CSE241A
VLSI Digital Circuits
Winter 2003

Recitation 6: Formal Verification
Formal Verification - Definition

- Formal verification:
  - Logic equivalence checking
  - It verifies the logical equivalence of RTL, gate or transistor level netlists to each other.
  - Does not guarantee that the initial design met the design specification
    - Ignores timing information
    - Only boolean equivalence
Formal Verification Advantages

- Does not test functional correctness
  - No input vector creation necessary
  - No test vectors for logical function
- Faster verification cycle for the design!
- Only generates functional vectors for simulation to identify the bugs
Technology Libraries

- Verifies logic function independent of technology and timing
- There is no timing check
  - Timing issues must be checked by static timing analysis or dynamic timing simulation.
- Can compare different technology and different hierarchical structure.

Clock Period = 2.5 ns  

Clock Period = 5 ns
Formal Verification Application

- Verifies equivalence of design at different stages of the ASIC flow
- Makes sure that no logical changes are made
Formal Verification - Flow

- Equivalence checking is a branch of static verification
  - Employs formal, mathematical techniques
  - Proves two versions of a design are, or are not, functionally equivalent
  - EC flow consists of four primary stages:
    - Read
    - Match
    - Verification
    - Debug

- Match and verification stages are those most impacted by design transformations
Read:

- 1. During the read stage, both versions of the design are read into the EC tool.
- 2. Segmented into manageable sections called logic cones.
  - Logic cones (Figure 1) are groups of logic bordered by registers, ports, or black boxes. The output border of a logic cone is referred to as the compare point.

![Diagram of Logic Cone](Diagram of Logic Cone)
Match:

- During the match phase, tools attempt to match, or map, compare points
- Reference (golden) design is used for comparison

Types of comparisons:

- Non-function (name-based)
- Function-based matching methods
- Best performance is by the more efficient name-based methods
Figure 2: Cones from the reference design are matched to corresponding cones within the implementation design during the matching phase of the EC flow.
Verification Comparison Types

- RTL to RTL Verification
  - Gated clocks addition
  - Critical path optimization
  - Other RTL to gate constraints

- RTL to Gate Verification
  - Post synthesis of an HDL design

- Gate to Gate Verification
  - Buffer insertion for re-timing
  - P&R buffers
  - Test logic
  - Clock trees
  - Scan chains
Formal Proof Issues

- Formal tools are great at proving that one logic cone is equivalent to another
  - Problem: matching up the cones in one input file with the cones in the other input file.
  - If you can't figure out which cones are supposed to be equivalent, then the tool can't help you

- Normally done by matching up the signal names
  - Done automatically
  - Some tools alter signal names.
  - Change the borders of cones
    - No direct equivalence between cones in one file and the other
    - Causes formal tool to do tedious process of matching up cones based on the topology of the network
    - Very time-consuming
Types of Input

- Formality, Formal Pro and others use
  - .db,
  - Verilog, VHDL
  - Spice
  - Edif

- Functional descriptions of library cells:
  - Used for mapping the designs into generic models

- RTL and the gate level design netlists:
  - Are converted into generic models at the mapping stage
Mapping Methods

- Name based mapping for design with mostly same names.
- No name mapping for designs with mostly different names.
- Extra points that exist in one design but not in other are not mapped.
Generic Tool Function - Diagnosis

- Locate the error pattern
  - What is the cause of the mismatch?
  - List the possible sources of the mismatch
    - All paths and probable mismatch points
  - Report gates to trace the fan-in or fan-out of the mismatch points
    - Check for gated designs