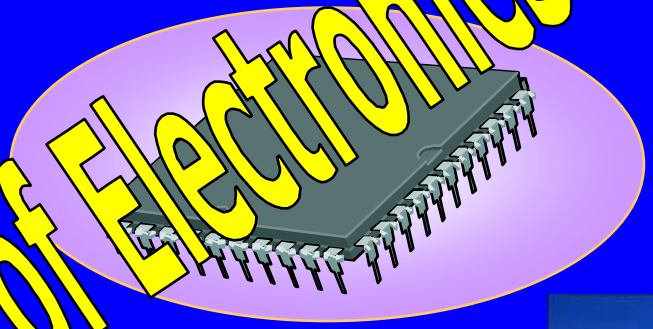
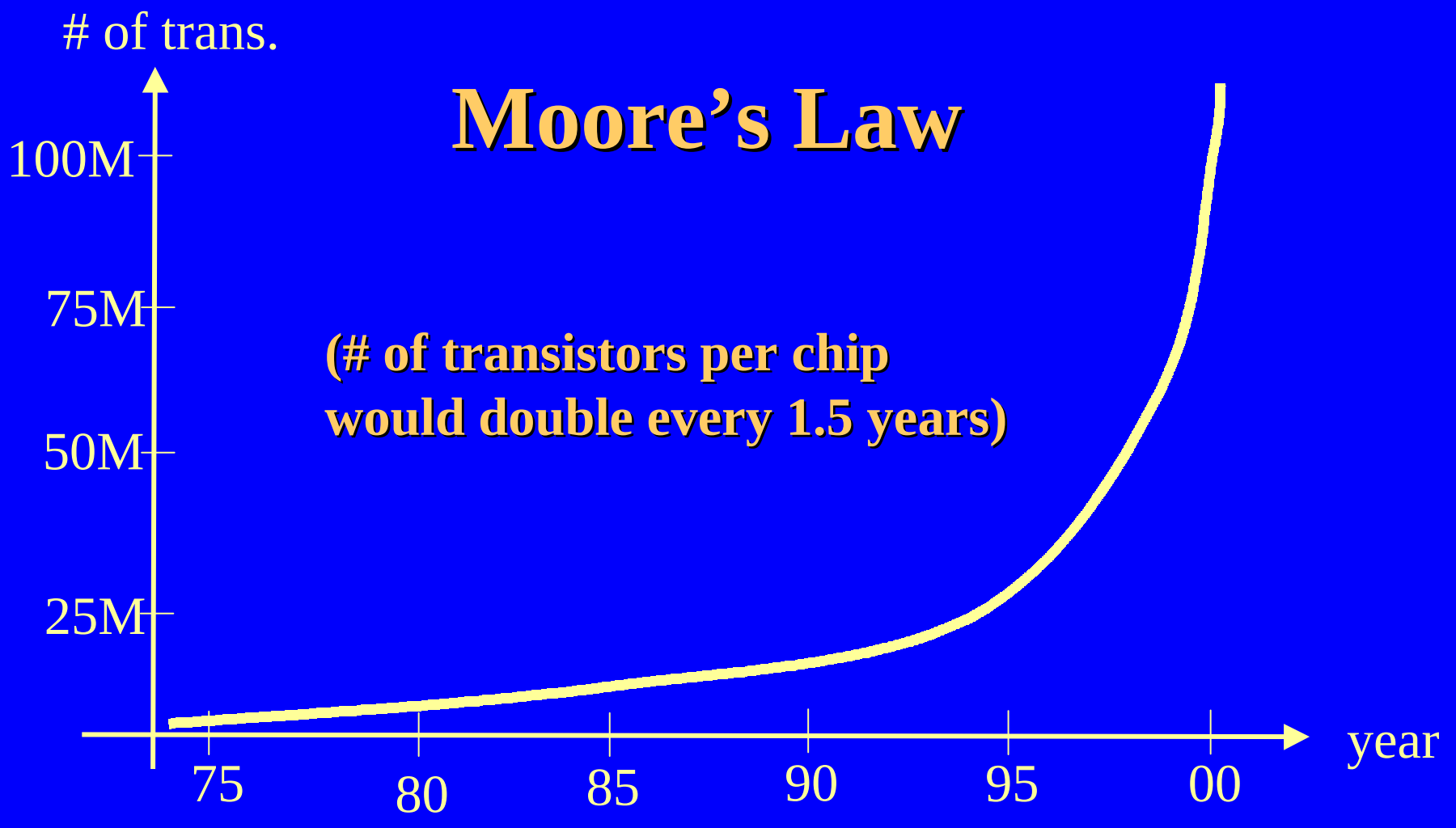
Electronics Systems



Design of Electronics Systems

- Trend in microelectronics
- The design process and tasks
- Different design paradigms
- Basic terminology
- The test problems

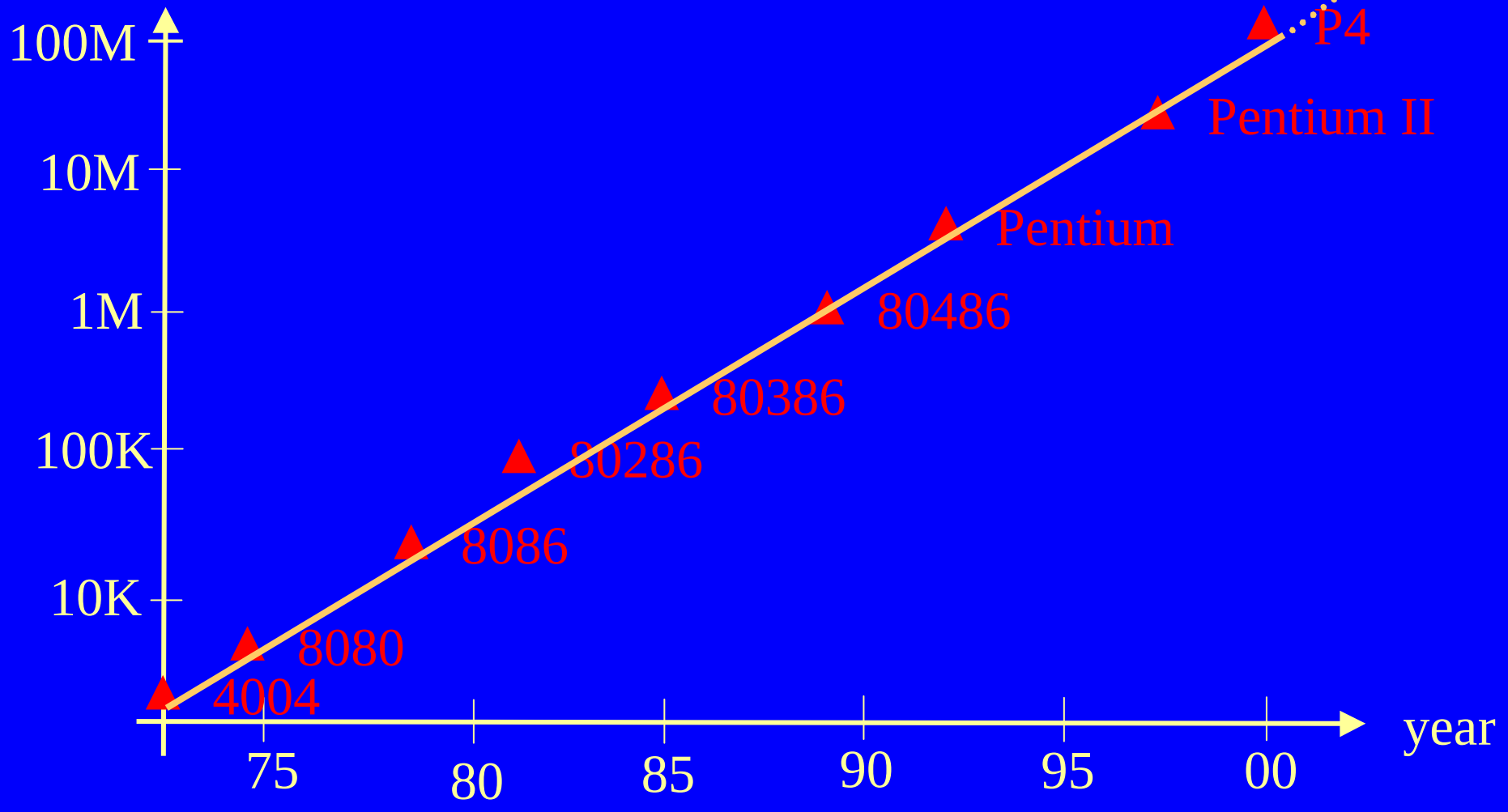
The Technological Trend



Intel Microprocessor Evolution

	Year/Month	Clock =1/tc.	Transistors.	Micras
I4004	1971/11	108 KHz.	2300	10
I8080	1974/04	2 MHz.	6000	6
I8086	1978/06	10 MHz.	29000	3
I80286	1982/02	12 MHz.	0.13 m.	1.50
I486DX	1989/04	25 MHz.	1.2 m.	1
Intel DX2	1992/03	100 MHz.	1.6 m	0.8
Pentium	1993/03	60 MHz.	3.1 m	0.8
Pentium Pro	1995/11	200 MHz.	5.5 m	0.35
Pentium II	1998/	450 MHz	7.5 m.	0.25
Pentium III	2000/01	1000 MHz.	28 m.	0.18
P4	2000/09	1400 MHz.	42 m.	0.18

Intel Microprocessor Evolution



Technology Directions: SIA Roadmap

Year	2002	2005	2008	2011	2014
Feature size (nm)	130	100	70	50	35
Logic: trans/cm^2	18M	44M	109M	269M	664M
Trans/chip	67.6M	190M	539M	1523M	4308M
#pads/chip	2553	3492	4776	6532	8935
Clock (MHz)	2100	3500	6000	10000	16900
Chip size (mm^2)	430	520	620	750	900
Wiring levels	7	7-8	8-9	9	10
Power supply (V)	1.5	1.2	0.9	0.6	0.5
High-perf pow (W)	130	160	170	175	183
Battery pow (W)	2	2.4	2.8	3.2	3.7

System on Chip (SoC)

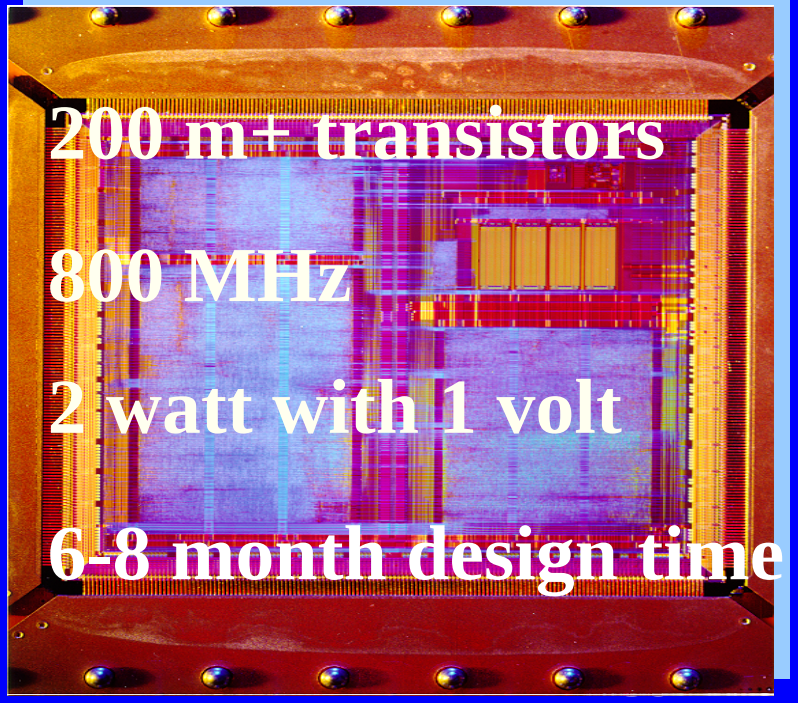
Hardware

Software

microprocessor

ASIC

Analog circuit



Source: Stratus Computers

Embedded memory

DSP

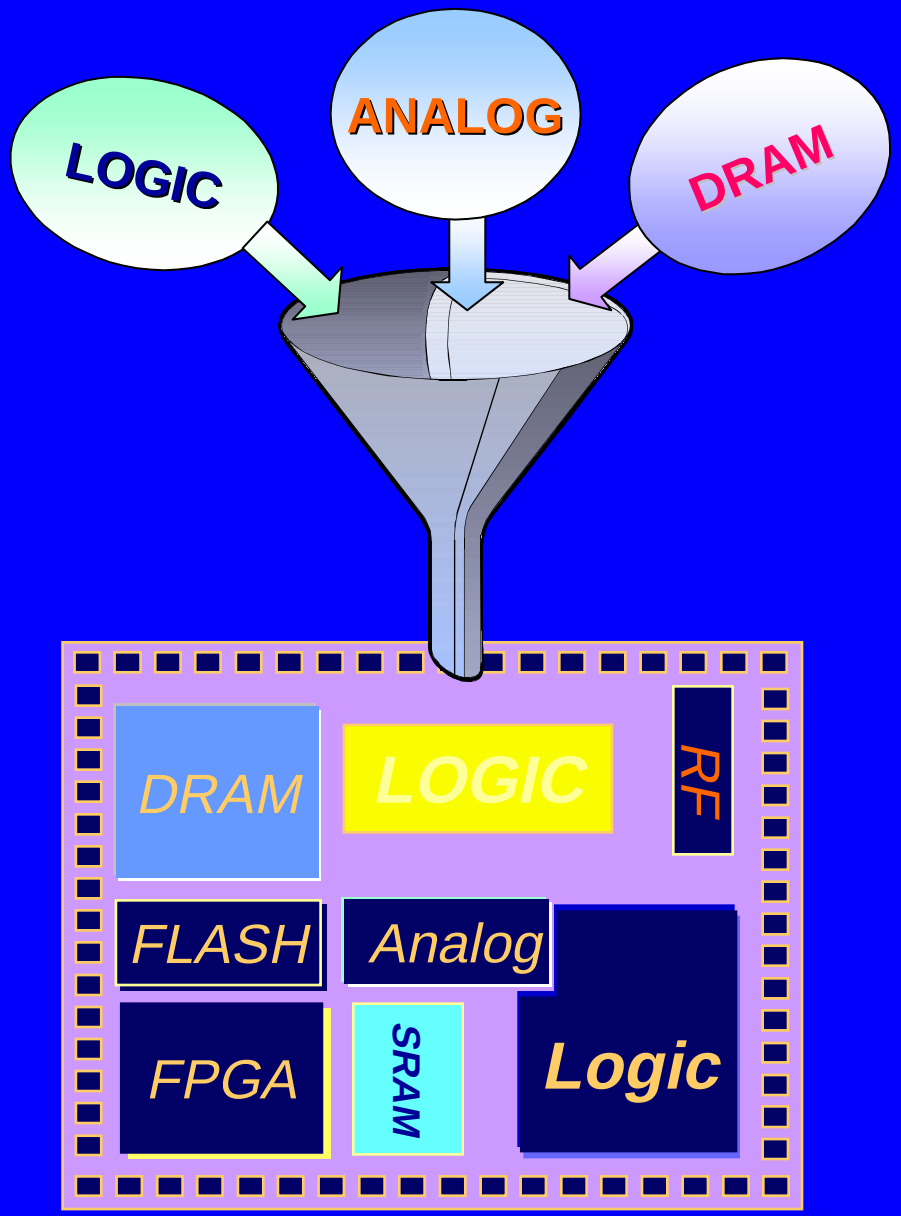
Network

Sensor

High-speed electronics

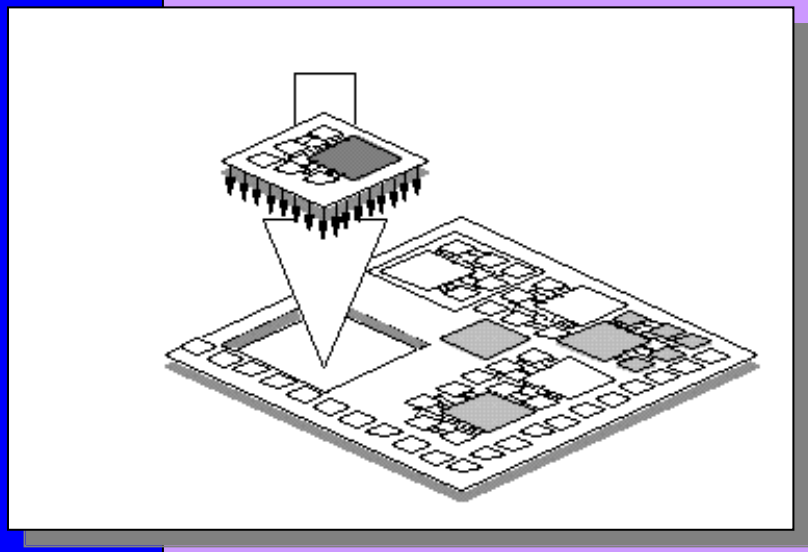
Mixed Technologies

- Embed in a single chip: Logic, Analog, DRAM blocks
- Embed advanced technology blocks:
  - FPGA, Flash, RF/Microwave
- Beyond Electronic
  - MEMS
  - Optical elements



IP-Based Design

- Intellectual Property: pre-designed and pre-verified building blocks.
- Design reuse
- Hard v. soft IPs
- Interface synthesis
- Verification
- Testing

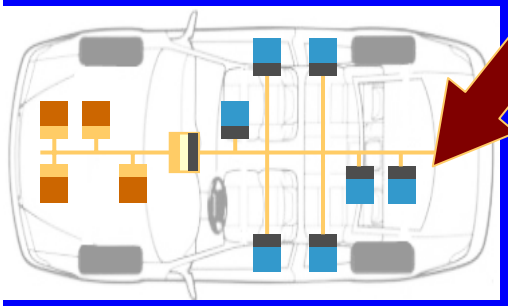


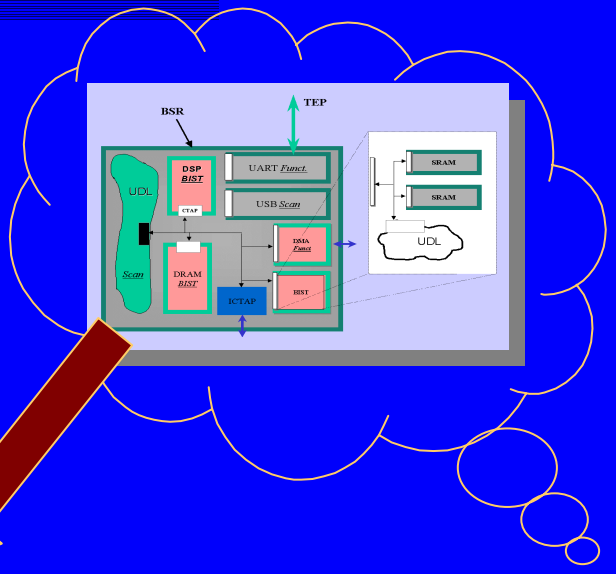
Source: VSI Alliance

Design Requirements

- Technology-driven:
  - Greater Complexity
  - Higher Density
  - Increased Performance
  - Lower Power Dissipation
- Market-driven:
  - Shorter Time-to-Market (TTM)

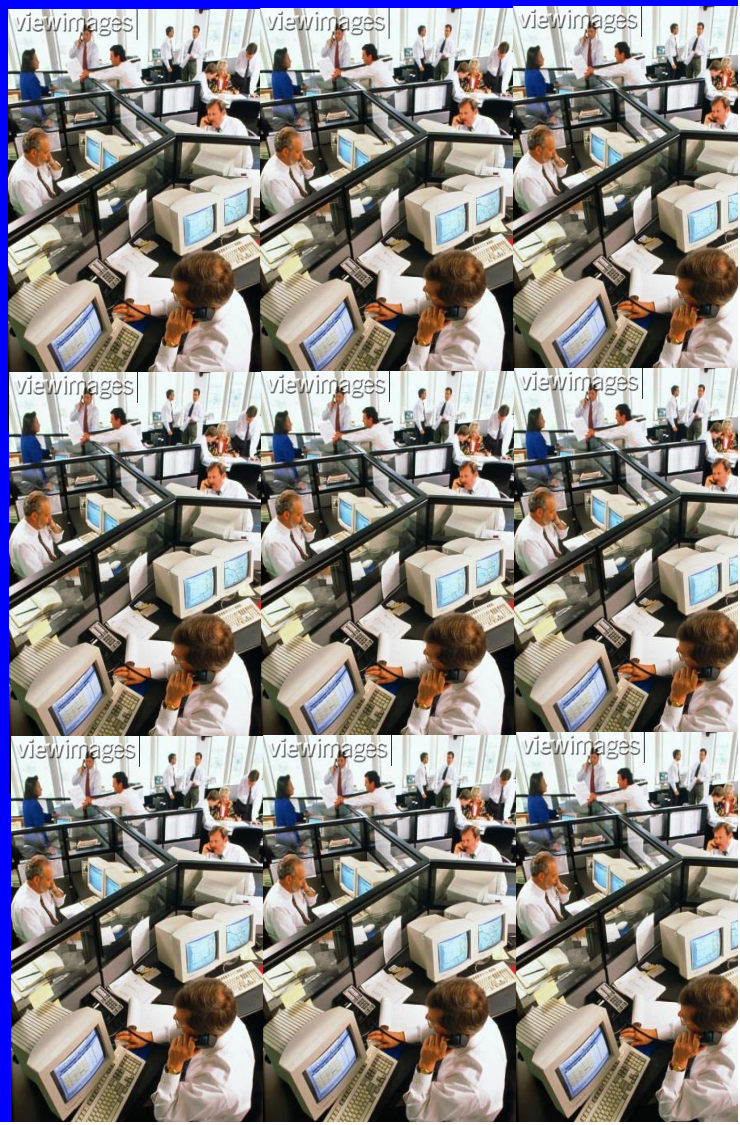
The Electronics System Designer





The Design Challenges

- Complexity implication:
  - 300 gates/person-week
  - 15 000 gates/person-year
  - For a 12-million gate system:
    - 800 designers for one year
    - $120 million design cost ($150K salary)



What are the Solutions?

- Powerful design methodology and tools.
- Advanced architecture (modularity).
- Extensive design reuse.

Design Paradigm Shift

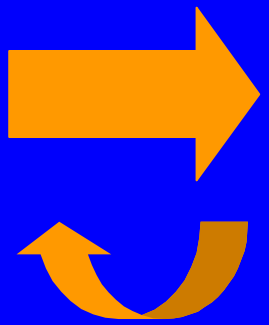
Basic Terminology

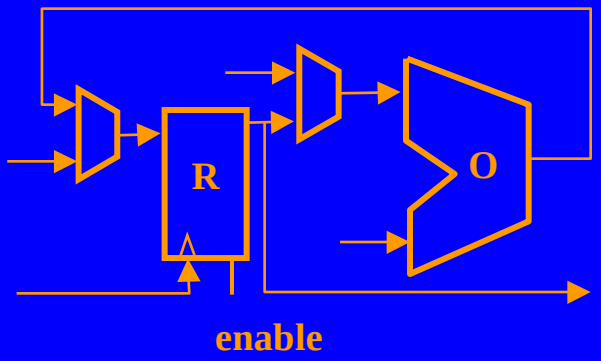
- Design — A series of transformations from one representation to another until one exists that can be fabricated.
- Synthesis — Transforming one representation to another at a lower abstraction level or a behavioral representation into a structural representation at the same level.
- Analysis — Studying a representation to find out its behavior or checking for certain property of a given representation.
- Simulation — Use of a software model to study the response of a system to input stimuli.
- Verification — The process of determining that a system functions correctly.
- Optimization — The change of a design representation to a new form with improved features.

High-Level Describe and Synthesize

- Description of a design in terms of behavioral specification.
- Refinement of the design towards an implementation by adding structural details.
- Evaluation of the design in terms of a cost function and the design is optimized for the cost function.

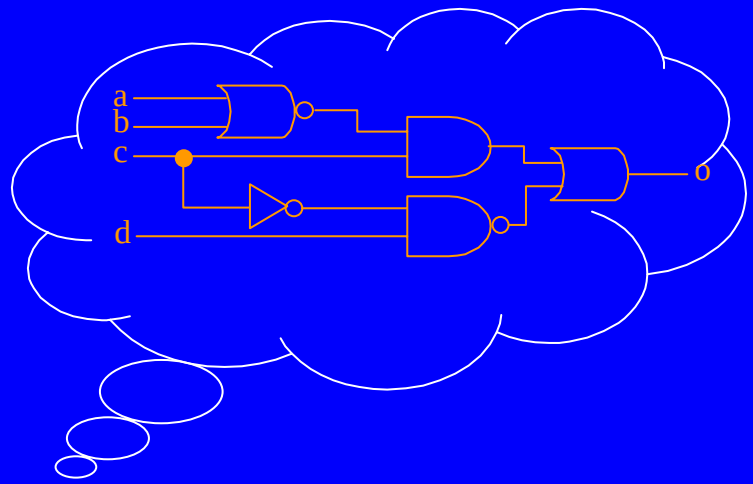
```
For I=0 To 2 Loop
Wait until clk'event
  and clk='1';
  If (rgb[I] < 248) Then
    P=rgb[I] mod 8;
  ...
```

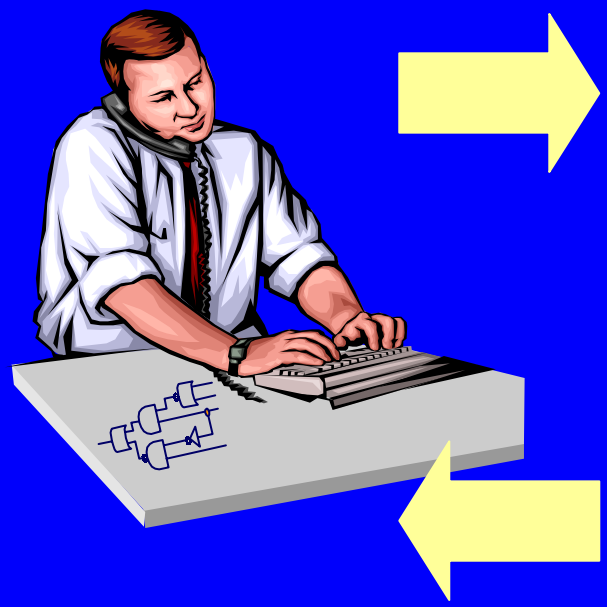


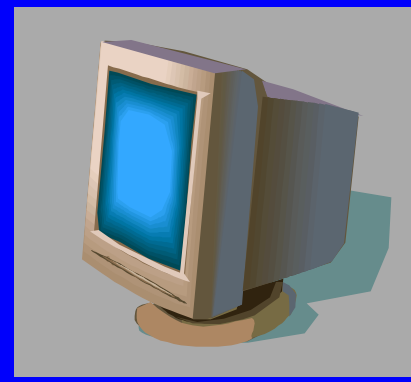


Capture and Simulate

- The detailed design is captured in a model.
- The model is simulated.
- The results are used to guide the improvement of the design.
- All design decisions are made by the designers.

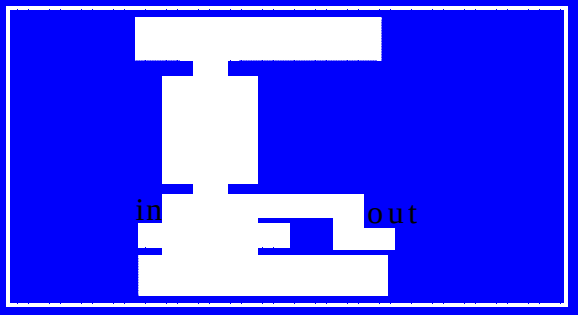


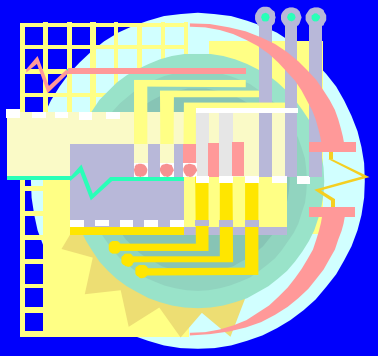


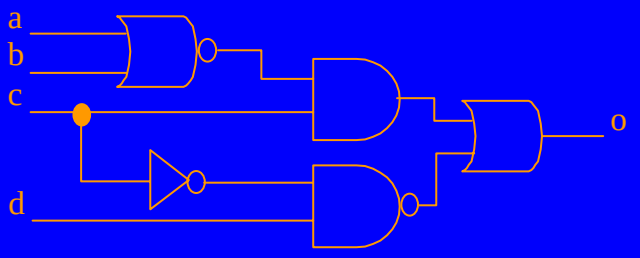


Abstraction Hierarchy

- Layout/silicon level — The physical layout of the integrated circuits is described.
- Circuit level — The detailed circuits of transistors, resistors, and capacitors are described.
- Logic (gate) level — The design is given as gates and their interconnections.

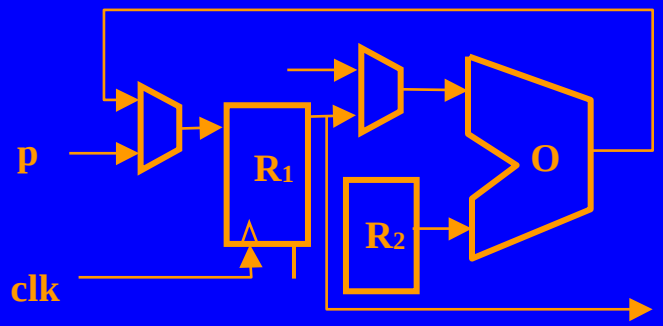




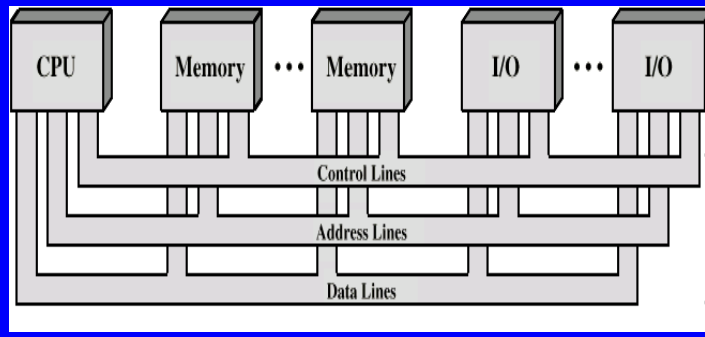


Abstraction Hierarchy (Cont'd)

- Register-transfer level (RTL) — Operations are described as transfers of values between registers.
- Algorithmic level — A system is described as a set of usually concurrent algorithms.
- System level — A system is described as a set of processors and communication channels.

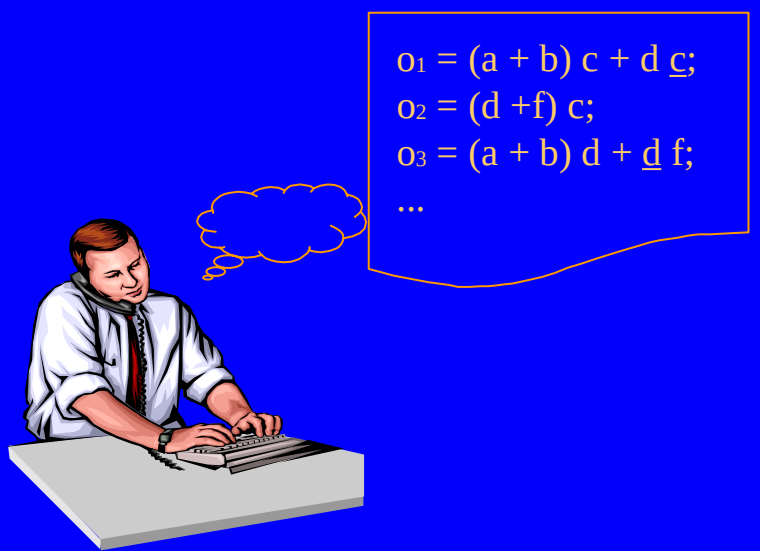


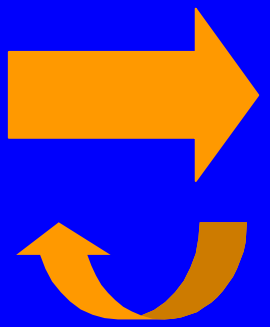

```
For I=0 To 2 Loop
Wait until clk'event and clk = ´1´;
 If (rgb[I] < 248) Then
    P = rgb[I] mod 8;
    Q = filter(x, y) * 8;
 End If;
```

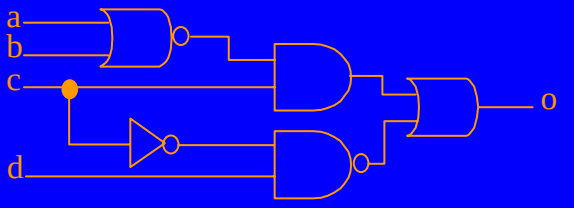


Describe and Synthesize

- Description of a design in terms of behavioral specification.
- Refinement of the design towards an implementation by adding structural details.
- Evaluation of the design in terms of a cost function and the design is optimized w.r.t. the cost function.







Conclusion Remarks

- Much of design of digital systems is managing complexity.
- What is needed: new techniques and tools to help the designers in the design process, taking into account different aspects.
- We need especially design tools working at the higher levels of abstraction.
- If the complexity of the microelectronics technology will continue to grow, the migration towards higher abstraction level will continue.

Challenges to the CAD Communities

- System specification with very high-level languages.
- Modeling techniques for heterogeneous system.
- Testing must be considered during the design process.
- Design verifications -> get the whole system right the first time!
- Very efficient power saving techniques.
- Global optimization.

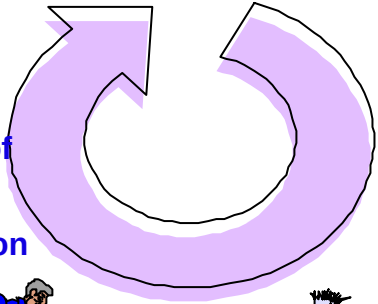
Design Verification

Design Process

Design : specify and enter the design intent

Verify:

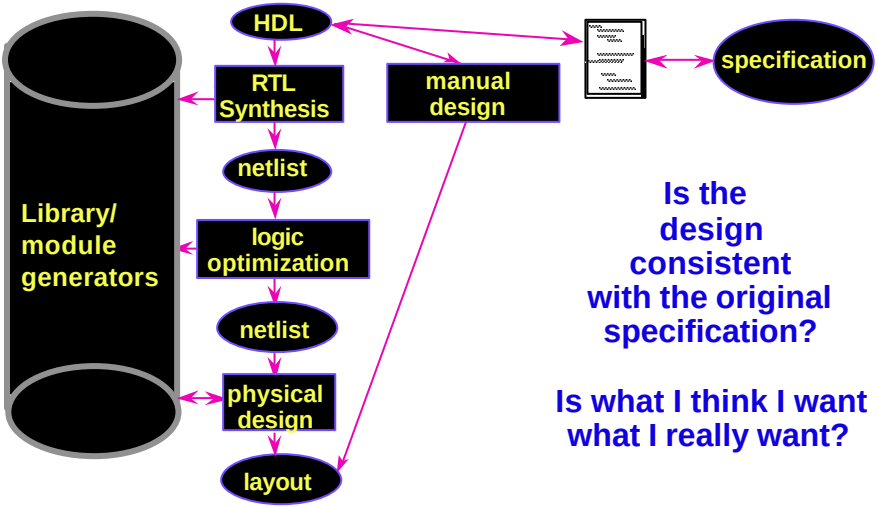
verify the correctness of design and implementation



Implement:

refine the design through all phases

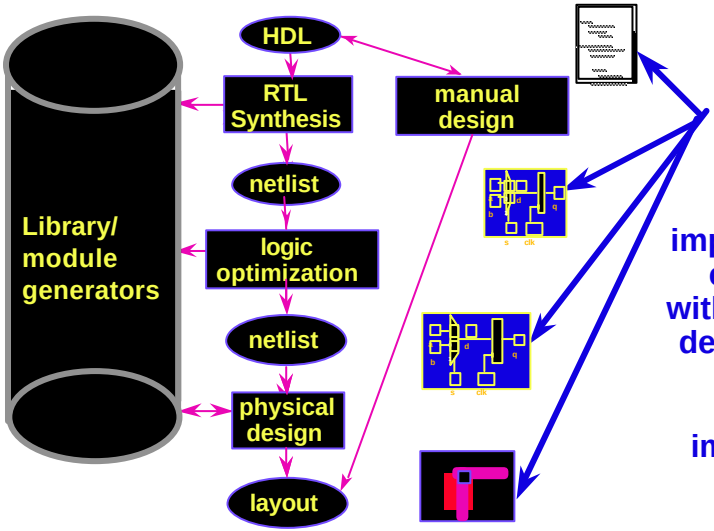
Design Verification



Is the design consistent with the original specification?

Is what I think I want what I really want?

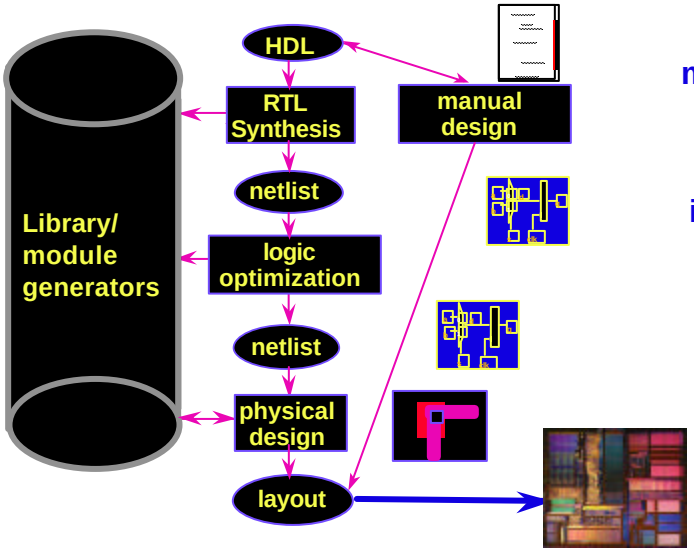
Implementation Verification



Is the implementation consistent with the original design intent?

Is what I implemented what I wanted?

Manufacture Verification (Test)



Is the manufactured circuit consistent with the implemented design?

Did they build what I wanted?

Verification is an Industry-Wide Issue

Intel: **Processor project verification:**
- "Billions of generated vectors"
- "Our VHDL regression tests take 27 days to run. "

Sun: **Sparc project verification:**
- Test suite ~1500 tests > 1 billion random simulation cycles
- "A server ranch ~1200 SPARC CPUs"

Bull: **Simulation including PwrPC 604**
- "Our simulations run at between 1-20 CPS."
- "We need 100-1000 cps."

Cyrix : **An x86 related project**
- "We need 50x Chronologic performance today."
- "170 CPUs running simulations continuously"

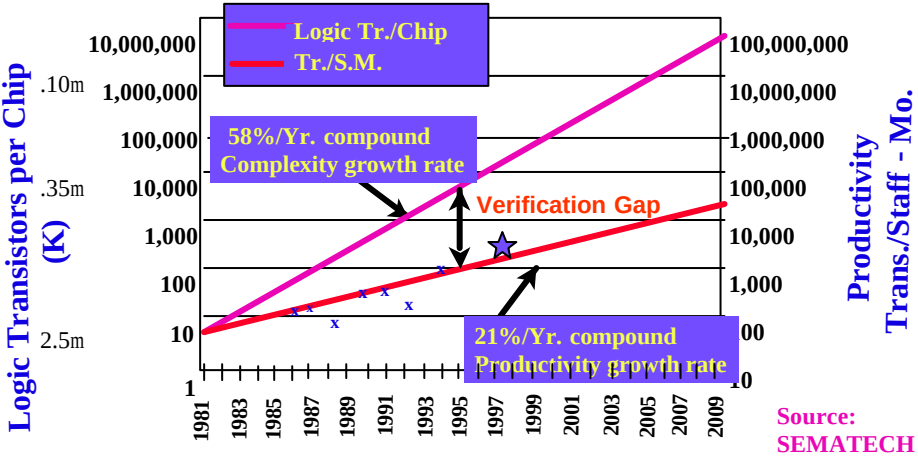
Kodak: "hundreds of 3-4 hour RTL functional simulations"

Xerox: "Simulation runtime occupies ~3 weeks of a design cycle"

Ross: 125 Million Vector Regression tests

Design Teams are *Desperate* for Faster Simulation

Verification Gap



Why the Gap?

$$\frac{\text{logic\_transistors}}{\text{chip}} \times \frac{\text{lines\_in\_design}}{\text{logic\_transistors}} \times \frac{\text{bugs}}{\text{line\_of\_design}}$$

$$= \frac{\text{bugs}}{\text{chip}}$$

Filling in Reasonable Numbers

$$\frac{\text{logic\_transistors}}{\text{chip}} \times \frac{\text{lines\_of\_design}}{\text{logic\_transistors}} \times \frac{\text{bugs}}{\text{lines\_of\_design}}$$

$$\frac{10{,}000{,}000 \text{ trs}}{\text{chip}} \times \frac{1}{10} \times \frac{1}{10{,}000}$$

$$= \frac{100 \text{ bugs}}{\text{chip}}$$

Raising the Level of Abstraction

$$\frac{\text{logic\_transistors}}{\text{chip}} \times \frac{\text{lines\_of\_design}}{\text{logic\_transistors}} \times \frac{\text{bugs}}{\text{lines\_of\_design}}$$

$$\frac{10{,}000{,}000 \text{ trs}}{\text{chip}} \times \frac{1}{100} \times \frac{1}{10{,}000}$$

$$= \frac{10 \text{ bugs}}{\text{chip}}$$

this year!!

Moore's Law Implies More Bugs

$$\frac{\text{logic\_transistors}}{\text{chip}} \times \frac{\text{lines\_of\_design}}{\text{logic\_transistors}} \times \frac{\text{bugs}}{\text{lines\_of\_design}}$$

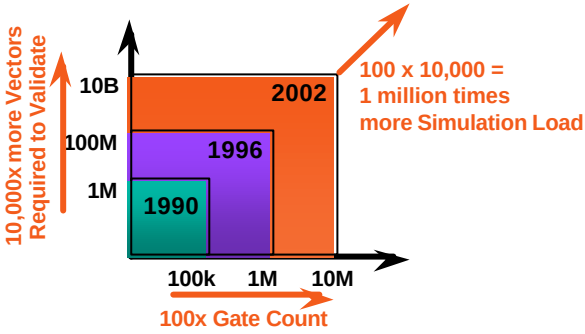
$$\frac{100{,}000{,}000 \text{ trs}}{\text{chip}} \times \frac{1}{100} \times \frac{1}{10{,}000}$$

$$= \frac{100 \text{ bugs}}{\text{chip}}$$

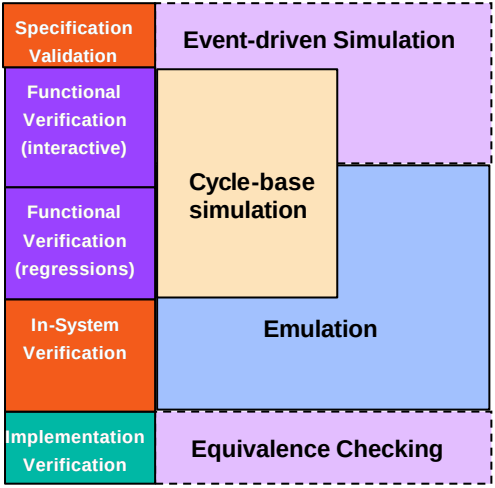
within 5 years!!

The Verification Bottleneck

Verification problem grows even faster due to the combination of increased gate count and increased vector count



Aspects of Design Verification



Event Driven

- Interactive Phase
- High flexibility
- Quick turnaround time
- Good debug capabilities

Cycle-based simulation

- Regression Phase
- Highest performance
- Highest capacity

Emulation and Acceleration

- In-System Verification
- Highest performance
- Highest Capacity
- Real system environment

Approaches to Design Verification

Software Simulation

- **Application of simulation stimulus to model of circuit**

Hardware Accelerated Simulation

- **Use of special purpose hardware to accelerate simulation of circuit**

Emulation

- **Emulate actual circuit behavior - e.g. using FPGA's**

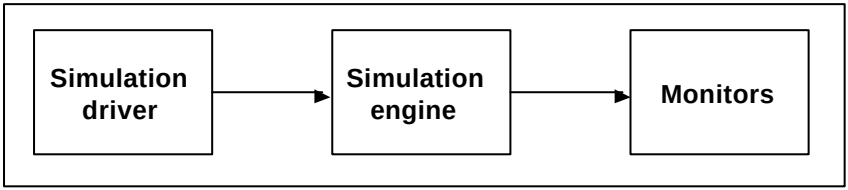
Rapid prototyping

- **Create a prototype of actual hardware**

Formal verification

- **Model checking - verify properties relative to model**
- **Theorem proving - prove theorems regarding properties of a model**

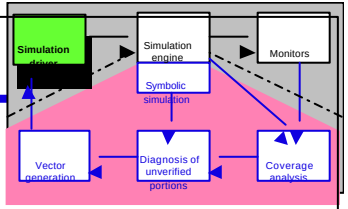
Simulation: The Current Picture



SHORTCOMINGS:

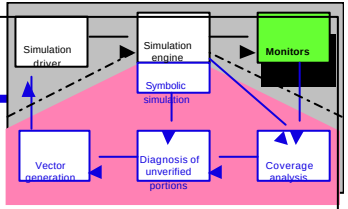
- Hard to generate high quality input stimuli
  - A lot of user effort
  - No formal way to identify unexercised aspects
- No good measure of comprehensiveness of validation
  - Low bug detection rate is the main criterion
    - Only means that current method of stimulus generation is not achieving more.

Simulation Drivers



- Input stimuli consistent with circuit interface must be generated
- Environment of circuit must be represented faithfully
- Tests can be generated
  - pre-run (faster, hard to use/maintain)
  - on-the-fly (better quality: can react to circuit state)
- Environment and input generation programs written in
  - HDL or C, C++, or
  - Object-oriented simulation environment
    - VERA, Verisity
- Sometimes verification environment and test suite come with product, e.g. PCI implementations, bridges, etc.

Monitors



- Reference models (e.g. ISA model)
- Temporal and snapshot “checkers”
  - Can be written in C, C++, HDLs, and VERA and Verisity: A lot of flexibility
  - Assertions and monitors can be automatically generated: 0-in’s checkers
- Protocol specification can be given as
  - a set of monitors
  - a set of temporal logic formulas
- (recent GSRC work)

Types of software simulators

- Circuit simulation
  - Spice, Advice, Hspice
  - Timemill + Ace, ADM
- Event-driven gate/RTL/Behavioral simulation
  - Verilog - VCS, NC-Verilog, Turbo-Verilog, Verilog-XL
  - VHDL - VSS, MTI, Leapfrog
- Cycle-based gate/RTL/Behavioral simulation
  - Verilog - Frontline, Speedsim
  - VHDL - Cyclone
- Domain-specific simulation
  - SPW, COSSAP
- Architecture-specific simulation

Approaches to Design Verification

- **Software Simulation**
  - **Application of simulation stimulus to model of circuit**
- **Hardware Accelerated Simulation**
  - **Use of special purpose hardware to accelerate simulation of circuit**
- **Emulation**
  - **Emulate actual circuit behavior - e.g. using FPGA's**
- **Rapid prototyping**
  - **Create a prototype of actual hardware**
- **Formal verification**
  - **Model checking - verify properties relative to model**
  - **Theorem proving - prove theorems regarding properties of a model**

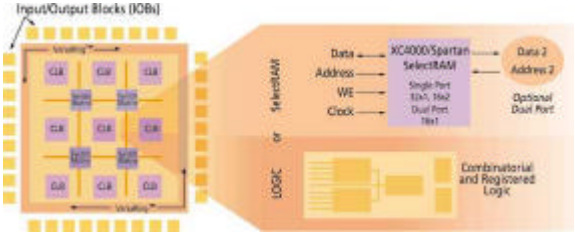
FPGAs as logic evaluators

Today: 2 trillion gate evaluations per second per FPGA (200K gates, 10M cps)

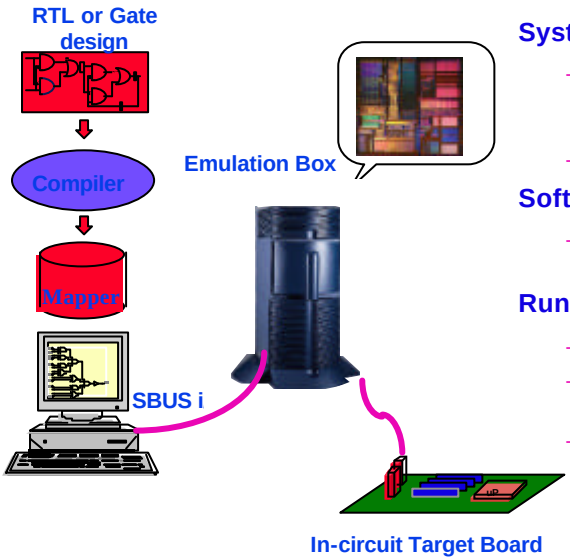
- **Growing with Moore's Law as designs do**
- **$1.5B industry behind it (XLNX+ALTR+ACTL)**

Potent tool for logic verification and validation

How best to put the FPGA to use?



Verification using Emulation



System Hardware

- Customized parallel processor system for emulating logic
- In-circuit target interface

Software Compiler

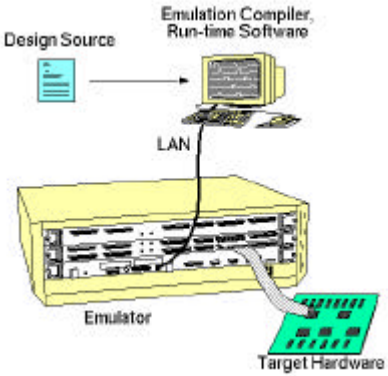
- Mapping RTL & Gate designs to emulator

Runtime Software

- C-API
- Open SW architecture for tight integration
- Flexible modes of stimulus

Logic Emulation

M. Butts - Synopsys



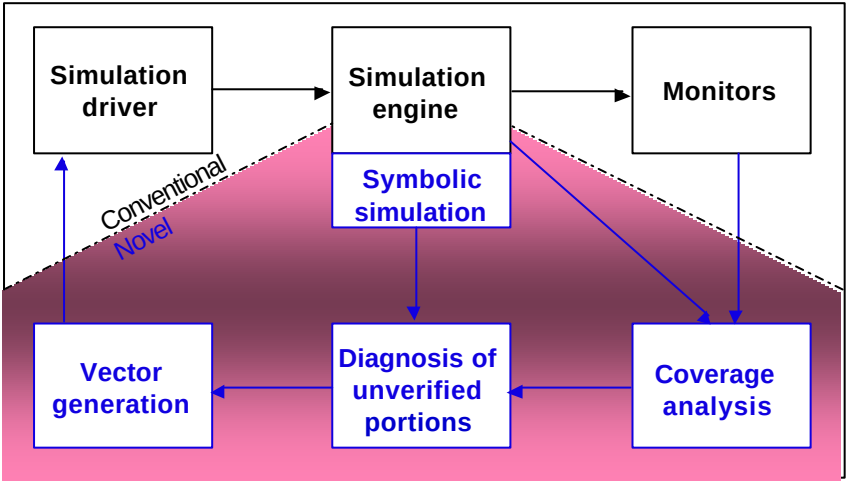
- Ultra-large “FPGA”
- Live hardware, gate-for-gate.
- Entire design or major module is flattened, and compiled at once into multi-FPGA form.
- Logically static circuit-switched interconnect.
- In-circuit or vector-driven
- Regular clock rate, > 1M cps.

Emulation Conclusions

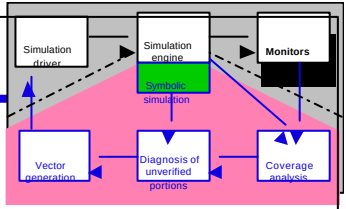
M. Butts - Synopsys

- Market is flat at $100M/year
- Expensive HW, SW, cost of sales
  - High-end supercomputer-like business
- Current competition
  - Simulation farms have similar $/cycle/sec for regression vector sets
  - FPGA-based rapid prototyping for validation, SW execution
- Good solution for large projects that can afford it
- Ultimately the basic concept is limited by IC packaging

How to make it smarter: Intelligent Simulation



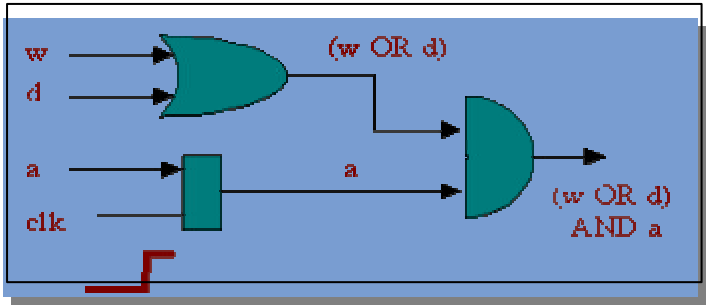
Symbolic Simulation



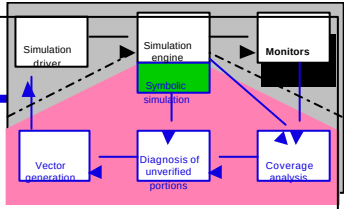
IDEA: One symbolic run covers many runs with concrete values.

Some inputs driven with symbols instead of concrete values

- $2^{(\#\,\text{symbols})}$ **equivalent binary coverage**



Symbolic Simulation



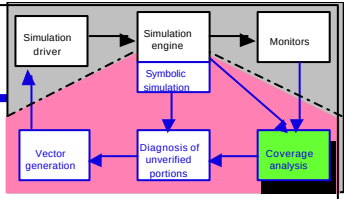
INNOLOGIC: Limitations

- **Capacity limits:**
  - **~ 1 million gate equivalents**
  - **# of symbols - design dependent.**
    - **< 50 in worst cases (multipliers)**
    - **several thousand in the best cases (memory, data movement).**
    - **When out of memory, turn symbols into binary values - coverage lost but simulation completes.**
- **Roughly 10 times slower than Verilog-XL**
- **Can't use in conjunction with Vera or Verisity currently.**

➡ **Definitely worth a shot: Extra cost of symbols offset quickly, doesn't require major change in framework.**

➡ **Full benefits of technology have not been realized yet.**

Coverage Analysis



Why?

- **To quantify comprehensiveness of validation effort**
  - **Tells us when not to stop**
  - **Even with completely formal methods, verification is only as complete as the set of properties checked**
- **To identify aspects of design not adequately exercised**
  - **Guides test/simulation vector generation**
- **Coordinate and compare verification efforts**
  - **Different sets of simulation runs**
  - **Different methods: Model checking, symbolic simulation, ...**

Status of Design Verification

- Software Simulation
  - Too slow
  - Moving to higher levels is helping – but not enough
- Hardware Accelerated Simulation
  - Too expensive
- Emulation
  - Even more expensive
- Rapid prototyping
  - Too ad hoc
- Formal verification
  - Not robust enough
- Intelligent Software Simulation
  - Symbolic simulation – not robust enough
  - Coverage metrics – useful, but not useful enough
  - Automatic vector generation – not robust enough