Lecture 1
Overview of ASIC and FPGA Design

Alex Jones

ECE 2120

Hardware Design Methodologies

Fall 2007

1

Today's Topics

- Administrivia
  - Course Overview
- Introduction to ASICs
- Physical Design 101
- ASIC Design Flow
- ASIC Alternatives
- READING: Bhatnagar Chapter 1

2

Class Administration

- Lectures Once a Week,
  - W 5:20-7:50pm
  - BEH 370
- Instructors
  - Alex Jones, Gayatri Mehta
  - Office 334 BEH, 272 BEH
  - Email: akj8@pitt.edu, gayatrimehta1@gmail.com
- Web Page
  - TBA

3

Class Prerequisites

- CoE: 1502, EE/CoE 142 or permission
- Topics for review
  - Need a familiarity with VHDL
  - Need a solid background with digital logic structures (e.g. gates, registers, memory)
  - Understand Finite State Machines
  - Background in Logic Simplification
  - Understand area performance power tradeoffs of various implementation options
  - Familiarity with UNIX/CAD tools like Mentor
- A UNIX account (signup sheet)

4

Class Textbooks and References

- Required Textbooks
  - J. Bhasker, "A VHDL Synthesis Primer," Second Edition, Star Galaxy Press, 1998.
- Supplementary Textbooks
  - H. Bhatnagar, "Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler Physical Compiler and PrimeTime," Second Edition, Kluwer, 2002.
  - P. Ashenden, "The Designer's Guide to VHDL," Second Edition, Morgan Kaufmann, 2001.
- Classnotes
  - Online Slides
  - Tutorials
- Online Documentation to CAD Tools

5

Grades

- Tentative Grading Schedule
  - 35% Homeworks and Labs
  - 35% Exams
  - 30% Project
- Structure
  - First Half of Class
    - Homeworks/Mini Projects each week
    - Individual Work
  - Second Half
    - Single Larger Project
    - Teams of at least 2 depends
- Late Work will have minimum of 10% Penalty per day late.
  - Not guaranteed to accept late work at all

6

Summary

- I will walk away from this class knowing
  - How hardware synthesis tools work
  - How to write synthesizable VHDL
  - The design flow for ASICs
    - Synthesis: Synopsys Design Compiler
    - Simulation: Mentor Graphics Modelsim
    - Placement and Routing: Cadence SoC Encounter
  - The design flow for FPGAs
    - Synthesis: Synplicity Synplify Pro
    - Simulation: Modelsim
    - Placement and Routing: Xilinx ISE
  - How to optimize my design for area, performance, and power.

7

What this class is NOT

- A course in designed CAD tools and algorithms (ECE 3130)
- A course in physical design (ECE 1192/2192)
- A course in computer architecture and processors (ECE 2162)
- A course in the formal verification of hardware designs (ECE 2141)
- A course in real-time or embedded processing (ECE 2160)
- A course in hardware/software co-design (ECE 2140)

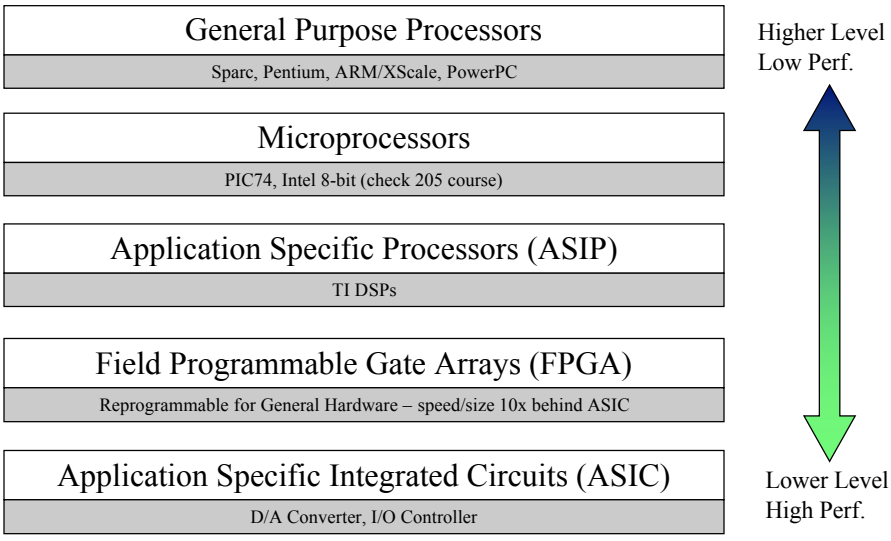
8

What is an ASIC?

- What is an Integrated Circuit (IC)?
  - ICs are basically "chips"
    - Silicon Wafers
    - Transistors, resistors, capacitors fabricated
  - Can be either Digital or Analog
  - Microprocessors, Amplifier, Memory
- ASICs are Application Specific ICs
  - Designed for a special application
  - ASICs may be customized or mass-produced
  - Digital to Audio Converter
  - Mpeg2 Decoder

9

Hierarchy of IC Design



10

ASIC Design Methodologies

- Gate Level Design (ECE 132/501)
  - Implement Logic at Gate Level
  - Schematic Editor (Mentor Graphics)
  - Hardware Description Language (Gates)
  - Logic Synthesis to Netlist
  - Implement in Silicon
- Full Custom Layout (ECE 1192/2192)
  - Design Logic at the Transistor Level
  - Hierarchical Design
  - Build Design from Bottom Up
  - Connect Components Manually

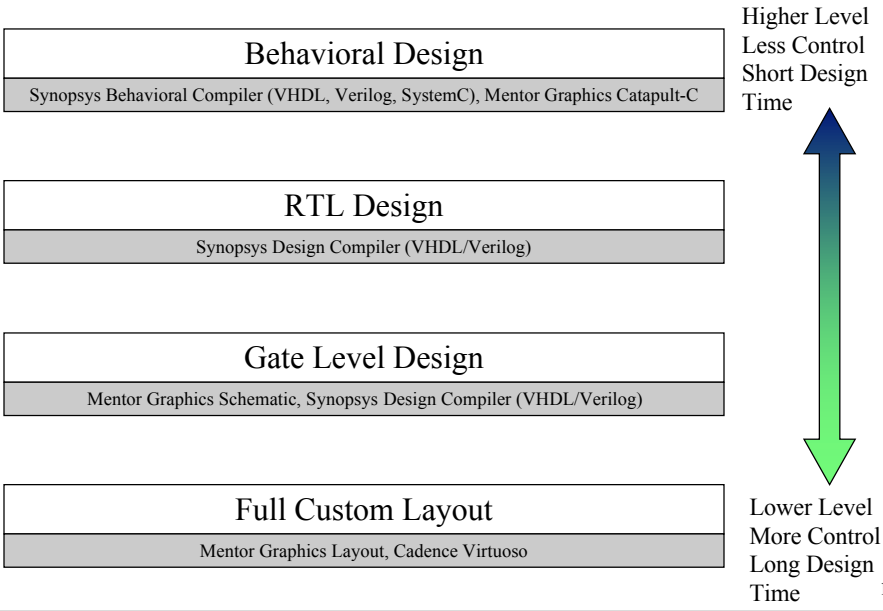
11

ASIC Design Methodologies

- Behavioral Design (Seminar course)
  - Describe Algorithm Behavior (C like code)
  - Use Behavioral Synthesis tools
  - Result is RTL-HDL or Gate-level Netlist
  - Implement in Silicon
- RTL Design (THIS COURSE!)
  - Describe Behavior
  - Synthesizable Subset of Hardware Description Language
  - Use RTL Synthesis tools for Gate-level Netlist
  - Implement in Silicon

12

Hierarchy of ASIC Design



13

ASIC Implementation Techniques

- I've got a Gate-Level Netlist, now what?
- Most Common Implementation Technique is to use Standard Cells
- Commercial CAD tools map the Netlist to the particular technology
- A die size is created based on the logic required from the netlist
- Cells are placed on the die using CAD Placement Algorithms (ECE 3130)
- Wiring between the cells also uses CAD routing algorithms (ECE 3130)

14

Standard Cells

- During Placement: Cells are placed in rows on the die
  - Horizontal routing channels between rows
  - Vertical routing around outside
  - Vertical routing can also use "filler" cells
  - Placement algorithms attempt to minimize the distance between cells connected in the netlist
- During Routing: Cells are connected using wires

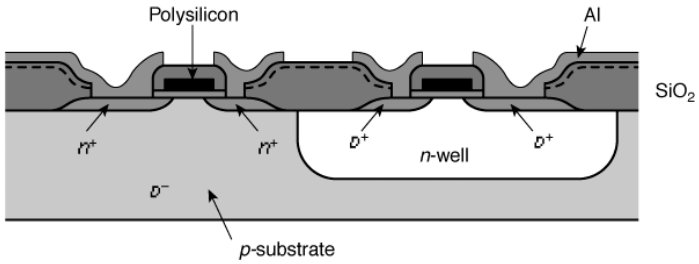
15

Physical Design 101

- Ok great, but what's a Standard Cell?
- Before we talk about that, let's discuss the fundamentals of Physical Design
- In ECE 132 we learned that gates are implemented using transistors

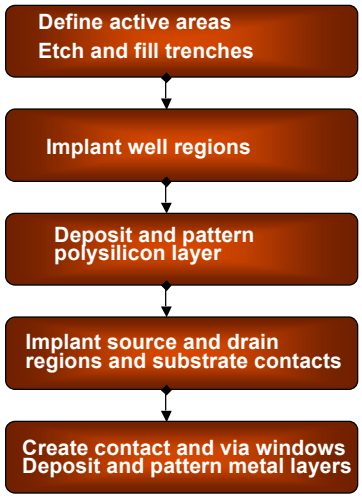
16

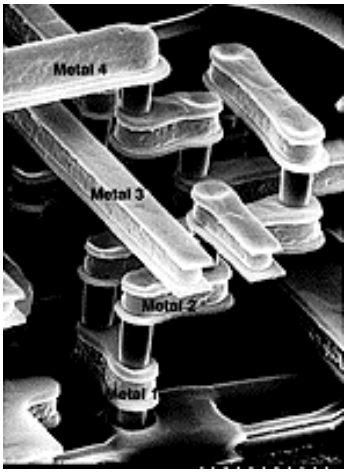
CMOS Process



17

CMOS Process at a Glance





18

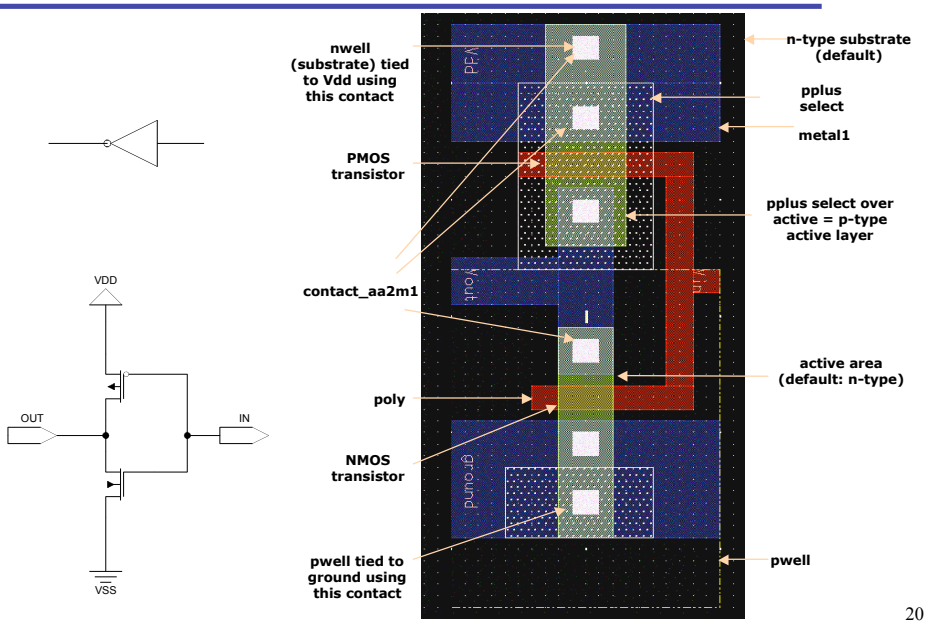
Design Rules

- Interface between designer and process engineer
- Guidelines for constructing process masks
- Unit dimension: Minimum line width
  - scalable design rules: lambda parameter
  - absolute dimensions (micron rules)

Layer	Color	Representation
Well (p,n)	Yellow	
Active Area (n+,p+)	Green	
Select (p+,n+)	Green	
Polysilicon	Red	
Metal1	Blue	
Metal2	Magenta	
Contact To Poly	Black	
Contact To Diffusion	Black	
Via	Black	

19

Example: Inverter



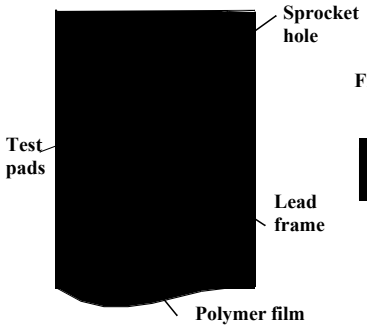
20

Packaging Requirements

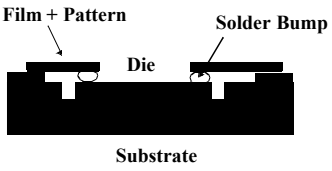
- Electrical: Low parasitics
- Mechanical: Reliable and robust
- Thermal: Efficient heat removal
- Economical: Cheap

21

Tape-Automated Bonding (TAB)



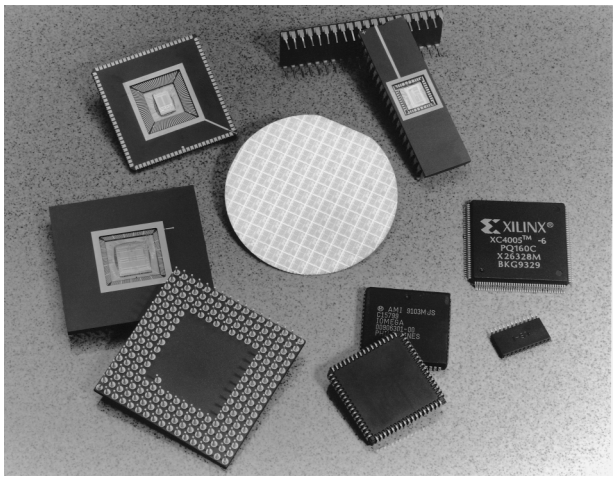
(a) Polymer Tape with imprinted wiring pattern.



(b) Die attachment using solder bumps.

22

Package Types



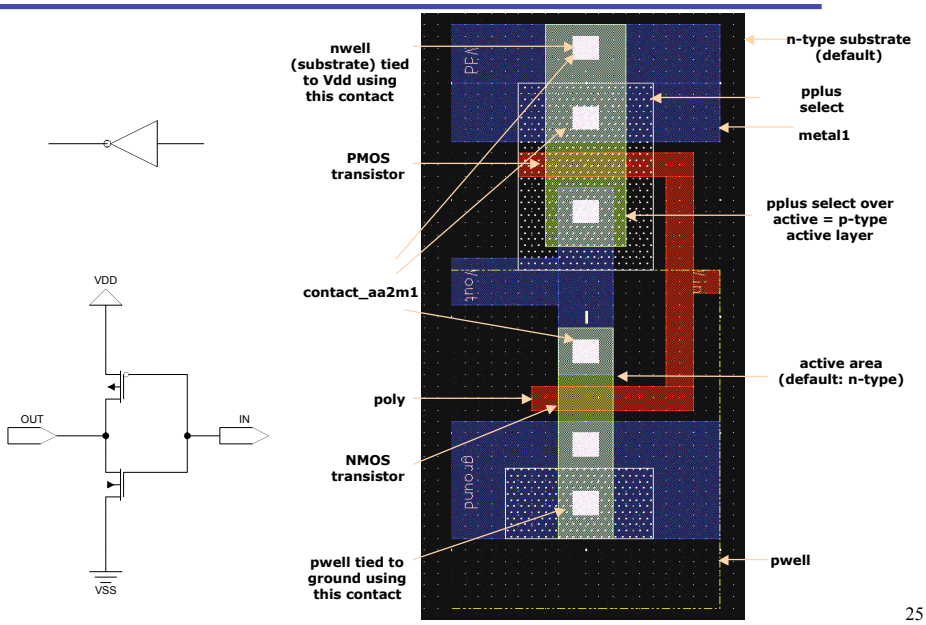
23

Standard Cells

- Ok great, but what's a Standard Cell?
- Standard Cells are custom physical layouts of basic logic building blocks
  - Ex: basic gates, RAM blocks, I/O devices
- There are restrictions on cell design
  - Fixed Height
    - based on technology feature size or λ design rule
  - Variable width (based on logic complexity)
  - Fixed locations for logic inputs, outputs, Vdd, Vss
- Standard Cells are kept in libraries tied to a particular technology process

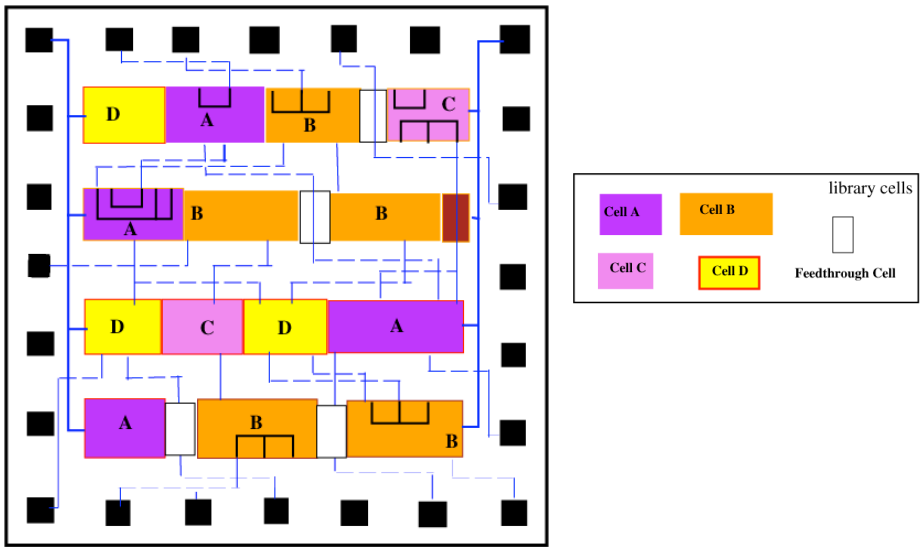
24

Standard Cell: Inverter



25

Standard Cell Example



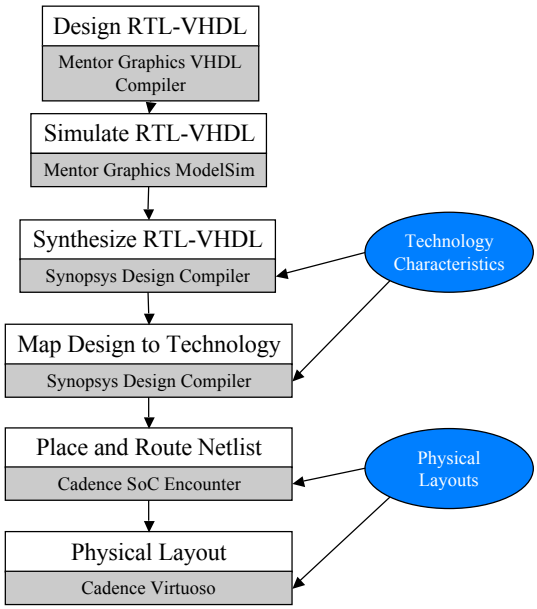
26

ASIC Implementation Techniques

- We now have a physical layout of the ASIC chip for fabrication
  - Omitted some specifics
  - I/O Pins
  - Clock Trees
  - Power Nets
- Design Verification
  - Timing, Area, Power requirements
- Tape Out, and Fabrication
- Testing

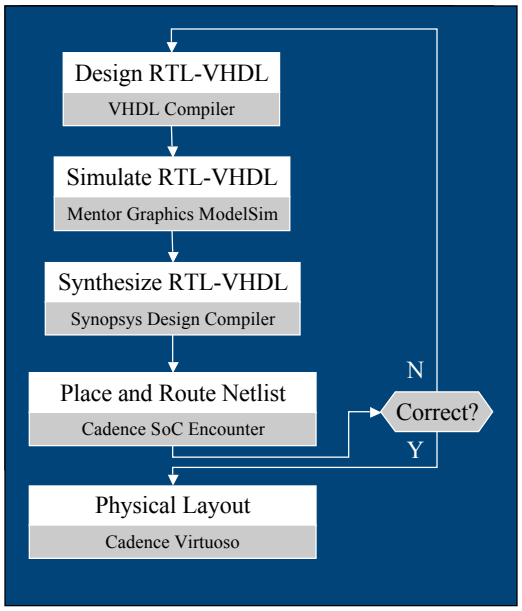
27

ASIC Design Flow Overview



28

Detailed ASIC Design Overview



- Large Part of ASIC Design is Verification and Testing

- Correctness
- Area
- Timing
- Power

29

Verification

- Originally HDLs designed for verification
  - Designs written in HDL and translated manually to schematics
  - Mentor Design Architect, Altera Quartus II
  - RTL Synthesis Tool replaces this step
- Test Benches still used for verification
  - Test bench written in behavioral HDL
  - RTL Design Simulated with test bench
- Determines correctness of design intent
  - Several other factors not considered
  - Timing, manufacturing process, power

30

Formal Verification

- Verification through simulation is problematic
  - Long simulation times
  - Likely does not test all possible permutations
- Formal Verification (ECE 2141)
  - Reduces time to do verification
  - Proves the structure of the two designs are logically equivalent
  - Can test original RTL against successive iterations of RTL
  - Test RTL against synthesized netlist
  - Test pre vs. post layout netlist

31

Timing Verification

- Course topic all by itself!
- One of the most important aspects of ASIC design
- Analyze critical paths to verify that they meet timing constraints
- Can be done pre or post layout
  - Pre-layout uses wire load models to estimate delays
  - Post-layout back annotates actual wire delays extracted from layout tools

32

Design for Test

- Design for test (DFT) adds logic to the design to verify the function of the chip.
- Built in self test (BIST) is synthesizable logic added to the chip
  - Some EDA tools generate this logic automatically
  - Generates logic vectors to test the memory or logic of the hardware
- Boundary Scan (JTAG)
  - For testing board connections
- Scan Insertion
  - Multiplexed FF's, turns into shift style register, applying logic vectors

33

Alternate Hardware Implementation

- Gate Array
  - Transistors are laid out on the die
    - Arrays of p and n transistors
  - User connects the transistors to create logic
  - Generally this is done by mapping logic gates down the the transistors
  - Remember NAND/NAND networks?
    - NAND gates map efficiently to gate arrays
    - Any combinational logic is possible with NAND
  - Sequential Logic?
    - Just add registers!

34

Alternate Hardware Implementation

- Field Programmable Gate Arrays (FPGA)
  - Hardware-like Speeds (10-100x slower)
  - Software-like Programmability (10-100x harder)
- Design Flow similar, but easier than ASIC flows
- Design logic at the Gate-Level or higher
  - ASIC can go down to polysilicon level
- Logic is traditionally mapped to LUTs
- Routing is mapped to pre-existing lines and switches

35

Field Programmable Gate Arrays

- Originally Designed as "Reprogrammable" Hardware solution
  - Can be retargeted in "the field"
  - Early devices, extremely small and slow
  - 1000 gates? < 20 MHz
- Now billed as "ASIC replacement"
  - >> 1 Million Gates, > 500 MHz
  - Pay a fixed price per part
  - ASIC Fabrication Start-up cost becoming prohibitive
  - FPGAs also used for testing ASICs before fabrication
- But more about this next time….

36

Via Programmable Gate Arrays

- LSI Logic and Carnegie Mellon
- Middle ground between FPGA and Gate Array (ASIC)
- Use Gate Array approach for logic implementation
- Fixed Route Structures
  - Horizontal Lines in Metal 1
  - Vertical Lines in Metal 2
- You place the vias to do interconnects
- Like a "program once" FPGA
  - Also called programmable ASIC

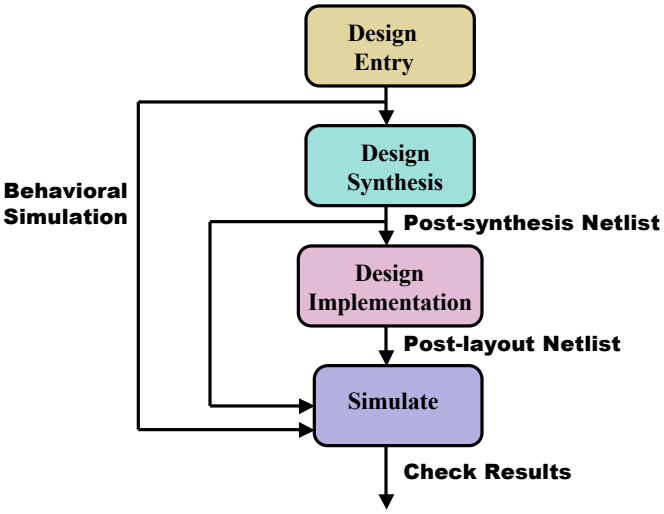
37

Design Flow and EDA Tools

Gayatri Mehta

38

Design Flow



39

Design Flow

- Design Entry
  - VHDL/Verilog
  - Schematic
- Synthesis
  - FPGA- Synplify Pro
  - ASIC- Design Compiler/Design Analyzer
- Place & Route
  - FPGA- Xilinix ISE
  - ASIC- SoC Encounter
- Simulate
  - ModelSim

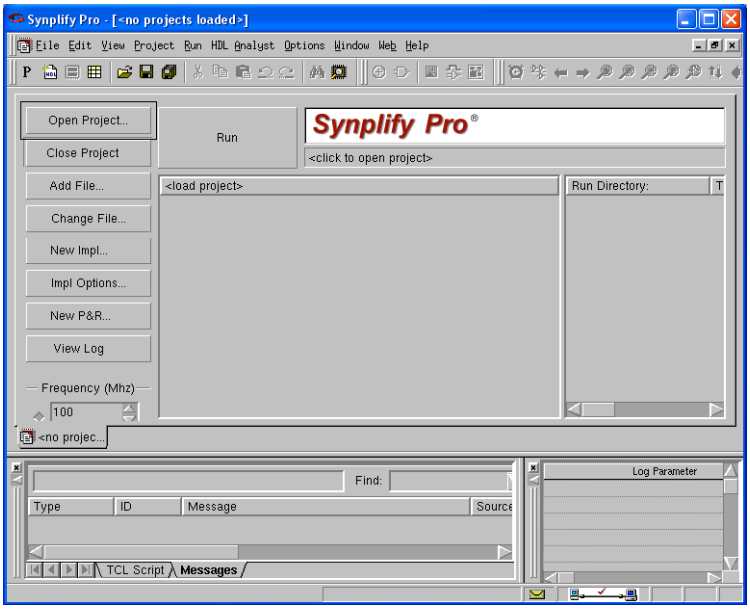
40

Synplify Pro (FPGA)

- Synthesizes the HDL code
- Provides optimized netlist for the target FPGA technology
- Does synthesis in two steps:
  - Compile-
    - syntax check
    - creates technology independent optimized netlist
  - Technology Map-
    - based on the timing constraints specified by the user, optimizes the netlist generated by the compiler to the technology

41

Synplify Pro



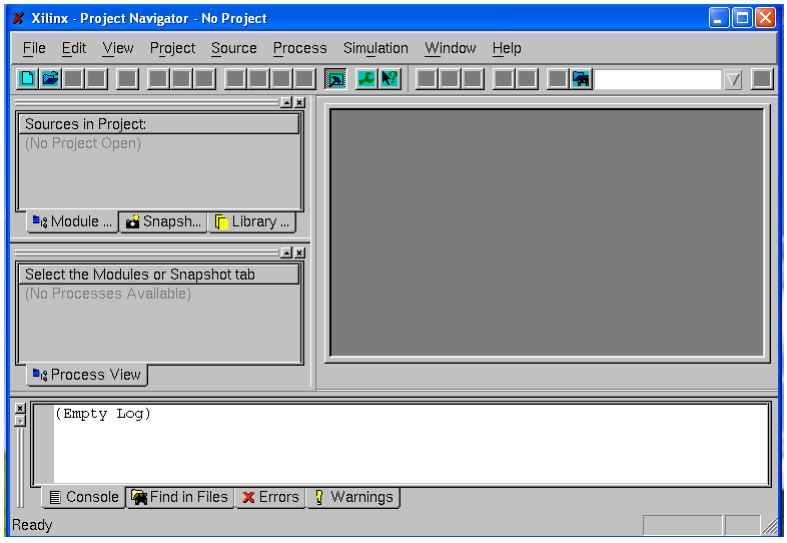
42

Xilinx ISE (FPGA)

- Design Implementation:
  - Translate- merges the incoming netlists and constraints into a Xilinx design file (.ngd file)
  - Map- maps the design into the available resources (CLBs and IOBs) on the target device (.ncd file)
  - Place and Route- places and routes the design to the timing constraints

43

Xilinx ISE



44

Design Compiler (ASIC)

- Takes an RTL hardware description and a standard cell library as input
- Produces optimized netlist as output
- Synthesis involves many steps:
  - High-level RTL optimizations
  - Technology independent optimizations
  - Technology mapping to the standard cells

45

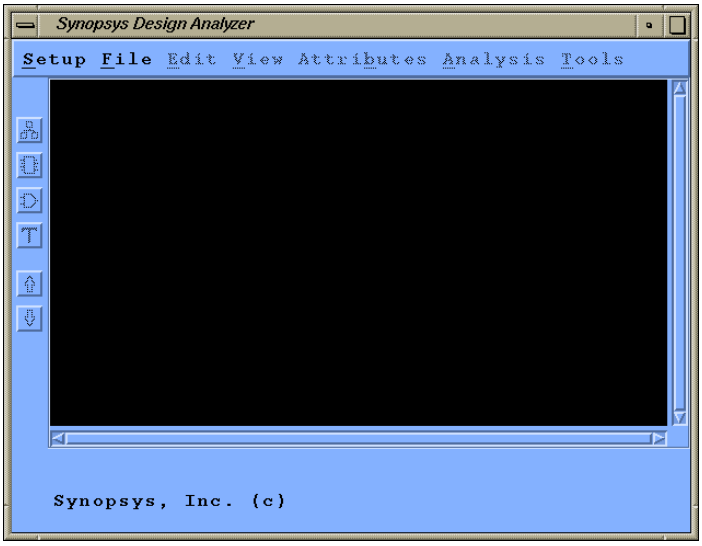
Design Compiler (ASIC)

File types

- Script files (.tcl)
  - Use Synopsys commands to execute synthesis
- RTL Verilog (.v)
- Synthesized Verilog (.sv)
- RTL VHDL (.vhd)
- Synthesized VHDL (.svhd)
- Synopsys Database (.db)
  - Binary format that represents RTL, gates, or internal libraries
- Reports (.rpt)
- Logs (.log)

46

Design Analyzer



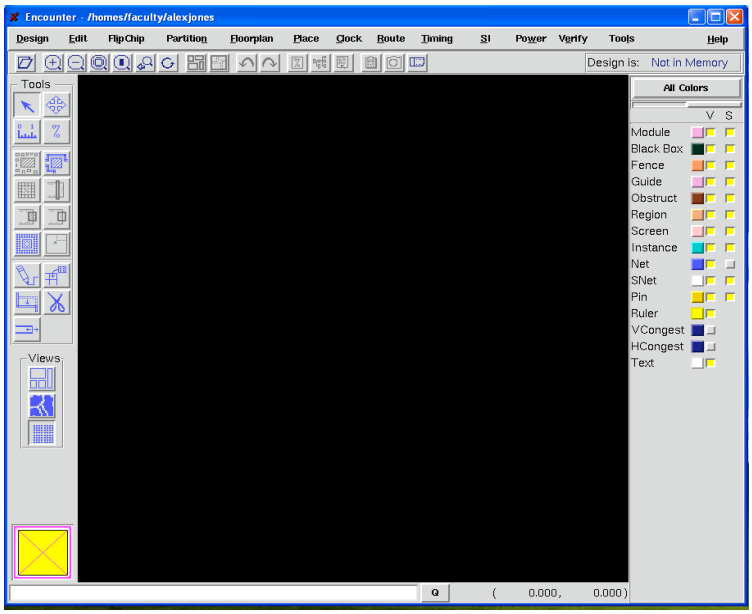
47

SoC Encounter (ASIC)

- SoC Encounter provides an integrated solution for an RTL-to-GDSII design flow
  - Advanced RTL synthesis
  - Silicon virtual prototyping
  - Mixed-signal support
  - Nanometer routing

48

SoC Encounter



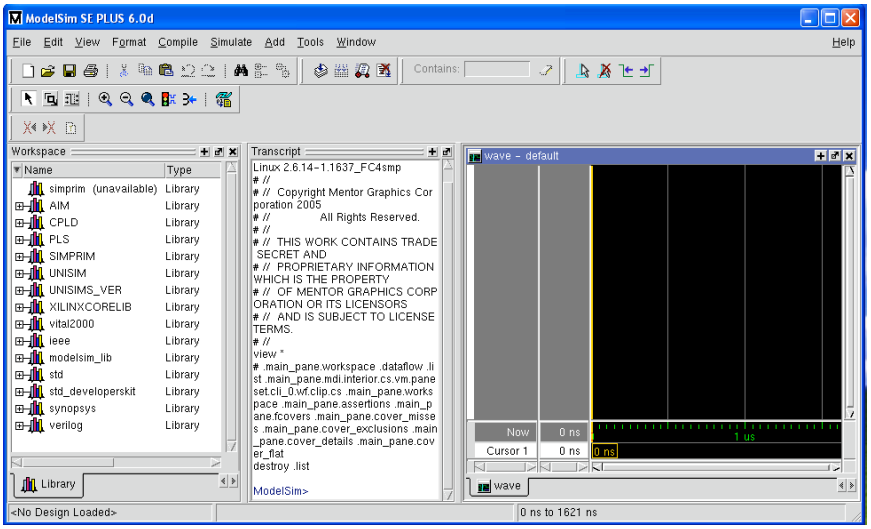
49

ModelSim

- Provides a comprehensive simulation and debug environment for ASIC and FPGA designs
- Can be used for functional and timing simulation of the designs

50

ModelSim



Summary

- Introduction to ASICs
- Physical Design 101
- ASIC Design Flow
- ASIC Alternatives
- Course Overview
- Design Flow and EDA Tools
- NEXT LECTURE: Review of VHDL
- READING: Bhasker Chapter 1