

CS184a: Computer Architecture (Structure and Organization)

Day 4: January 12, 2005
Memories....



Caltech CS184 Winter2005 -- DeHon

Last Time

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

Caltech CS184 Winter2005 -- DeHon

2

Today

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

Caltech CS184 Winter2005 -- DeHon

3

Memory

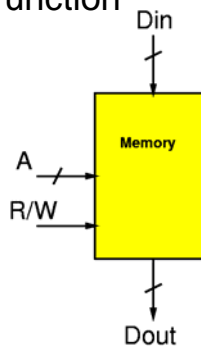
- What's a memory?
- What's special about a memory?

Caltech CS184 Winter2005 -- DeHon

4

Memory Function

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



Caltech CS184 Winter2005 -- DeHon

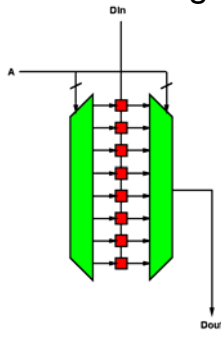
Memory

- 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?

Caltech CS184 Winter2005 -- DeHon

6

Collection of Registers



Caltech CS184 Winter2005 -- C

7

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

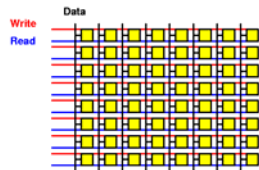
Caltech CS184 Winter2005 -- DeHon

8

Memory Organization

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

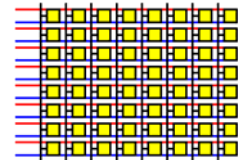
Memory bit cell



Caltech CS184 Winter2005 -- DeHon

Memory Organization

- **Share:** Interconnect
 - Input bus
 - Output bus
 - Control routing
- **very** topology/wire cost aware design
- **Note:** local, abutment wiring

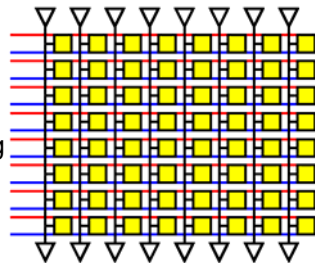


Caltech CS184 Winter2005 -- DeHon

10

Share Interconnect

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

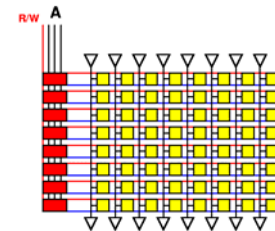


Caltech CS184 Winter2005 -- DeHon

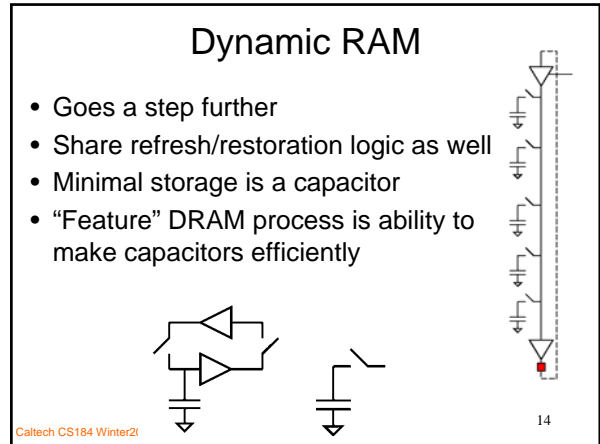
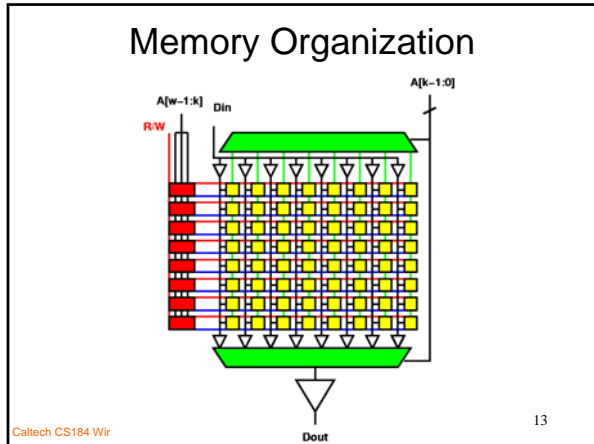
11

Address/Control

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

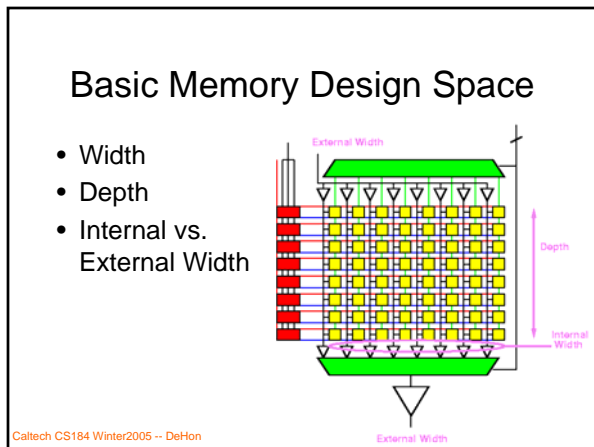


Caltech CS184 Winter2005 -- DeHon



- ### Some Numbers (memory)
- Unit of area = λ^2
 - [more next time]
 - Register as stand-alone element $\approx 4K\lambda^2$
 - e.g. as needed/used last two lectures
 - Static RAM cell $\approx 1K\lambda^2$
 - SRAM Memory (single ported)
 - Dynamic RAM cell (DRAM process) $\approx 100\lambda^2$
 - Dynamic RAM cell (SRAM process) $\approx 300\lambda^2$
- Caltech CS184 Winter2005 -- DeHon 15

- ### Memory
- **Key Idea**
 - Memories hold state compactly
 - Do so by minimizing key state storage and amortizing rest of structure across large array
- Caltech CS184 Winter2005 -- DeHon 16



- ### System Memory Design
- Have a memory capacity to provide
 - What are choices?
- Caltech CS184 Winter2005 -- DeHon 18

System Memory Design

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

Caltech CS184 Winter2005 -- DeHon

19

Yesterday vs. Today (Memory Technology)

- What's changed?

Caltech CS184 Winter2005 -- DeHon

20

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

Caltech CS184 Winter2005 -- DeHon

21

Important Technology Cost

- IO between chips \ll 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

Caltech CS184 Winter2005 -- DeHon

22

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

Caltech CS184 Winter2005 -- DeHon

23

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**

Caltech CS184 Winter2005 -- DeHon

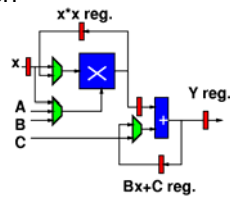
24

5/12/05 Lecture Ended Here

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

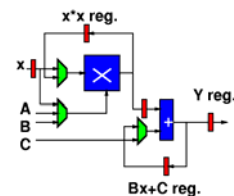
- Might seem we need one of each different type of operator

..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:

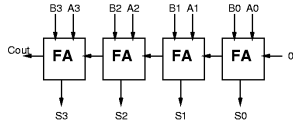
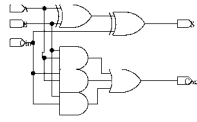
– (big assumption as we'll see later)

– Consider:

A: 001a

B: 000b

S: 00cd



Caltech CS184 Winter2005 -- DeHon

- 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

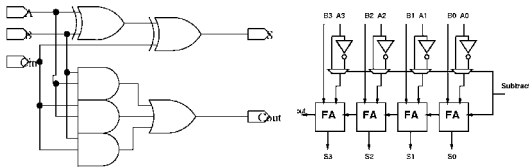
Caltech CS184 Winter2005 -- DeHon

32

Arithmetic Logic Unit (ALU)

- Observe:

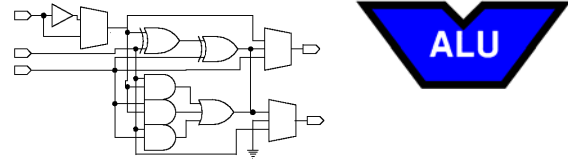
– with small tweaks can get many functions with basic adder components



Caltech CS184 Winter2005 -- DeHon

33

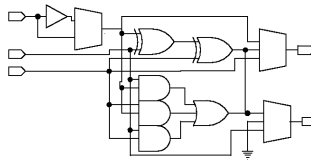
ALU



Caltech CS184 Winter2005 -- DeHon

34

ALU Functions



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

Caltech CS184 Winter2005 -- DeHon

35

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

Caltech CS184 Winter2005 -- DeHon

36

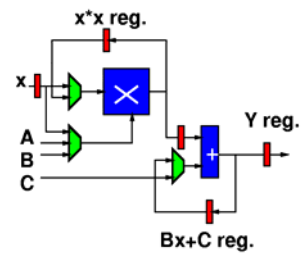
What does this mean?

- With only one active component
 - ALU, **nand** gate, LUT
- Can implement **any** function
 - given appropriate
 - state registers
 - muxes (interconnect)
 - Control
- Compare: Universal Turing Machine

Caltech CS184 Winter2005 -- DeHon

37

Revisit Example



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

Caltech CS184 Winter2005 -- DeHon

38

Virtualization

Caltech CS184 Winter2005 -- DeHon

39

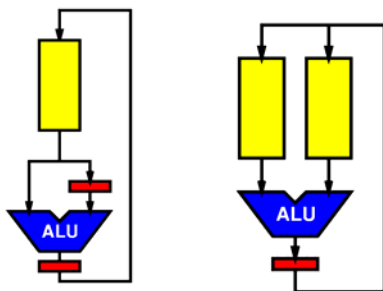
Back to Memories

- 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

Caltech CS184 Winter2005 -- DeHon

40

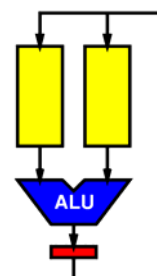
ALU + Memory



Caltech CS184 Winter2005 -- DeHon

41

What's left?

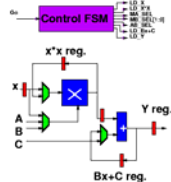


Caltech CS184 Winter2005 -- DeHon

42

Control

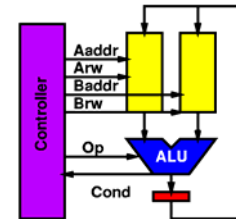
- Still need that controller which directed which state, went where, and when
- Has more work now,
 - also say what operations for compute unit



Caltech CS184 Winter2005 -- DeHon

Implementing Control

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

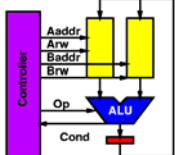


44

Caltech CS184 Winter2005 -- DeHon

...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



Caltech CS184 Winter2005 -- DeHon

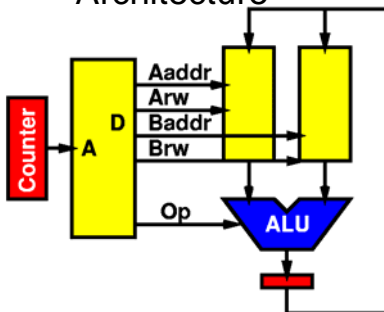
Simplest Programmable Control

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



Caltech CS184 Winter2005 -- DeHon

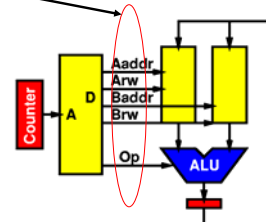
Our “First” Programmable Architecture



Caltech CS184

Instructions

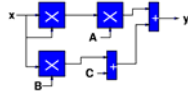
- Identify the bits which control the function of our programmable device as:
 - *Instructions*



Caltech CS184 Winter2005 -- DeHon

What have we done?

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



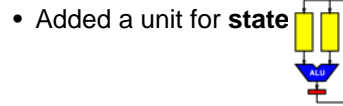
- Decomposed operators into a basic primitive: Additions, ALU, ...nand

Caltech CS184 Winter2005 -- DeHon

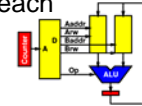
49

What have we done?

- Said we can implement it on as few as one of **compute unit** {ALU, LUT, nand}



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



Caltech CS184 Winter2005 -- DeHon

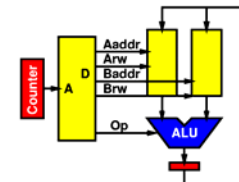
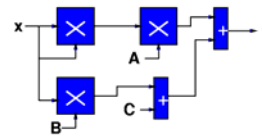
Virtualization

- 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

Caltech CS184 Winter2005 -- DeHon

51

Virtualization



Caltech CS184 Winter2005 -- DeHon

52

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)

Caltech CS184 Winter2005 -- DeHon

53

Admin Comments

- Do have class on Friday
- Reading for next Wed. is online

- From Info handout:

Writeups should be done in electronic form, using CAD or drawing tools where appropriate. Electronic submission will be preferred (and may be required for some assignments).

Caltech CS184 Winter2005 -- DeHon

54

Big Ideas [MSB Ideas]

- Memory: efficient way to hold state
- State can be \ll 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

Caltech CS184 Winter2005 -- DeHon

55

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

Caltech CS184 Winter2005 -- DeHon

56