VLSI Design Methodology

Ming-Hwa Wang, Ph.D.
COEN 388 Principles of Computer-Aided Engineering Design
Department of Computer Engineering
Santa Clara University

Topics

- Introduction
- Design flow and technology
- Design languages
- System approach
- Front-end design tools
- Back-end design tools
- Analog design tools
- Design verification tools
- Companies and recession
- Questions and answers

Introduction

High-tech Industry

- 20% of all US revenues are directly from high-tech industry; even more are indirectly from high-tech industry
- High profit margin
- Dow Jones index and NASDAQ index
- The heart of this industry: Silicon Valley

Design flow and technology

- Design process and design flow
- Performance
- Flexibility
- Power
- EDA

Design process and design flow

- Design process
- Presentation of design process
- Tools for design automation
- Engineers for design process
- Flow automation

design process	presentation	tools
Design idea	Flow graph	Graphic/spreadsheet capturing
Behavior design	Pseudo/high-level code	Behavior compiler
Data path design	Bus & register structure	Synthesis tool
Logic design	Gate wire list, netlist	Simulator, Timing tool
Physical design	Transistor list, layout	Place and route, Design rule check, Schematic capturing
Manufacturing	Chip or board	Signal integrity

Engineers for design process

- Chief Technical Officer (CTO)
- Project manager
- Architecture
- Logic designer
- Circuit designer
- Verification engineer
- CAD engineer
- DFT engineer
- Layout engineer
- Test engineer
- System engineer
- QA engineer
- Process engineer
- Product engineer

Flow Automation

- Perl (Practical Extraction and Report Language) script to automate design flow
- Tcl front-end interface and code in C/C++
- Python
- Configuration file to make automation more flexible
- Command line interface for Unix/Linux users
- GUI interface for Window or novice users

Perl

- Perl interpreter and compiler
- Regular expressions
- Scalar variable $, array variable @, and hash or associated array %
- C-like operators, controls, and statements
- Input/output, file tests and operations, directory and system interactions, networking, DB, etc.
- Object-oriented since Perl5

Tool command language and X-window extension (tcl/tk)

- Tcl script and interpreter
- A "glue" language
- Writing tcl applications in C/C++
- GUI using tk

Performance

- Design for faster chips
  - Vacuum tube, transistor, IC, VLSI
  - Double the performance every 18 months*

- Design for smaller chips using better process
  - Quadrupled capacity every 3 years*
- Technology: ASIC, custom design
  - RTL vs. Gate, semi-custom vs. full-custom flow
  - Tradeoff between performance and cost
- Process
  - 0.18, 0.13, 90, 65, 45

* Patterson & Hennessy "Computer Organization & Design" 1998

Flexibility
- Combining ASIC with DSP
- FPGA
- ASSP
- Reconfigurable ASIC

Combining ASIC with DSP
- General-purpose processor
- Digital Signal Processor (DSP)
  - Discrete Fourier transformation
  - Multiplication and accumulation
- Application-Specific Integrated Circuit (ASIC)
- Tradeoff between performance and flexibility
- Vendor: TI, Motorola

Field Programmable Gate Arrays
- Flexibility
  - PROM, EPROM, EEPROM, PLD/PAL/PLA (sum-of-product), MPGA/FPGA (mask-/field-programmable)
- Time-to-market
  - Reduce development and production time
- Low-cost prototype
  - Without integrated circuit fabrication facility
- Performance, program time (minutes), CAD tools
- Vendor: Xilinx, Altera, etc.

ASSP
- Application-Specific Standard Products (ASSP)
- Generic ASIC blocks/modules
- Manufactured in high volume
- Program by re-route or final-manufacturing
- Vendor: SemiView

Reconfigurable ASIC
- Adaptive Computing Machine (ACM)
- ASIC performance with programmability
- Shared hardware components
- ASIC flow and technology
- Difficult to developing and difficult to programming
- Vendor: Quicksilver Technology

Power
- Power consumption and heat
  - Fan or cooling device
- Portable and handheld devices
  - Cellular phone, PDA, camcorder, laptop, etc.
- Long-life battery
- Low power design
  - Power estimation
  - Shutdown clock when not in use

EDA (Electronic Design Automation)
- What is EDA?
  - What does EDA think of itself?
  - What do users think of EDA?
- What are the EDA emphases?
- How is EDA methodology changing?
- EDA industry problems
- Who are the EDA players?

"About half of ASIC designs are now SoCs, and that percentage is growing rapidly." - Bryan Lewis, Chief Analyst - Gartner/Dataquest
"ASIC design starts peaked at an annualized rate of 10,500 in 1996 and will decline to under 2,500 in 2005" - Erich Desai, Analyst, American Tech. Research

What Is EDA?
- What does EDA think of itself?
  - EDA industry views itself as the head of the electronics food chain
  - EDA continues to be undervalued
  - EDA values proposition
- What do users think of EDA?
  - Consumer view of EDA

EDA values proposition
- Productivity, Productivity, Productivity!!!
- Establishing standards, but striving to maintain proprietary value
- Large companies position solutions for DFM, physical layout, verification
- Emerging companies position flow consolidation i.e. RTL to GDSII tools, SL formal tools, FPGA Design.
- Companies offering point tools are being swallowed up at a fierce rate.

Consumer View of EDA
- Design Gap
  - Divergence of design size growth and tool capacity growth.
- Financial interaction changing from capital to expense
  - Trial, time-based, and annual licensing versus perpetual licensing
  - Majors use "all you can eat" to stifle competition
- Support more often turning into design services

- Time zone support expected

What are the EDA emphases?
- Closing the "design gap"
- Moving up the abstraction level
  - Early 90s - Schematics for HDL capture and simulation
  - Mid, late-90s – IP (Intellectual Property)
  - Now - ESL (Electronic System Level)
    - Silicon Virtual Prototype
    - Algorithmic Design
    - Co-Design (H/W - S/W convergence)
- Control the supply chain
  - DFM (Design For Manufacturing)

  "The things you do after you think the design is done" - Lucio Lanza (partner, USVP)
- Design Services and IP development
  - Taligent and Intrinsix

How is EDA methodology changing?
- IP, SoC, DSP, VSP (Silicon Virtual Prototype) methods change the EDA scope
- FPGA design emerges as growth driver
- Classic ASIC design methods shrink as ASIC design starts to plummet
  - RTL becomes intermediate format
  - RTL sim too slow
- ESL is taking hold
  - System C and System Verilog battle for position
  - CoWare, Monterey, Magma, Verisity, Forte emerge
  - Majors rush to integrate system language extensions to ASIC/FPGA environments

EDA industry problems
- Current licensing models and customer relationships limit growth
- Design automation now a more extensive problem, requiring multiple technologies to solve
- More designs incorporate mixed signal, embedded, and linear/RF on the same chip
- Successful design strategy must include signal integrity and power optimization
- FPGAs are becoming the dominant methodology in pure digital design

Who are the EDA players?
- Vendors
  - Majors: Synopsys, Cadence, Mentor
  - Fast movers: Magma, Verisity, Simplicity, Nassda, Sequence, Axis
  - Emerging: Forte, Hier, Celoxica, Aldec, Accelchip, Virage, Mathworks, Tera
  - New: Europe/Asia: esp. mixed signal, ESL, Anasift, EVE, Chipvison
- Conferences
  - DAC (Design Automation Conference), IEEE-CAD
- Editors
  - EETimes, Richard Goering, Mike Santarini
  - Chip Design Mag, Tets Maniwa
- Analysts
- Gary Smith/Daya Nathramundi, Gartner/Dataquest
- John Cooley, Deepchip
- Erich Desai, Soundview Financial

Design Languages
- Try-and-error approach
- Hardware Design Languages (HDL)
- Modeling languages
- Merge modeling languages and HDL

Try-and-error approach
- Cost
- Time-consuming

Hardware Description Languages (HDL)
- Parallel vs. sequential
- Mixed-level HDL
- Proprietary HDL
- Verilog HDL
- VHDL

Parallel vs. Sequential
- Software programming languages
  - Sequential execution high-level languages
  - Parallel/vector compilers convert sequential to parallel
  - Parallel languages: not very popular
- Hardware description languages
  - Parallel execution, concurrency

Mixed-Level HDL
- Behavior or functional level
- Register Transfer Level (RTL)
- Gate level
- Transistor or switch level

Proprietary HDL
- A Hardware Programming Languages (AHPL)
- Computer Design Language (CDL)
- Consensus Language (CONLAN)
- Interactive Design Languages (IDL)
- Instruction Set Processor Specification (ISPS)
- Test Generation And Simulation (TEGAS)
- Texas Instructions Hardware Description Language (TI-HDL)
- ZEUS hardware description language

- Others: SGI newt/reptile HDL, etc.

Verilog HDL
- A C-like de facto standard language (IEEE std 1364-1995/2001)
- Verilog
  - Modules, instances, ports, width, connections, gates, delay (min/typ/max), continuous assignment, expression, operators, initial statements, always statements, blocking/non-blocking statements, conditional statements, loops, sequential and nested blocks, tasks, functions, PLI/VPI, etc.
- Vendor: Gateway

Verilog Example
```
module inv (in, out);
    input in;
    output out;

    assign out = ~in;
endmodule

module test;
    wire out;
    reg in;
    inv i (.in(in), .out(out));

    initial begin
        $monitor("in = %b,
        out = %b\n", in, out);
        #1  in = 1/b0;
        #1  in = 1'b1;
    end
endmodule
```

VHDL
- Very High-speed IC Hardware Description Language (VHDL)
- IEEE standard 1076-1987/2002
- Ada-like, better language but too verbose
- VHDL
  - Entity, architecture, package, configuration, process, concurrency, component, port, initial/transport/delta delay, procedure, function, operator, signal, variable, constant, generic, generate, type, array, subtype, record, attribute, while, for, if/then/elsif/else, case, guarded, select, when, begin, end, wait on/for/until, read, write, assert, report, severity, library, use, etc.

VHDL Example
```
ENTITY inv IS
    PORT (
        i1: IN BIT; o1: OUT BIT
    );
END inv;

ENTITY test IS
END test;

ARCHITECTURE gate OF inv IS
BEGIN
    o1 <= NOT i1 AFTER 1 ns;
END gate;

ARCHITECTURE gate OF test IS
    COMPONENT inv1 PORT (
        i1: IN BIT; o1: OUT BIT
    );
    END COMPONENT;
    SIGNAL s1, s2 : BIT;
BEGIN
    g1 : inv1 PORT MAP (s1, s2);
    apply_data (s1, 0& 1&, 10 NS);
END gate;

CONFIGURATION gate OF test IS
    FOR g1 : inv1 USE ENTITY WORK.inv (gate);
    END FOR;
END gate;
```

Modeling Languages
- High level, functional level, behavior level
- No timing vs. cycle accurate
- C/C++
  - Need co-simulation
  - PLI (Programming Language Interface) and VPI (Verilog Procedural Interface) for Verilog

Merge Modeling Languages and HDL
- SystemC
- System Verilog

SystemC
- SystemC with master/slave library and TLM library
- Based on C++ object-oriented paradigm and added HDL features, most used for verification
- Single language for both modeling and HDL
- No simulator needed: simply use gcc
- Untimed/timed functional level, bus cycle accurate (BCA) level, cycle accurate (CA) level, RTL level, gate level
- Emphasize more on the modeling side, executable specification
- Synthesis to Verilog/VHDL gate level design

- Open source

http://www.systemc.org

SystemC Example

```
#include "systemc.h"
SC_MODULE(test) {
    sc_in_clk clk;
    void doit ( ) {
    cout << clk << endl;
    }
    SC_CTOR(test) {
    SC_METHOD(doit);
        sensitive << clk;
    }
}

int sc_main(int argc, char *argv[ ]) {
    sc_clock clk("clk", 20);
    test *t = new test("t");
    t->clk(clk);
    sc_start(41);
    return 0;
}
```

System Verilog

- Originated from HDL and toward the system level
- Most used for implementation
- Constraint random test generation
- Emphasize on design verification
- Assertions

System Approach

- Chip company vs. system company
- Hardware ECO vs. software patch
- System guided design
- System approach and SoC

Chip Company vs. System Company

- Tradeoff between develop effort and profit
- Chip company
  - Hardware only
  - e.g., hub, switch
- System company
  - Hardware and software
  - e.g., router

Hardware ECO vs. Software Patch

- Hardware changes (ECOs) are expensive
- Software changes are cheap
- Hardware less bugs
- Software more bugs
- Hardware bugs patched by software

System Guided Design

- Embedded system design
  - System debugging, task swapping, etc.
- VLIW chip
  - Very Long Instruction Word (VLIW)
  - Transmeta
- Wireless communication DSP chips
  - Convolution code and bit processing (scrambling, interleaving, puncturing, etc)

System approach and SoC

- System on Chip (SoC)
  - CPU, bus, I/O
  - Hardware and software

Front-End Design Tools

- Simulator
- Sanity Checking
- Productivity tools
- Coverage tools
- Formal verification
- Synthesis
- Design For Test (DFT)
- Debug and waveform viewing tools

Simulator

- Major tools vs. complement tools
- Digital vs. analog
- RTL/gate/switch/timing simulation
- Two-state vs. four-state
- Event-driven vs. cycle-based
- Interpreter vs. compiled code
- Hardware accelerator and emulation

Major Tools vs. Complement Tools

- Major tools
  - The primary tools; can't be replaced
  - Much slower than real hardware
  - Bottleneck of the whole design process
- Complement tools
  - Can help reduce simulation time, sometimes by a lot

Digital vs. Analog

- Digital
  - Discrete

- Analog
  - Continuous

RTL/Gate/Switch/Timing Simulation
- RTL
- Gate
- Switch
  - Meta Hspice
- Timing
- Speed comparison

Two-State vs. Four-State
- Two-state
  - 0 and 1
- Four-state
  - 0, 1, x (don't care), z (high impedance)
- Speed and memory requirement comparison

Event-Driven vs. Cycle-Based
- Event-driven
  - Time wheel
  - Event queue
- Cycle-based
  - Activity percentage
  - Parallel evaluation
- Speed comparison and cache locality

Interpreter vs. Compiled Code
- Interpreter simulator
  - Cadence VerilogXL
- Compiled code (native code) simulator
  - Synopsys VCS
  - Cadence NC-Verilog
- Speed comparison

Hardware Accelerator and Emulation
- Hardware/software co-simulation/verification
  - Integration and verification of hw/sw is performed in parallel
- Hardware Accelerator
  - Parallel processing
  - Zycad vs. Ikos
  - Cycle-based vs. event-driven
  - Testbench and Amdahl's law
- Emulation
  - FPGA and setup time
  - Quickturn (now division of Cadence) Palladium
- Cost vs. performance
- Small designs use accelerator, big designs use emulation

Sanity Check
- Syntax check
  - Verilog lint utility
- Style enforcement
  - Warning message
  - Regular expression filtering
- Race condition detection
  - Switch simulator?
  - Portability
- Static property checking

Static Property Check
- Static vs. dynamic properties
  - Static property check is a sanity check
  - Dynamic property check is a formal verification
- Intention property check
  - Realintent

Productivity Tools
- Revision control
- Job queue
- Verilog/Perl or SystemC/Perl
- State machine generator
- Parameterized design code generator
- Spreadsheet or graphic code generator

Revision Control
- Free software
  - SCCS, RCS, CVS, Subversion
- Commercial revision control system
  - RRCS, perforce, Clearcase, Synchronicity, Cadence TDM, etc.
  - Efficiency vs. functionality

RRCS
- Recursive Revision Control System
- Repository, release, and working trees
- Binary file handling
- Release/snapshot mechanisms
- Operations with pre/post processing
- Links
- Multiple views and re-locatable database

Job queues
- Free tool : gnu queue
- Commercial tools
  - Load sharing facilities (LSF)
  - Gridware

Verilog/Perl or SystemC/Perl

- Mixture code with Verilog/SystemC and Perl
- A translator translates the mixture code into pure Verilog/SystemC
- Automatic port declarations and connections

Vperl : the Verilog/Perl mixture language
- Sun internal design productivity tool
- &module( ), &portList( ), &ports( ), &wires( ), &instance( ), &connect( ), &force( ), &comment( ), etc.

Vperl Example

Source code : buf.vp

```
// VPERL: PERL_BEG
// print "// vperl buf\n";
// &MODULE     ("buf");
// &INSTANCE   ("inv.v", "ia");
// &INSTANCE   ("inv.v", "ib");
// &CONNECT ("ib.in",   "out_l");
// &CONNECT ("ib.out_1","out");
// &FORCE       ("output"    "out_1");
// &ENDMODULE;
// VPERL: PERL_END
```

Generated code : buf.v

```
// VPERL: GENERATED_BEG
// vperl buf
module buf (
     in,
     out_l,
     out
     );
input        in;
output       out;
output       out_l;
inv ia (
     .in       (in),
     .out_l    (out_l)
     );
inv ib (
     .in       (out_l),
     .out_l    (out)
     );
endmodule
// VPERL: GENERATED_END
```

State Machine Generator
- State machine generator
  - Input: state transition table
  - Output: state machine in HDL
- Output code in either RTL or gate design

Input Example

```
# Soft drink machine -- Price is 15 cents
#
# Assumptions: Only one coin is deposited at a time
#   and slower than the clock for the machine
# Design name
.design soft_drink_machine
# Input port names
.inputnames clk reset nickel_in dime_in quarter_in
# Output port names
.outputnames nickel_out dime_out dispense
# Clock signal's name and type
.clock clk rising_edge
# Reset signal's name, type, and resulting state
.asynchronous_reset reset rising IDLE
# State table body
100  IDLE      FIVE 000
010  IDLE      TEN  000
001  IDLE      IDLE 011
100  FIVE      TEN  000
010  FIVE      IDLE 001
001  FIVE      IDLE 111
100  TEN       IDLE 001
010  TEN       IDLE 101
001  TEN       OWE_DIME  011
000  OWE_DIME  IDLE  010
# Wait in current state until money is deposited
000  IDLE      &     000
000  FIVE      &     000
000  TEN       &     000
# Preferred state encoding
.encoding
IDLE      2#00
FIVE      2#01
TEN       2#11
OWE_DIME  2#10
```

RTL-level Code Generated

```
`define OWE_DIME 2'b10
`define FIVE 2'b01
`define IDLE 2'b00
`define TEN 2'b11
module soft_drink_machine (
   clk, reset, nickel_in, dime_in, quarter_in,
   nickel_out, dime_out, dispense
);
input clk, reset, nickel_in, dime_in, quarter_in;
output nickel_out, dime_out, dispense;
```

```
reg nickel_out, dime_out, dispense;
reg [1:0] fsm_state;
always @(posedge clk or reset) begin
   if ( reset ) begin
      fsm_state = `IDLE;
   end else begin
      case ( fsm_state )
         `OWE_DIME:
    case ( {nickel_in, dime_in, quarter_in} )
        3'b000: begin
       nickel_out = 1'b0;    dime_out = 1'b1;
       dispense = 1'b0;    fsm_state = `IDLE;
              end
    endcase
        `FIVE:
           case ( {nickel_in, dime_in, quarter_in} )
        3'b100: begin
       nickel_out = 1'b0;    dime_out = 1'b0;
       dispense = 1'b0;    fsm_state = `TEN;
              end
              3'b010: begin
       nickel_out = 1'b0;    dime_out = 1'b0;
       dispense = 1'b1;    fsm_state = `IDLE;
              end
              3'b001: begin
       nickel_out = 1'b1;    dime_out = 1'b1;
       dispense = 1'b1;    fsm_state = `IDLE;
              end
              3'b000: begin
       nickel_out = 1'b0;    dime_out = 1'b0;    dispense = 1'b0;
              end
           endcase

          `IDLE:
           case ( {nickel_in, dime_in, quarter_in} )
              3'b100: begin
       nickel_out = 1'b0;    dime_out = 1'b0;
       dispense = 1'b0;    fsm_state = `FIVE;
              end
              3'b010: begin
       nickel_out = 1'b0;    dime_out = 1'b0;
       dispense = 1'b0;    fsm_state = `TEN;
              end
              3'b001: begin
       nickel_out = 1'b0;    dime_out = 1'b1;    dispense = 1'b1;
              end
              3'b000: begin
       nickel_out = 1'b0;    dime_out = 1'b0;    dispense = 1'b0;
              end
           endcase
          `TEN:
             case ( {nickel_in, dime_in, quarter_in} )
                3'b100: begin
       nickel_out = 1'b0;    dime_out = 1'b0;
       dispense = 1'b1;    fsm_state = `IDLE;
              end
              3'b010: begin
       nickel_out = 1'b1;    dime_out = 1'b0;
       dispense = 1'b1;    fsm_state = `IDLE;
              end
              3'b001: begin
       nickel_out = 1'b0;    dime_out = 1'b1;
       dispense = 1'b1;    fsm_state = `OWE_DIME;
              end
              3'b000: begin
       nickel_out = 1'b0;    dime_out = 1'b0;    dispense = 1'b0;
              end
          endcase
      endcase
     end
  end
  endmodule
```

Gate-level Code Generated

```
  module soft_drink (nickel_out, dime_out,  food_out, nickel_in, dime_in,
  quarter_in, clk);
  output      nickel_out;
  output      dime_out;
  output      food_out;
  input       nickel_in;
  input       dime_in;
  input       quarter_in;
  input       clk;
  // forward declare the state names
     wire idle;
     wire five;
     wire ten;
     wire owe_dime;
  // identify the degenerate state - the assumption is
  // that the inputs will either be all off or only
  // that the inputs will either be all off or only
  // one on at a time.
     wire no_money = ~(
        nickel_in
      | dime_in
      | quarter_in
      );
  // next state evaluation - written as simple AND/ORs.
```

```
// this would be a lot faster as AOI or as simple
// inverting gates
   wire five_a1 =
      idle & nickel_in
    | five & no_money
    ;
   wire ten_a1 =
      idle & dime_in
    | five & nickel_in
    | ten & no_money
    ;
   wire owe_dime_a1 =
      ten & quarter_in
    ;
   wire idle_a1 = ~(
      five_a1
    | ten_a1
    | owe_dime_a1
    );

// generate outputs
   wire nickel_out =
      five & quarter_in
    | ten & dime_in
    ;
   // idle, five, and ten out straight out of a local
   // flop, so do the OR ahead of time while we wait
   // for quarter_in to arrive.  this is *much* faster
   // than the straight AND/OR.
   wire dime_out =
      quarter_in & (
         idle
       | five
       | ten )
    | owe_dime
    );
   wire food_out =
      idle & quarter_in
    | five & (dime_in | quarter_in)
    | ten & ~no_money
    ;
// state registers
   dff five_reg (five, five_a1, clk);
   dff ten_reg (ten, ten_a1, clk);
   dff owe_dime_reg (owe_dime, owe_dime_a1, clk);
   dff idle_reg (idle, idle_a1, clk);
endmodule
```

Parameterized Design Code Generator

- Generate design code depending on input parameters
  - Source code is usually a mixture of Perl and Verilog/SystemC
  - Generated code is pure Verilog/SystemC

Spreadsheet or Graphic Code Generator

- Input is spreadsheet or graphic symbol
- Output is generated HDL code
- Vendor : Escalade

Coverage Tools

- PLI/VPI and embedded comment instructions
- Code/statement/line coverage
- Branch or condition sub-expression coverage
- Toggle coverage
- User expression coverage

Formal Verification

- Model checker
- Property checker
- Theorem proofing
- Equivalency checker
- Symbolic simulation

Equivalency Checker

- RTL and gate equivalency
  - State points mapping
- Design and ECOs equivalency
- Explosion when doing multiplier
  - Graphic chips have a lot of multipliers
  - Cycle-by-cycle state points comparison tool by using PLI to dump state points and by using simulation to verify equivalency

Symbolic Simulation

- Vector based simulation vs. symbolic simulation
- Good for memory verification
- Vendor : Innologic (now division of Synopsys)

Synthesis

- RTL to gate
  - Using compiler techniques
  - Technology/library mapping
  - Optimization
  - Vendor : Synopsys DC, Gate2Chip (now division of Cadence)
- Behavior to gate
  - Vendor : Synopsys BC

Design For Test (DFT)

- Use configuration file to automate flow
- Hierarchical donut scan stitch

- DFT constraint generator to parallelize runs
- Vector capturing and playback

Use Configuration File To Automate Flow
- Ordering
- Stitching
- Scan chaining
- ATPG
- Verification/simulation
- Assembly

Hierarchical Donut Scan Stitch
- When chip becomes huge, use hierarchical approach
- Scan stitch on each block
- Donut scan stitch

DFT Constraint Generator to Parallelize Runs
- ATPG is slow; when chip is huge, it runs forever and never terminates
- Split design into mutually exclusive design from scan muxes, by recursive tracing back and evaluating forward the cones
- Generate constraint file and feed into ATPG
- Parallel execute *N* serial runs

Vector Capturing and Playback
- Facilitate vector exchange between the HDL simulation environment and the various test and debug facilities
- Vector capturing by using PLI on simulation
- Playback to reproduce the simulation

Debug and Waveform Viewing Tools
- Snapshot and debugging tools
- Waveform viewing and evaluation tools
- Debugging environment

Snapshot and Debugging Tools
- Manual debugging flow
  - Run without dump for efficiency
  - Re-run with dump when error
- Snapshot and debugging tools
  - Run without dump but save snapshots
  - When error, restart from previous snapshot and run with dump on

Waveform Viewing and Evaluation Tools
- Waveform viewing/displaying tools
- Waveform evaluation tools

Debugging Environment
- Vendor : Debussy
- Trace
- Fan out
- Value tagging
- State machine recognizing/displaying

Back-End Design Tools
- Back-end design flow
- Back-end design tools

Back-end Design Flow
- Many iterations, time-consuming
- Flow
  - Floor planning
  - Place & route
  - RC extraction and delay calculation
  - Timing closure
  - Physical verification
- Timing-driven flow

Timing-Driven Flow
- Combine layout and synthesis
  - Synopsys Physical Compiler (PC)
- Timing-driven physical prototyping
  - Fast turn-around P&R and timing
  - Vendor : Silicon Perspective (now division of Cadence)

Back-End Design Tools
- Schematic capturing tool
- Power estimation tool
- Floor planning tool
- Place and route tool
- RC extraction tool
- Static timing tool
- Physical verification tool
- Signal integrity tool

Power Estimation Tool
- Static power estimation tool
- Dynamic power estimation tool

Floor Planning Tool
- Most manual work

Place and Route Tool
- Most algorithmic
- Minimize total area, total length of connection wire, and total number of crossing of connections
- NP-hard problem and heuristic
- Vendor : Apolo, Silicon Ensemble

RC Extraction Tool
- RC extraction
- Vendor: Synopsys StarRCXT, Cadence simplex

Static Timing Tool
- Set-up time and hold time
- Pre-timing (estimation) vs. timing-closure (realistic)
- The trend is that estimation is becoming further away from reality
- Vendor : Synopsys PrimeTime, Sequence Saphire

Physical Verification Tool
- Design rule check (DRC)
  - For fab manufacturing the chips
- Layout verification tool (LVS)
  - Verify the layout has the same functionality as the logic
- Vendor : Cadence DRC and LVS, Mentor Calibre, Synopsys Hercules

Signal integrity tool
- Signal Integrity
  - Fight with noise to get better signal quality
  - Signal integrity trade-offs
  - On chip, board, or system level
  - Bit error, timing shift, IR drop, electric migration
- Tools
  - Field extraction parasitics (Ansoft, simplex)
  - Circuit simulation (Hspice)

Analog Design Tools
- Craftsmanship
- Circuit tuning

Craftsmanship
- Analog design needs length/width adjustment of transistors in order to achieve desired performance
  - Switch level simulation
  - Measure, adjust, and cycle again
  - Don't know where is the optimal
  - Time consuming

Circuit Tuning
- Local optimization
  - Gradient
  - Simplex
- Global optimization
  - Simulation annealing
- Automatic circuit turning
- Analog optimal design
  - Vendor : Barcelona

Design Verification Tools
- Automatic regression tools
- Verification tasks
- Language interface and co-simulation
- Transaction-based verification
- Design verification language approach
- Coverage-driven verification

Automatic Regression Tools
- Regression tests
- Triggered by revision control tools
- Triggered by daemon
- Bug tracking system
  - Free tool : Gnat
  - Commercial tools
  - Integrated with revision control tools

Verification tasks
- Stimulus
- Design Under Test (DUT)
- Compare results against expected results
- Testbench
  - Stimulus generation
  - DUT instantiation
  - Result analysis
  - Automatic checking

Language Interface and Co-simulation
- Socket programming
- C/C++ and Verilog
  - PLI v1.0
  - PLI v2.0 (VPI)
- Direct kernel calls
- Performance comparison

Transaction-based verification
- Raises the level of abstraction from signals to transactions
  - Makes developing more effective and efficient test suites easier
- Bus functional model (BMF)
  - Concentrate on verification, coverage, and corner cases instead of detailed signal transitions and timing

Design Verification Language Approach
- Design a verification language for design verification
  - Enhanced high-level language to do verification tasks
  - Random test generator
- Vendor : Synopsys VERA

Coverage-driven verification

- For confidence on verification, use coverage-driven approach
  - Automatic random test generation
  - Guided test generation by constraint random generator
- Vendor : Verisity Specman and e language

Companies and Recession

- Famous companies and their products
- How to get VC funding to start a new company
- Recession at Silicon Valley and job lost
- Engineer's future?

How to Get VC Funding to Start a New Company

- Business plan
  - Idea, returns, teammates, technology, etc.
- Relationship between VCs and startups
- Startups
  - Fund raising, office setup, equipment purchasing, computer setup, execution, projection, planning, administrating, etc.

Recession at Silicon Valley and Job Lost

- Recession in USA
  - 1982
  - 1991
  - 2001 : dotcom bubble
- Job lost
  - Hardware migrates to Taiwan, China
  - Software and services move to India

Engineer's Future?

- Engineer, medical, lawyer, business, or science?
- Computer hardware/software engineer, electric engineer, or bioinformatics?
- Next bubble
  - Bioinformatics
  - Artificial intelligence, smart machine

Questions and Answers