ASL: Auburn Simulation Language
and
AUSIM: Auburn University SIMulator

Chuck Stroud
Professor
Electrical & Computer Engineering
Auburn University
Example of Digital System Design

The integrated circuit design process

- Note all the simulation
  - Ensure design works and assists in debugging design errors (design verification)
  - Detect defects in manufacturing process or faults in system (testing)

- To simulate a circuit, we must describe it in a manner that can be interpreted and understood by the simulator
Design Capture with HDLs

- All of these captured design descriptions can go to simulation:
  - **Netlist**
    - Connections of components made via signal name
      - ASL for example
  - **Schematic**
    - Connections explicit (via wires) or via signal name
    - Produces a netlist for simulation
  - **Higher level language (VHDL or Verilog)**
    - Synthesis to gate level netlist
Netlist Example

- **Circuit statement:**
  - Circuit name
  - Inputs
  - Outputs

- **Component statements:**
  - Component type
  - Component instantiation
  - Inputs signals
  - Output signals

```plaintext
# ASL description of MUX ;
ckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
and: G2 in: A SN out: G2 ;
and: G3 in: S B out: G3 ;
or: G4 in: G2 G3 out: Z ;
```
Design Verification

**Schematic (or logic diagram)**

```
ckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
and: G2 in: A SN out: G2 ;
and: G3 in: S B out: G3 ;
or: G4 in: G2 G3 out: Z ;
```

**Logic Simulation**

- Netlist
- Stimuli

Logic Simulator

Results
Simulation

Design Capture Input

*Schematic diagram*

\[
\begin{align*}
\text{ckt: } & \text{MUX in: } A B S \text{ out: } Z ; \\
\text{not: } & G1 \text{ in: } S \text{ out: } SN ; \\
\text{and: } & G2 \text{ in: } A SN \text{ out: } G2 ; \\
\text{and: } & G3 \text{ in: } S B \text{ out: } G3 ; \\
\text{or: } & G4 \text{ in: } G2 G3 \text{ out: } Z ;
\end{align*}
\]

Computer Emulation

Pointers to source logic values

Data structures hold logic values

logical operations performed by computer
Design Verification

Logic simulation results used to:

- Verify proper operation of design
- Find & fix problems (errors) in design

aka: debugging

<table>
<thead>
<tr>
<th>Input Stimuli (or vectors)</th>
<th>Simulation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td># ABS ;</td>
<td># ABS Z ;</td>
</tr>
<tr>
<td>000</td>
<td>000 0</td>
</tr>
<tr>
<td>001</td>
<td>001 0</td>
</tr>
<tr>
<td>010</td>
<td>010 0</td>
</tr>
<tr>
<td>011</td>
<td>011 1</td>
</tr>
<tr>
<td>100</td>
<td>100 1</td>
</tr>
<tr>
<td>101</td>
<td>101 0</td>
</tr>
<tr>
<td>110</td>
<td>110 1</td>
</tr>
<tr>
<td>111</td>
<td>111 1</td>
</tr>
</tbody>
</table>

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>S</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

compare simulation results to truth table
Types of Simulators

 Compiled Simulator (AUSIM)
 - Simulation continues until circuit is stable
   - No changing logic values within circuit
 - aka: unit delay or logic simulator
   - All gates in circuit have a finite unit delay
 - Good for initial design verification
   - Shorter simulation times

 Event-Driven Simulator
 - Simulation events scheduled in time
   - Circuit may not be stable when input changes
 - aka: timing simulator
   - Gates have real delays base on intrinsic & extrinsic factors
 - More accurate for real circuits
   - Longer simulation times and more computer intensive
Overview of ASL

- **ASL**: Auburn Simulation Language
  - A positional notation hardware description language for digital logic
    - Also referred to as a *netlist*
  - Used with AUSIM (Auburn University SIMulator) for
    - Logic simulation and design verification
Delimiters

- Delimiters include:
  - Space, tab, and new line
- Used to separate all:
  - Keywords
  - Some reserved characters
    - Specifically ‘#’ and ‘;’
  - Names
    - Gate names
    - Signal (net) names
- Can be used freely
  - Missing delimiter is a frequent source of syntax errors
Reserved Characters

- Can not (or should not) be used in names:
  - ‘#’ (pound sign)
    - Denotes beginning of comment statement
  - ‘;’ (semicolon)
    - Denotes end of statement
  - ‘:’ (colon)
    - Denotes keyword
Keywords

- Keywords include:
  - CKT: (ckt:)
    - Denotes circuit statements
  - IN: (in:) and OUT: (out:)
    - Denotes inputs and outputs of circuit or gate
  - AND: (and:), OR: (or:), NAND: (nand:), NOR: (nor:), & NOT: (not:)
    - Denotes elementary logic gate component statements

- Keywords always end with colon ‘:’
- Can be all uppercase or all lowercase letters
  - But not a mixture of upper and lower case
Statements

- Three types include:
  - Comment statements
    - Include textual information or remove portions of circuit without deleting
    - **Cannot** be nested inside circuit or component statements
  - Circuit statements
    - Define attributes of circuit
      - Name of circuit
      - Primary inputs and outputs of circuit
  - Component (gate) statements
    - Defines attributes of gate
      - Unique name of gate
      - Inputs and output connections of gate
Statements

- All statements can be multiple lines
  - Good for large comments
  - Good for large number of inputs or outputs
    - In circuit statements
    - In component statements
  - Good for readability and debugging

- Common mistakes:
  - Missing ‘;’ at end of statement
  - Missing delimiter before ‘;’ at end of statement
Statement Formats

- All statements end with a **delimiter** then ‘;’

- **Comment statements**
  - Begin with ‘#’ followed by space
  - **Example:**
    - # this is a valid comment ;

- **Circuit statement**
  - **Example:**
    - ckt: FullAdder in: A B Cin out: SUM Cout ;
  - **Only one circuit statement per ASL file**
Statement Formats

Component statements

- **GATE**: name IN: input(s) OUT: output(s) ;
  - where **GATE**: can be:
    - AND: (and:), OR: (or:), NAND (nand:), NOR: (nor:)
      - Any number of inputs
    - NOT: (not:)
      - Only one input for NOT:

- **Example:**
  - nand: G1 in: A B C D E F out: Z ;

- One component statement for every gate in circuit
Naming Conventions

- Need unique names for:
  - Circuit, Gates, Signals (nets)
- Any combination of characters & numbers
  - Except reserved characters & keywords
- Case sensitive
  - Example: X1 and x1 are two different names
- Signal name can be same as gate name
  - Common practice by designers to:
    - Reduce number of names
    - Aid in debugging
- Recommend use of short but meaningful names
ASL Example

logic diagram

ASL

# ASL description for MUX ;
ckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
and: G2 in: A SN out: G2 ;
and: G3 in: S B out: G3 ;
or: G4 in: G2 G3 out: Z ;
Input Stimulus for Simulation

- aka “input vectors” or just “vectors”
- Vectors are used for design verification
  - *The most important part of our work!!!*
- Each vector is a multi-bit binary value to be applied to inputs of circuit during simulation
  - Vectors are chosen to expose design errors
  - Vectors applied one at a time with output responses reported by simulator for each input vector
  - We study output responses to input vectors to find design errors
- Design verification is only as good as the set of vectors used for simulation
AUSIM Input Vectors

- One vector = a string of 1s and 0s
  - No delimiters in a vector
- Comments are optional and can be added between vectors
  - Comments are passed to simulation results file (.out) by AUSIM
- Bits in vector are applied in ordered of inputs in circuit (CKT:) statement

Example vector
File for MUX

```plaintext
# Z=0 ;
000
001
010
101
# Z=1 ;
011
100
110
111
```
Checks for Common Problems

- Typically part of CAD tool
  - Schematic capture (when generating netlist)
  - Simulator (AUSIM)

- AUSIM checks for potential design errors
  - Common syntax errors
    - Cannot always identify exact error
      - Actual error may be just before point of complaint
  - Unconnected gate inputs and outputs
  - Multiple gates driving same net
Other AUSIM Checks

- Duplicate names for gates, inputs, outputs
- Reserved characters (#, _, ;) in names
- Multiple or missing CKT: statement in ASL
- Missing IN: keyword after circuit/gate name
- Some rare errors
  - Incorrect number of inputs to inverter
  - Net names too long
    - Maximum length currently 35 characters
AUSIM Files

- All all input and output files are ASCII text
- Any text editor can be used to create files
  - Be sure to save as text file
    - When using default file names be sure to delete .txt suffix from some editors (like NotePad) before executing AUSIM
      - Example: change name.asl.txt to name.asl
    - Suggest turn off “hide extensions for file types”
      - Windows Explorer -> Tools -> Folder Options -> View
        - Uncheck “hide extensions for file types”

- AUSIM checks for missing input files
AUSIM Files (default names)

- **Input files (files you generate)**
  - ASL file (*name.asl*)
  - Input vector file (*name.vec*)

- **Output files (files AUSIM generates)**
  - Errors and Warnings (*ausim_errs.txt*)
    - Checks for syntax and potential design errors
    - Generated when processing ASL
  - Circuit audit (*name.aud*)
    - Circuit statistics and analysis (can help with debug)
    - Generated when processing ASL
  - Simulation results file (*name.out*)
    - Generated during logic simulation
Accessing AUSIM L2.2

- Go to my ELEC 2200 web page
  - www.eng.auburn.edu/~strouce/elec2200.html
- Click download AUSIM L2.2 executable for PC
  - Download is ausiml22.zip file that contains:
    - ASL and AUSIM version L2.2 manual
    - Windows PC executable (ausim.exe)
    - Multiplexer example files discussed in manual
- Unzip, extract, and save files on your computer or desktop
Starting AUSIM L2.2

- In My Computer or Windows Explorer, go to directory where you saved ausim.exe
  - ASL & vector files must be in this directory
    - MUX example files should already be there
- Double click ausim.exe
- Type file prefix `name`
  - Using default names
- Click Process
  - Fix any errors
- Click Simul8
  - Results in `name.out`
    - Use text editor to view
# multiplexer ;
ckt: mux in: a b s out: z ;
not: sn in: s out: sn ;
and: a1 in: a sn out: a1 ;
and: a2 in: b s out: a2 ;
or: o1 in: a1 a2 out: z ;

# vectors give Z=1 ;
000
001
010
101
# vectors give Z=1 ;
011
100
110
111
# end of vectors ;

# AUSIM Simulation Results ;
# abs z ;
000 0
001 0
010 0
101 0
# vectors give Z=1 ;
011 1
100 1
110 1
111 1
# end of vectors ;
AUSIM Audit File

Area analysis
- Number of gates, \( G \)
- Number of gate I/O, \( G_{IO} \)

Gate usage

Loading and delays
- Nets with no driver help find unconnected gate inputs
  - Also nets with no loads help
- Intrinsic + extrinsic propagation delay

Timing path analysis
- For all paths
  - Number gate delays, \( G_{del} \)
  - Propagation delay, \( P_{del} \)
- Worst case path