Development of a Symbolic Model Checker

A thesis presented for the degree of
Master Of Science

Contents

Abstract

Abstract goes here Abstract goes here

List of Acronyms

BDD Binary Decision Diagram

ROBDD Reduced Ordered Binary Decision Diagram

SAT Satisfiability

CNF Conjunctive Normal Form

DNF Disjunctive Normal Form

POS Product of Sums

SOP Sum Of Products

List of Algorithms

List of Figures

Chapter 1

Introduction

The later half of the 20th century had seen a sea change in field of Integrated Circuits. The complexity of design had increased drastically from few hundred to billions of gates as of now. This made the conventional method of testing and simulation to be incomplete as one can never test the complete functionality of a system. This led to the idea of using Formal Verification in the field of Integrated Circuits."Formal verification is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics"[**?**]. Formal verification provides exhaustive exploration of all possible behaviours rather than the traditional approach of simulation/testing which explore only few possible behaviours of a system. There are several methods of formal verification available out of which we will be looking into a method of model checking. The main advantages of the method is,

- The ability to perform verification in a completely automatic manner without any intervention from the user.
- The result of model checking is always either True or False
- If the property is failed to satisfy, it always produces a counterexample which provides an insight about the failure of the system

Chapter 2

Model Checking

Model checking is the process of verifying the given model with the given set of specifications. The process of model checking involves

- Modelling
- Specification
- Verification

2.1 Model

The first step in model checking is to generate a model for the system. Model should capture all the properties of the system. The most important feature of a system is its state. State is the snapshot of the values of the variables of a system at a particular instant of time. State upon which an action is performed results in a new state. Pair of such states determines the transition of the system. To capture these behaviours of a system, we use kripke structure, a state transition graph.

2.1.1 Kripke Structure

We use Kripke structure for modelling our system. Kripke structure is a tuple M = (S,R,I,AP,L).

- S is set of states

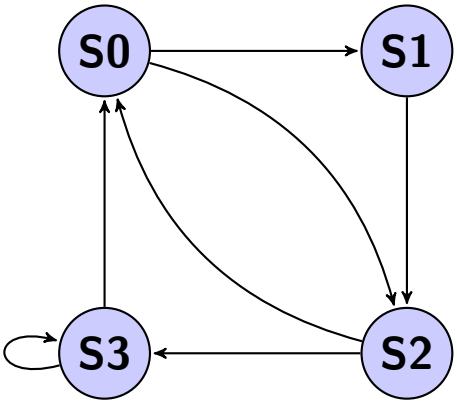

Figure 2.1: Kripke structure

- $R \subseteq S \times S$ is the transition relation
- $I \subseteq S$ is the set of possible initial states
- AP is a set of atomic propositions
- $L : S \rightarrow 2^{AP}$ is a labeling function: each state is labeled with the atomic propositions that are true in that state

S is also called as the state space of the system.

The Figure 2.1 is a kripke structure. S0,S1,S2 and S3 are the states of the system. {S0,S1},{S0,S2},{S3,S3}.. are the transitions of the system.

Till early 1980's, transition relation of a system was represented using adjacency list[**?**]. Due to the increase in the number of states, state transition graph became too large to handle using adjacency list. [**?**] suggested a novel approach to solve the state exploration problem by using a symbolic representation of the state transition graph. It was based on ROBDD. For a given order, ROBDD of a boolean expression is always canonical[**?**].

2.1.2 Binary Decision Diagram

Boolean expressions can be represented using a rooted, direct, acyclic graph called the BDD. BDD was first introduced by Lee[**?**, **?**]. An efficient data structure for representing the BDD and efficient algorithms for manipulating

BDD's were later developed by [**?**]. He also introduced the concept of ordering of variables in a BDD, which would affect the size significantly.

Algorithm 1 Symbolic computation of E U

procedure SYMBOLIC COMPUTATION OF$(E(\phi 1\, U\, \phi 2))$
  $f_0(V) = \chi_{\phi 2}(V)$
  $j = 0$
  repeat
    $f_{j+1}(V) = f_j(V) \vee (\chi_{\phi 1} \wedge \exists_{V'}[N(V, V') \wedge f_j(V')])$;
    j = j+1;
  until $f_j(V) = f_{j-1}(V)$;
  return $f_j(V)$
end procedure

Algorithm 2 Symbolic computation of E G

procedure SYMBOLIC COMPUTATION OF$(EG(\phi 1))$
  $f_0(V) = \chi_{\phi 1}(V)$
  $j = 0$
  repeat
    $f_{j+1}(V) = f_j(V) \wedge (\chi_{\phi 1} \wedge \exists_{V'}[N(V, V') \wedge f_j(V')])$;
    j = j+1;
  until $f_j(V) = f_{j-1}(V)$;
  return $f_j(V)$
end procedure

Algorithm 3 Relational Product algorithm

procedure RELPROD(f,g:BDD,$E : set\_of\_variables$): BDD
  if $f = false \vee g = false$ **then**
    $return\ false$
  else if $f = true \wedge g = true$ **then**
    $return\ true$
  else if $(f, g, E, h)\ in\ result\ cache$ **then**
    $return\ h$
  else
    $let\ x\ be\ the\ top\ variable\ of$ f
    $let\ y\ be\ the\ top\ variable\ of$ g
    $let\ z\ be\ the\ topmost\ of\ f$ and g
    $h_0 = RelProd(f\ |_{z\to 0}, g\ |_{z\to 0}, E)$
    $h_1 = RelProd(f\ |_{z\to 1}, g\ |_{z\to 1}, E)$
    if $z \in E$ **then**
      $h = OR(h_0, h_1)\ //BDD for h_0 \vee h_1$
    else
      $h = IfThenElse(z, h_0, h_1)$
      $/ * BDD for (z \wedge h_1) \vee (\neg z \wedge h_0) * /$
    end if
    $insert(f, g, E, h) in$ result cache
    return h
  end if
end procedure

Chapter 3

Partitioned Transition Relation

kdbhajfb

Chapter 4

Cone Of Influence Reduction

efewf

Chapter 5

SAT Solver

rfnefl