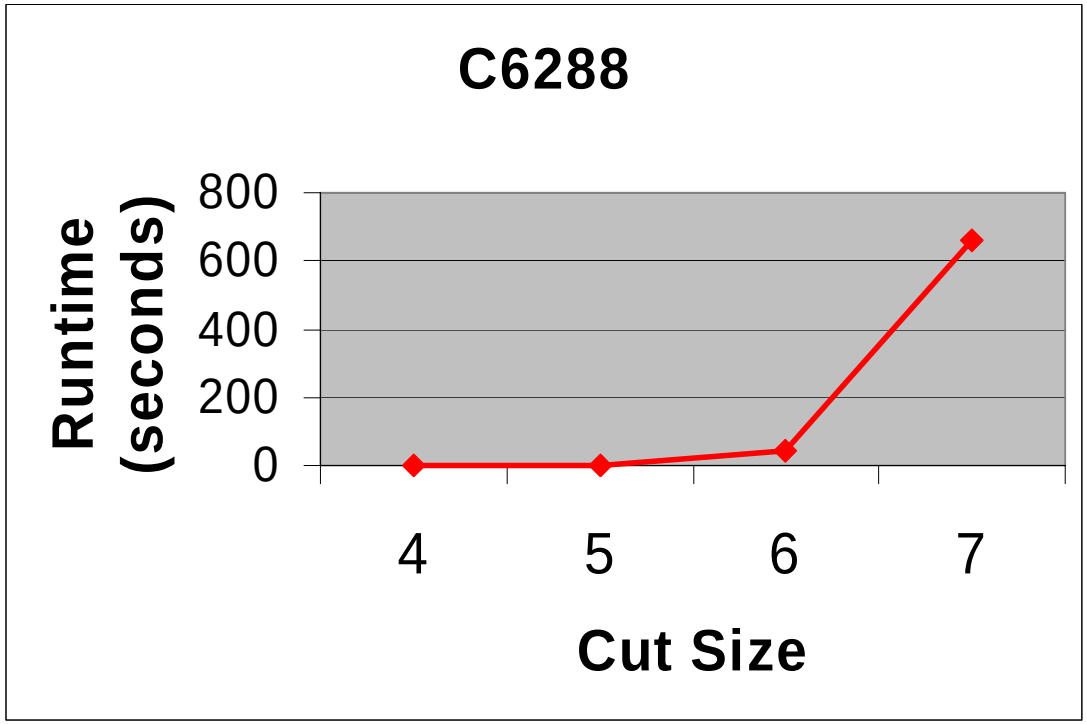
BddCut: Towards Scalable Cut Enumeration

Andrew C. Ling Ph.D. Candidate (Speaker)
University of Toronto
Professor Jianwen Zhu
University of Toronto
Professor Stephen D. Brown
University of Toronto and Altera Corporation

Cut Enumeration: Applications

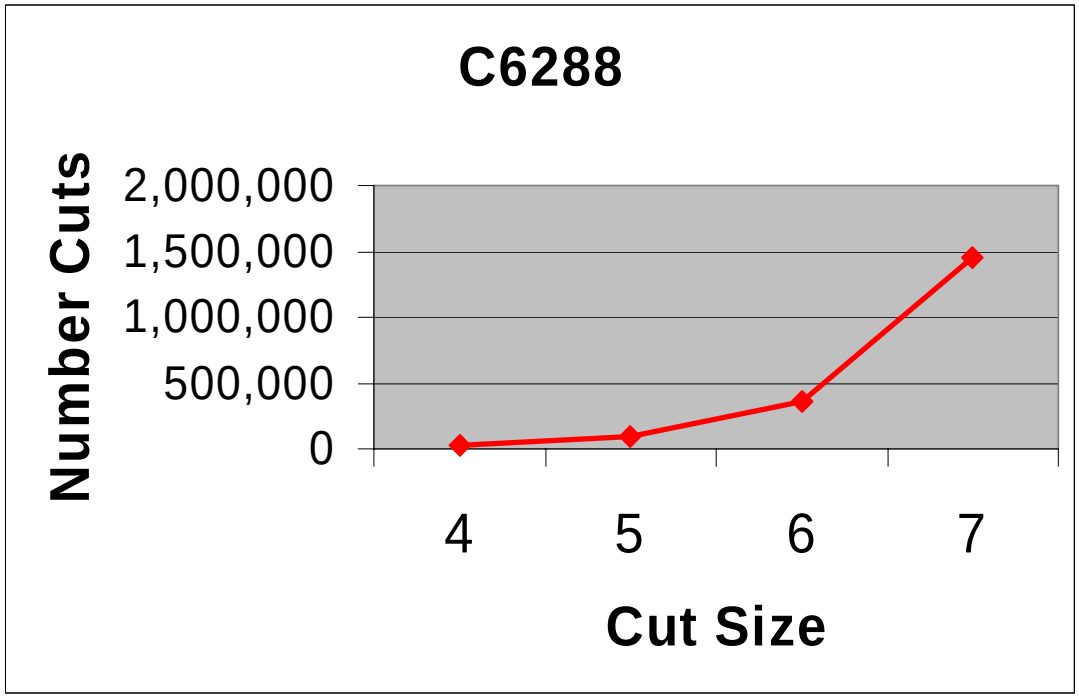
- Applications:
  - K-LUT technology mapping ($K \leq 6$)
  - Rewriting ($K \leq 6$)
  - *Resynthesis* ($K > 6$)
  - *Macrocell Technology mapping* ($K > 6$)

Cut Enumeration: Applications



- IMap Technology Mapper (TCAD'06), cut enumeration runtime in seconds.

Cut Enumeration: Applications



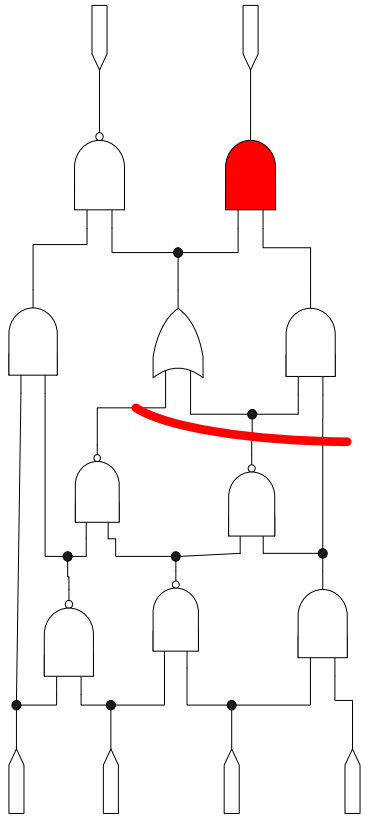
- IMap Technology Mapper (TCAD'06), number cuts generated.

Cut Enumeration: Applications

- Applications:
  - K-LUT technology mapping ($K \leq 6$)
  - Rewriting ($K \leq 6$)
  - *Resynthesis* ($K > 6$)
  - *Macrocell Technology mapping* ($K > 6$)
  - **Elimination** ($K > 7$)

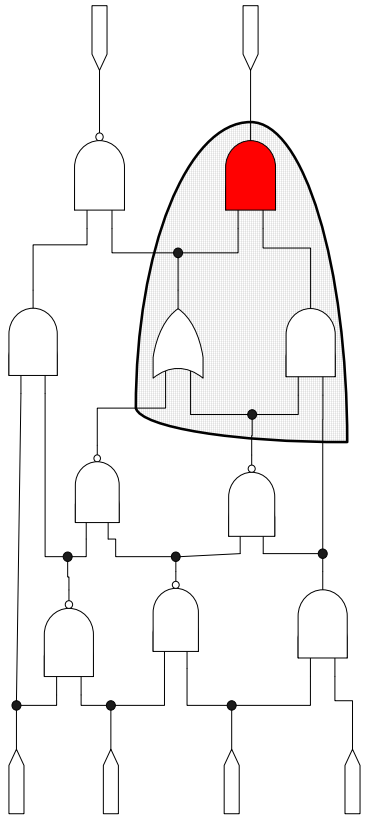
Definition: Cut

- A cut, for a root node, of size K is the set of fanin nodes where all paths from the fanin nodes converge at the root node.



Definition: Cut

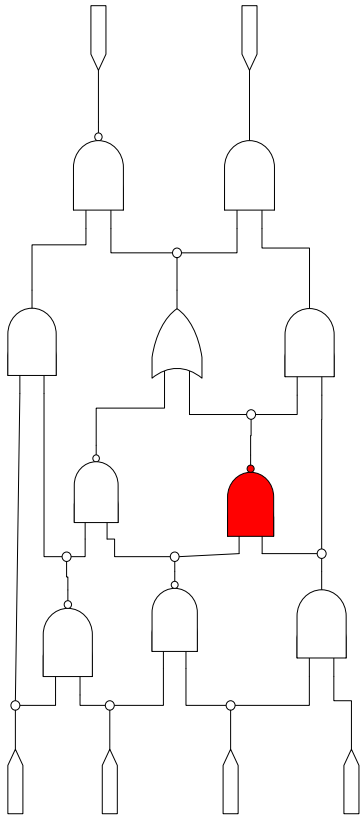
- One to one mapping between cuts and cones.



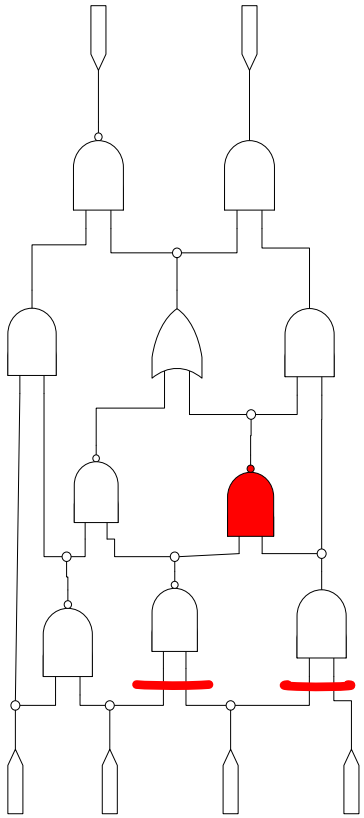
Cut Enumeration

- Cong et al. “Cut ranking and pruning: enabling a general and efficient FPGA mapping solution”, FPGA’99
- All cuts of size K are generated for every node.
- Generates cuts by combining the cuts of the fanin nodes in every possible way.
- Cuts must be duplicated every time they are used to form a new cut from one of its transitive fanouts.

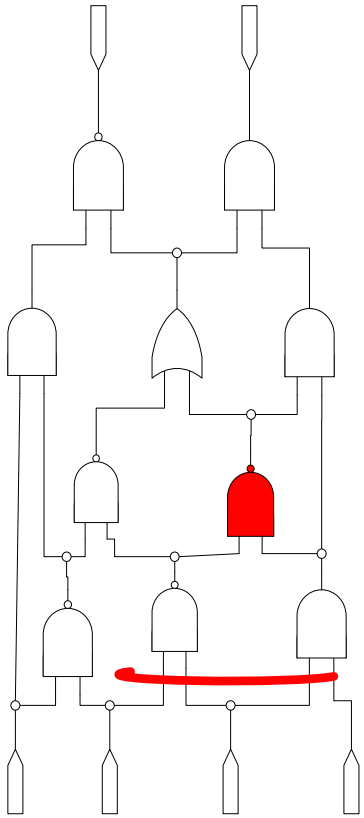
Cut Enumeration Example



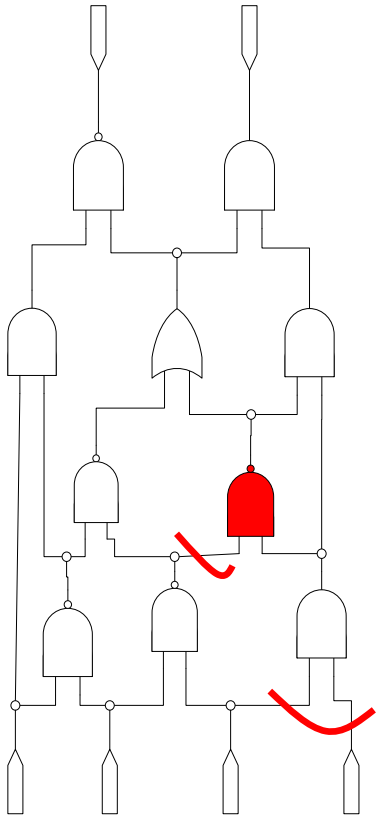
Cut Enumeration Example



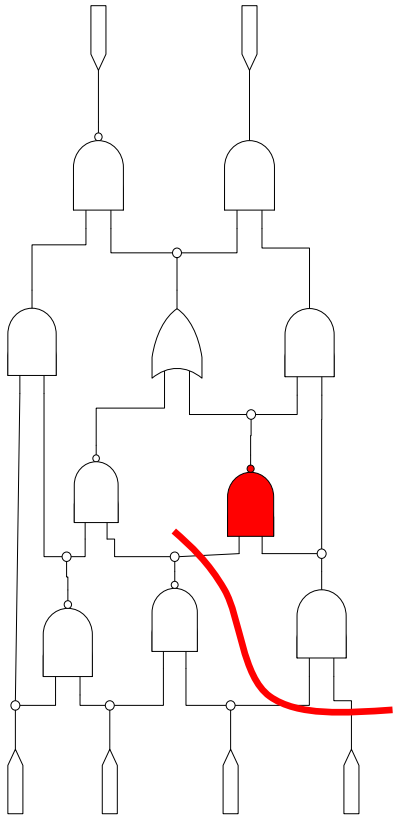
Cut Enumeration Example



Cut Enumeration Example



Cut Enumeration Example

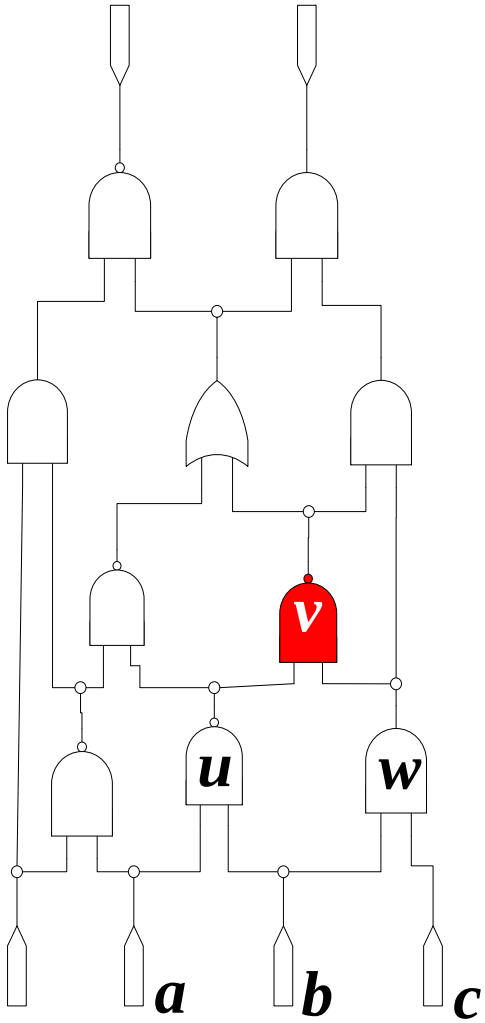


Cut Enumeration Set Relation

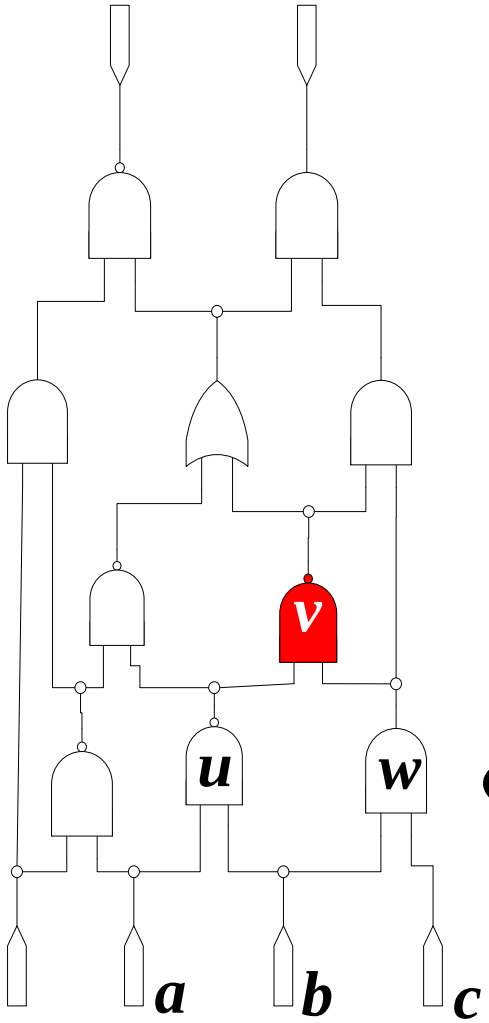
$$\Phi(v) = \{c_u * c_w \mid$$
$$c_u \in \{\{u\} \cup \Phi(u) \mid u \in fanin(v)\},$$
$$c_w \in \{\{w\} \cup \Phi(w) \mid w \in fanin(v)\},$$
$$u \neq w, \|c_u * c_w\| \leq K\}$$

- Cong et al., “Cut ranking and pruning”, FPGA’99, Monterey, California, USA.

Cut Enumeration Set Relation



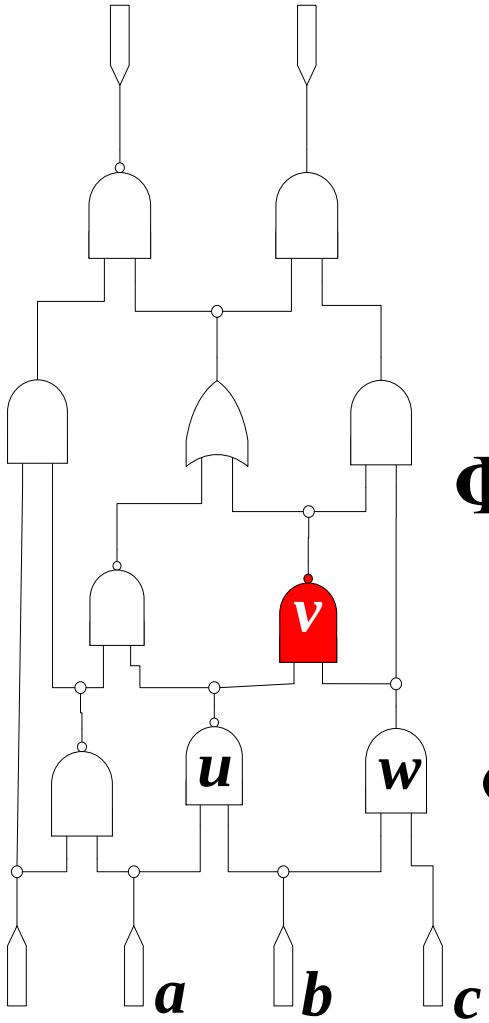
Cut Enumeration Set Relation



$\Phi(w)=\{\{bc\}\}$

$\Phi(u)=\{\{ab\}\}$

Cut Enumeration Set Relation

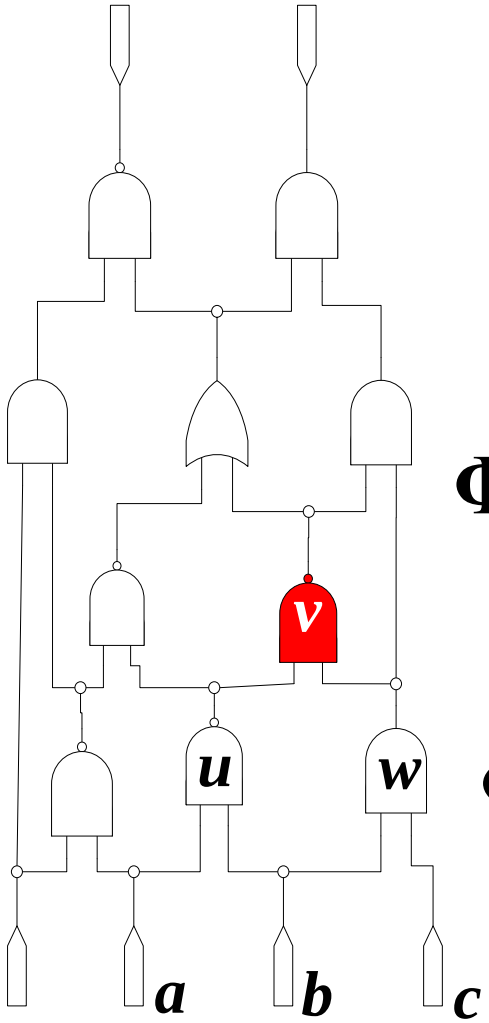


$$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$$

$$\Phi(w)=\{\{bc\}\}$$

$$\Phi(u)=\{\{ab\}\}$$

Cut Enumeration Set Relation

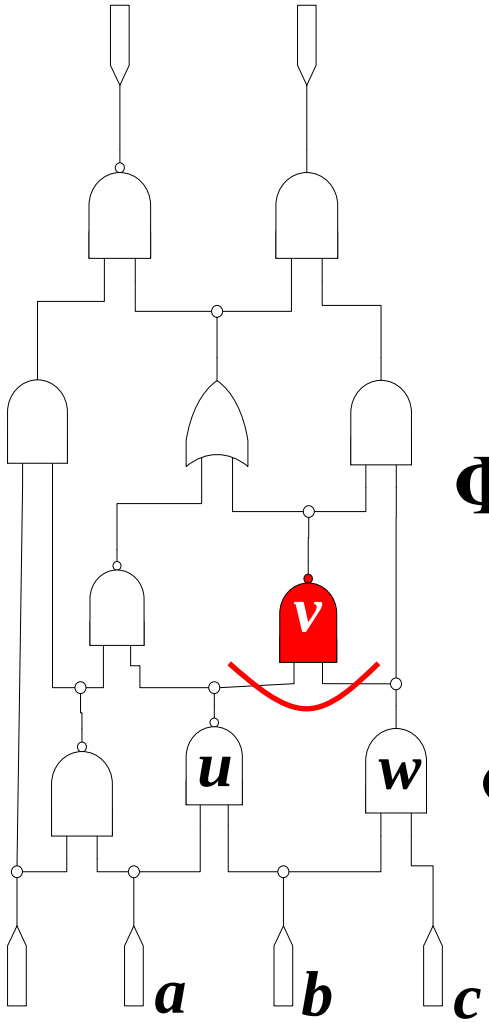


$$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$$
$$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$$

$$\Phi(w)=\{\{bc\}\}$$

$$\Phi(u)=\{\{ab\}\}$$

Cut Enumeration Set Relation

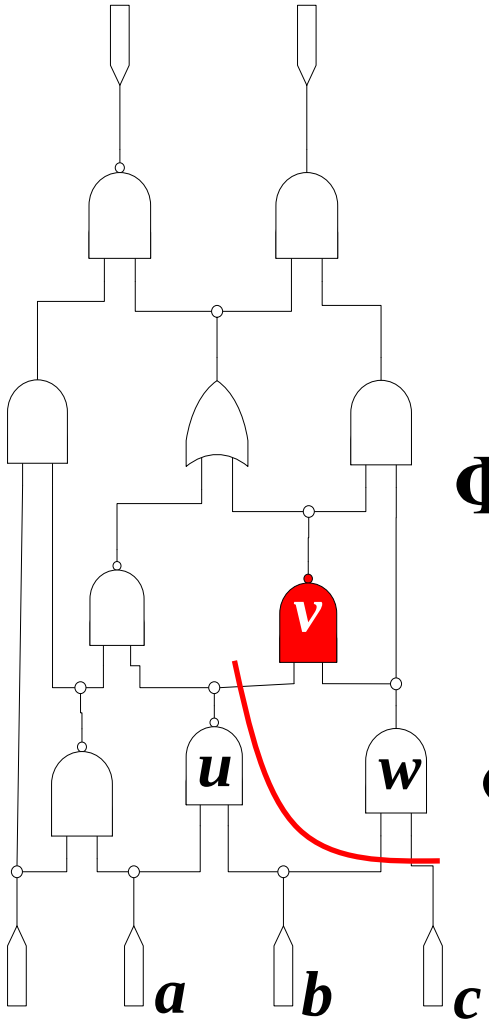


$$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$$
$$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$$

$$\Phi(w)=\{\{bc\}\}$$

$$\Phi(u)=\{\{ab\}\}$$

Cut Enumeration Set Relation

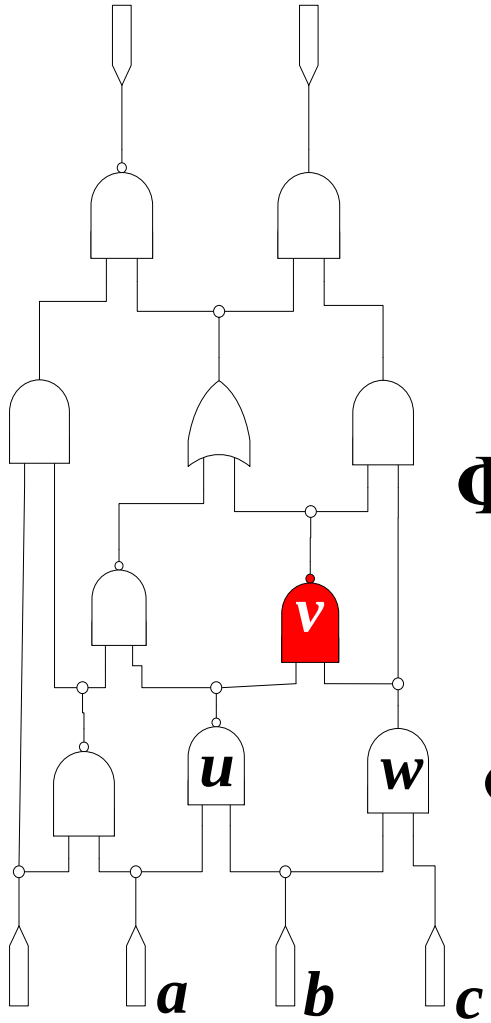


$$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$$
$$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$$

$$\Phi(w)=\{\{bc\}\}$$

$$\Phi(u)=\{\{ab\}\}$$

Cut Enumeration Set Relation



$$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$$
$$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$$

$$\Phi(w)=\{\{bc\}\}$$

$$\Phi(u)=\{\{ab\}\}$$

Cut Enumeration: heuristics

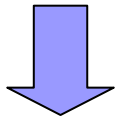
- Subcut duplication is not scalable for large values of K (K > 5).
- Work arounds:
  - Pruning – stop creating new cuts for a node after ‘N’ cuts have been created for a node (Cong et al. FPGA’99)
  - Removal of redundant cuts – throw away “useless” cuts during generation process (Mishchenko et al. FPGA’06)
  - Cut Dropping – throw away cuts after they have been processed (Mishchenko et al. FPGA’06)
  - Factor Cuts – ignore cuts within trees, expand factor cuts within trees if necessary (Chatterjee et al. ICCAD’06)

Symbolic Representation

- Represent our cut sets as BDDs.
- Reduces duplication problem dramatically:
  - Duplication of subcuts dramatically reduces since BDD managers reuse BDDs stored in memory to form new ones.
  - Subcuts are shared to form larger cuts in the BDD.
  - Redundant cuts are automatically removed.

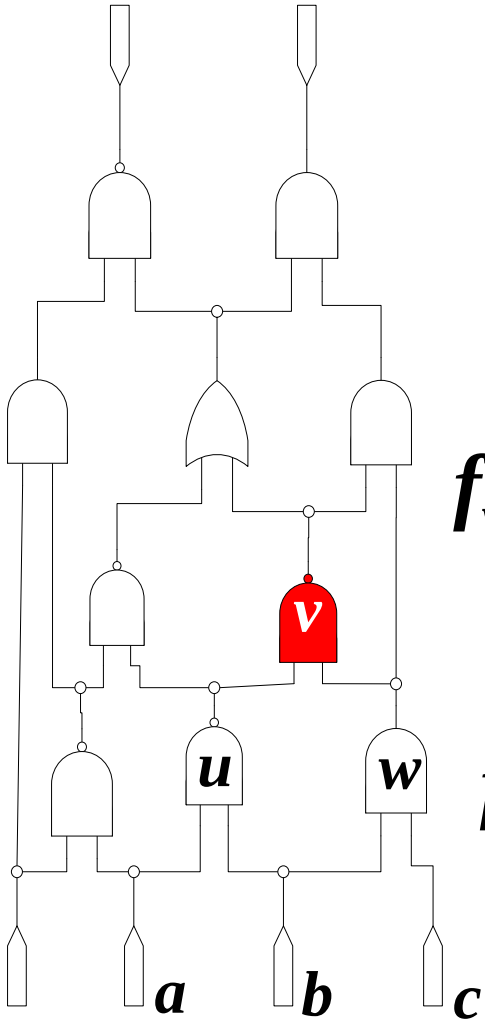
Symbolic Representation

$$\Phi(v) = \{c_u * c_w \mid$$
$$c_u \in \{\{u\} \cup \Phi(u) \mid u \in fanin(v)\},$$
$$c_w \in \{\{w\} \cup \Phi(w) \mid w \in fanin(v)\},$$
$$u \neq w, \|c_u * c_w\| \leq K\}$$



$$f_v = c_u \bullet c_w \mid$$
$$c_u = (u + f_u) \mid u \in fanin(v),$$
$$c_w = (w + f_w) \mid w \in fanin(v),$$
$$u \neq w, \|c_u \bullet c_w\| \leq K$$

Symbolic Representation



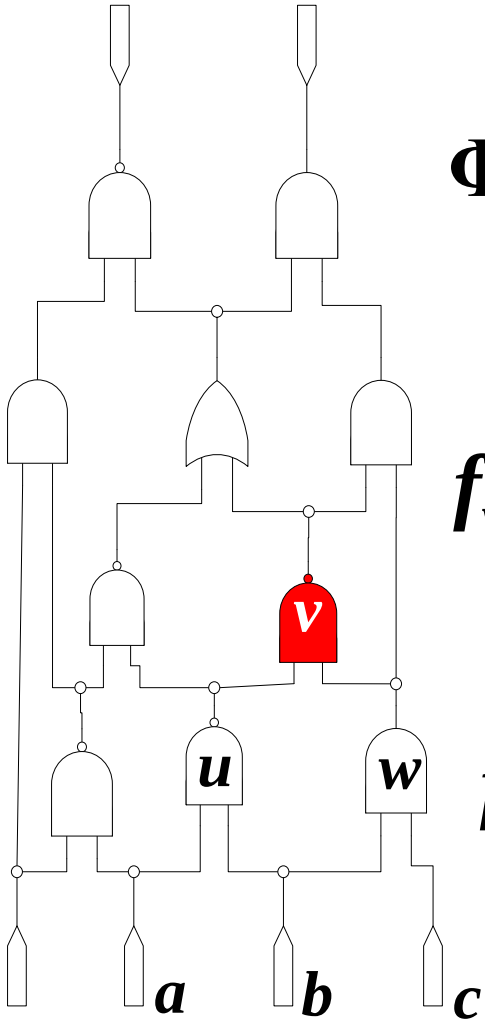
$f_v=(w+bc)(u+ab)$

$=uw+ubc+wab+abc$

$f_u=ab$

$f_w=bc$

Symbolic Representation



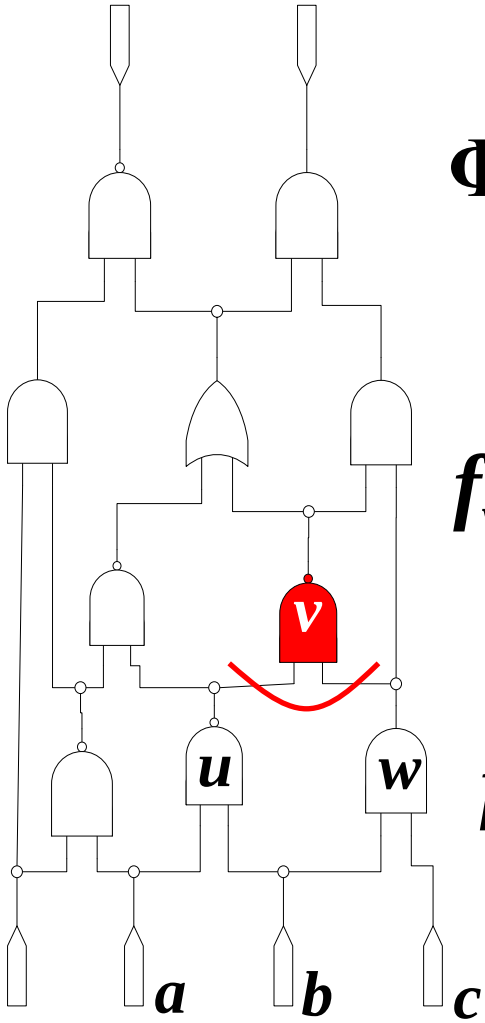
$$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$$
$$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$$

$$f_v=(w+bc)(u+ab)$$
$$=uw+ubc+wab+abc$$

$$f_u=ab$$

$$f_w=bc$$

Symbolic Representation



$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$

$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$

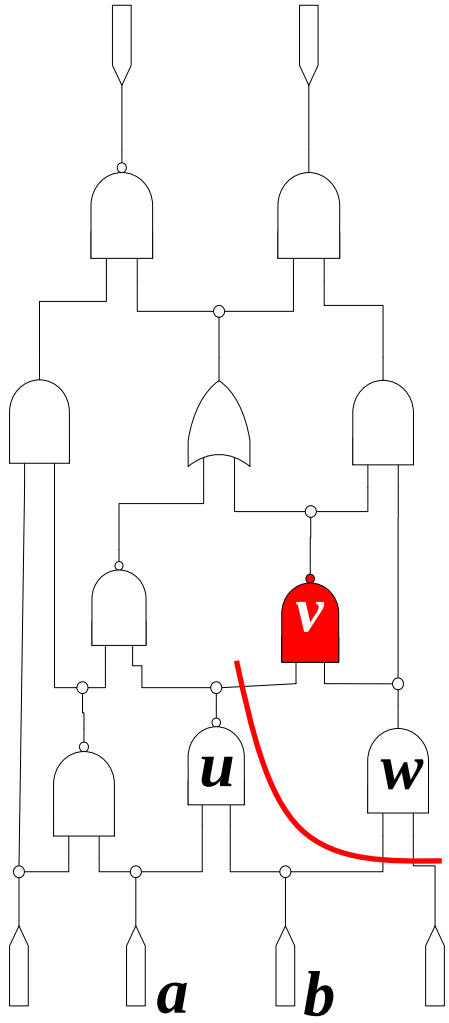
$f_v=(w+bc)(u+ab)$

$=uw+ubc+wab+abc$

$f_u=ab$

$f_w=bc$

Symbolic Representation



$\Phi(v)=\{\{w\},\{bc\}\}\times\{\{u\},\{ab\}\}$

$=\{\{uw\},\{ubc\},\{wab\},\{abc\}\}$

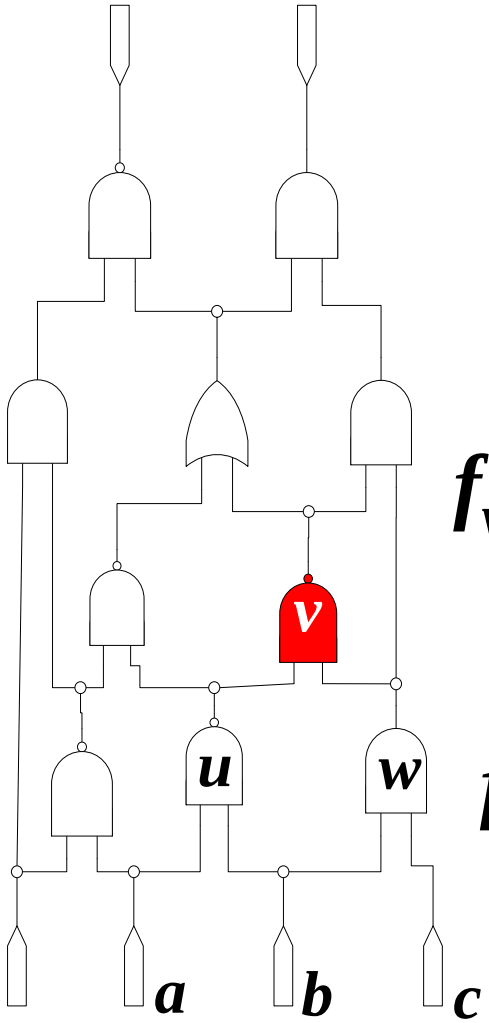
$f_v=(w+bc)(u+ab)$

$=uw+ubc+wab+abc$

$f_u=ab$

$f_w=bc$

Symbolic Representation



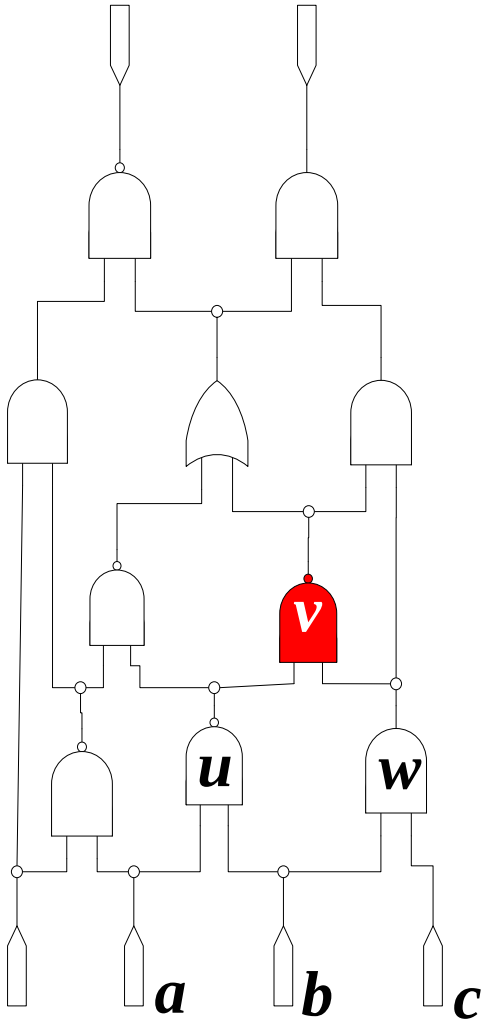
$f_v=(w+bc)(u+ab)$

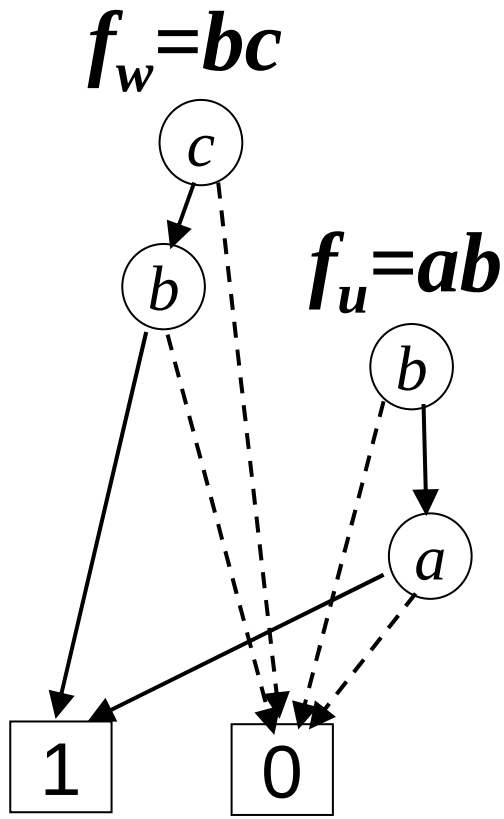
$=uw+ubc+wab+abc$

$f_u=ab$

$f_w=bc$

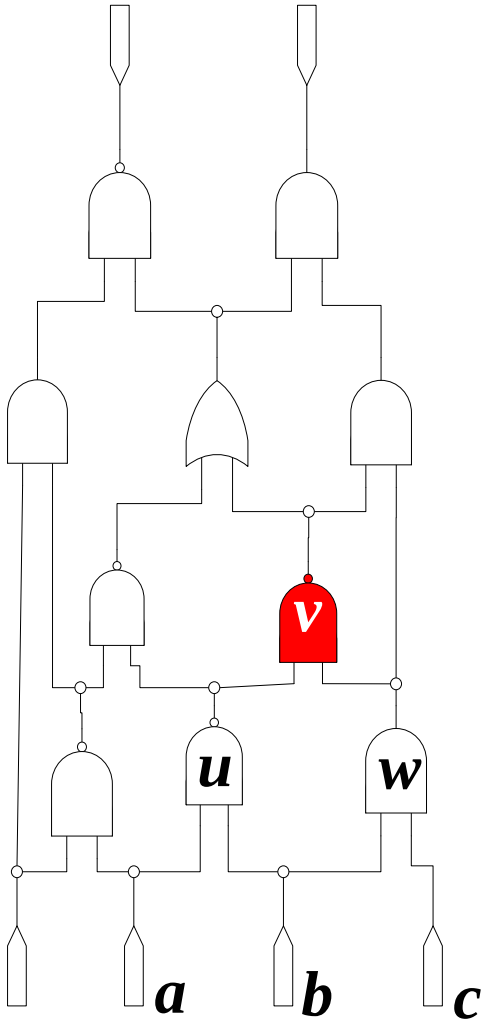
Symbolic Representation

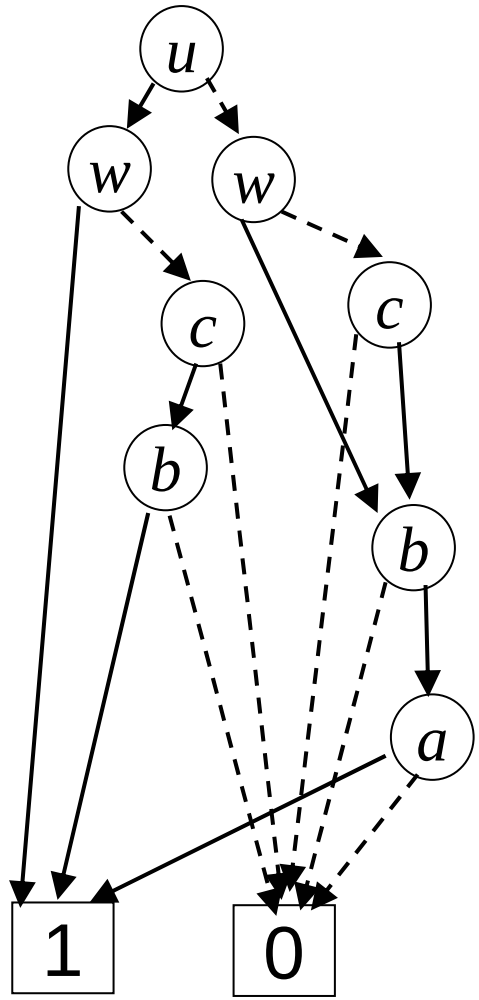




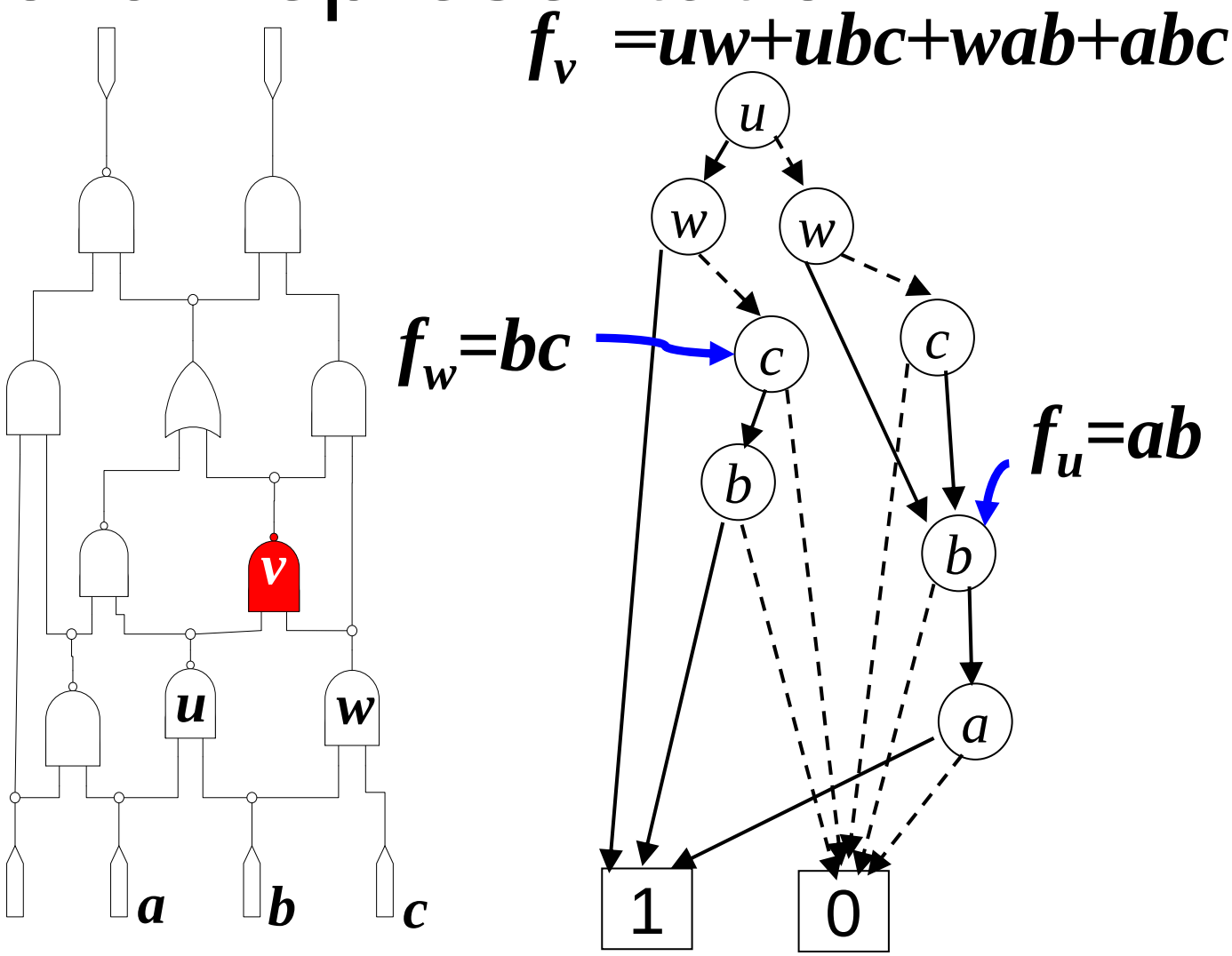
Symbolic Representation

$f_v = uw + ubc + wab + abc$

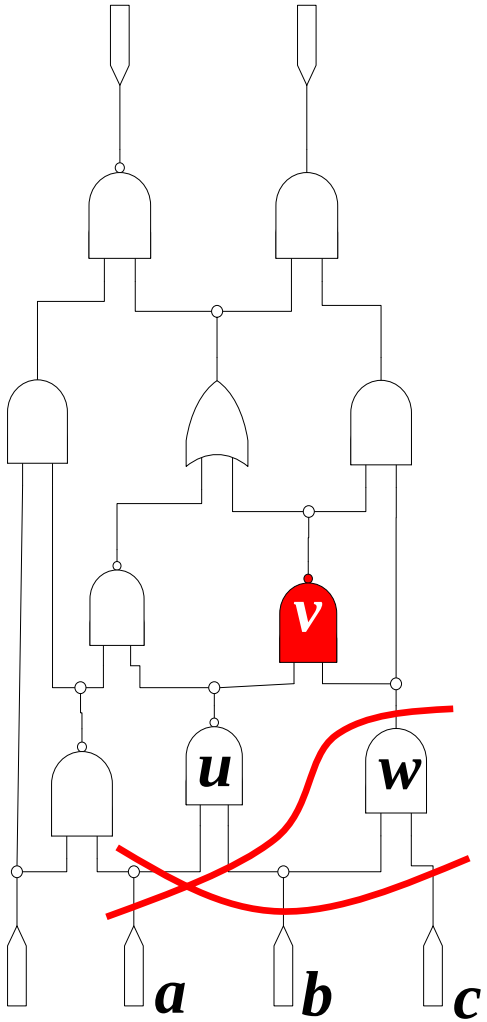


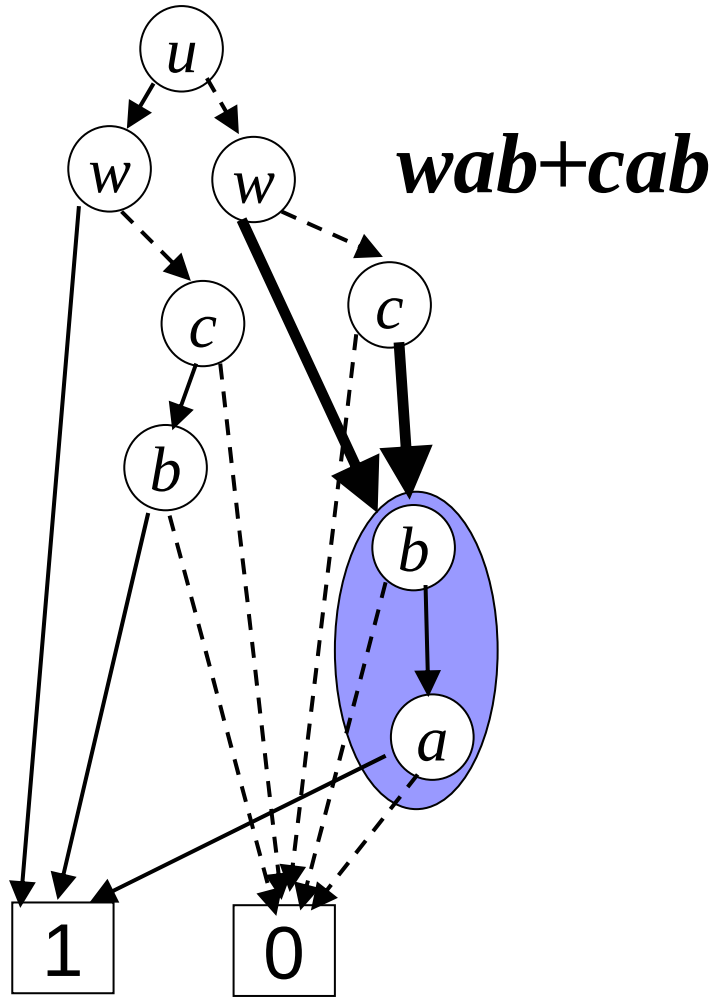


Symbolic Representation



Symbolic Representation





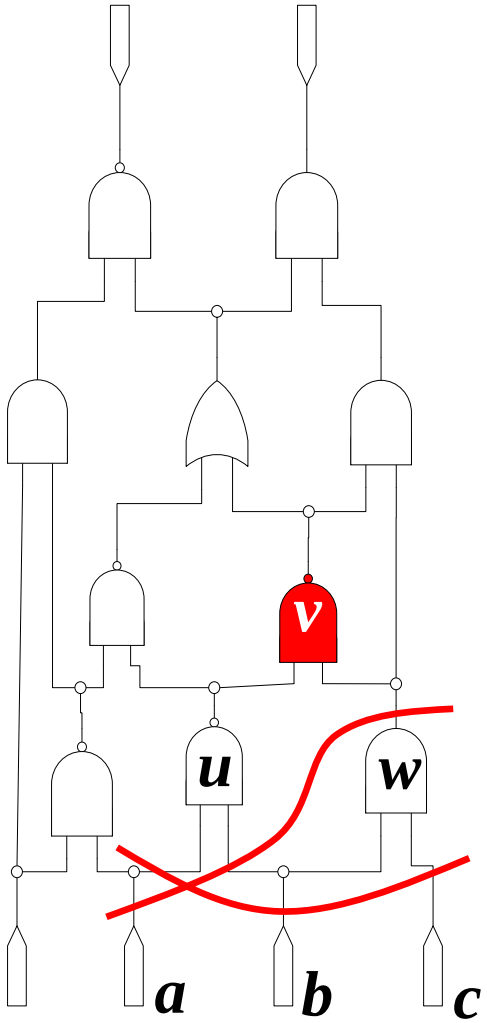
wab+cab

Cut Evaluation

- How do we evaluate our cuts in our symbolic approach?

$$\boldsymbol{cost}_{\boldsymbol{cut}} = \sum_{i \in cut} \boldsymbol{cost}_{i}$$

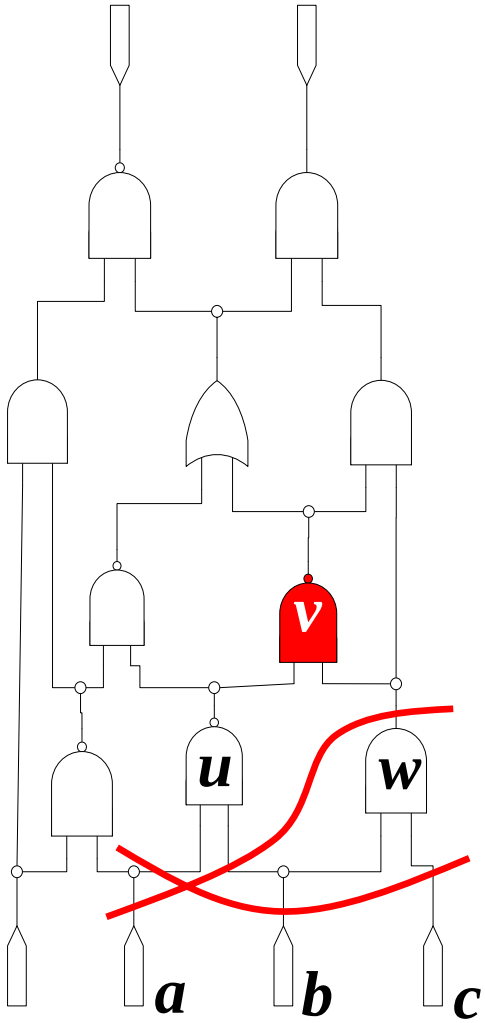
Cut evaluation: example



$$cost_{wab} = cost_w + cost_a + cost_b$$

$$cost_{cab} = cost_c + cost_a + cost_b$$

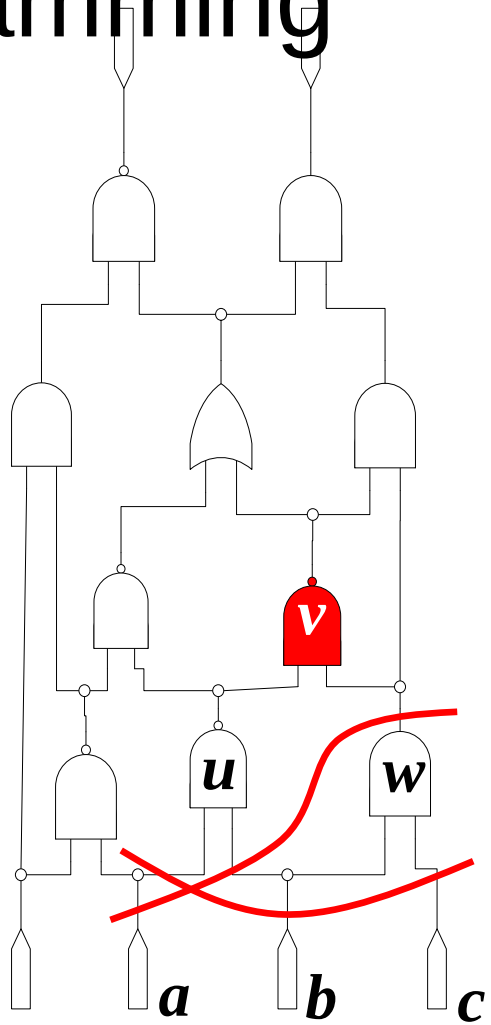
Cut evaluation: example

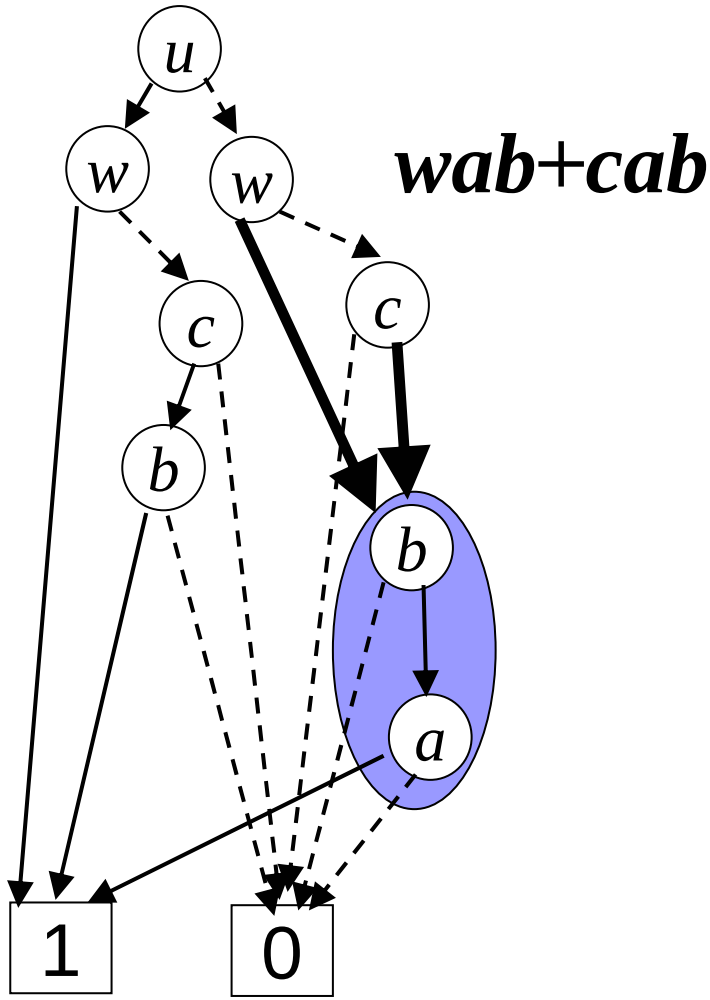


$$cost_{wab} = cost_w + cost_a + cost_b$$

$$cost_{cab} = cost_c + cost_a + cost_b$$

Cut evaluation: dynamic programming





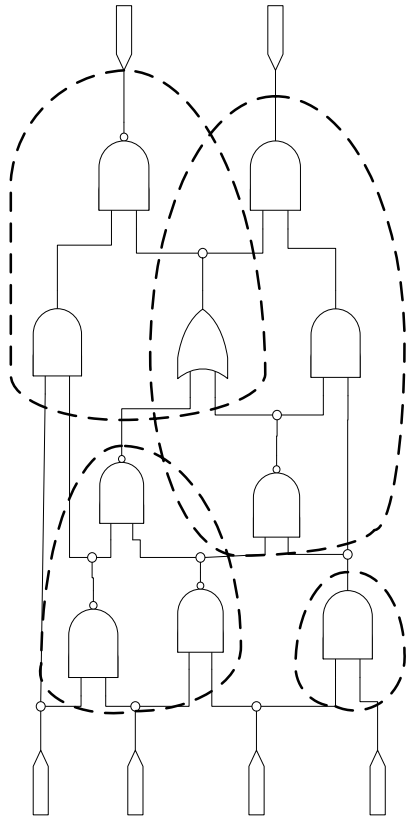
wab+cab

Cuts Applied to BDD Based Synthesis

- FBDD, which represents logic functions as BDDs, has sped up logic transformations over cube-based techniques
  - decomposition,Boolean matching, etc.
- Elimination has become the primary bottleneck
  - Takes up 70% of runtime in FBDD

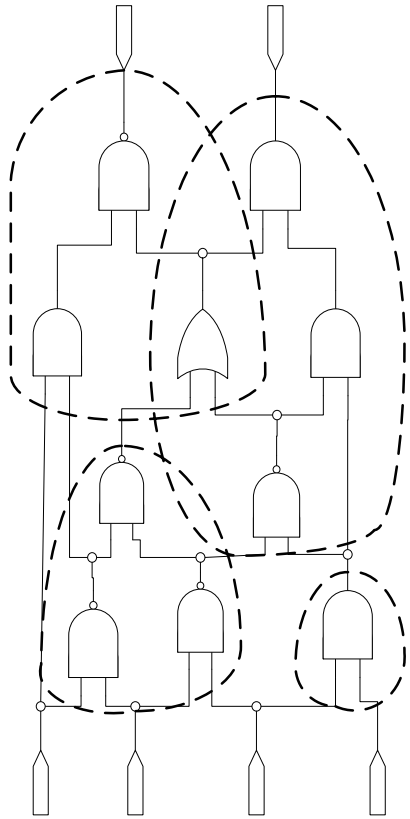
Elimination

- Removes redundancies and identifies resynthesis regions.



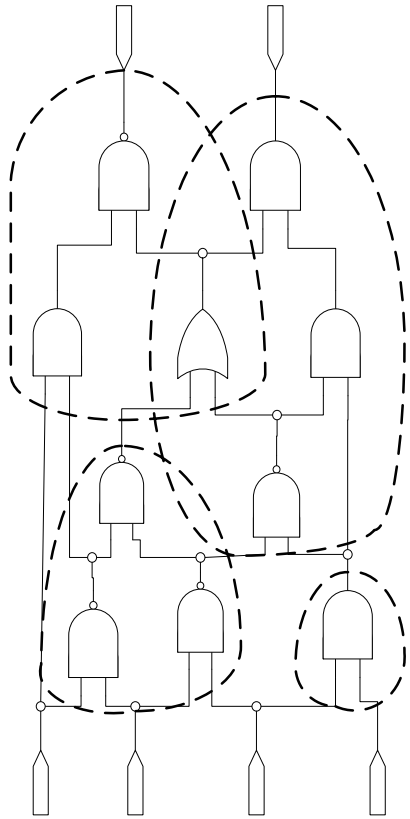
Elimination

- Currently, both cube based and BDD based synthesis rely on trial and error to create elimination regions
- Many compose and decompose operations necessary in BDD based elimination.



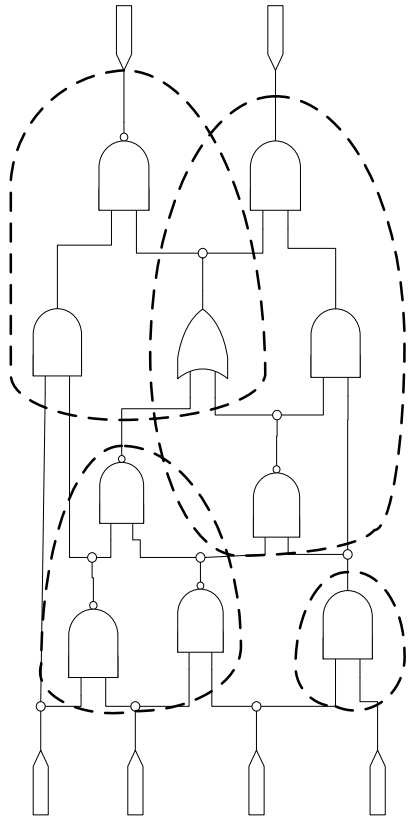
Elimination Solution

- Treat elimination as a covering problem
  - Use cuts to estimate area for each elimination region
  - Elimination regions are relatively large ($K \geq 8$)



Elimination Solution

- Treat elimination as a covering problem
  - Symbolism applied to Cut Enumeration will help



Results: Cut Enumeration

- Dell Pentium D. 3.2 GHz, 2GB RAM
- Compared against IMap (V. Manohararajah et al. TCAD'06) and ABC (Mishchenko et al. FPGA'06).

Results Cut Enumeration

Circuit	K=6 (sec)			K=7 (sec)		
	BddCut	*IMap*	*ABC*	*BddCut*	*IMap*	*ABC*
c6288	0.20	40.64	0.52	0.67	660.76	5.66
i10	0.22	14.27	0.25	1.58	98.27	2.00
b22_1	2.17	107.16	1.38	8.81	n/a	10.3
s4863	0.11	19.27	0.11	0.36	269.07	0.84
s6669	0.11	15.73	0.09	0.33	197.76	0.63
Ratio		63x	0.83x		225x	1.8x

Results: Cut Enumeration

Circuit	K=8 (sec)		K=9 (sec)		K=10 (sec)	
	BddCut	*ABC*	*BddCut*	*ABC*	*BddCut*	*ABC*
C6288	2.48	14.49	9.91	150.13	41.86	1758.44
des	9.05	10.7	74.66	105.16	828.44	1126.5
i10	2.83	6.06	11.41	57.17	50.78	581.09
b20	42.01	73.53	200.27	889.92	895.63	n/a
b21	44.03	80.34	205.25	942.88	920.22	n/a
b22_1	41.22	84.36	180.58	924.38	766.63	n/a
s4863	1.45	4.99	6.53	50.66	30.77	555.59
s6669	1.2	3.53	5.88	32.63	31.61	295.38
Ratio		2.5x		4.9x		10x

Results: Cut Enumeration

leon2 (size: 278,292 4-LUTs)

Cut Size	*BddCut* (sec)	*ABC* (sec)
6	23.33	77.94
7	58.92	n/a
8	152.88	n/a
9	547.63	n/a

Results: Covering Problem Applied to Elimination

Circuit	Runtime (sec)		Area (4-LUTs)	
	***FBDD*$_{new}$**	FBDD	***FBDD*$_{new}$**	FBDD
s38417	1.9	7.2	3560	3559
s38584	3	13.7	4289	4152
s35932	3.9	4.1	3264	3360
s15850	0.8	9.1	1282	1270
b20	5.5	44.8	4514	4324
b22_1	6.2	38.4	5788	6505
systemcdes	3.1	11.3	1152	1207
vga_lcd	38.9	585.2	40680	40676
wb_conmax	18.6	104.2	19135	19479
Ratio		5.7x		1.0x

Conclusion

- Symbolic Approach To Cut Enumeration using BDDs
  - Order of magnitude faster than previous approaches
- Application of the Covering Problem to Elimination in FBDD.