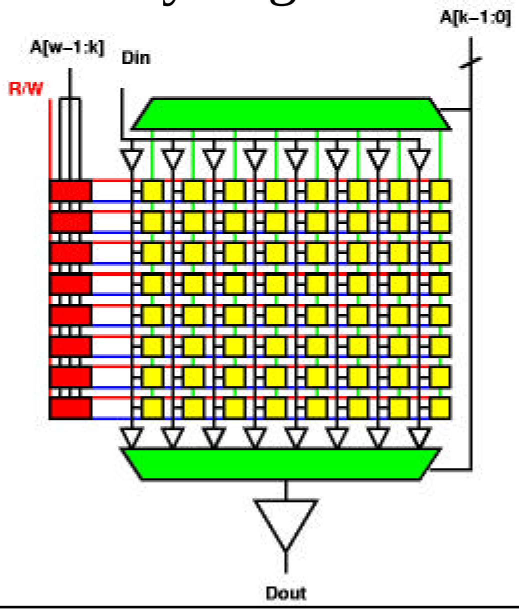


Address/Control

- Addressing and Control
  - an overhead
  - paid to allow this sharing

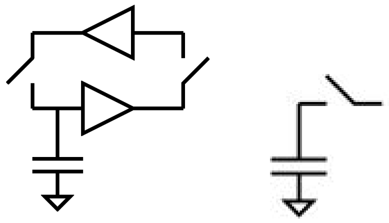


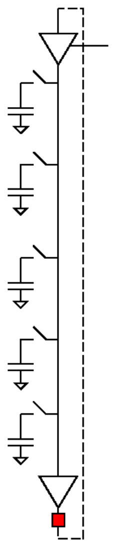
Memory Organization



Dynamic RAM

- Goes a step further
- Share refresh/restoration logic as well
- Minimal storage is a capacitor
- "Feature" DRAM process is ability to make capacitors efficiently



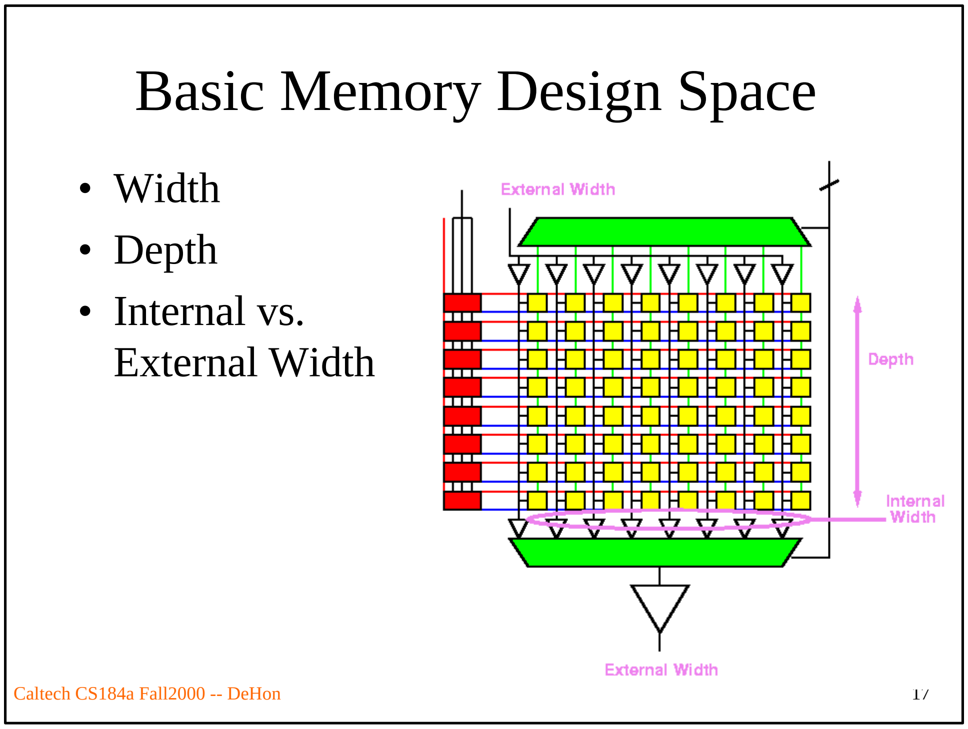


Some Numbers (memory)

- Unit of area = λ^2
  - [more next time]
- Register as stand-alone element ≈ $4K\lambda^2$
  - *e.g.* as needed/used last two lectures
- Static RAM cell ≈ $1K\lambda^2$
  - SRAM Memory (single ported)
- Dynamic RAM cell (DRAM process) ≈ $100\lambda^2$
- Dynamic RAM cell (SRAM process) ≈ $300\lambda^2$

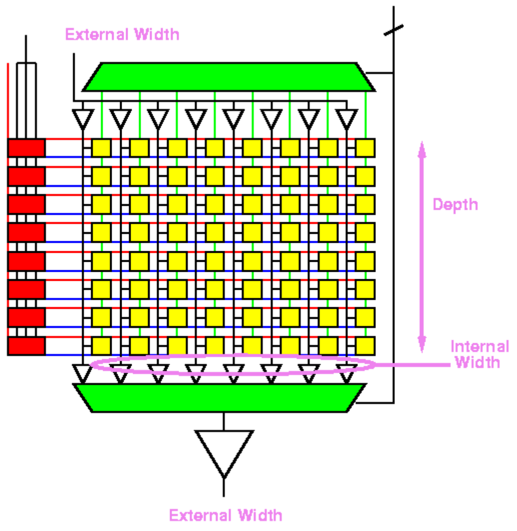
Memory

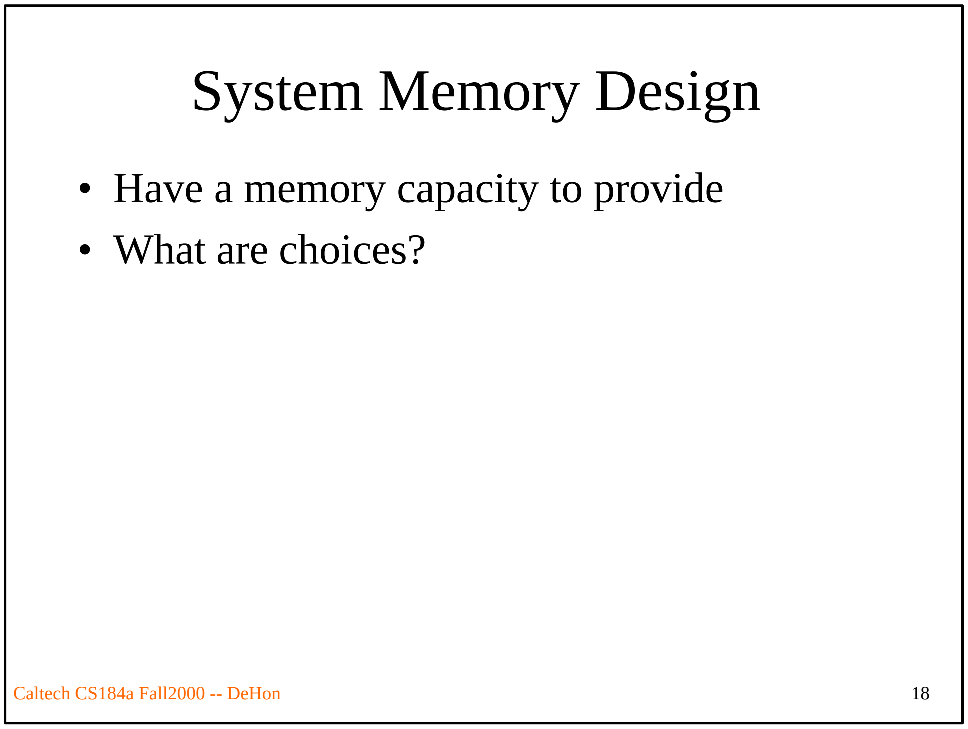
- Key Idea
  - Memories hold state compactly
  - Do so by minimizing key state storage and amortizing rest of structure across large array



Basic Memory Design Space

- Width
- Depth
- Internal vs. External Width





System Memory Design

- Have a memory capacity to provide
- What are choices?

System Memory Design

- One monolithic memory?
  - Internal vs. external width
  - internal banking
- External width
- Separate memory banks (address ports)

Yesterday vs. Today (Memory Technology)

- What's changed?

Yesterday vs. Today (Memory Technology)

- What's changed?
  - Capacity
    - single chip
  - Integration
    - memory and logic
    - dram and logic
    - embedded memories
  - Room on chip for big memories
  - Don't have to make a chip crossing to get to memory

Important Technology Cost

- IO between chips << IO on chip
  - pad spacing
  - area vs. perimeter (4s vs. s^2)
  - wiring technology
- BIG factor in multi-chip system designs
- Memories nice
  - very efficient with IO cost vs. internal area

Costs Change

- Design space changes when whole system goes on single chip
- Can afford
  - – wider busses
  - – more banks
  - – memory tailored to application/architecture
- Beware of old (stale) answers
  - – their cost model was different

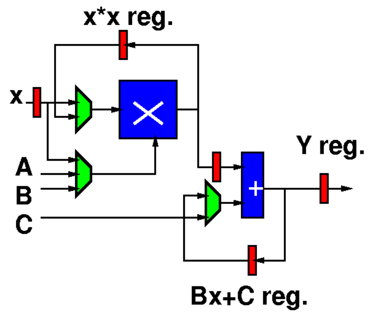
What is Importance of Memory?

- Radical Hypothesis:
  - – Memory is simply a very efficient organization which allows us to store data compactly
    - (at least, in the technologies we've seen to date)
  - – A great engineering trick to optimize resources
- Alternative:
  - – memory is a **primary**

Sharing

Last Time

- Given a task: $y=Ax^2 +Bx +C$
- Saw how to share primitive operators
- Got down to one of each



Very naively

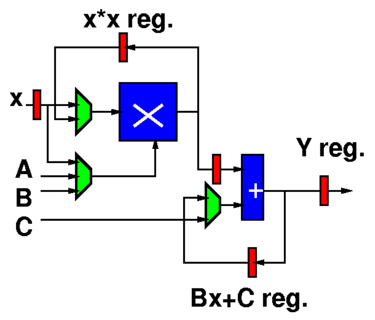
- Might seem we need one of each different type of operator

..But

- Doesn’t fool us
- We already know that nand gate (and many other things) are universal
- So, we know, we can build a universal compute operator

This Example

- $y=Ax^2 +Bx +C$
- Know a single adder will do

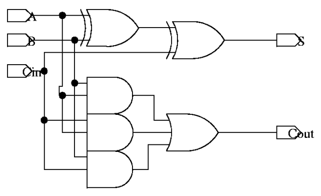


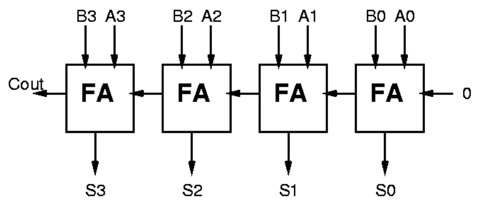
Adder Universal?

- Assuming interconnect:
  - (big assumption as we'll see later)
  - Consider:

A: 001a
B: 000b
S: 00cd

- What's c?



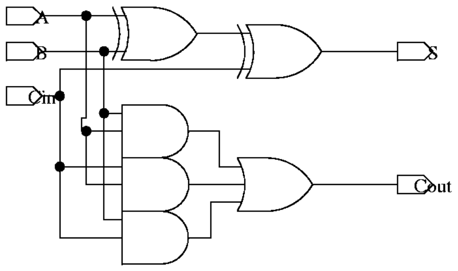


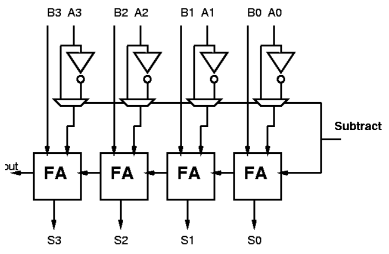
Practically

- To reduce (some) interconnect
- and to reduce number of operations
- do tend to build a bit more general "universal" computing function

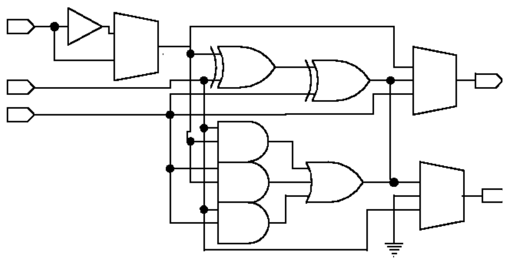
Arithmetic Logic Unit (ALU)

- Observe:
  – with small tweaks can get many functions with basic adder components

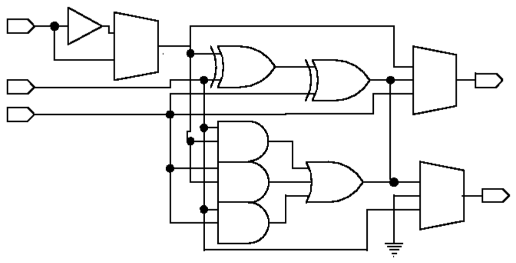




ALU



ALU Functions



- A+B w/ Carry
- B-A
- A xor B (squash carry)
- A*B (squash carry)
- /A
- B<<1

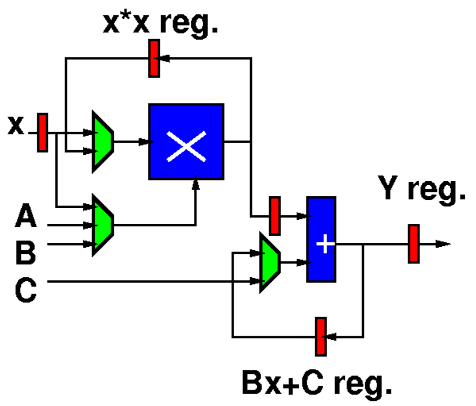
Table Lookup Function

- Observe 2: only 2^{2^3}=256 functions of 3 inputs
  - 3-inputs = A, B, carry in from lower
- Two, 3-input Lookup Tables
  - give all functions of 2-inputs and a cascade
  - 8b to specify function of each lookup table
- LUT = LookUp Table

What does this mean?

- With only one active component
  - ALU, nand gate, LUT
- Can implement any function
  - given appropriate
    - state registers
    - muxes (interconnect)
    - control

Revisit Example

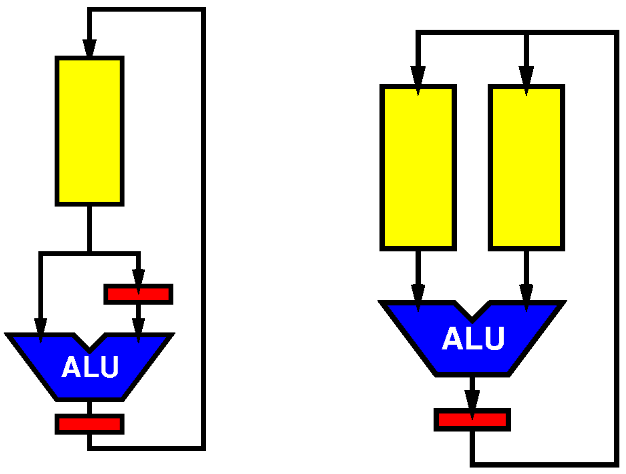


- We do see a proliferation of memory and muxes -- what do we do about that?

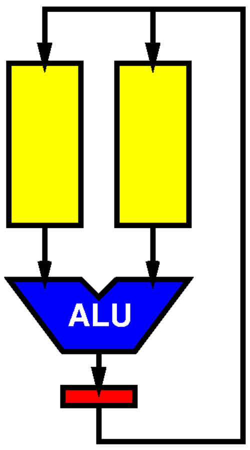
Back to Memories

- State in memory more compact than “live” registers
  – shared input/output/drivers
- If we’re sequentializing, only need one (few) at a time anyway
  – I.e. sharing compute unit, might as well share interconnect
- Shared interconnect also gives muxing function

ALU + Memory

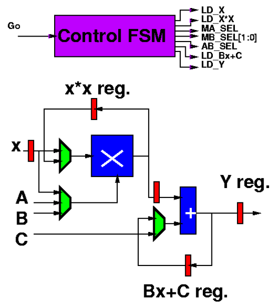


What’s left?



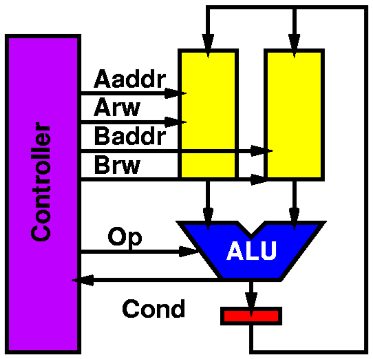
Control

- Still need that controller which directed which state, went where, and when
- Has more work now,
  - also say what operations for compute unit



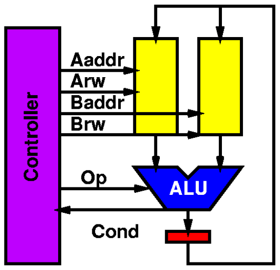
Implementing Control

- Implementing a single, Fixed computation
  - might still just build a custom FSM



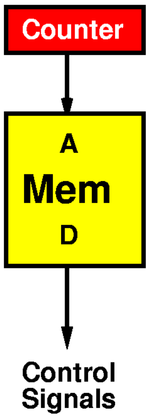
…and Programmable

- At this point, it’s a small leap to say maybe the controller can be programmable as well
- Then have a building block which can implement anything
  - within state and control programmability bounds

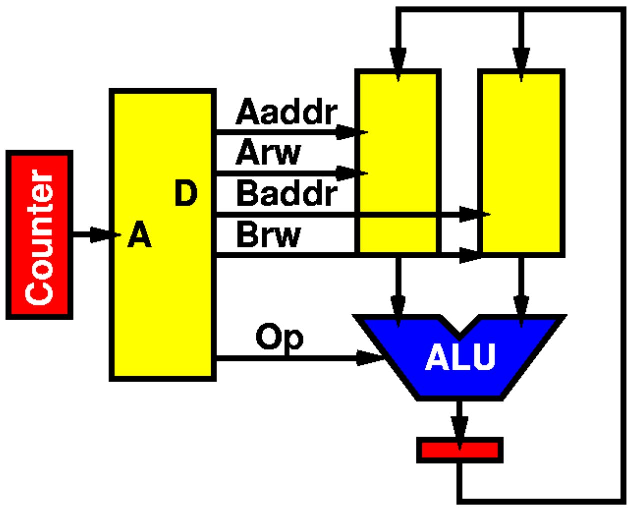


Simplest Programmable Control

- Use a memory to “record” control instructions
- “Play” control with sequence

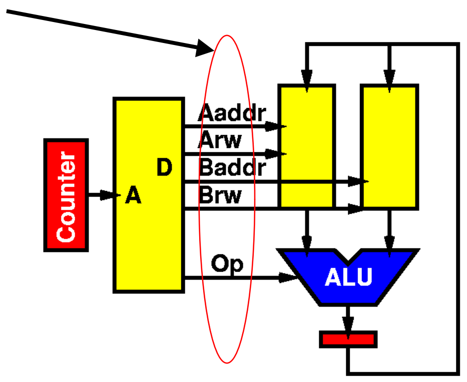


Our "First" Programmable Architecture



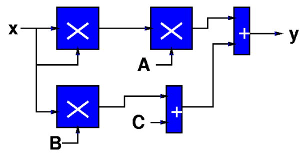
Instructions

- Identify the bits which control the function of our programmable device as:
  - *Instructions*



What have we done?

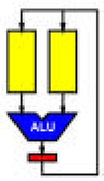
- Taken a computation: $y=Ax^2+Bx+C$
- Turned it into operators and interconnect



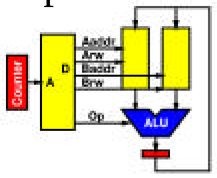
- Decomposed operators into a basic primitive: Additions, ALU, ...nand

What have we done?

- Said we can implement it on as few as one of these
- Added a unit for state



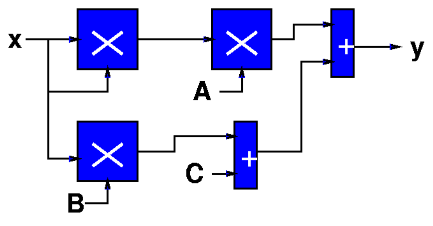
- Added an instruction to tell single, universal unit how to act as each operator in original graph

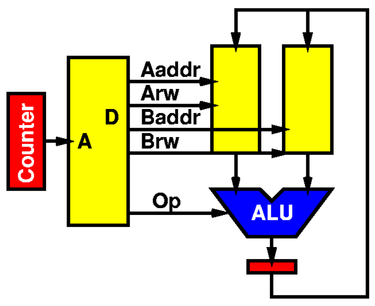


Virtualization

- We've virtualized the computation
- No longer need one physical compute unit for each operator in original computation
- Can suffice with shared operator(s)
- and a description of how each operator behaved
- and a place to store the intermediate data between operators

Virtualization





Why Interesting?

- Memory compactness
- This works and was interesting because
  - the area to describe a computation, its interconnect, and its state
  - is much smaller than the physical area to spatially implement the computation
- *e.g.* traded multiplier for
  - few memory slots to hold state
  - few memory slots to describe operation
  - time on a shared unit (ALU)

Finishing up

- Coming Attractions
- Administrivia
- Big Ideas
  - MSB
  - MSB-1

Coming Attractions: Three Talks by Tom Knight

- Thursday 4pm (102 Steele)
  - Robust Computation with Capabilities and Data Ownership (computer architecture)
- This Fri 4pm (102 Steele)
  - Reversibility in Digital, Analog, and Neural Computation (physics of computation)
- Next Mon 3pm (Beckman Institute Auditorium)
  - Computing with Life (biological computers)

Administrative

- CS184 mailing list -- sent test message
  - if didn't receive, you should mail me
- CS184 questions:
  - please put CS184 in the subject line
- Homework Instructions:
  - read the info handout!
- Course web page
- Comment Reading

Caltech CS

- Want to talk with undergrads about
  - department
  - classes
  - great CS community
  - …concentration/major…
- Fora (plural of forum?)
  - small group discussion in houses?
  - small group dinner
  - ???

Big Ideas [MSB Ideas]

- Memory: efficient way to hold state
- State can be << computation [area]
- Resource sharing: key trick to reduce area
- Memories are a great example of resource sharing
- Memory key tool for Area-Time tradeoffs
- “configuration” signals allow us to generalize the utility of a computational operator

Big Ideas
[MSB-1 Ideas]

- Tradeoffs in memory organization
- Changing cost of memory organization as we go to on-chip, embedded memories
- ALUs and LUTs as universal compute elements
- First programmable computing unit