Design Flows & Design Methodologies

ECE 152A – Fall 2006

Reading Assignment

- Brown and Vranesic (cont)
  - 3 Implementation Technology
    - 3.6 Programmable Logic Devices
      - 3.6.1 Programmable Logic Array (PLA)
      - 3.6.2 Programmable Array Logic (PAL)
      - 3.6.3 Programming of PLAs and PALs
      - 3.6.4 Complex Programmable Logic Devices (CPLD)
      - 3.6.5 Field Programmable Gate Arrays (FPGA)
      - 3.6.6 Using CAD Tools to Implement Circuits in CPLDs and FPGAs
    - 3.7 Custom Chips, Standard Cells, and Gate Arrays

Reading Assignment

- Roth
  - 9 Multiplexers, Decoders, and Programmable Logic Devices
    - 9.6 Programmable Logic Devices
      - Programmable Logic Array
      - Programmable Array Logic
    - 9.7 Complex Programmable Logic Device
    - 9.8 Field Programmable Gate Array

Ancient Design Flow

- Design on paper
  - Paper schematic, timing diagrams, etc.
  - Specify standard chips (54XX/74XX)
- Build prototype
  - Wire wrap on vector board
    - Test and debug prototype
- Build production version
  - Printed circuit board (PCB)
    - Any modifications done by cutting traces and using discrete wires to jump (“cut and jumper”)

Computer Aided Design

- Driven by the rising complexity of digital systems and the falling cost of computers
  - Microprocessors and programmable devices
  - PCs and workstations
- Computer Aided Design also referred to as Electronic Design Automation (EDA)
- Design Automation Conference (DAC)
  - Began in early 1960's (1964)
  - web site at www.dac.com

ASICs

- Application Specific Integrated Circuit (ASIC)
  - Programmable chips (one way or another) that perform a specific function
    - As opposed to discrete / fixed logic TTL type devices
      - Standard Chips (B&V, 3.5)
  - Multiple, configurable gates on a single integrated circuit
  - May be field programmable (by user) or custom programmable (by manufacturer)

Custom ASICs

- Custom ASICs
  - Not reprogrammable
    - Mask programmable by manufacturer
  - Many more gates and much higher performance than field programmable devices
  - Two types of custom ASICs
    - Gate Arrays
    - Standard Cell

Custom ASICs (continued)

- Gate Array
  - Array of uncommitted transistors (base array)
    - Architecture may be “channeled” or “sea-of-gates”
  - Macrocells (gates, flip-flops) defined by metal interconnection
  - Chip constructed by “placing” macrocells on the base array and “routing” (interconnecting) them

Custom ASICs (continued)

- Standard Cell
  - More gates and higher performance than gate array
  - Allows simpler incorporation of megacells
    - RAM, ROM, Register Files, Analog Blocks (phase locked loops, A/D, D/A)
  - No base array
    - All macrocells and megacells are custom designs
    - As with gate array, physical design involves “place and route” of elements

CAD Flow and Tools

- Programmable Devices require computer supported design flow
  - Can program very simple devices with tables, more complicated devices require netlists
  - First generation CAD tools ran on mainframes or proprietary hardware
    - Next generation ran on high performance workstations
      - Current generation run on PC’s

CAD Tools

- The Monsters
  - Cadence Design Systems
  - Mentor Graphics
  - Synopsys
- Significant technical innovations often attributed to small startup companies
  - Startups then acquired by one of the monsters and technology incorporated into their tool suites

CAD Tools (continued)

- Schematic Capture Tools
  - Create netlist from graphical interface
    - Human readable input, tool usable output
  - Computer based schematic that other tools use
    - Simulators
    - Physical design tools
    - Verification tools

CAD Tools (continued)

- Simulators
  - SPICE
    - Simulation Program, Integrated Circuit Emphasis
    - Analog, transistor level simulator
  - Viewsim, Modelsim, Verilog XL, etc.
    - Digital simulators
      - Behavioral simulation
      - Functional simulation
      - Timing simulation
  - Mixed – Signal (analog/digital) simulators

CAD Tools (continued)

- Physical Design (Layout) Tools
  - "Place and Route" Tools
    - Circuit Board Layout
    - Integrated Circuit Layout
      - Standard Cell and Gate Array
    - FPGA Layout
  - Floorplanning Tools
    - High level pre-placement of functional elements

CAD Tools (continued)

- Design Verification Tools
  - Physical Design Rule Checkers (DRC)
  - Electrical Design Rule Checkers (ERC)
  - Layout v. Schematic (LVS)
  - Static Timing Analysis
  - Formal Verification
    - Gate implementation vs. RTL

Schematic Based ASIC Design Flow

- Front End (Logical) Design
  - Capture schematic using library elements for given technology
    - Elements may be primitive (generic) or structural (specific)
  - Generate netlist and simulate
    - Add structure to logical design
    - Functional simulation for logical correctness
    - Timing simulation with estimated gate and wire delays

Schematic Based ASIC Design Flow

- Back End (Physical) Design
  - Place and route design
    - Structural design mapped onto actual device
  - Extract netlist and parasitics from physical design
    - Create physical data base for “back annotation”
      - Simulation and verification
    - Gate delays and wire delays (RLC)
  - Re-simulate and verify
    - “Close timing”

Code Based ASIC Design

- Synthesis is key element in code based design (Front End)
  - Take hardware description language (HDL) code as input and “create” the complete design
  - Synthesis tool:
    - Generates logical model of design
    - Selects elements from library and defines interconnection
    - Maps design onto target architecture and physical implementation

Code Based ASIC Design

- Back End (Physical) Design
  - Essentially the same as schematic based, but design iterations performed via code rather than schematic
    - Place and route design
    - Extract netlist and parasitics from physical design
    - Create physical data base for “back annotation”
    - Re-simulate and verify
    - “Close timing”

Hardware Description Languages

- VHDL
  - VHSIC Hardware Description Language
    - VHSIC - Very High Speed Integrated Circuits
    - DOD/DARPA
- Verilog
  - Language and Simulator
    - Cadence Design Systems
      - Verilog XL (simulator)
    - Synopsys
      - Design Compiler (synthesis)

Hardware Description Languages

- Both languages support top - down and bottom - up design methodologies
  - Behavioral
  - Register Transfer
  - Gate
- Other proprietary and open hardware description languages have come and gone
  - VHDL and Verilog dominate

VHDL

- History
  - Developed by DOD/DARPA to describe very high speed integrated circuits
    - Begun in 1981
    - IBM, Texas Instruments, Intermetrics developed and released first version (Version 7.2) in 1985
  - IEEE Standard in 1987 (1076-1987)
    - Revised standard in 1993 (1076-1993)

VHDL

- DOD requires all digital ASICs and their subcomponents be delivered with both behavioral and structural descriptions in VHDL
- Levels of Abstraction
  - Behavioral (software abstraction)
  - Data Flow (register transfer)
  - Structural (gate level)

Verilog

- History
  - Proprietary hardware modeling language by Gateway Design Automation in 1984
  - Verilog simulator developed between 1985 and 1987
  - Cadence Design Systems acquired Gateway in 1990

Verilog

- Cadence continued to market Verilog as a language and a simulator
  - Language for modeling, not synthesis at this point
  - Verilog was a proprietary language
    - Other CAE vendors were not allowed to create and market a Verilog simulator
    - They needed a comparable product to sell, and VHDL was the only viable alternative.

Verilog

- Language opened up in 1991
  - Open Verilog Initiative (OVI)
  - IEEE standard in 1995
- Synopsys responsible for driving Verilog as a synthesis language
  - In 1988, Synopsys delivered the first logic synthesizer (Design Compiler)
    - Used Verilog as an input language

Verilog

- This was a major event, as now the top-down design methodology could actually be used effectively
  - The design could be done at the "register transfer level", and then Synopsys' Design Compiler could translate that into gates
  - With this event, the use of Verilog increased dramatically

Verilog Levels of Abstraction

- Behavioral (software abstraction)
  - Describes a system by concurrent algorithms
  - Each algorithm is sequential
    - No concept of "structural" realization
      - “What it does”
      - No implementation details
    - Modeling, simulation
      - Bus transactions, address, data, ALE, etc.
      - Allows simulation of peripheral circuits around microprocessor

Verilog Levels of Abstraction

- Register Transfer Level (RTL)
  - Specifies the characteristics of a circuit by register operations and the transfer of data between registers
  - Uses an explicit clock
    - Operations are scheduled to occur at certain times
  - Used for synthesis and simulation
  - Any code that is synthesizable is called RTL code

Verilog Levels of Abstraction

- Gate (structural)
  - System is described by logical links and timing properties
    - CAD tool or human generated netlist
      - Corresponds directly to schematic
    - Usable operations are predefined logic primitives
      - AND, OR, NOT, etc.
    - Rarely used in ASIC design
      - Used for post physical design (back end) gate level simulation

Verilog Levels of Abstraction

- Switch (transistor)
  - Transistors modeled as switches
  - (Almost) Never used in design
    - Used for back end simulation