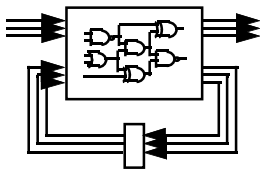
Computational and Efficiency Issues in VIS

VIS Technology Transfer Course
University of California, Berkeley

Rajeev K. Ranjan

Underlying Computation in VIS

- **A circuit is modeled as a finite state machine (FSM)**

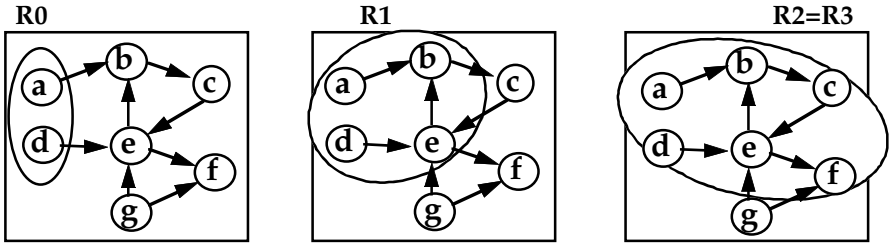


- **Verifying a property requires state space traversal of the FSM.**

State Space Traversal

One-step reachability: Given a set of states, find set of states which can be reached in one step

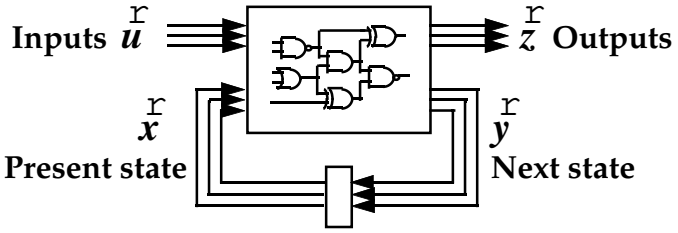
Reachability: Iteratively apply one-step reachability until no new states visited



Symbolic Computation in VIS

- **Given a circuit with n latches, a state of the circuit is defined by the value of these latches**
- **Using $V_0 V_1 V_2 .... V_{n-1}$ boolean variables, a state is represented by a minterm on these boolean variables, i.e., $V_0 V_1 V_2 .... V_{n-1}$ = 010 .... 1**
- **A set of states is a collection of minterms and is represented by a function: f(v) = 1 <=> v is in set**
- **Transition between states is also represented by boolean functions : T(p,i,n) = 1 <=> In state p, on input i there is a transition to state n.**
- **Boolean functions represented as Binary Decision Diagram (BDD)**
- **Size of the BDD data structure critical**

Finite State Machine



7-tuple : $\vec{x}, \vec{u}, \vec{y}, \vec{z}, \vec{f}, L, I$

$f_i\ (\vec{x},\vec{u})$
next state functions
(state transition graph)

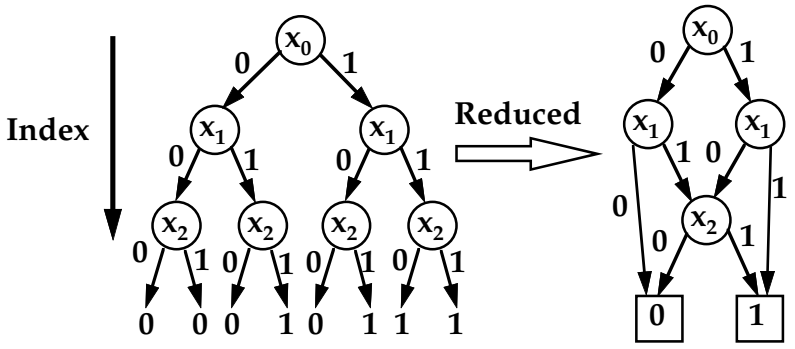
$L(\vec{x},\vec{u})$
output functions

$I(\vec{x})$
initial states

What are BDDs?

$f = x_0(x_1 + x_2) + x_1x_2$

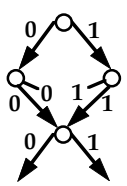
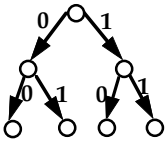
Akers'78, Bryant'86



- Canonical => Constant time equality checking
- Polynomial time Boolean operations e.g. h = f | g

Ordering of Variables in BDD

An n-bit comparator: $f = a \geq b;\ \ a = a_0 a_1 ... a_{n-1},\ \ b = b_0 b_1 ... b_{n-1}$

Ordering		BDD	BDD Size
Interleaving ordering from MSB to LSB	a_0 b_0 a_1		O(n)
Non-interleaving ordering	a_0 a_1 a_2		$O(2^n)$

Controlling the BDD Size : Space vs Time Trade-off

- **Verification of a design becomes difficult (large memory consumption and computation times) when the BDD sizes get too large**
- **In general, BDD size can be controlled at the expense of computational inefficiency**
- **Values of various parameters could be chosen suitably depending upon the working memory available**

Space vs Time Trade-off : Dynamic Variable Reordering

- **Transparent to the user**
- **Essential when:**
  - **Static ordering heuristics is not able to find the best ordering**
  - **No single good ordering exists for computing reachable states**
- **Helps the computation by :**
  - **Providing an ordering which results in less BDD nodes**
- **In general results in more computation time if kept enabled continuously**
  - **Need to invoke it judiciously**
  - **"enable" mode vs "force" mode**

Space vs Time Trade-off : Partitioning the Circuit

- **In some cases sizes of next state functions get too large**
- **BDD size can be controlled by partitioning the circuit and introducing intermediate variables to represent the partition.**
- **Number of BDD nodes decrease with increase in partition**
- **Reachability analysis time increases with increase in number of partitions**
- **Usage: vis> build_partition_mdds frontier**
- **Number of partitions is controlled by the parameter "partition_threshold" (size of the BDD at which new partition is created)**
- **Creates the BDDs for next state functions and intermediate variables : $T_1, T_2, \ldots, T_n, I_1, I_2 \ldots, I_k$**

Space vs Time Trade-off : Partitioned Transition Relation

- **Behavior of the system described by a vector of next state functions (# of latches), and functions for intermediate variables : $T_1, T_2, \ldots, T_n, I_1, I_2 \ldots, I_k$**
- **Some next state functions can be grouped together into components : (T_1, T_2), (T_3, T_4, T_5) ... (T_n, I_1, I_2) .....**
- **Number of BDD nodes roughly increases with decrease in number of components**
- **Time taken to perform reachability analysis is a convex function of number of components**
- **Number of components can be controlled by the parameter "image_cluster_size". Usage:**

  vis> set image_cluster_size 1000
- **"print_img_info" command creates the clusters**

Some Useful Tips - I

- **Use dynamic variable ordering after "init_verify" to reduce the BDD size.**
- **Save the ordering obtained from dynamic variable ordering to be used in future iterations of the design.**

```
vis> read_blif_mv  tlc.mv
vis> flatten_hierarchy;  static_order;  build_partition_mdds; print_img_info
vis> dynamic_var_ordering   -f  sift
vis> write_order  tlc.order
vis> mc tlc.ctl
```

Next iteration:

```
vis> read_blif_mv  tlc.mv
vis> flatten_hierarchy;  static_order tlc.order;  build_partition_mdds; mc tlc.ctl
```

Some Useful Tips - II

- **Do not keep dynamic variable ordering enabled.**
- **Typically invoke dynamic ordering after**
  - **Doing "print_img_info"**
  - **Use "compute_reach -s <step_size> -r <threshold_value>" to invoke dynamic ordering, when the BDD size of the reached state set crosses the threshold value. The size of the BDD is checked every "step_size" steps of reachability.**

Demonstration : Dynamic Variable Reordering

- **Read in a design**
  - **init_verify; print_img_info; compute_reach;**
  - **Note down the MDD size and analysis time**
  - **dynamic_var_ordering -f sift**
  - **write_order saved_order**
- **Read in the design again**
  - **flatten; static_order saved_order; print_img_info; compute_reach;**
  - **Note down the difference in MDD size and analysis time**