Direct Construction of ROBDDs

- In the past lectures, we learnt:
  - $ITE(f, g, h) = f \cdot g + f' \cdot h$
  - $f \odot g = x'(f_{x'} \odot g_{x'}) + x(f_x \odot g_x)$
  - ITE at top-nodes of $f, g, h \rightarrow$ ITE at their cofactors
  - Operate on graphs of f, g, h and derive the graph for $Z = ite(f, g, h)$
  - Shannon's expansion always w.r.t. top-node of f, g, h
- Use of a symbol table as a unique table
  - Every time you create a new node, Reduce it
  - Then, compute its *Key:* $\{low(v), v, high(v)\}$
  - No duplicate keys to be stored in the hash-table

The ITE Algorithm

$ITE(f, g, h)\{$

if (terminal case)

return trivial result;

else

$x =$ top variable of f, g, h;

$e = ITE(f_{x'}, g_{x'}, h_{x'})$;

$t = ITE(f_x, g_x, h_x)$;

if $(t == e)$/* redundant node */

return t; /* or return e */

/* Look-up the unique table for isomorphic subtrees */

$r = find\_or\_add\_in\_unique\_table(e, x, t)$;

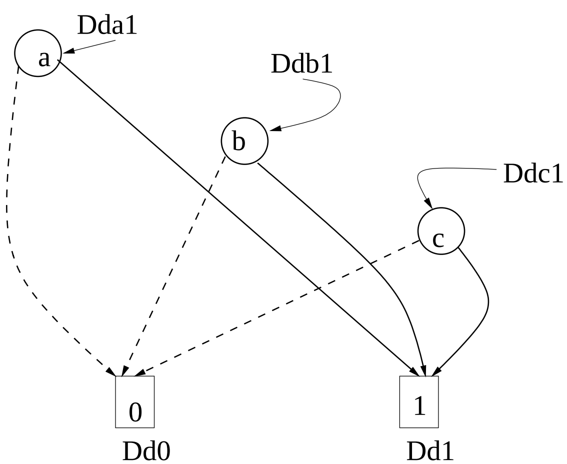
Update unique table, if required;

return r;

end if

ROBDD Construction Example

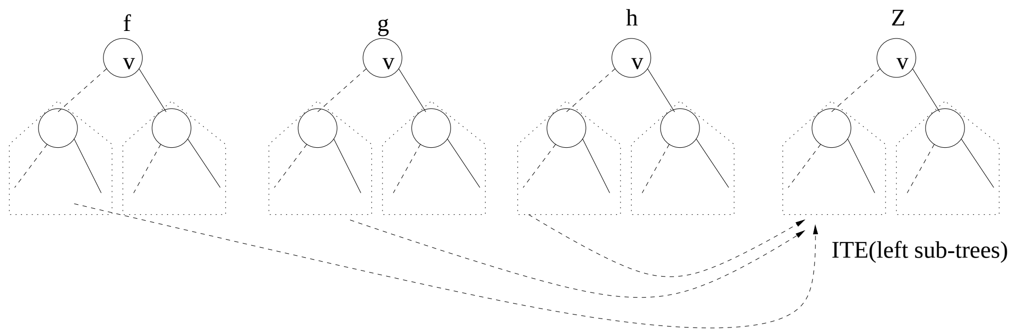
- $f = a + c; g = b + c; f \cdot g = ab + c$
- First construct trivial ROBDDs for a, b, c
- Assume var order a, b, c
- Fill-up the Unique Table (symbol/hash table)

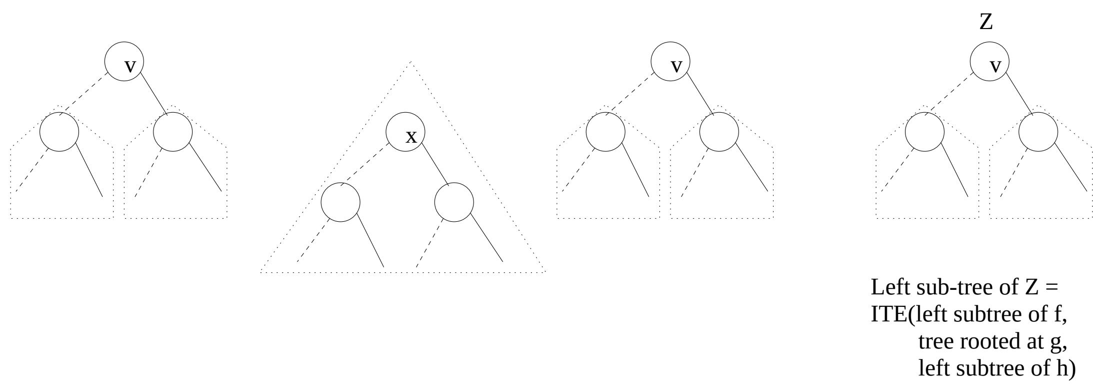


Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

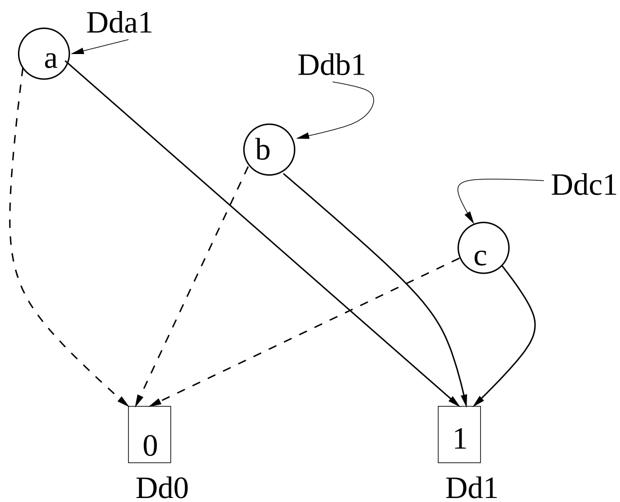
Constructing ROBDDs using ITE

- Few things to keep in mind....
- When f, g, h have same top variable...
- When the f, g, h may not have same top-vars



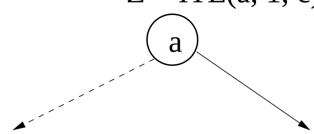


Construct $f = a + c$



Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

Z = ITE(a, 1, c) = ITE(Dda1, DD1, Ddc1)

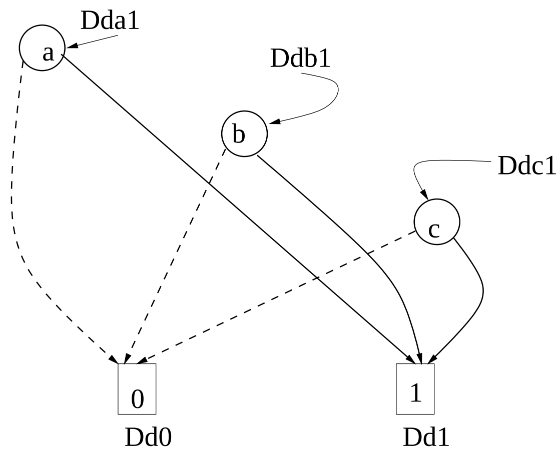


TOP-VAR = a

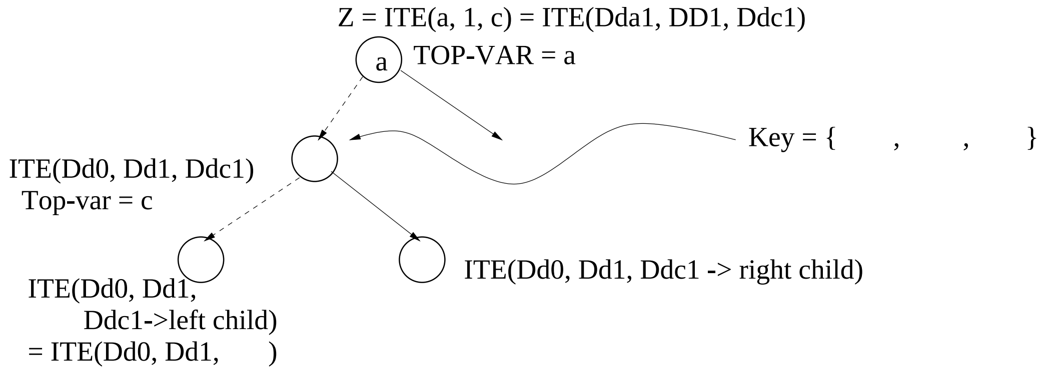
ITE(Dda1->left child, Dd1, Ddc1)

= ITE(Dd0, Dd1, Ddc1)

$f = a + c$ Contd....

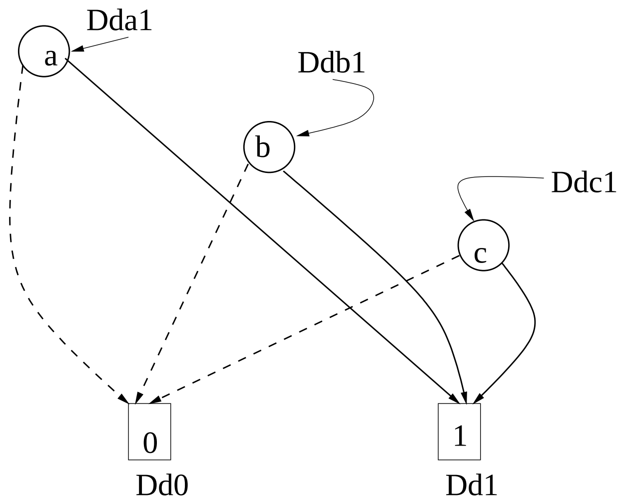


Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1



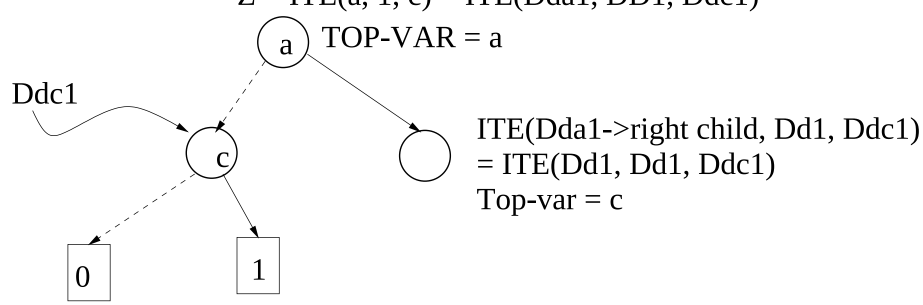
- Go-up a recursion level and compute the right sub-tree

$f = a + c$ Contd. further...



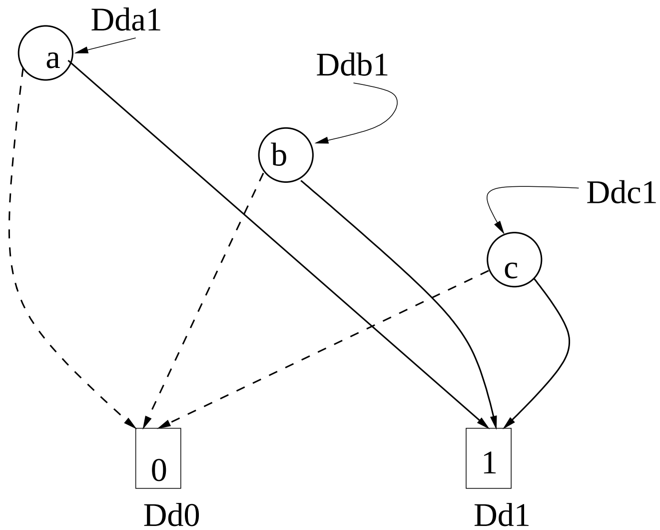
Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

Z = ITE(a, 1, c) = ITE(Dda1, DD1, Ddc1)

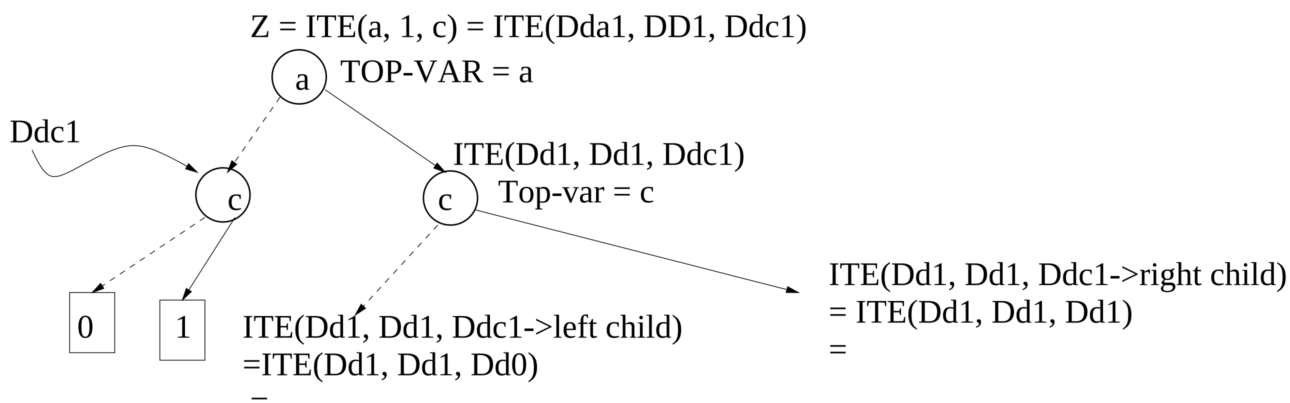


•

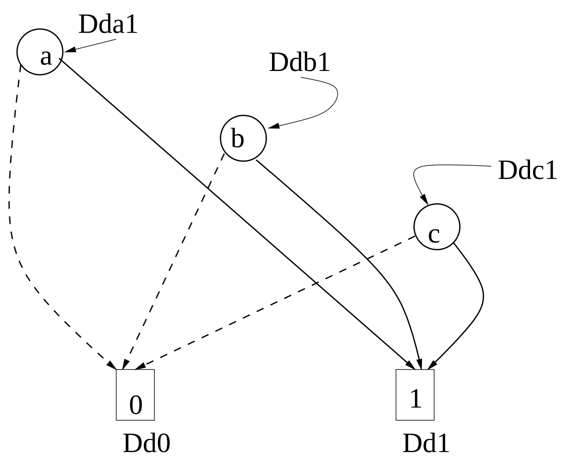
$f = a + c$ Contd. even further...



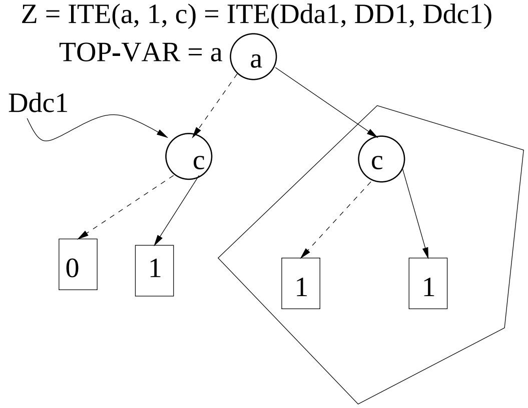
Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1



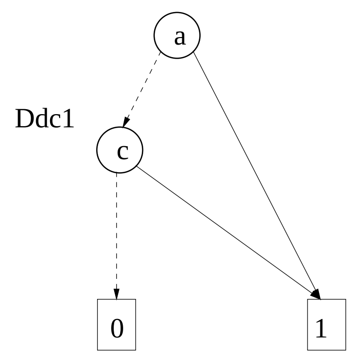
Final ROBDD for $f = a + c$



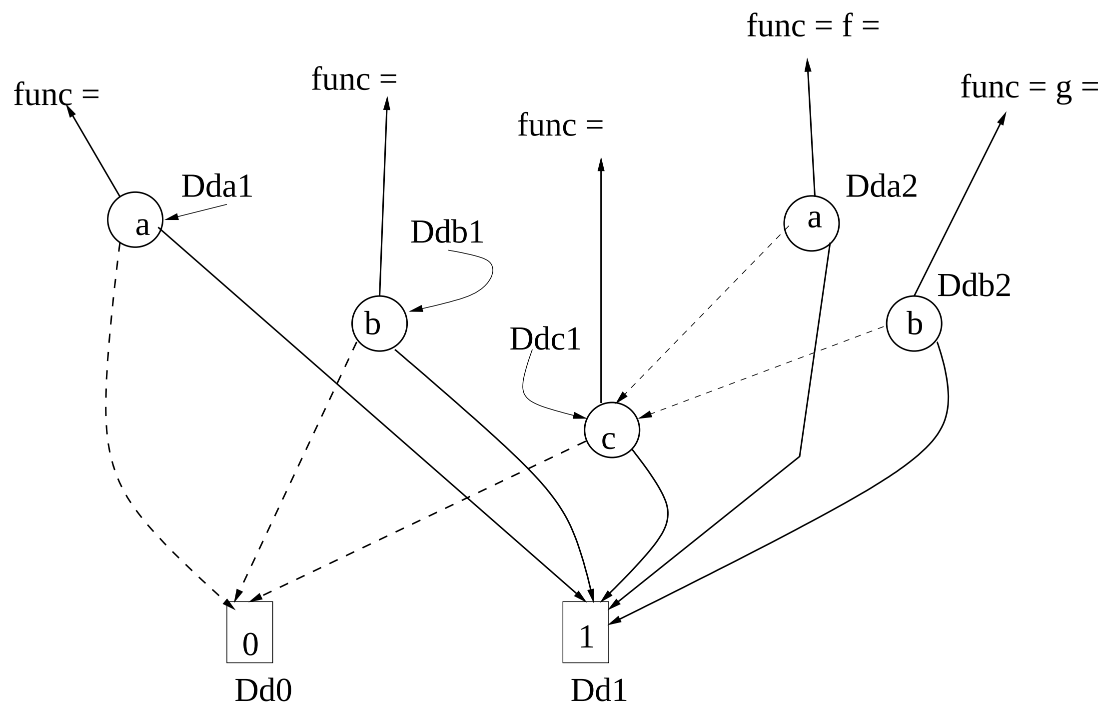
Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1



Key:{ , , }



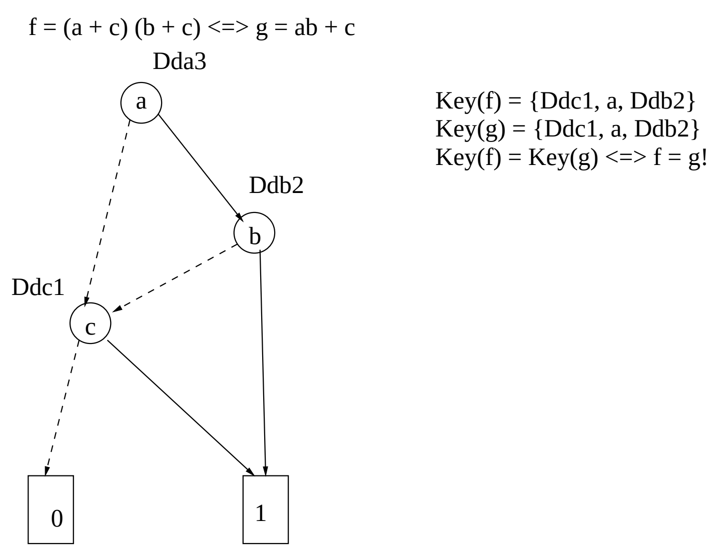
$f \cdot g = (a + c) \cdot (b + c)$



- Homework: Compute $f \cdot g$ yourself!
- Homework: Compute $(f = ab) \cdot (g = ab')$ yourself and notice how the graph reduces to $Dd0$.

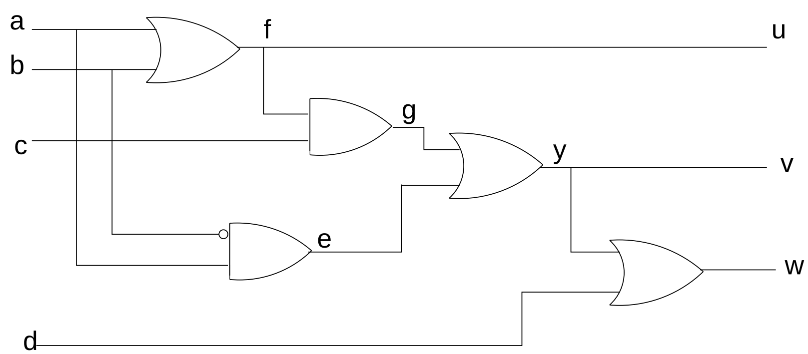
Equivalence Verification

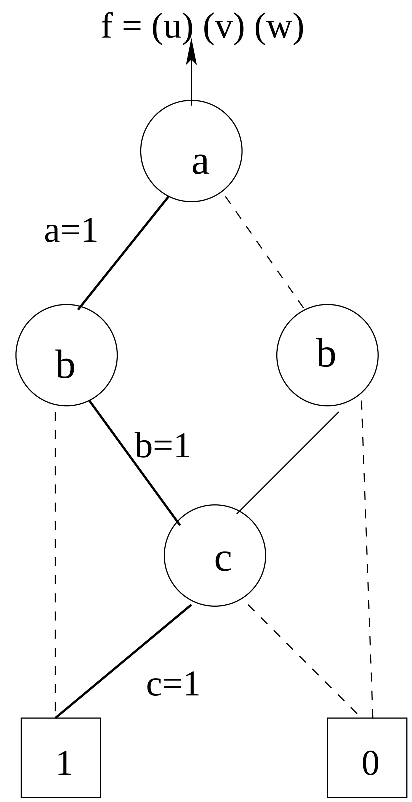
- $f = (a+c)(b+c)$; $g = ab + c$
- Compute f, compute keys of every node, update symbol table
- Do the same with g and prove to yourself that $f = g$



Application to SAT

- For the circuit shown, SAT(u=v=w=1) = ac + bc + ab'
- BDD $(u \cdot v \cdot w)$ shown. How to pick a solution?





BDD Size Problems and Variable Order

- Change Var order $\rightarrow$ ROBDD structure and size changes!
- Worst-case ROBDD size $\rightarrow$ no reduction $\rightarrow$ full-blown tree.
- Given a Boolean function, how do we forecast a good var order?
- Intractable Problem! Though some heuristics exist....
- Var $x \in$ many cubes $\rightarrow$ keep it up in the tree
- **Variable Ordering**: Interchange order of variables and see the change in size of ROBDD
- **Dynamic Var Ordering**: Do this while constructing OBDDs
- Careful: Var ordering of ALL the BDDs in the manager has to go through re-ordering.
- Multipliers: There exists NO good ordering. Take any order, at least one output would have worst-case scenario!

```
Script started on Sun Sep  8 20:52:52 2002
[1]panda:examples>> vis

vis release 1.3 (compiled 8-Jun-02 at 8:01 PM)
vis> read_blif m10x10.blif
Warning: Some variables are unused in model X_MULTIPLY_OP0.
Warning: Some variables are unused in model X_MULTIPLY_OP2.
Warning: Some variables are unused in model X_MULTIPLY_OP5.
Warning: Some variables are unused in model X_MULTIPLY_OP8.
Warning: Some variables are unused in model X_MULTIPLY_OP11.
Warning: Some variables are unused in model X_MULTIPLY_OP14.
Warning: Some variables are unused in model X_MULTIPLY_OP17.
Warning: Some variables are unused in model X_MULTIPLY_OP20.
Warning: Some variables are unused in model X_MULTIPLY_OP23.
Warning: Some variables are unused in model X_ADD_OP3.
Warning: Some variables are unused in model X_ADD_OP6.
Warning: Some variables are unused in model X_ADD_OP9.
Warning: Some variables are unused in model X_ADD_OP12.
Warning: Some variables are unused in model X_ADD_OP15.
Warning: Some variables are unused in model X_ADD_OP18.
Warning: Some variables are unused in model X_ADD_OP21.
Warning: Some variables are unused in model X_ADD_OP24.
Warning: Some variables are unused in model X_ADD_OP26.
vis>
vis>
vis>
vis> flatten_hierarchy
vis> static_order
vis> write_order
# vis release 1.3 (compiled 8-Jun-02 at 8:01 PM)
# network name: mpy2_temp_1
# generated: Sun Sep  8 20:53:28 2002
#
# name type            mddId vals levs
B_0  primary-input        0    2 (0)
A_8  primary-input        1    2 (1)
B_1  primary-input        2    2 (2)
A_7  primary-input        3    2 (3)
B_2  primary-input        4    2 (4)
A_6  primary-input        5    2 (5)
B_3  primary-input        6    2 (6)
A_5  primary-input        7    2 (7)
B_4  primary-input        8    2 (8)
A_4  primary-input        9    2 (9)
B_5  primary-input       10    2 (10)
A_3  primary-input       11    2 (11)
B_6  primary-input       12    2 (12)
A_2  primary-input       13    2 (13)
B_7  primary-input       14    2 (14)
A_1  primary-input       15    2 (15)
B_8  primary-input       16    2 (16)
A_9  primary-input       17    2 (17)
B_9  primary-input       18    2 (18)
A_0  primary-input       19    2 (19)
vis>
vis>
vis> build_partition_mdds inout
vis>
vis> print_partition_stats
Method Inputs-Outputs, 20 sinks, 20 sources, 40 total vertices, 148442 mdd nodes
vis>
vis>
vis>
vis>
vis>
vis> dynamic_var_ordering -f sift
Dynamic variable ordering forced with method sift....
vis>
vis>
vis>
vis> print_partition_stats
Method Inputs-Outputs, 20 sinks, 20 sources, 40 total vertices, 85793 mdd nodes
vis>
vis>
vis> write_order
# vis release 1.3 (compiled 8-Jun-02 at 8:01 PM)
# network name: mpy2_temp_1
# generated: Sun Sep  8 20:54:47 2002
#
# name type            mddId vals levs
A_9  primary-input       17    2 (0)
B_0  primary-input        0    2 (1)
B_1  primary-input        2    2 (2)
A_8  primary-input        1    2 (3)
B_2  primary-input        4    2 (4)
A_7  primary-input        3    2 (5)
A_6  primary-input        5    2 (6)
B_5  primary-input       10    2 (7)
A_5  primary-input        7    2 (8)
B_4  primary-input        8    2 (9)
B_3  primary-input        6    2 (10)
A_4  primary-input        9    2 (11)
A_3  primary-input       11    2 (12)
B_6  primary-input       12    2 (13)
B_7  primary-input       14    2 (14)
B_8  primary-input       16    2 (15)
B_9  primary-input       18    2 (16)
A_0  primary-input       19    2 (17)
A_2  primary-input       13    2 (18)
A_1  primary-input       15    2 (19)
vis> quit
[2]panda:examples>> quit

quit: Command not found.
[3]panda:examples>> exit


Script done on Sun Sep  8 20:55:14 2002
```