

# *In Silico Vox:* Towards Speech Recognition in Silicon

Edward C. Lin, Kai Yu, *Rob A. Rutenbar*, Tsuhan Chen  
Electrical & Computer Engineering  
{eclin, kaiy, rutenbar, tsuhan}@ece.cmu.edu

© R.A. Rutenbar 2006

CarnegieMellon

## Speech Recognition Today



■ Quality = *OK*    Vocab = *large*

■ Quality = *poor*    Vocab = *small*

■ **Commonality: all software apps**

© Rob A. Rutenbar 2006

Slide 2

## Today's Best *Software* Speech Recognizers

- **Best-quality recognition is computationally *hard***
  - ▼ For speaker-independent, large-vocabulary, continuous speech
  
- **1-10-100-1000 rule**
  - ▼ For ~**1X** real-time recognition rate
  - ▼ For ~**10%** word error rate (90% accuracy)
  - ▼ Need ~**100 MB** memory footprint
  - ▼ Need ~**100 W** power
  - ▼ Need ~**1000 MHz** CPU
  
- **This proves to be very *limiting* ...**

## The Carnegie Mellon *In Silico Vox* Project

- **The thesis: It's time to liberate speech recognition from the unreasonable limitations of software**
  
- **The solution: *Speech recognition in silicon***
  
- **Why...?**
  - ▼ Tomorrow's compelling apps need **100X – 1000X** performance improvements to accomplish. (Not going to happen in software)
  
  - ▼ We have some successful **historical** examples of this migration

## History: Graphics Engines

- **Nobody paints pixels in software anymore!**

- ▼ Too limiting in max performance. Too inefficient in power.

True on the desktop (& laptop)



<http://www.nvidia.com>

...and on your cellphone too



<http://www.mtekvision.com>

© Rob A. Rutenbar 2006

Slide 5

## Next-Gen Compelling Applications

Audio-mining

- **Very fast** recognizers – much faster than realtime
- **App:** search large media streams (DVD) quickly

FIND: "Hasta la vista, baby!"



© Rob A. Rutenbar 2006

Hands-free appliances

- **Very portable** recognizers – high quality result on << 1 watt
- **App:** interfaces to small devices, cellphone dictation



"send email to arnold - let's do lunch..."

Slide 6

## Our Focus: How to Get to *Fast...*

### Audio-mining

- **Very fast recognizers** – much faster than realtime
- **App: search large media streams (DVD) quickly**

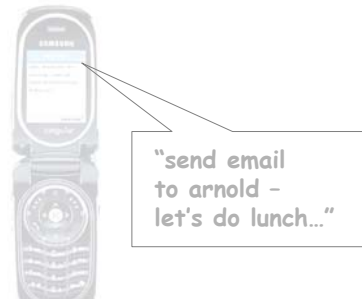
FIND: "Hasta la vista, baby!"



© Rob A. Rutenbar 2006

### Hands-free appliances

- **Very portable recognizers** – high quality result on  $\ll 1$  watt
- **App: interfaces to small devices, cellphone dictation**



Slide 7

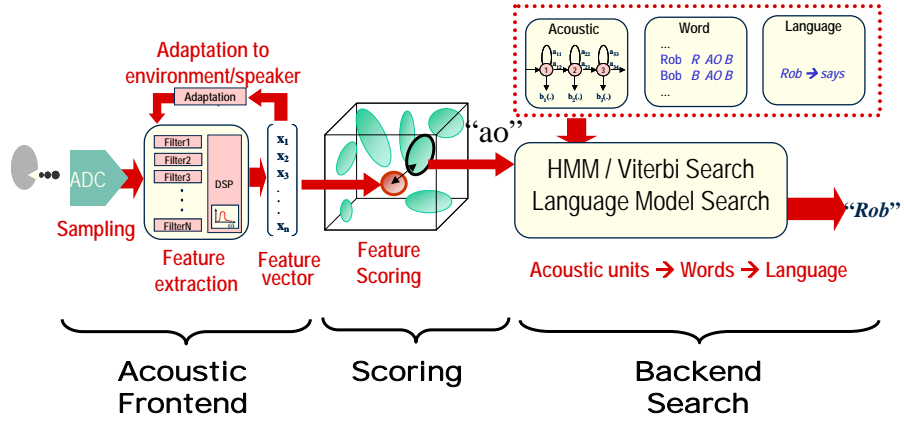
## About This Talk

- **The \$2 tour: How speech recognition works**
  - ▼ What happens in a recognizer
- **An ASIC architecture**
  - ▼ Stripping away all CPU stuff we don't need, focus on essentials
- **Results**
  - ▼ ASIC version: Cycle simulator results
  - ▼ FPGA version: Live, running hardware-based recognizer

© Rob A. Rutenbar 2006

Slide 8

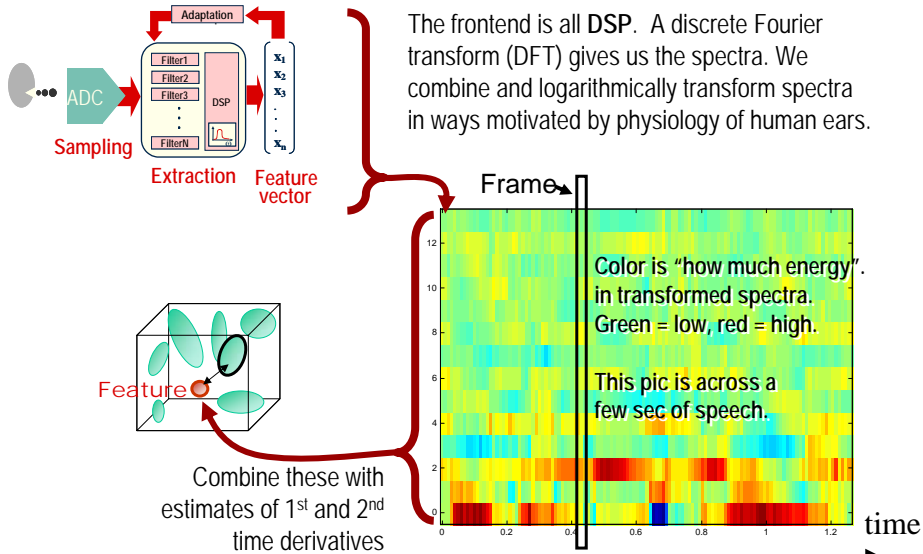
# How Speech Recognition Works



© Rob A. Rutenbar 2006

Slide 9

## (1) Acoustic Frontend

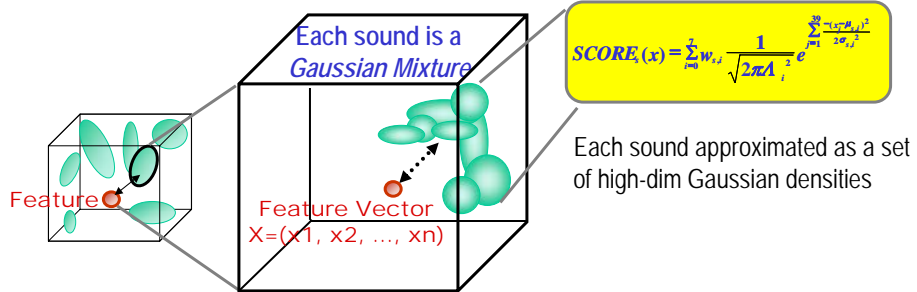


© Rob A. Rutenbar 2006

Slide 10

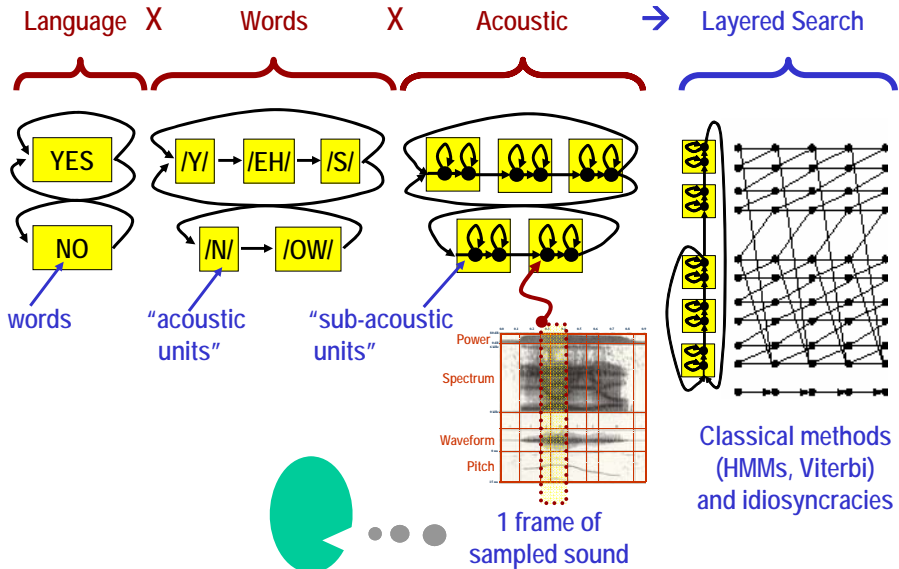
## (2) Scoring Stage

- Each feature is a point in high-dimensional space
  - ▼ Each "atomic sound" is a region of this space
  - ▼ Score each atomic sound with Probability(sound matches feature)



■ Note: (sounds) X (dimensions) X (Gaussians) = BIG

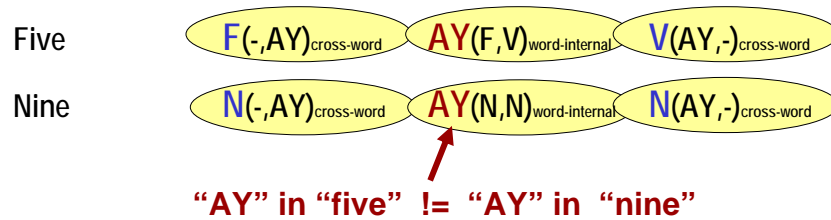
## (3) Search: Speech Models are Layered Models



## Context Matters: At Bottom -- *Triphones*

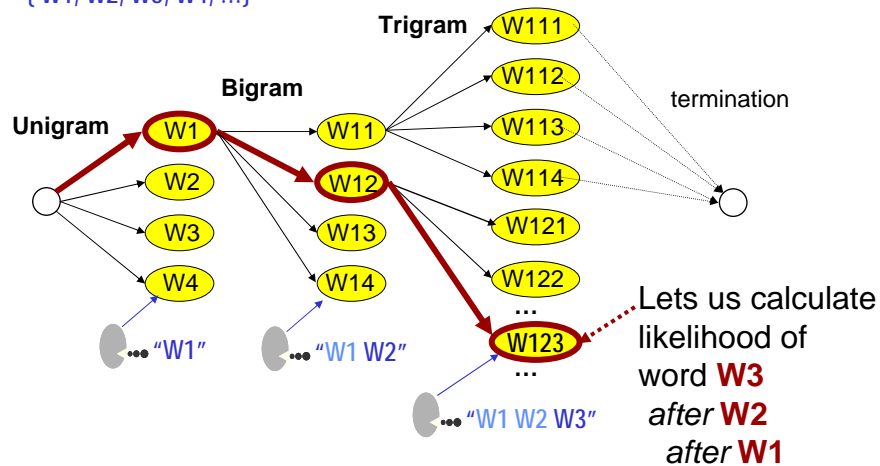
- English has ~50 atomic sounds (**phones**) but we recognize ~50x50x50 context-dependent **triphones**

▼ Because "AY" sound in "five" is different than the "AY" in "nine"

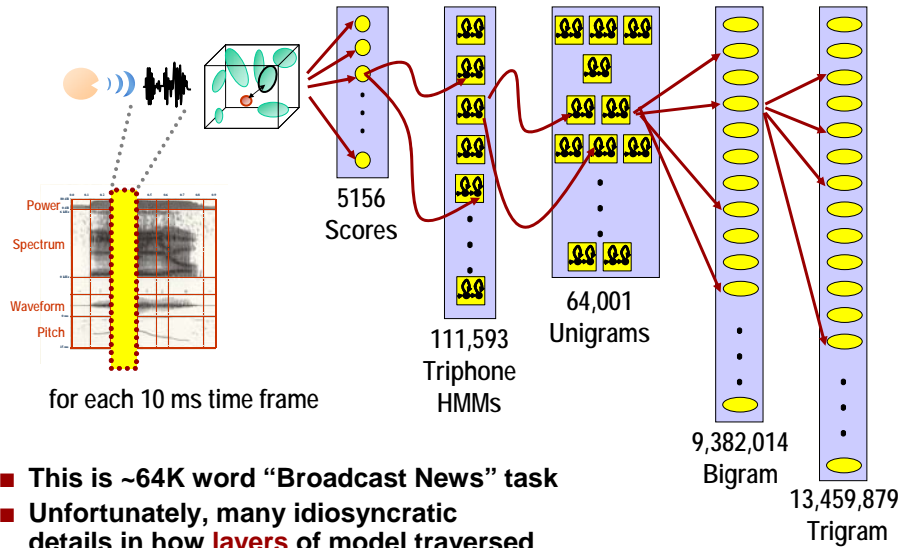


## Similar Idea at Top: *N-gram* Language Model

Suppose we have vocabulary  
 $\{W_1, W_2, W_3, W_4, \dots\}$



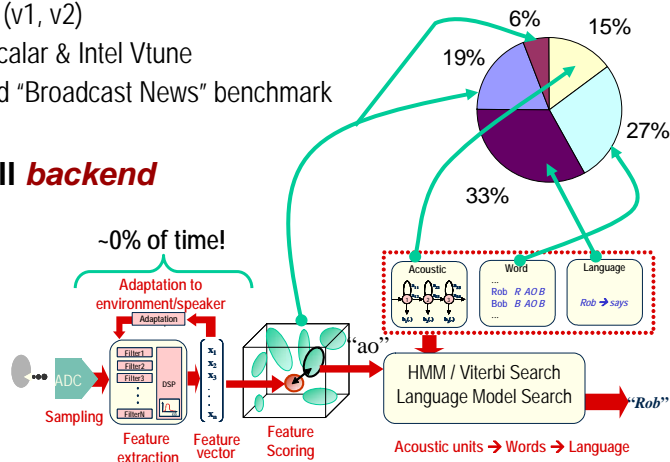
# Good Speech Models are BIG



# Where Does *Software* Spend its Time?

- CPU time for CMU Sphinx 3.0
  - ▼ Prior studies targeted less capable versions (v1, v2)
  - ▼ SimpleScalar & Intel Vtune
  - ▼ 64K-word "Broadcast News" benchmark

■ So: It's all **backend**



## Memory Usage? SPHINX 3.0 vs Spec CPU2000

### Cache sizes

- ▼ L1: 64 KB, direct mapped
- ▼ DL1: 64 KB, direct mapped
- ▼ UL2: 512 KB, 4-way set assoc

### So...

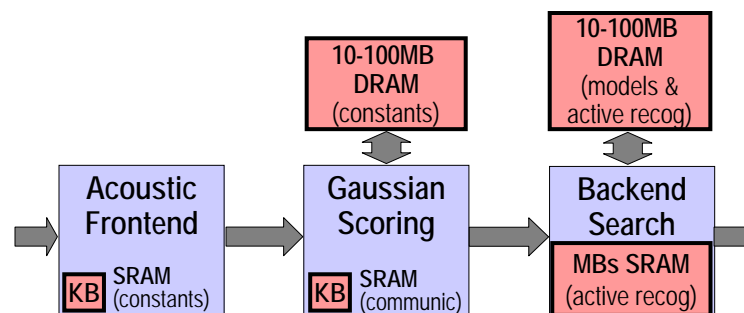
- ▼ **Terrible locality** (no surprise, graph search + huge datasets)
- ▼ **Load dominated** (no surprise, reads a lot, computes a little)
- ▼ Not an insignificant **footprint**

	SPHINX 3.0	Gcc	Gzip	Equake
<b>Cycles</b>	53 B	55 B	15 B	23 B
<b>IPC</b>	0.69	0.29	1.05	0.7
<b>Instruction Mixes</b>				
<b>Loads</b>	0.27	0.25	0.2	0.27
<b>Stores</b>	0.05	0.15	0.09	0.08
<b>Branch's</b>	0.14	0.2	0.17	0.12
<b>Branch Misprediction Rates</b>				
	0.025	0.07	0.08	0.02
<b>Cache Miss Rates</b>				
<b>DL1</b>	0.04	0.02	0.02	0.03
<b>L2</b>	0.48	0.06	0.03	0.30
<b>Memory Footprint</b>				
	64 MB	24 MB	186 MB	42 MB

© Rob A. Rutenbar 2006

Slide 17

## A Silicon Architecture: Big Picture



Computations (Ops)	Low	High	Medium
SRAM (size)	Small	Small	Large
DRAM (size)	--	Medium/Large	Large
DRAM (bandwidth)	--	High	High

© Rob A. Rutenbar 2006

Slide 18

## Essential Implementation Ideas

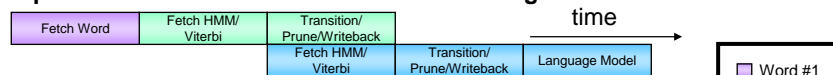
- **Custom precision, everywhere**
  - ▼ Every bit counts, no extras, no floating point – all fixed point
- **(Almost) no caching**
  - ▼ Like graphics chips: fetch from SDRAM, do careful data placement
  - ▼ (Little bit of caching for bandwidth filtering on big language models)
- **Aggressive pipelining**
  - ▼ If we can possibly overlap computations – we try to do so
- **Algorithm transformation**
  - ▼ Some software computations are just bad news for hardware
  - ▼ Substitute some “deep computation” with hardware-friendly versions

© Rob A. Rutenbar 2006

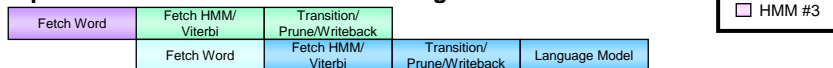
Slide 19

## Example: Aggressive Pipelining

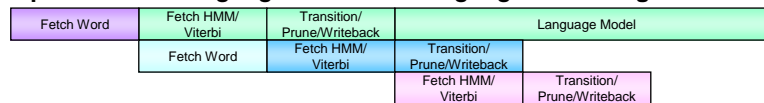
### Pipelined *Get-HMM/Viterbi* and *Transition* stages



### Pipelined *Get-Word* and *Get-HMM* stages



### Pipelined *non-LanguageModel* and *LanguageModel* stages



© Rob A. Rutenbar 2006

Slide 20

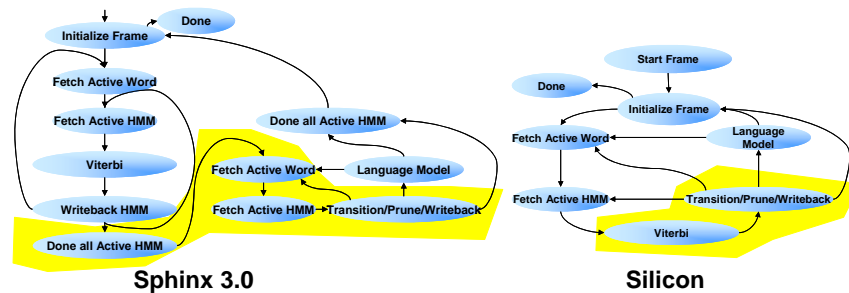
## Example: Algorithmic Changes

### ■ Acoustic-level pruning threshold

- ▼ Software: Use best score of *current* frame (after Viterbi on Active HMMs)
- ▼ Silicon: Use best score of *previous* frame (nixes big temporal bottleneck)

### ■ Tradeoffs

- ▼ Less memory bandwidth, can pipeline, little pessimistic on scores



© Rob A. Rutenbar 2006

Slide 21

## Predicted Performance: C++ Cycle Simulator

### ■ Benchmark: 5K-word "Wall Street Journal" task

### ■ Results:

- ▼ No accuracy loss; not quite 2X @ 125MHz ASIC clock
- ▼ Backend search needs: ~1.5MB SRAM, ~30MB DRAM



Recognizer Engine	Word Error Rate (%)	Clock (GHz)	Speedup Over Real Time (bigger is better)
Software: Sphinx 3.3 (fast decoder)	7.32%	1 GHz	0.74X
Software: Sphinx 4 (single CPU)	6.97%	1 GHz	0.82X
Software: Sphinx 4 (dual CPU)	6.97%	1 GHz	1.05X
Software: Sphinx 3.0 (single CPU)	6.707%	2.8 GHz	0.59X
Hardware: Our Proposed Recognizer	6.725%	0.125 GHz	1.67X

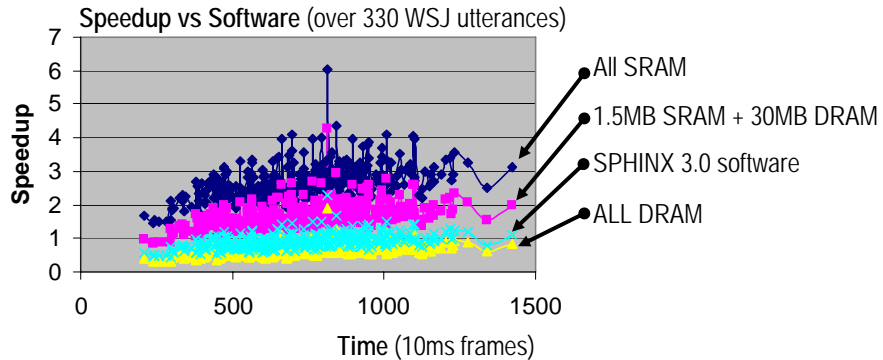
© Rob A. Rutenbar 2006

Slide 22

## Aside: Bit-Level Verification Hurts (A Lot)

- We have newfound sympathy for others doing silicon designs that handle large media streams

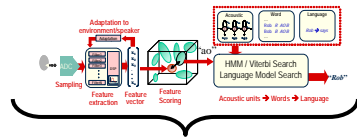
▼ Generating these sort of tradeoff curves: CPU days → weeks



© Rob A. Rutenbar 2006

Slide 23

## A Complete Live Recognizer: FPGA Demo



- In any “system design” research, you reach a point where you just want to see it work – *for real*



- Goal: *Full recognizer 1 FPGA + 1 DRAM*

Xilinx XC2VP30 FPGA  
 [13969 slices / 2448 Kb]  
 Utiliz: 99% of slices  
 45% of Block RAM  
 ~3MB DDR DRAM  
 50MHz clk, ~200Mb/s IO

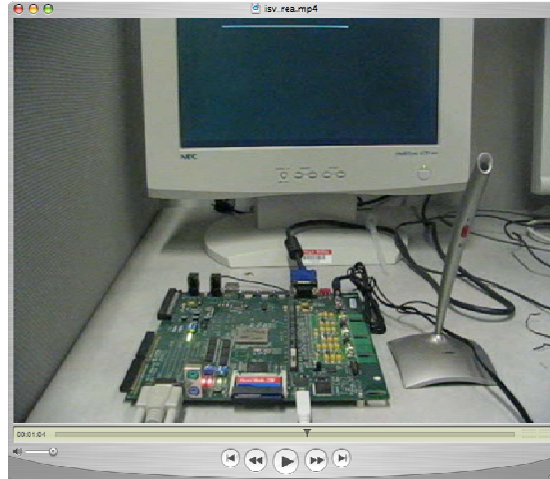
- A benchmark that fits on chip
  - ▼ 1000-word “Resource Mgt” task
  - ▼ Slightly simplified: no tri-grams
  - ▼ Slower: not real time, ~2.3X slower
  - ▼ Resource limited: slices, mem bandwidth

© Rob A. Rutenbar 2006

Slide 24

## FPGA Experimental Results

- **Aside:** as far as we know, this is the *most complex* recognizer architecture ever fully mapped into a hardware-only form



© Rob A. Rutenbar 2006

Slide 25

## Summary

- **Software is too constraining for speech recognition**
  - ▼ Evolution of graphics chips suggests alternative: **Do it in silicon**
  - ▼ Compelling performance and power reasons for silicon speech recog
- **Several “*in silico vox*” architectures in design**
  - ▼ ASIC version: ~1.6X realtime for 5K-word task; 10X version in progress
  - ▼ FPGA version: tiny design successfully running 1000-word benchmark
- **Directions**
  - ▼ Exploit Berkeley BEE2 emulation engine : ~25X more FPGA resources
  - ▼ Detailed architecture/performance/power tradeoffs for mobile apps

WHEN IS WINDOWS AVAILABLE

## Acknowledgements

- **Work supported by**

- ▼ National Science Foundation
- ▼ Semiconductor Research Corporation
- ▼ MARCO/DARPA Center for Circuit & System Solutions (C2S2)

- **We are grateful for the advice and speech recognition expertise shared with us by**

- ▼ Richard M. Stern, CMU
- ▼ Arthur Chan, CMU