

# COMP 617 Fall08 Proposal

## Structural and Behavioral Verilog Synthesizability

Cherif Salama

### 1 Goal

My main goal for this fall is to be able to check for the synthesizability of any hardware description that might contain structural, behavioral, and/or generate-like constructs. This will include array bounds and wire size checks as well as conflicting assignment checks.

### 2 Talks

#### Talk 1 (Sep 29, 2008)

**Title:** Array Bounds Checking for Verilog

**Abstract:**

Verilog's padding semantics allow designers to write descriptions where wires of different widths can be interconnected. However, many of the descriptions using such constructs are bogus and will synthesize into incorrect circuits. A similar situation occurs when wires are incorrectly indexed by values (or ranges) that exceed their bounds. In this presentation we show how these kinds of errors can be pinned down using static analysis. The key idea is in mapping the problem to a constraint satisfiability problem and formally specifying how constraints should be extracted from a Verilog description.

**Papers:**

TBA: papers about arrays bounds checking

TBA: papers about integer constraints satisfiability

#### Talk 2 (Oct 28, 2008)

**Title:** Behavioral Constructs Synthesizability

**Abstract:**

Converting a hardware description written in Verilog into a graph of primitive logic components can be an extremely complex task. Usage of behavioral con-

structs in undisciplined ways will often lead to synthesize failure. We aim to extend our Verilog type system to support behavioral constructs. In this stage we also aim at supporting the full Verilog 2005 syntax in our implementation.

**Papers:**

[1] IEEE Standards Board. IEEE Standard for Verilog Register Transfer Level Synthesis. Number 1364.1-2002 (IEC 62142:2005) in IEEE Standards. IEEE, 2005

**Talk 3 (Nov 26, 2008)**

**Title: Short Circuit Checking for Verilog**

**Abstract:**

Synthesizable Verilog descriptions are not necessarily free from undesirable artifacts like short-circuits. In other instances, some outputs might remain unassigned and/or some inputs might be unused. A type system that guarantees that each wire is assigned exactly once can be used to statically check for this kind of inconsistencies.

**Papers:**

TBA: Papers about linear type systems