Design Verification

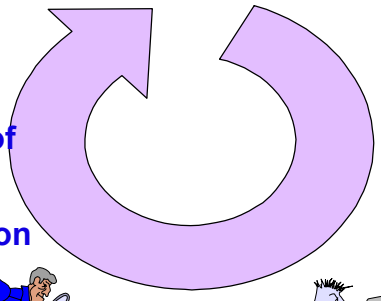
Mike Butts
Synopsys
Prof. Kurt Keutzer
EECS
UC Berkeley

Design Process

Design : specify and enter the design intent

Verify:

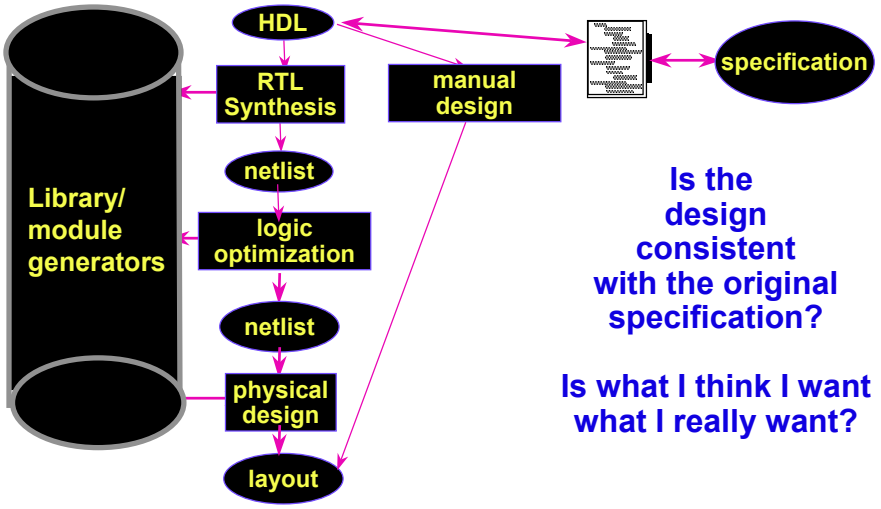
verify the correctness of design and implementation



Implement:

refine the design through all phases

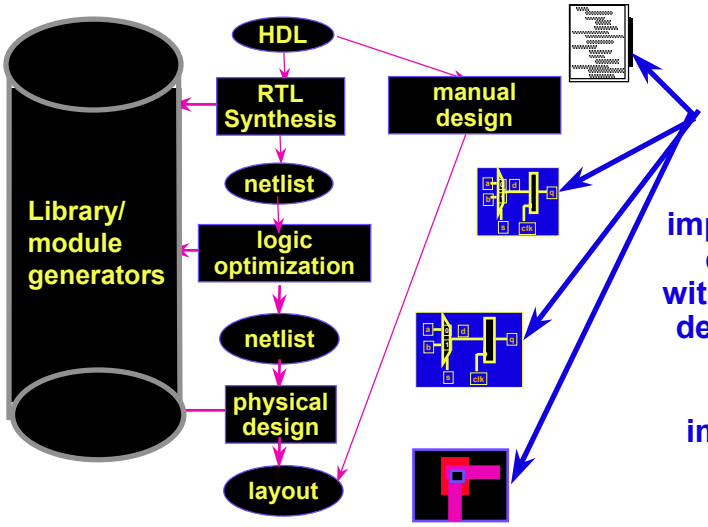
Design Verification



Is the design consistent with the original specification?

Is what I think I want what I really want?

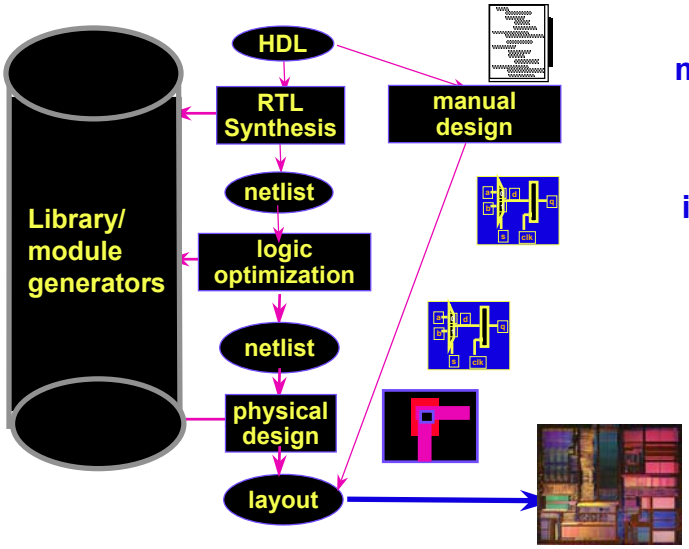
Implementation Verification



Is the implementation consistent with the original design intent?

Is what I implemented what I wanted?

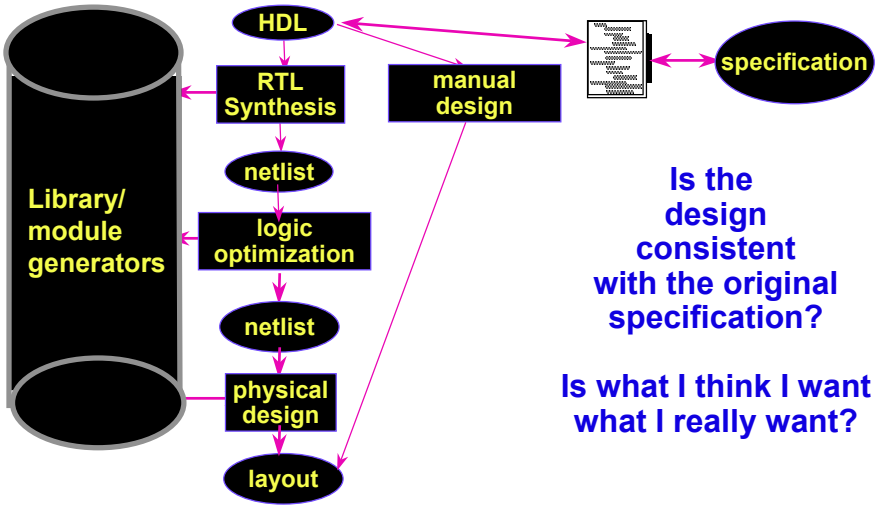
Manufacture Verification (Test)



Is the manufactured circuit consistent with the implemented design?

Did they build what I wanted?

Design Verification



Is the design consistent with the original specification?

Is what I think I want what I really want?

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"

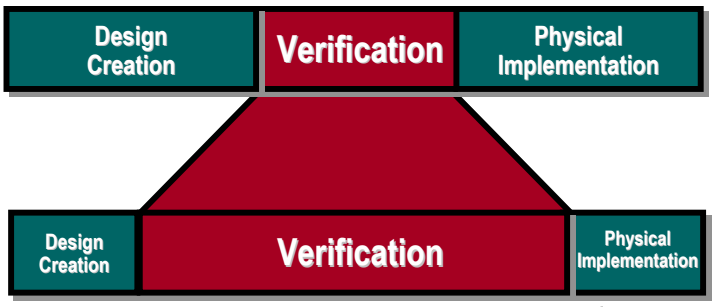
Xerox: "Simulation runtime occupies ~3 weeks of a design cycle"

Ross: 125 Million Vector Regression tests

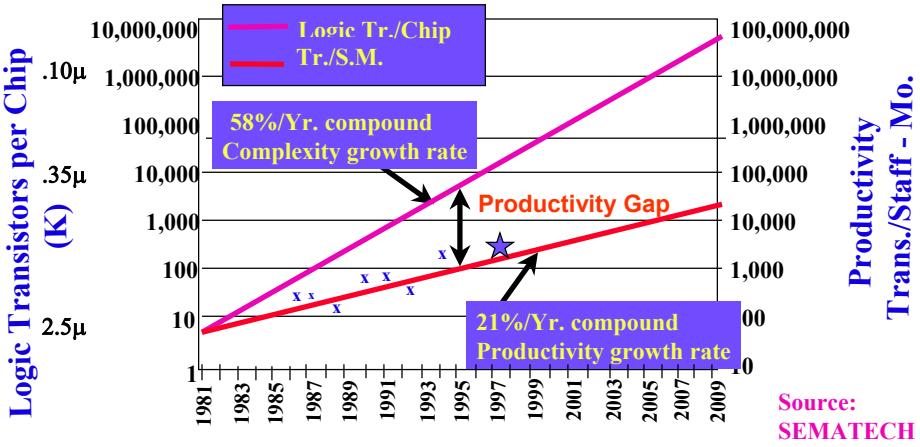
Design Teams are *Desperate* for Faster Simulation

The Verification Crisis

Verification Consumes Hardware Design Cycle

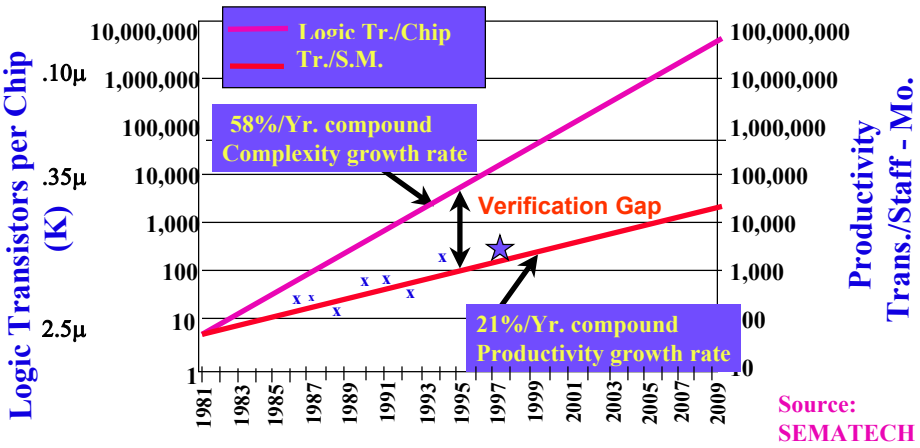


Productivity Gap



Source: SEMATECH

Verification Gap



Source: SEMATECH

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

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

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

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

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

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

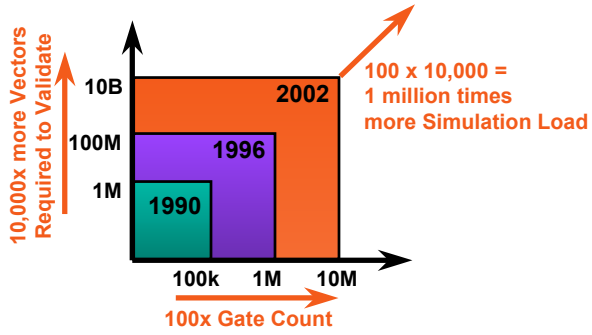
$$\frac{1{,}000{,}000{,}000 \text{ trs}}{\text{chip}} \times \frac{1}{100} \times \frac{1}{10{,}000}$$

$$= \frac{1000 \text{ bugs}}{\text{chip}}$$

5 years!!

The Verification Bottleneck

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



Time to boot VxWorks

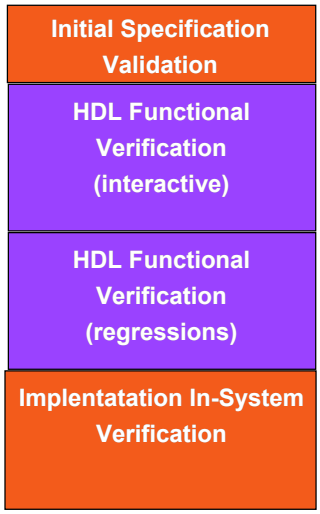
M. Butts - Synopsys

1 million instructions, assume 2 million cycles
Today's verification choices:

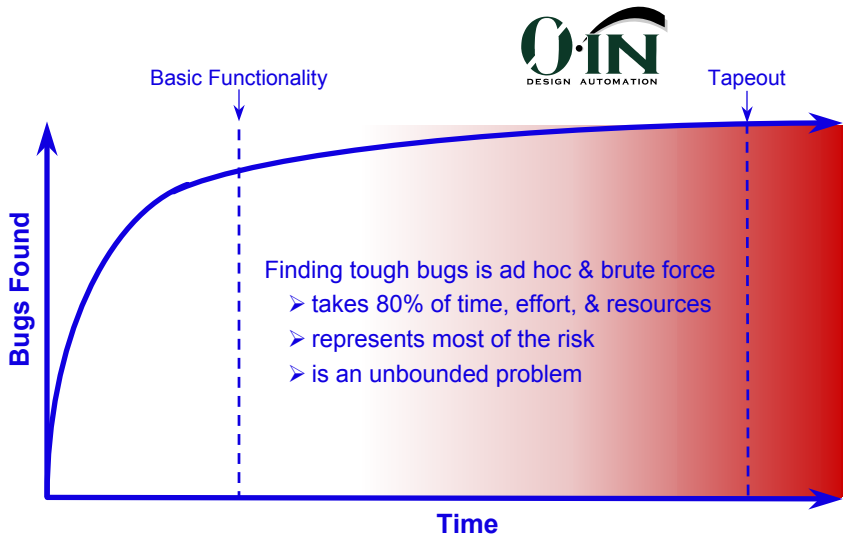
50M cps: 40 msec	**Actual system HW**
5M cps: 400 msec	**Logic emulator[1] (QT Mercury)**
500K cps: 4 sec	**Cycle-based gate accelerator[1] (QT CoBALT)**
50K cps: 40 sec	**Hybrid emulator/simulator[2] (Axis)**
5K cps: 7 min	**Event-driven gate accelerator[2] (Ikos NSIM)**
500 cps: 1.1 hr	
50 cps: 11 hr	**CPU and logic in HDL simulator[3] (VCS)**
5 cps: 4.6 days	

1: assumes CPU chip 2: assumes RTL CPU 3: assumes HDL CPU

Aspects of Design Verification



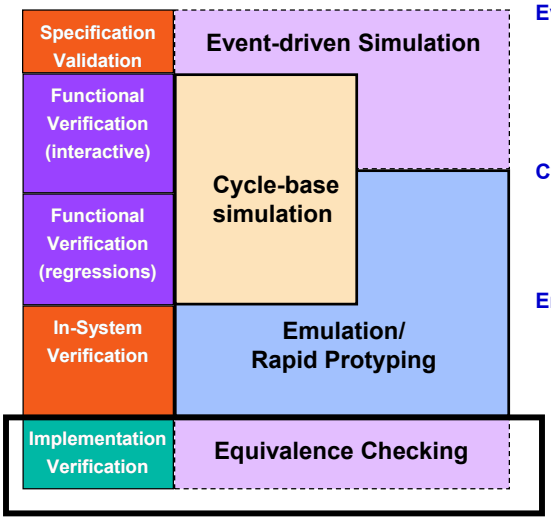
Phases of Design Verification



Technologies for Design Verification

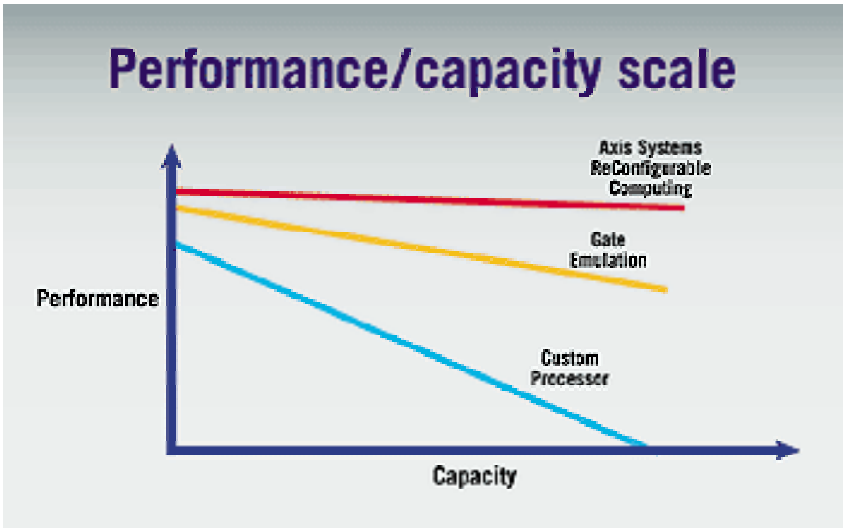
- Software Simulation
  - Application of simulation stimulus to model of circuit
- Hardware Accelerated Simulation
  - Use of special purpose hardware to accelerate simulation of circuit
- Emulation
  - Emulate actual circuit behavior - e.g. using FPGA's
- Rapid prototyping
  - Create a prototype of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

Matching Problems and Technologies



- 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

Axis (Emulator) View



Approaches to Design Verification

Software Simulation
- Application of simulation stimulus to model of circuit

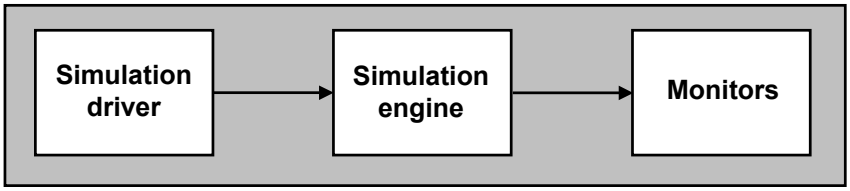
Hardware Accelerated Simulation
- Use of special purpose hardware to accelerate simulation of circuit

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

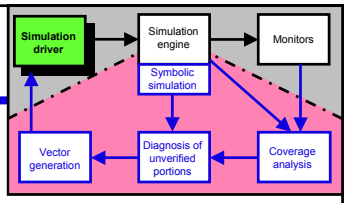
Simulation: The Current Picture



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

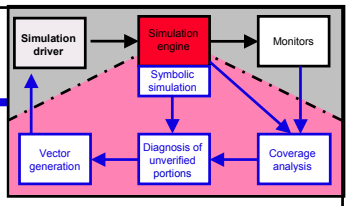
- pre-run (faster, hard to use/maintain)
- on-the-fly (better quality: can react to circuit state)

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.

Simulators



EVENT DRIVEN

- **VCS**
- **Affirma**
- **Verilog-XL, ...**

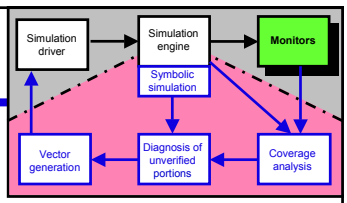
CYCLE-BASED

- **Cyclone VHDL**
- **Cobra, ...**

HYBRID

- **VSS**

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)

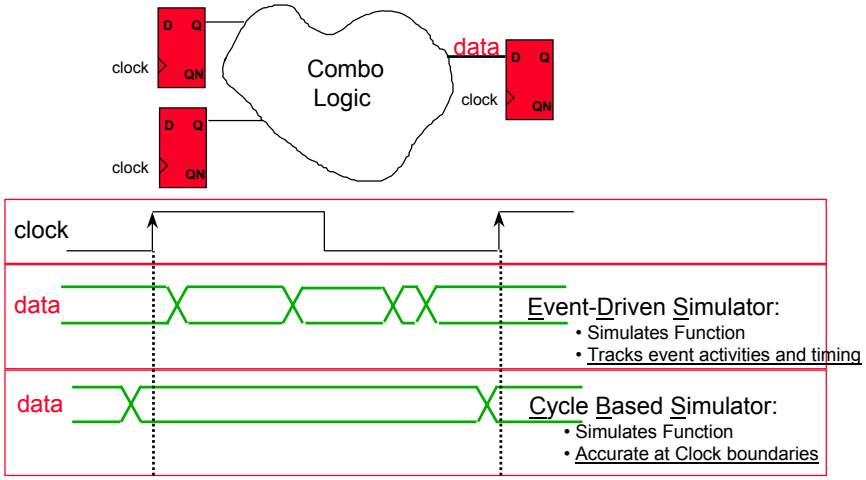
Types of software simulators

- 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, VCC, COSSAP,
- Architecture-specific simulation

Event-driven simulation

- Key elements:
  - Circuit models and libraries
    - cells
    - interconnect
  - Event-wheel
    - Maintains schedules of events
    - Enables sub-cycle timing
- Advantages
  - Timing accuracy
  - Handles asynchronous
- Disadvantage - performance and data management

Event versus cycle-based simulation



Which approach is faster?

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**

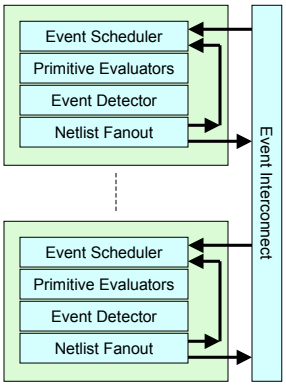
Rapid prototyping

- **Create a prototype of actual hardware**

Formal verification

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

Gate-level Event-driven Sim Acceleration



HW implementation of gate-level event-driven algorithm

- Full timing, many states
- Exploits low-level parallelism (pipelining)

Design partitioned for high-level parallelism

- Limited: irregular topology, event distribution
- Much work in the 1980's: order 10X, not 100X

Performance

- 5G/eval * 100 MHz * 10 procs @ Max. 5B eps
- "7-25X HDL simulator", "500 to 5K cps" (NSIM)

Usability

- Easy to use, quick compilation
- Full timing and states

M. Butts - Synopsys

Gate-level Event-driven Simulation Accelerator

Just one: Ikos NSIM

- 4-input table primitives, RTL synthesis front-ends
- 8 to 64 processors, 0.5M to 15M gates

Value

- Much faster than unaccelerated simulators
- Not quite fast enough to run much code on the design

Competition

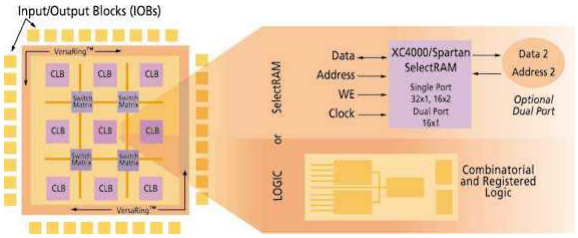
- Modern compiled or cycle-based SW on standard multi-processor platforms
- Gate-level event-driven HW accelerator usually isn't enough better
  - Today's GP multiprocessors exploit low and high-level parallelism

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

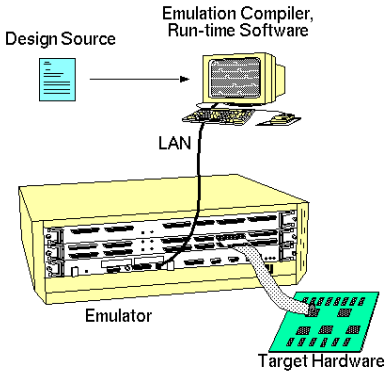
- Today: 2 trillion gate evaluations per second per FPGA (200K gates, 10M cps)
  - Growing with Moore's Law as designs do
  - $1.5B industry behind it (XLNX+ALTR+ACTL)
- Potent tool for logic verification and validation
- How best to put the FPGA to use?



M. Butts - Synopsys

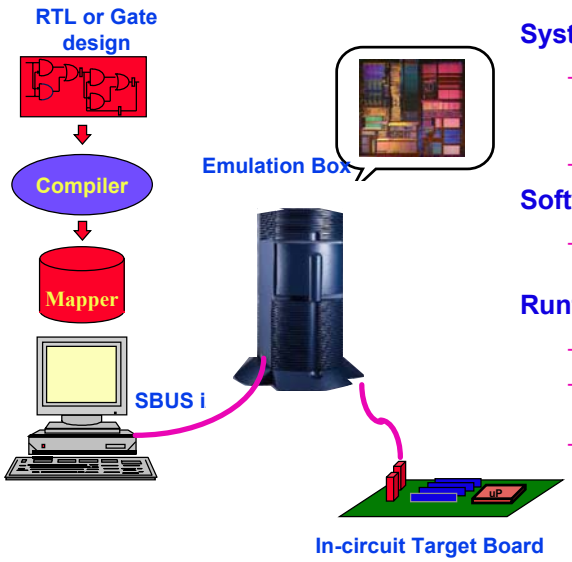
Logic Emulation

M. Butts - Synopsys



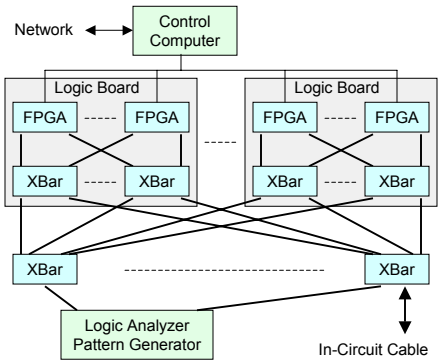
- Ultra-large "FPGA"
- Live hardware, gate-for-gate.
- Entire design or major module is flattened, and compiled at once into multi-FPGA form.
- Logically static circuit-switched interconnect.
- In-circuit or vector-driven
- Regular clock rate, > 1M cps.

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

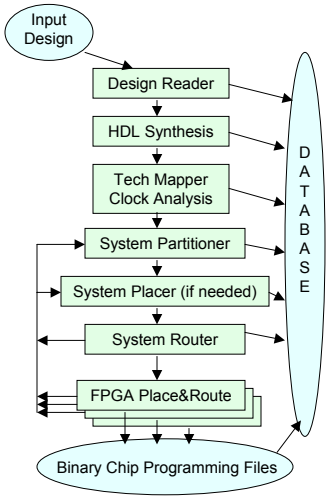
General Logic Emulation HW



- Tens to hundreds of large FPGAs
- Interconnect, either:
  - Programmable crossbars (QT), or
  - Nearest-neighbor with time-multiplexing (Ikos).
- SRAMs for modeling memory
- CPUs for behavioral simulation & testbenches (QT Mercury)
- Dedicated logic analyzer / pattern generator for visibility & vectors
- In-circuit cable plugs into target

M. Butts - Synopsys

General Logic Emulation SW



- Entire design is flattened
  - Emulation-specific HDL synthesis
- Clock tree timing analysis
  - To avoid functional errors when gated clocks get split across FPGAs
- Multi-level, multi-way partitioning
  - NP-hard, very compute intensive
- System placement (Ikos only)
- Place & route for every FPGA
  - Can be run in parallel
  - Interdependent due to interconnect
- Design database system
- Needs to be automatic and totally successful

M. Butts - Synopsys

Cycle-based Emulator

Levelized compiled simulation in massively parallel hardware form

- All gates evaluate every cycle
- No run-time data dependencies, so processors and IPC network are scheduled at compile time

Severe design constraints

- No asynchronous feedback, latches, etc.
- No timing: multiple related clock domains only by LCD slowdown
- Commonly OK for microprocessors, much less so in general

Compilation

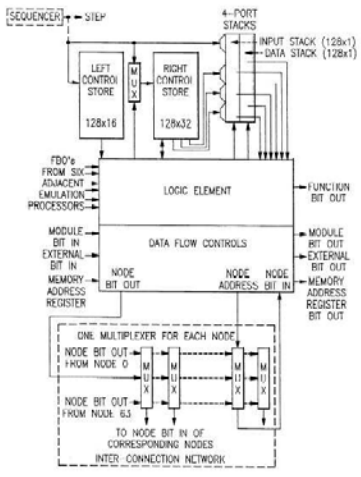
- Given design constraints, relatively easy to use
- Fast: 2M gates per hour (CoBALT)

History

- IBM: Yorktown Simulation Engine, ET3 / Quickturn CoBALT
- Arkos=> Synopsys => Quickturn => Cadence

M. Butts - Synopsys

Cycle-based Emulator



M. Butts - Synopsys

Just one: Cadence/Quickturn CoBALT

IBM Poughkeepsie ET3 technology

- 500 MHz custom chip, compiler core
- Up to 20M ASIC gates:
  - 128 3-input prims / processor (CE, new CL: 2.5X)
  - 64 processors per chip
  - 64 chips per board, 8 boards

Performance

- 32 trillion gate evaluations /sec (max)
  (2 gate equivalents / processor cycle * 64 processors/chip * 64 chips/board * 8 boards * 500 MHz)
- 10K to 500K cps in actual practice

Usage

- Vector memories or in-circuit cable
- PCI link to workstation simulator

Cycle-based Emulator

Much faster than SW or event-driven accelerator

Runs actual code and data, in actual target systems

Harder to use than SW or event-driven accelerator, but easier than emulator

Severe restrictions on design style

- Purely synchronous design OK, else No.

Expensive, complex, proprietary HW, SW

- Custom chips, interconnect, PCBs, connectors, chassis, instrumentation
- Compiler is substantial effort to develop & maintain

Isolated from simulation, separate environment, proprietary simulator

Conclusion:

- Good solution for large fully synchronous projects that can afford it
- Not a mainstream technology

M. Butts - Synopsys

State-of-art in CBE: Cobalt Ultra

- 112 Million ASIC gates
- 64 Gbytes Memory
- 4224 Bi-Directional I/O
- Compiles 4M ASIC gates/hour
- Performance up to 600KHz

Characteristics of Logic Emulation

Maximum Validation, fastest runtime speed

- **Runs actual code and data, in actual target systems**

No restrictions on design style

- **Gated clocks split across FPGAs may cause correctable functional errors**

Expensive, complex, proprietary HW, SW

- **Interconnect, PCBs, connectors, instrumentation; big FPGA tech. lag**
- **Compiler is hard to develop & maintain, user must be full-time expert**

Inflexible

- **Interconnect architecture makes FPGAs interdependent - changes often cause long recompile**

Isolated from simulation or integrated with proprietary simulator

It's HW speed, but not design speed; target HW slowdown required

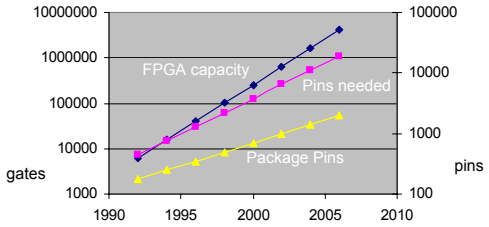
FPGA logic capacity tracks Moore's Law...

But interconnect capacity does not track Moore's Law.

M. Butts - Synopsys

The Emulation Interconnect Problem

M. Butts - Synopsys



FPGA capacity is emulation usage: 8 gates / 4-LUT+FF, 75% packing.

Pins needed is for emulation usage: $p = 2.75g^{0.58}$

Package pins are Xilinx FPGA IOBs (1991-2000, extrapolated afterwards).

Rent's Rule ($p = Kg^{r}$) applies to partitioned designs.

FPGA logic capacity: 2X / 1.5 yr (Moore's Law)

FPGA pins needed by emulator: 2X / 2.5 yr (Moore + Rent)

Package pins: 2X / 4 yr - Can't keep up.

Vendors are time-multiplexing pins more and more to compensate.

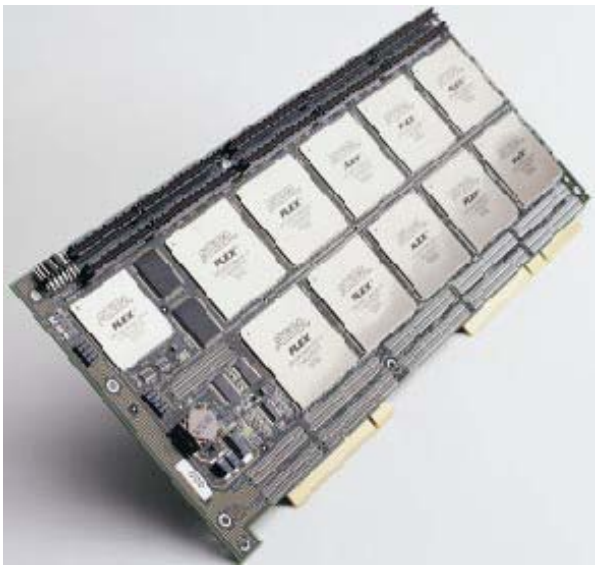
- **But that's only a linear effect; it does not change the doubling time.**

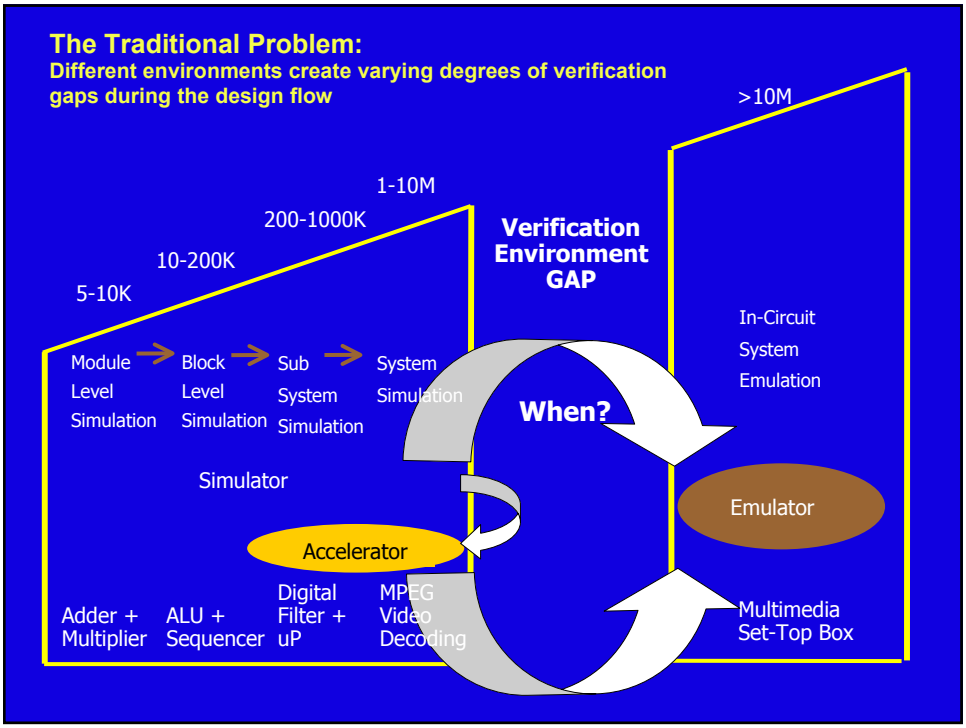
Emulation Conclusions

M. Butts - Synopsys

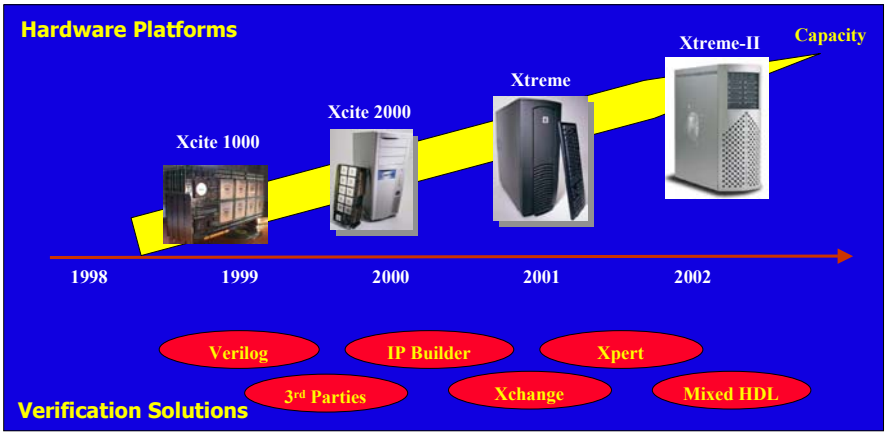
- Market is flat at $100M/year
- Expensive HW, SW, cost of sales
  - High-end supercomputer-like business
- Current competition
  - Simulation farms have similar $/cycle/sec for regression vector sets
  - FPGA-based rapid prototyping for validation, SW execution
- Good solution for large projects that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

Axis Reconfigurable Computing





Axis Product Overview



Recently announced Xtreme-II & Mixed HDL

Axis Product Characteristics

Xcite 1000	2M gate	100K cycle/sec
Xcite 2000	10M gate	200K cycle/sec
Xtreme	10M gate	500K cycle/sec
Xtremell	100M gate	1,000K c/sec

Axis Extreme Characteristics

1. Hardware Architecture
2. Design Format
3. Verilog Software Simulator: Xsim
4. Capacity
5. Simulation Performance
6. Emulation Performance
7. Memory

- Debugger Native Verilog interface. Built-in Clock Generators Supports up to 48 clock domains
- Programmable Trigger Generators Supports up to 1024 probes per trigger and up to 48 separate trigger generators

1. Re-configurable computing engine
2. RTL and Gate Level
3. Native compiled with event look-ahead. Supports all Verilog constructs including PLI
4. Up to 20M gates
5. Up to 100K cycles/second
6. Up to 500K cycles/second
7. Up to 384M bits of on-board RCC memory. Up to 18.75M bits of internal cache. Expanding up to 4G of memory mapped workstation memory

- All internal nodes visible. Real time emulation/simulation state swap between software and hardware

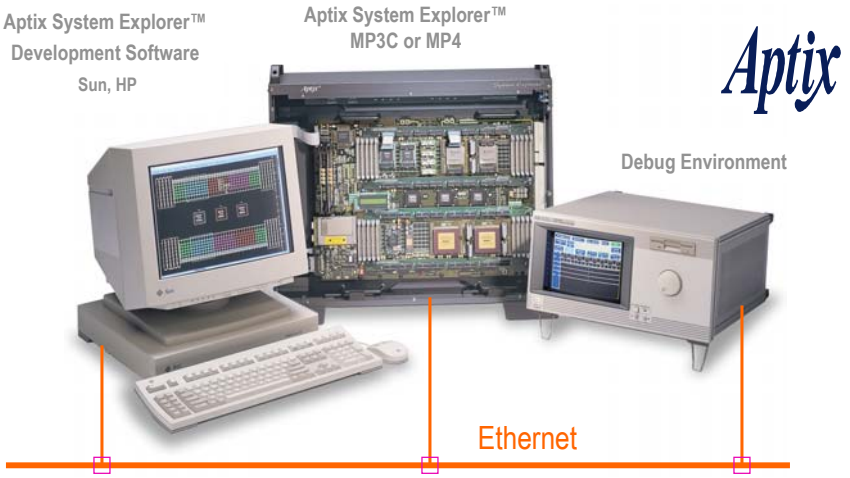
Axis Xtreme

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

Rapid System Prototyping Environment

- Need low-cost, instrument-like system prototyping environment
- Must be well-integrated into overall component-based flow



Rapid Prototyping of ASICs and SoCs

Target-specific tools

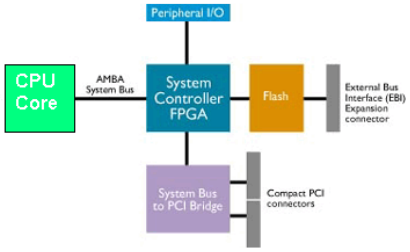
- **ASIC/core+FPGA: Philips/VLSI Velocity, ARM ($5K)**
- **FPGA+RAM: Altera/ARC "SoC" board (100KG, $5K)**

GP tool

- **Aptix: daughtercards, prog. breadboard, > $100K**

Rapid Prototyping Characteristics

- **Real HW running at MHz, low cost HW**
- **Isolated from simulation, throwaway effort**



M. Butts - Synopsys

Summary

Design Verification IS the biggest problem in IC design today

Verification teams ARE getting larger than design teams

No silver-bullet solutions on the horizon

Successful groups

- Intel, NVidia, IBM – use a bit of everything
- Leading adopters of new technology

Buying behaviors in software verification are poor

- All software solutions seen as ``simulators'' – poor ASPs

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

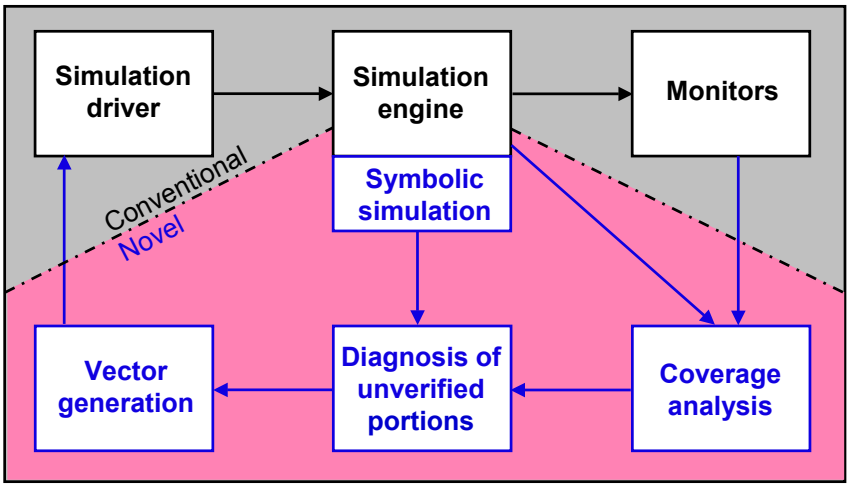
Rapid prototyping

- Create a prototype of actual hardware

Formal verification

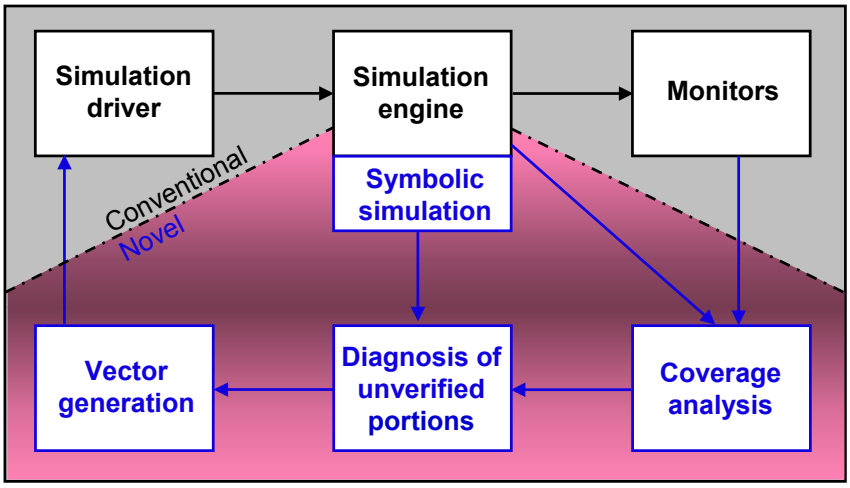
- Model checking - verify properties relative to model
- Theorem proving - prove theorems regarding properties of a model

How to make it smarter: Intelligent Simulation

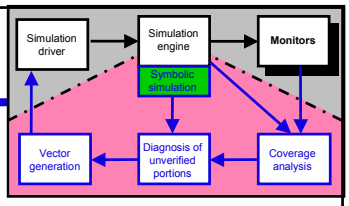


How to make it smarter: Intelligent Simulation

CLOSED FEEDBACK LOOP



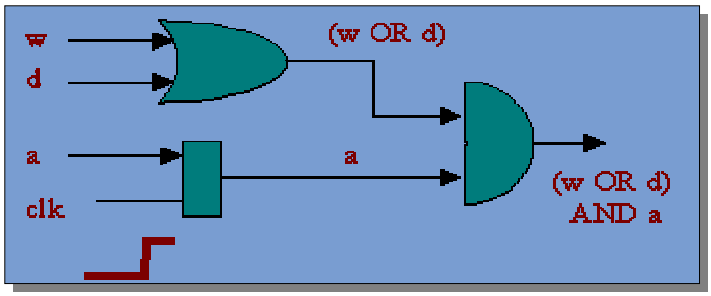
Symbolic Simulation



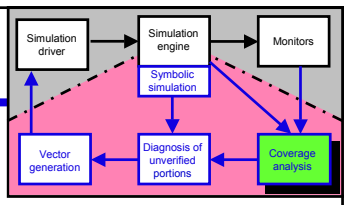
IDEA: One symbolic run covers many runs with concrete values.

Some inputs driven with symbols instead of concrete values

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



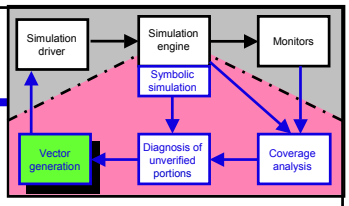
Coverage Analysis



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, ...**

Vector Generation

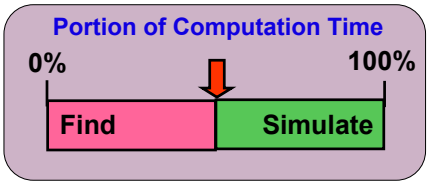


Classification:

- Algorithmic methods
  - Guided search of state-space
    - Traverse "more relevant" portion
  - Vector generation aimed at coverage
    - Generate input stimuli to
- "Randomized" methods

Trade-off between

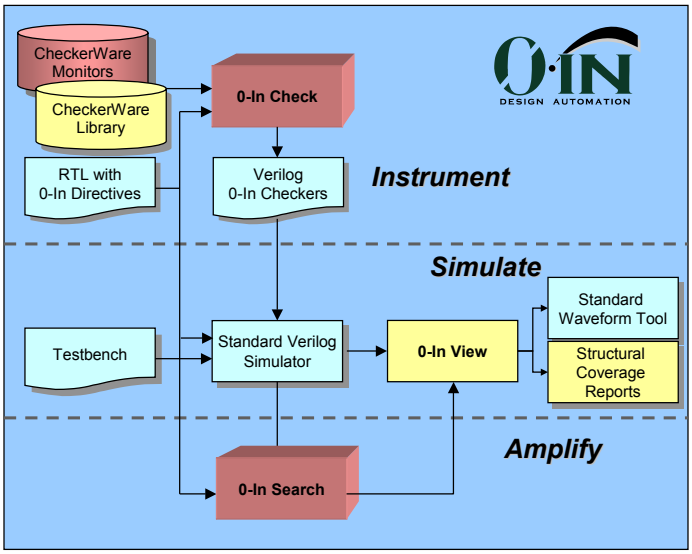
- Time to find "good" vectors
- Time to simulate vectors



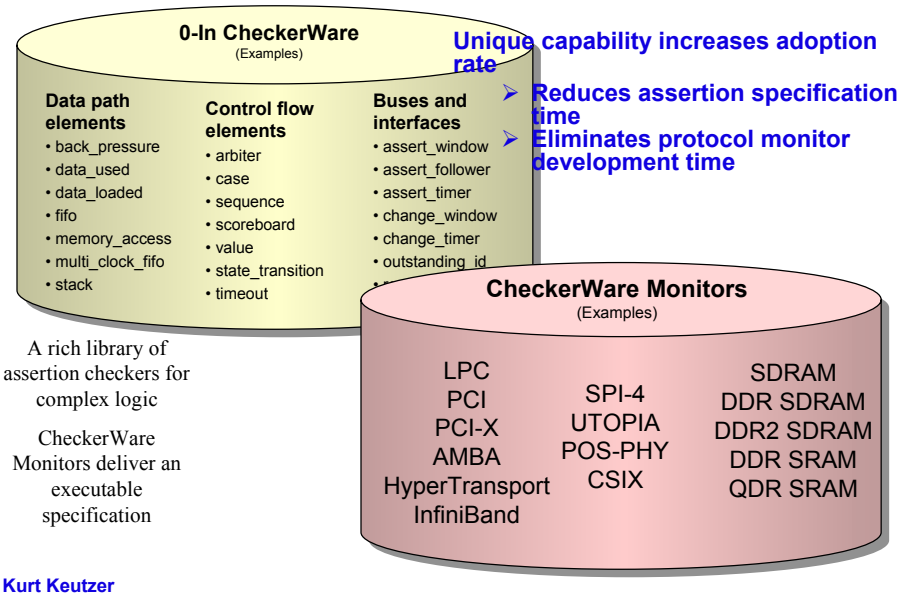
Improved Time to Market: More Efficient & Effective Verification

0-In extends the value of simulation with white-box verification

0-In brings the power of formal verification to a simulation-based methodology



CheckerWare Library



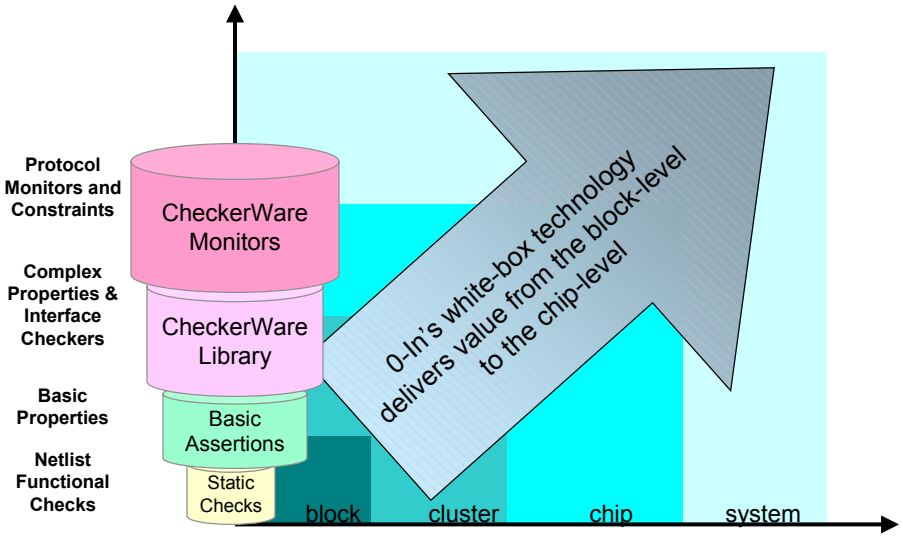
Unique capability increases adoption rate

- **Reduces assertion specification time**
- **Eliminates protocol monitor development time**

A rich library of assertion checkers for complex logic

CheckerWare Monitors deliver an executable specification

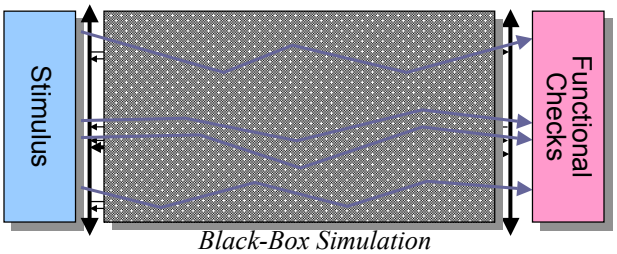
Solving the Critical Problems



0-In Check Makes Simulation More Efficient

Detects bugs earlier when they are less expensive

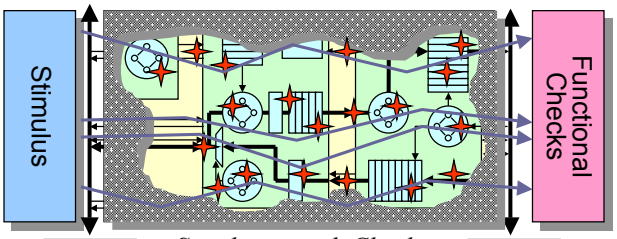
Reduces debugging time



Black-Box Simulation

Reduces test redundancy

Reduces reliance on brute-force methods



checker

bus monitor

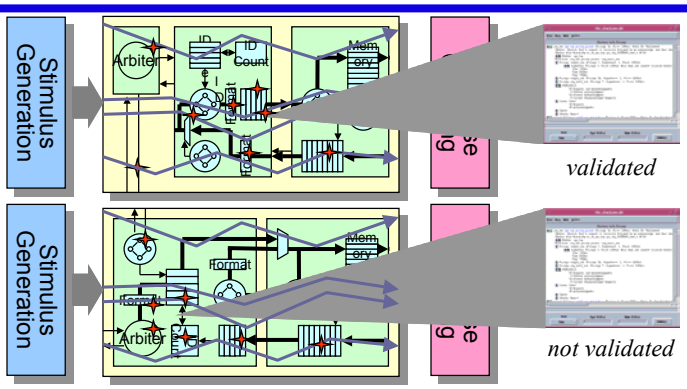
Simulation with Checkers

bus monitor

Structural Coverage

Coverage metric drives methodology

Testbenches are developed with Specman, Vera, C/C++, Superlog, Verilog, etc.



validated

not validated

Structural Coverage is built into CheckerWare

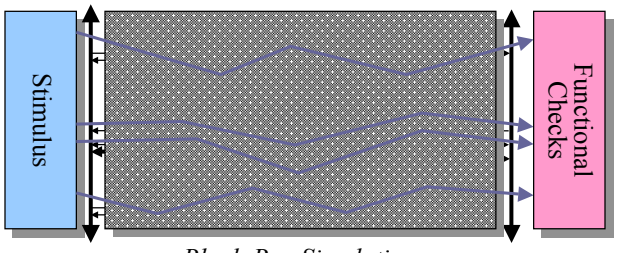
- Effective method for testbench grading
- Implementation-specific
  - Checkers capture structural characteristics of design
  - Familiar RTL structures (memories, FIFOs, state machines, etc.)
  - Checker-specific corner cases (FIFO full, FIFO empty, etc.)
- Objective and actionable
  - Guide development of additional tests to plug verification holes

checker

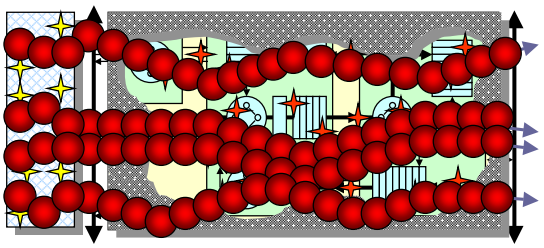
0-In Search Makes Simulation More Effective

Reduces reliance on manually-written directed tests

Finds bugs simulation misses



Black-Box Simulation



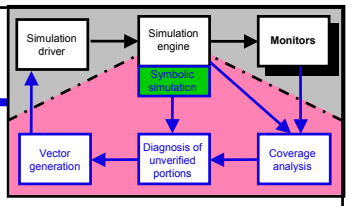
checker
constraint

0-In Search

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 compehensive enough
  - Automatic vector generation – not robust enough

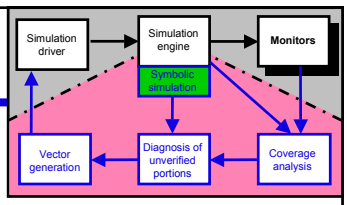
Symbolic Simulation



INNOLOGIC:
BDD-based symbolic Verilog simulators

- **ESP-XV: For processor and networking applications**
- **ESP-CV: For memory verification and sequential equivalence checking**
- **Monitors can have symbolic expressions**
- **Can symbolize time, e.g., event occurring after time T, 10 < T < 20.**
- **If bug is found, computes actual values exercising it**
- **Current "sweet-spots" of technology**
  - **Memory verification: CAMs, caches, register files**
  - **Unit level RTL functional verification: DMA, PCI,100-1000K gate blocks**
  - **Data movement, datapath**

Symbolic Simulation



INNOLOGIC: Limitations

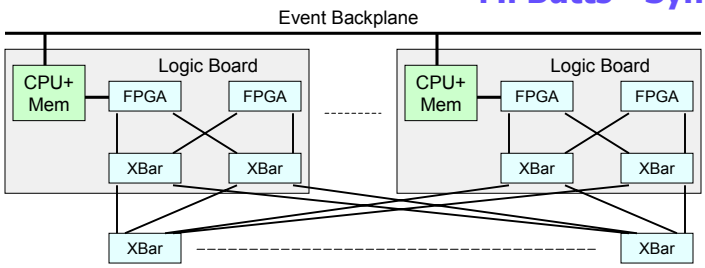
- **Capacity limits:**
  - **~ 1 million gate equivalents**
  - **# of symbols - design dependent.**
    - **< 50 in worst cases (multipliers)**
    - **several thousand in the best cases (memory, data movement).**
    - **When out of memory, turn symbols into binary values - coverage lost but simulation 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.**

Emulation + Accelerated Simulation

M. Butts - Synopsys



QT Mercury SimServer

Bauer, Bershteyn, Kaplan, Vyedin. A Reconfigurable Logic Machine for Fast Event-Driven Simulation, *Proc. 35th DAC*, 1998.

- **Multiprocessing HW-accelerated Verilog simulator + emulator**
- **Automatic HDL partitioning: synthesizable modules to emulator, behavioral modules to PowerPC CPUs (up to 10)**
- **Accelerated time wheel, event detection in emulator FPGAs**