



*Sun*<sup>®</sup>  
microsystems



# Managing EDA Complexity

To Deliver Industry Leading Microprocessors

**Dwayne Lee**  
**Engineering Manager**  
**Sun Microsystems, Inc.**

# Goals

- **Functional chips which meet specification with first silicon (timing, power, die-size and functionality)**
- **Meet tight development schedules**
- **Shortest possible design cycle**

# Challenges

- **Larger, more complex designs**
- **Multiple development projects**
  - **Several simultaneous projects**
  - **Different project schedules**
- **Many EDA tools**
- **Managing change**
- **Lack of tool interoperability**
- **Larger compute resources required**

# Increasing Design Complexity

	Year	'91	'95	'99	'03*
Cycle Time		20n	6n	1.6n	.33n
# transistors		3M	5M	29M	200M
Logic transistors		–	2M	4M	50M
Library cells		400	2000	800	–
Megacells		30	50	150	–
Routing Layers		3	4	7	9
Verilog source lines		–	300k	1.8M	–
Diagnostics		–	5M	400M	–

\* Source: EDA Roadmap Taskforce Report <http://www.si2.org/ic>

# Multiple Project Development

- Several projects underway at any one time
- Each project at a different stage
  - demands placed on resources vary tremendously
- Some projects at multiple sites

# Utilization of EDA Tools

- Tools are very expensive and getting more expensive
- Good accessibility is important for high utilization
  - floating licenses
  - unlimited licenses for key tools
  - triple redundant license servers

# Many EDA Tools

- **Multiple projects, long timeline**
  - result, many tools and multiple versions
- **More than 125 commercial tools and increasing, over 250 including internally developed tools**
- **Tools used in all phases of design**
  - architecture, logic design & verification, circuit design & verification, layout design & verification

# Tool Interoperability

- **With complexity, no single EDA vendor can provide the best tools in all areas**
- **Interfacing tools from different vendors is critical**
- **Moving data from one tool to another is non-trivial**

# Tool Non-Interoperability

- Teams of engineers must be dedicated to create flows/convertors to connect tools together
- Difficult to make seamless
- Time consuming
- Better solutions are needed

# An Good Example

## VCS

- Interoperability easy
- VCS used only for functional simulation
- No backannotation
- Static timing analysis using Pearl
- PLIs do cause slowness and non-portability
  - monitors, signal dumps, hooks into architectural simulator, ...

# An More Difficult Example

## Design Compiler

- Interoperability difficult, interface to several tools
- Synthesis, Place&Route, Extraction, Optimization loop
- Issues:
  - Naming conventions - signals
  - Wireload models
  - Timing differences
  - Different tools in flow give different critical path results

# Verification Needs

- Full verification run is >150M simulated cycles
  - Run many times per week
  - Projects simulate ~200B cycles total
  - Current tools simulate ~45 cycles/sec.
  - Full run requires ~35 CPU years
- Represents a major portion of our computing
- Need to shorten turnaround time

# Complexity Requires More Compute Capability

	Year	'91	'95	'99
Tools		40	100	250
Tools Avg. Run Time		1 hr	2 hrs	3 hrs
Tools Peak Run Time		10 hrs	48 hrs	150 hrs
Process Size		800 MB	2 GB	4 GB
File Size		600 MB	1 GB	2 GB
Logic Simulation Cycles		200M total	2B total	2B/week
Batch Jobs		–	1M	1M/month

# Larger Design Size

64-bit is now a Requirement

- **The problem:**
  - Processors are getting larger
  - Hierarchical design/analysis breaks down because of deep sub-micron effects
- **UltraSPARC-III:**
  - Complete release is 23GB + 90,000 files
  - The DFII database consists of 124 libraries and 17,000 cells
  - Design broken into 12 clusters for streaming
  - The chip contains 29 million transistors

# The Solution . . .

## EDA Tool Interoperability

- **Highly desired:**
  - Use one (each) schematic, layout, verilog, waveform viewer
  - Reduce time to dump/load data (we communicate with files, but with 2GB files that is a problem)
- **EDA companies have made progress**
  - SPINE, Tap-in good programs, need more
- **Create tighter links between tools**

# The Solution . . .

## Virtual Computing Engine

- **Transparent network**
  - Shared file name space
  - Uniform tool server setup
  - Uniform machine setup
- **Resource sharing software**
  - **DReAM - Distributed Resource Allocation Manager**
  - Programmable and dynamic allocation

# Virtual Compute Engine

## Typical System

### User system

- Uni-processor  
(U2, U30, U60)
- 256-512MB Ram
- 8 GB of disk space (more in some cases)
- Fast Graphics
- Used for:
  - reviewing results
  - interactive apps.

### Ranch system

- Multi-processor  
(U2, U60, U80, E4500)
- 850MB RAM/CPU
- 9GB - 2TB of disk space
- average over 3 CPUs/engineer
- Used for:
  - heavy computing
  - most work

# Compute Usage

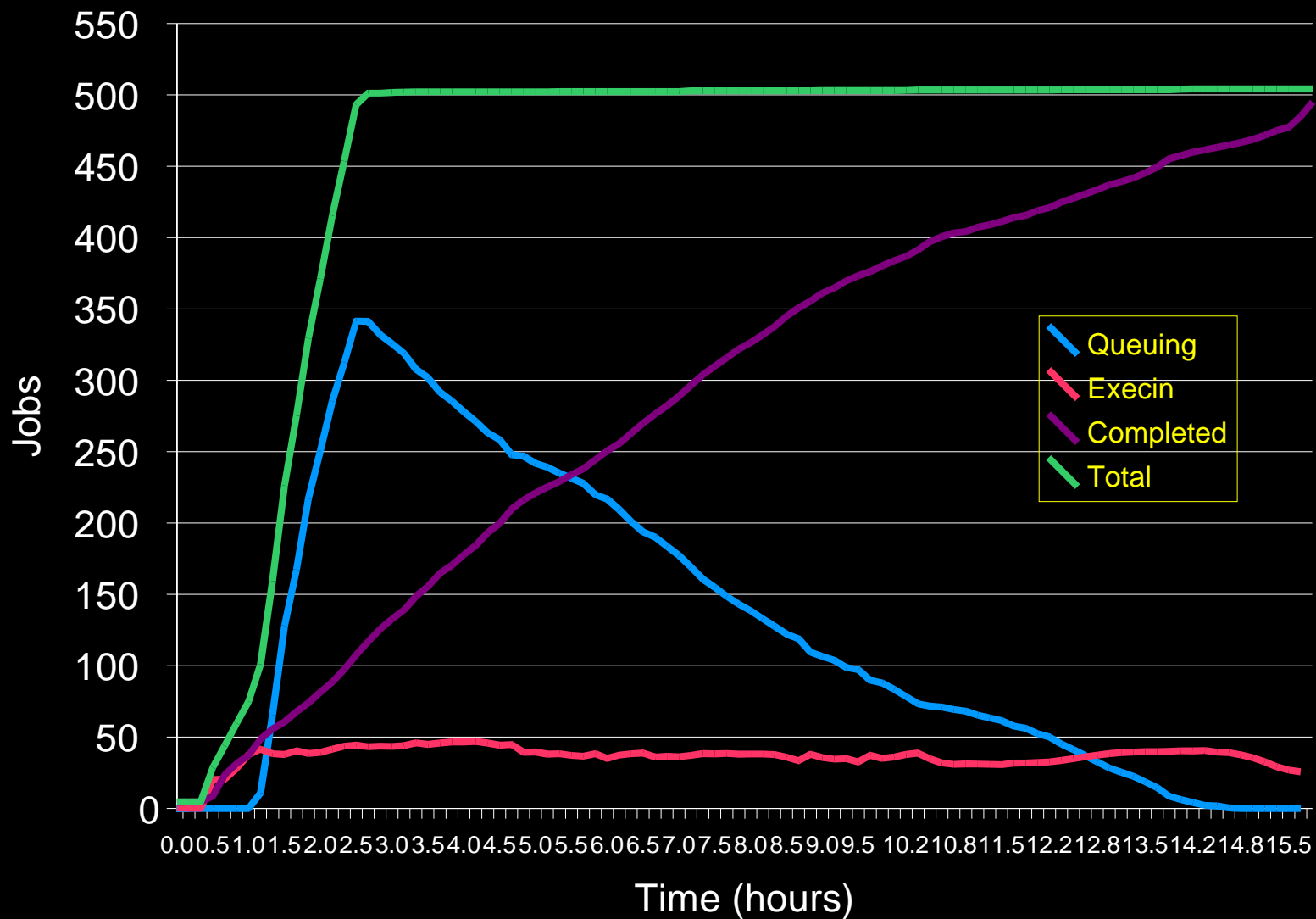
24 by 7, 365 days per year

	Aug '00	Jul' 00	Jun '00	May '00
Total jobs dispatched	1249540	1192322	1245699	1167112
Avg. run time	0.9	0.85	0.92	0.99
Total run time	1113886	1014293	1060272	1063601

\* All times in hours

- Avg. close to 40,000 jobs per day
- Avg. utilization of servers >95%

# Typical Regression Run



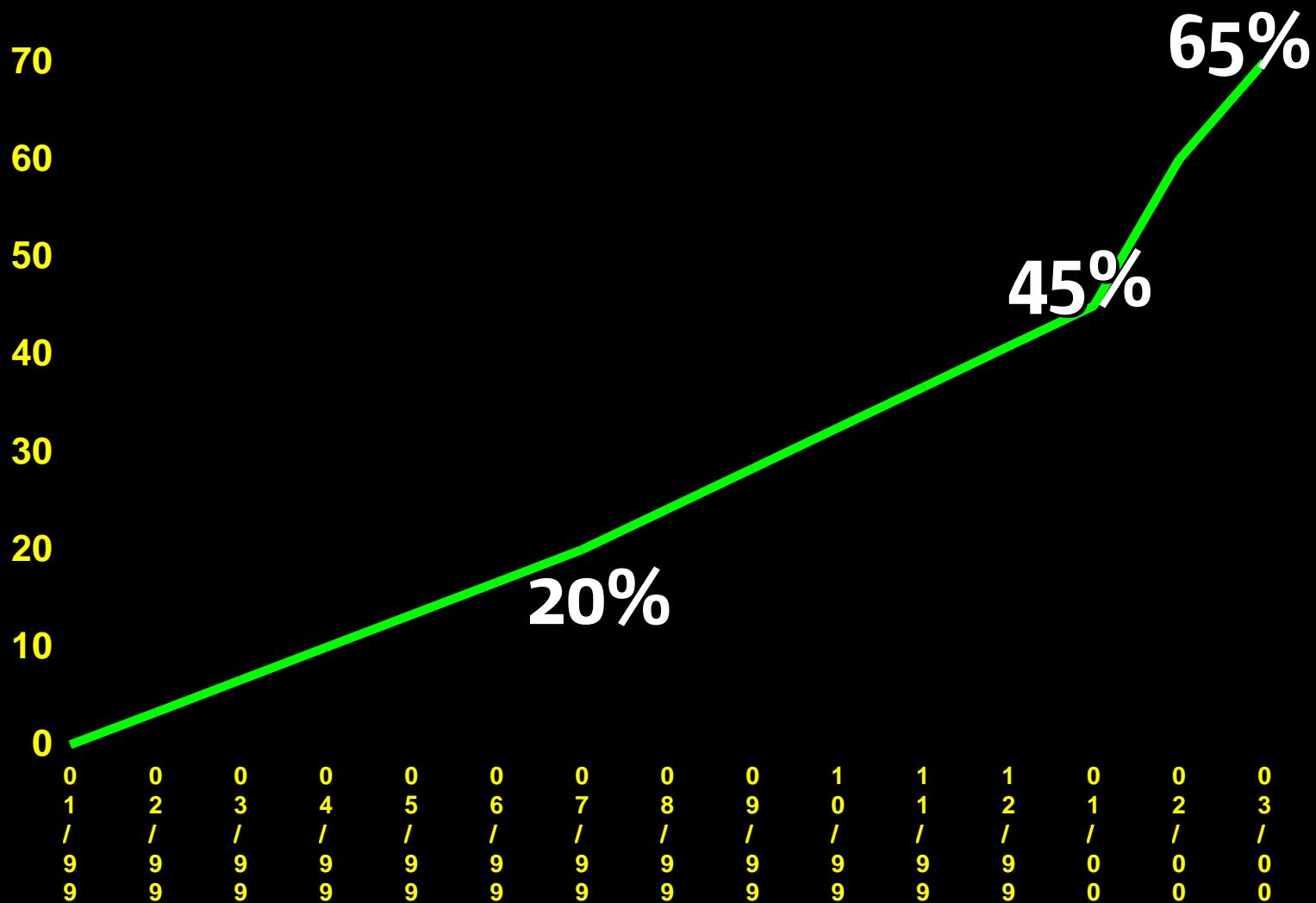
# Managing Change

## Operating System Transitions

- **Aggressively move to the newest operating system, binary compatibility makes this possible**
- **Deploy in ranch first**
- **DReAM allows control of where jobs go**
- **After use in ranch move to desktops**

# Desktops @ Solaris 7

Jan. '99 Thru Feb. '00

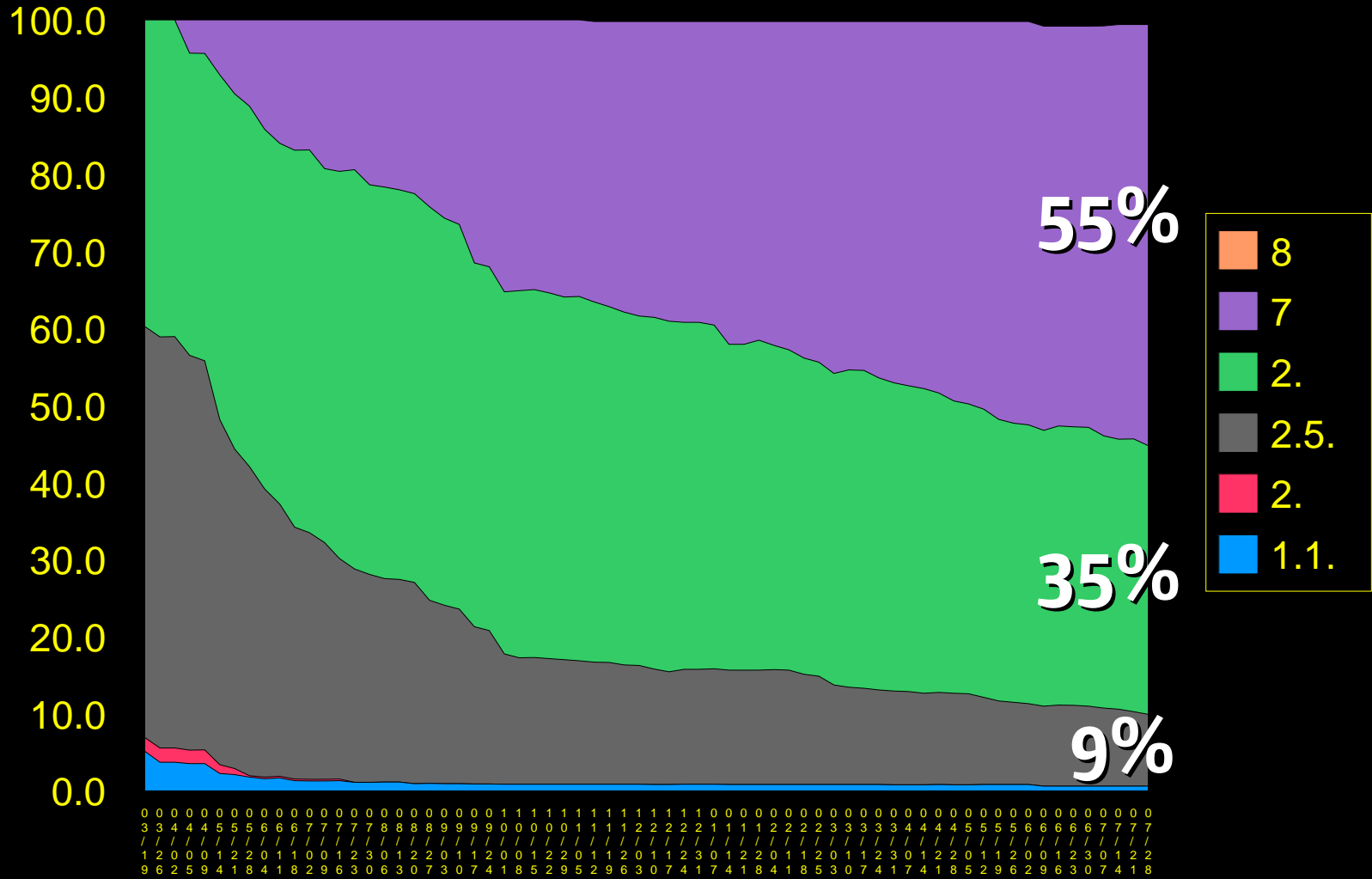


# Automated Patch Distribution

- **Centralized patch distribution**
  - Standard patch configurations centralized
  - Unique patch configurations supported
  - All machines have OS have the same patch set
- **Dispatchable reboot via DReAM**

# Ranch Operating Systems

Mar. '99 Thru Jul. '00



# Managing Change

## Multiple Tool Versions

- Project can never transition to a tool quickly enough, project milestones forces transition to be stagger
- TRE (Tool Run Environment) create to handle tool versioning
- Any project can run a different version of a tool
- Leaves a trail behind on every run

# Architecture for RAS

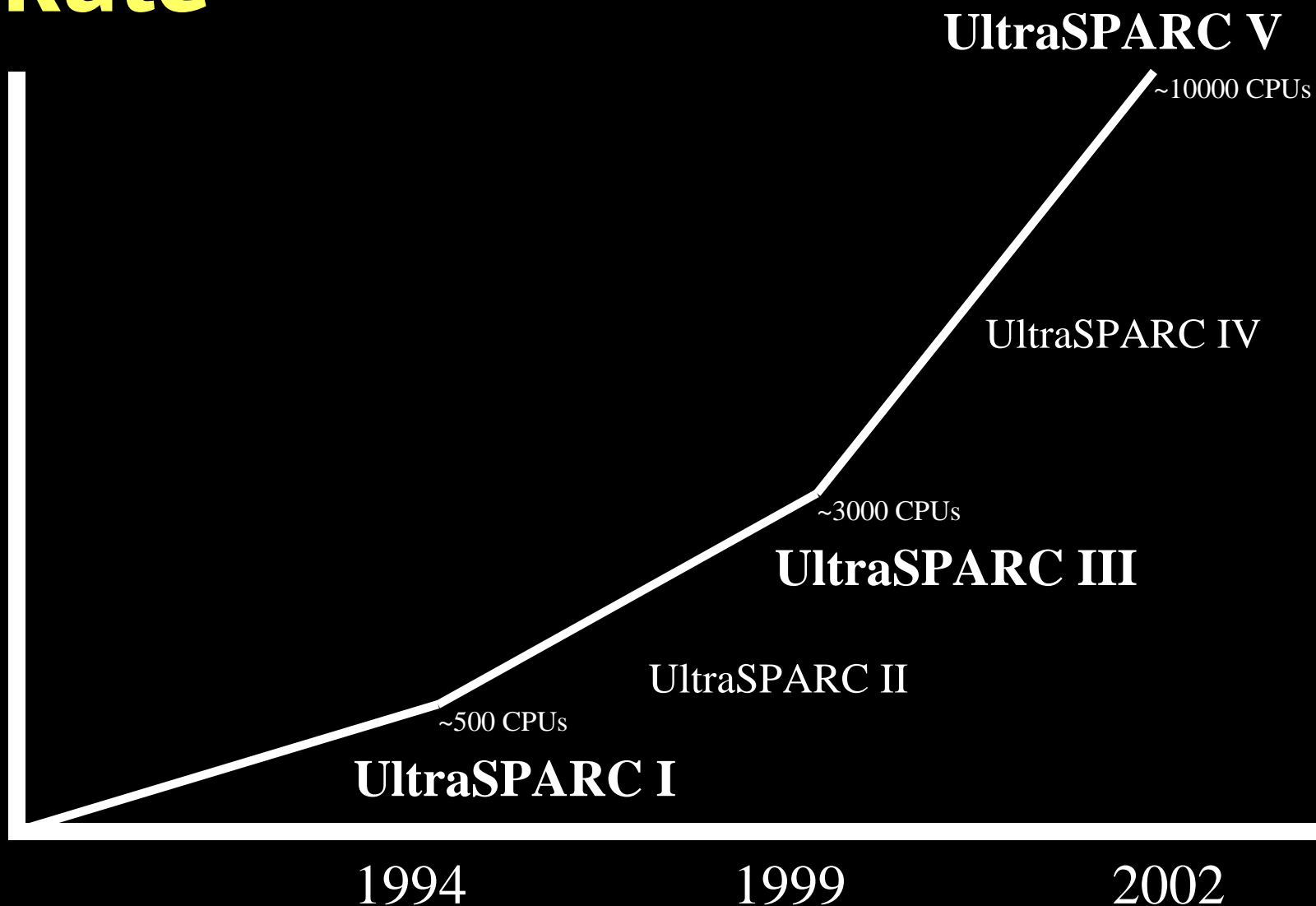
- **Reliability, Availability and Serviceability must be designed in at all levels. Within the Ranch:**
  - **Infrastructure**
  - **Networking**
  - **Systems**
  - **Backup**
  - **Applications**

# Conclusion . . .

Larger, more complex designs require:

- **Cooperation among EDA vendors**
  - Good, tightly knit interfaces between tools
  - Reduce need for customer generated interfaces
- **Larger compute resources**
  - Compute ranch
  - 64 bit compute resources
- **Managing change**
- **Result is more usable cycles or a gain in productivity**

# Demand Accelerates Growth Rate



We're the  
dot in .com™

