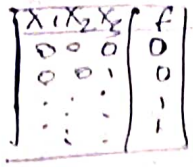
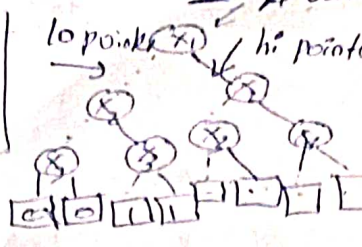
BDD Basics

#Idea : BDD



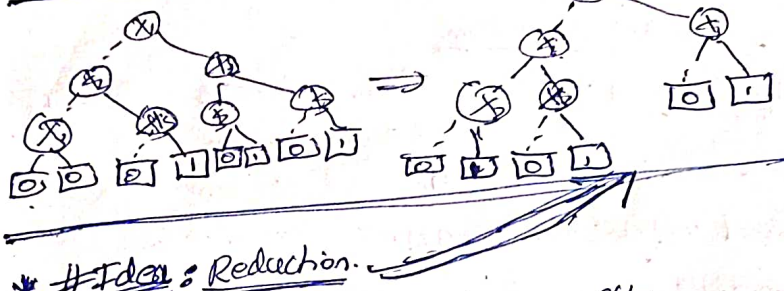


Canonical form [doesn't depend of gate level representation]

- we want canonical form data structure.
- but it is too big

#Idea : Ordering

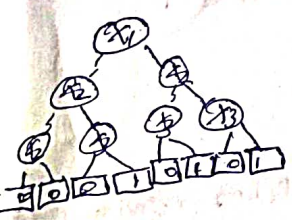
"Every path visits the root with same order"

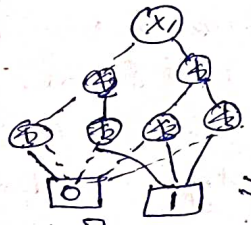


#Idea : Reduction.

Data structure as small as possible..

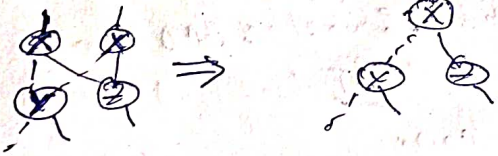
Reduction rules [Merge Equivalent leaves]

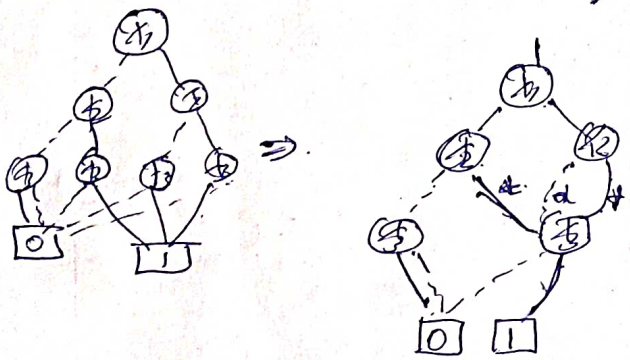




[Merge Isomorphic Nodes]

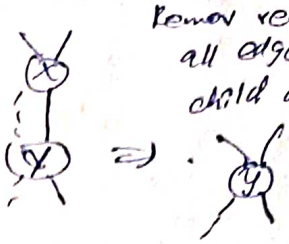
Same variable has identical children - 2 pointers.

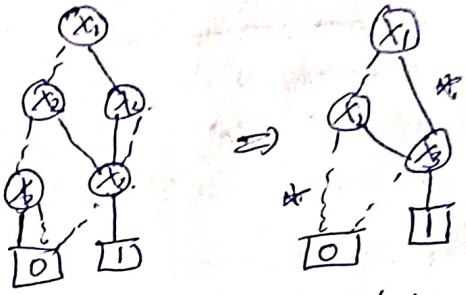




[Eliminate Redundant Tests]

Remove redundant node & redirect all edges into redundant node into child of removed node





Iteratively apply rules

[ROBDD]

Reduced ordered BDD.

ROBDD is a canonical form data structure for any boolean function.

BDD sharing

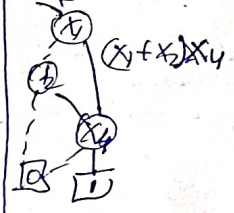
Any function as ROBDD

ROBDD → $f(x_1, x_2, \ldots x_n) = 0$ → f_1 (0)

ROBDD → $f(x_1, x_2 \ldots x_n) = 1$ → f_2 (1)

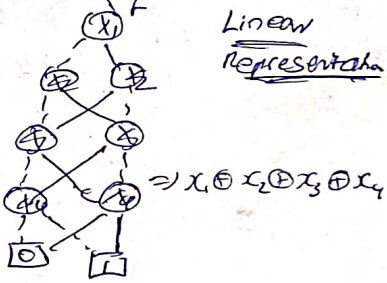
ROBDD → $f(x_1 \ldots x_i \ldots x_n) = x$

Typical Function



odd Parity

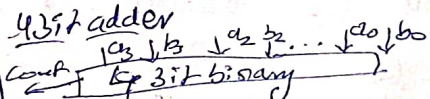
Linear representation



BDD parity = 1 if odd no. of 1's exist

#Every BDD node represents some function.

4 bit adder



BDD can have multirooted BDD

Sum, Cout ⇒ S_3 S_2 S_1 S_0 Cout

Seperately
51 nodes for 4-bit adder
1248 for 64-bit adder

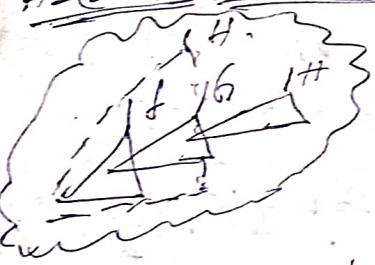
Shared
31 nodes for 4-bit adder
571 nodes for 64-bit adder

→ Shared data structure reuse multirooted graphs to do savings.

BDD Ordering, Implementation

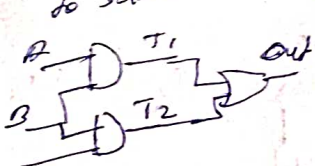
- Recursive methods like ORP
  [Shannon cofactor divide conquer is key]
- ops on BDD → And, Or, Not, Exor, cofactor
  ∀Quant, ∃Quant, Satisfy etc
- Boolean functions
  → Shared, Reduced, Ordered

H = (bdd) op (bdd F, bdd G)

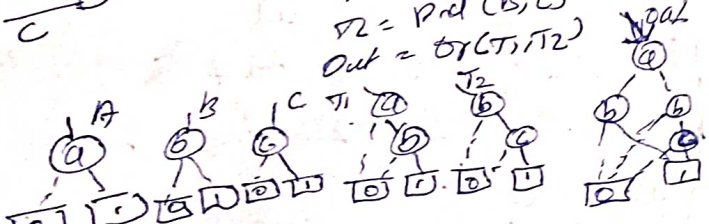


Build up Incrementally
Each i/p is BDD, each gate is operator give op BDD

- Build BDD for out as a script of calls to basic BDD operators

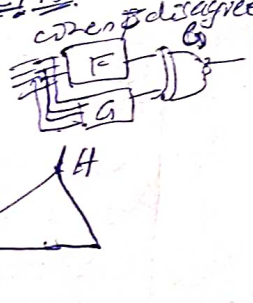


BDD operator script
A = Create bdd ("A")
B = " ("B")
C = " ("C")
T1 = And(A, B)
T2 = And (B, C)
Out = Or(T1, T2)



BDD Logics
F == G
Are boo funcs same

H = F ⊕ G
Satisfiability of H.



Tautology checking
With BDD, this Trivial, Given BDD check == 1

Satisfiability
Find values (0,1) for vars F == 1
Any path that's a solution.

Satisfiability: X_1 X_2 X_4
1 — — 1
0 1 — 1
Another OP.

Good Ordering	Bad ordering
Linear Growth	Exponential Growth

→ Variable ordering heuristics
→ Characterization
→ Dynamic Ordering

Variable Ordering

Carry chain Circuits: Adders, Subtractors, Comparators, Priority Encoders.

BDD order: $a_0 b_0 \cdots a_n b_n$.

General Experience with BDD: ~100s nodes.

Summary:

Reduced, Ordered, BDDs, ROBDD

- Canonical - data structure - Boolean functions.
- Identical BDD → F == G.
- Every node is a function.
- Boolean function is a pointer
- Basis for today's general manipulation of Boolean stuff.

Problems:

- Variable ordering matters: Sometimes BDD is too big.
- We want to know SAT don't need to build the whole function.