

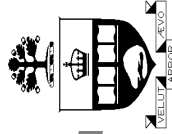
Using BDDs to Design ULMs for FPGAs

Zeljko Zilic and Zvonko Vranesic

Department of Electrical and Computer Engineering

University of Toronto

Work Supported by ITRC - Ontario



University of Toronto

Overview

Motivation

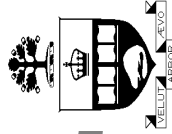
Universal Logic Modules (ULMs)

ULMs for FPGAs

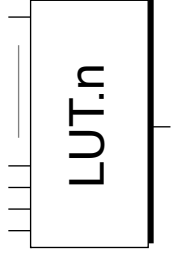
New Design Method for ULMs useful in FPGAs

Design Examples: 3-input and 4-input blocks

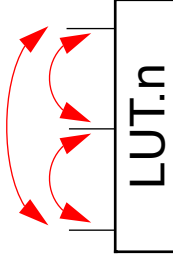
Applications



More Efficient FPGAs?

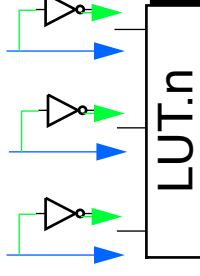


- Given LUT architecture:



- Input Permutations:

Provided by routing resources



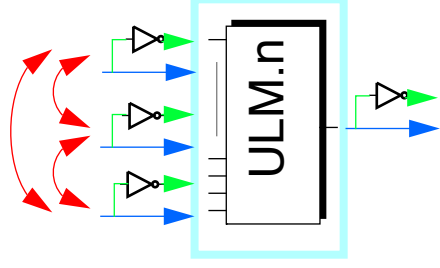
- Input Inversions:

Provided by previous blocks

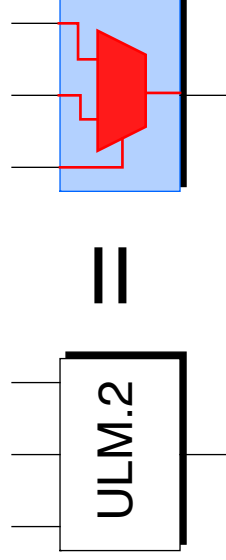
- LUT replacements that exploit permutations and inversions?

ULMS

- Logic Blocks that realize any function of n variables
- Assumed that input permutations and inversions are free

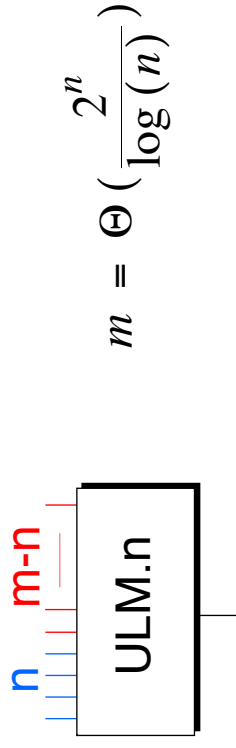


- Example: ULM.2 - used in Actel



Standard ULMs

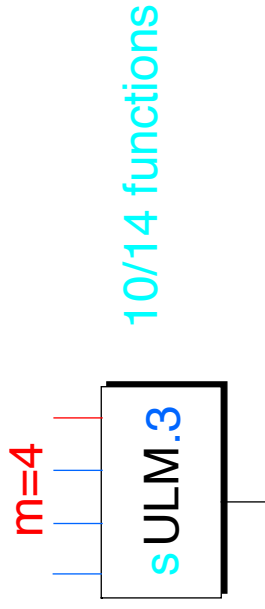
- Many pins needed:



$$m = \Theta\left(\frac{2^n}{\log(n)}\right)$$

- Attempts to use in FPGAs: incomplete functionality

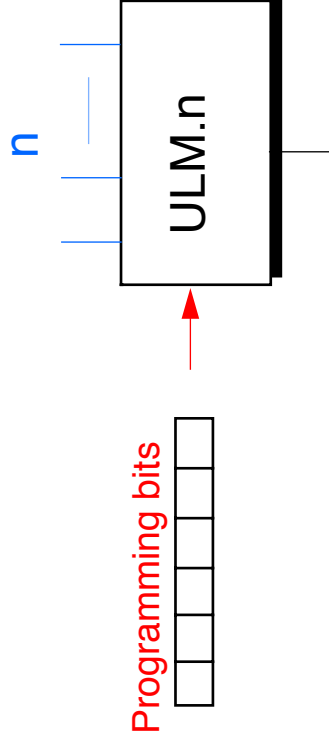
- ULM.3: [Lin, Sadowska, Gatlin 94]



- ULM.4: [Thakur, Wong 95] - $n=4$, $m=8$, 201/208 functions

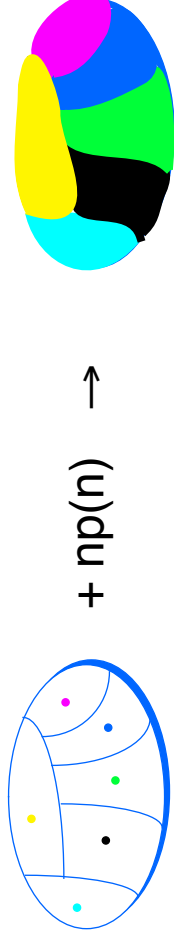
Need Different ULMs

- No additional inputs
 - Just like LUTs:
- Design procedure for these ULMs?



Design of ULMs - Idea

- Equivalence classes: n, npn, np
 - Input negation
 - Output negation
 - Input permutation
- Sufficient to realize only class representatives in ULM

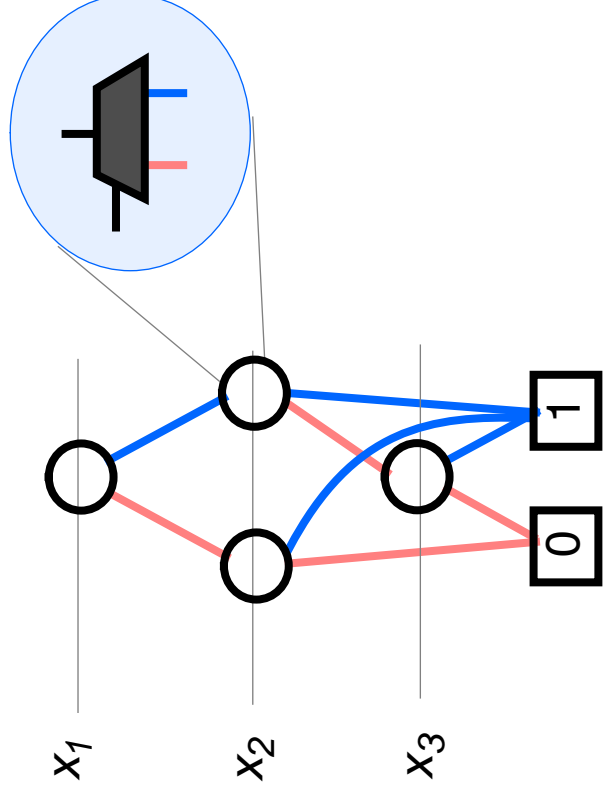


- Programming bits encode only class

n	3	4	5
n Classes	14	222	616126
Bits - n	4	8	20
Bits - np	5	9	21
Bits- LUT	8	16	32

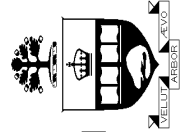
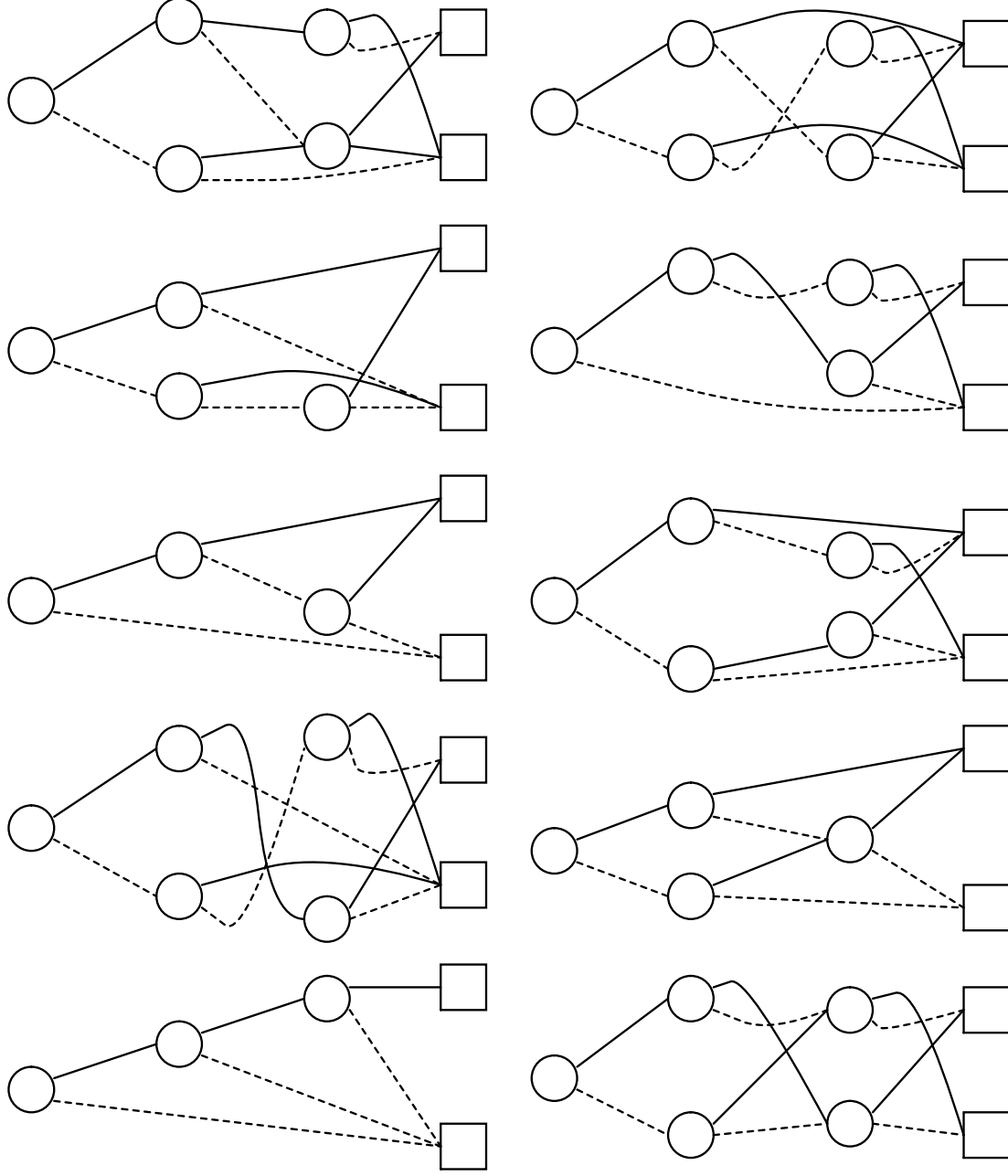
ULM Design - Using BDDs

- Want to both classify and physically realize
- Use BDDs



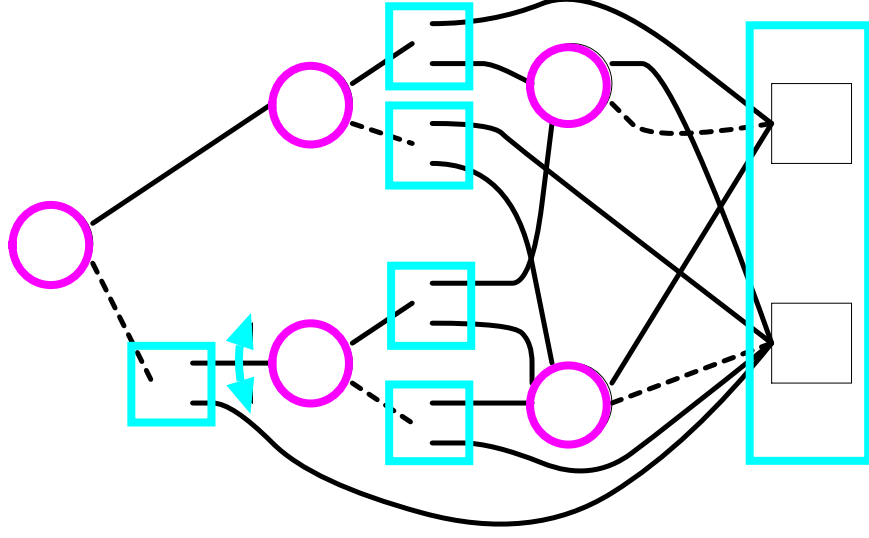
- Unique representation
- Realization: each node is a MUX:

BDDs for all classes: $n=3$



Super BDD: n=3

- SBDD.3: can realize all 10 BDDs



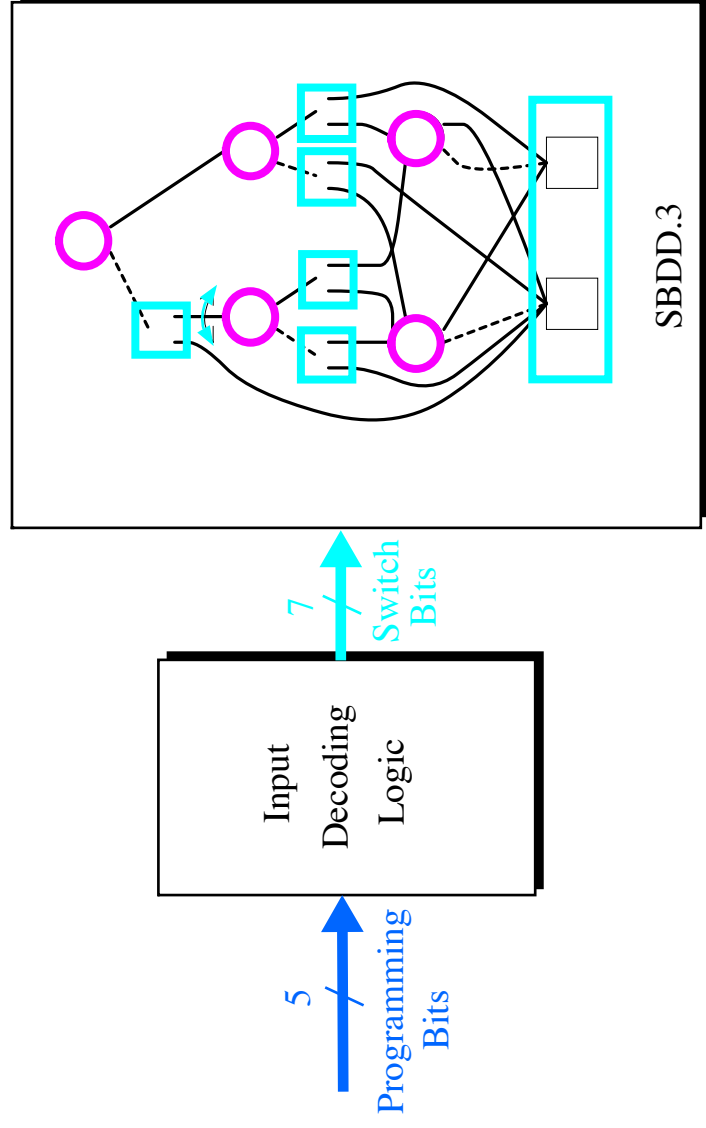
Edges optimized by
inversions, permutations

5 nodes

7 switches - 1 bit saved

Complete ULM

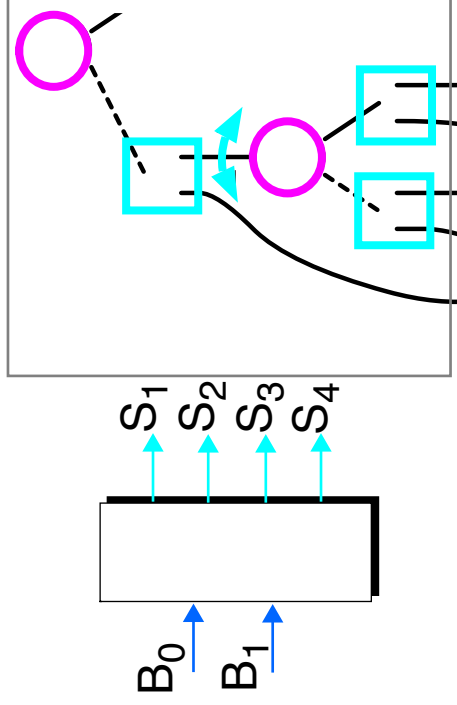
- Encode all possible input combinations



- Input Encoding - No influence on speed

Input Encoding

- First four switches: only 4 combinations → 2 bits



S ₁	S ₂	S ₃	S ₄	B ₀	B ₁
0	x	x	x	0	0
1	0	0	1	0	1
1	0	0	0	1	0
1	1	1	1	1	1

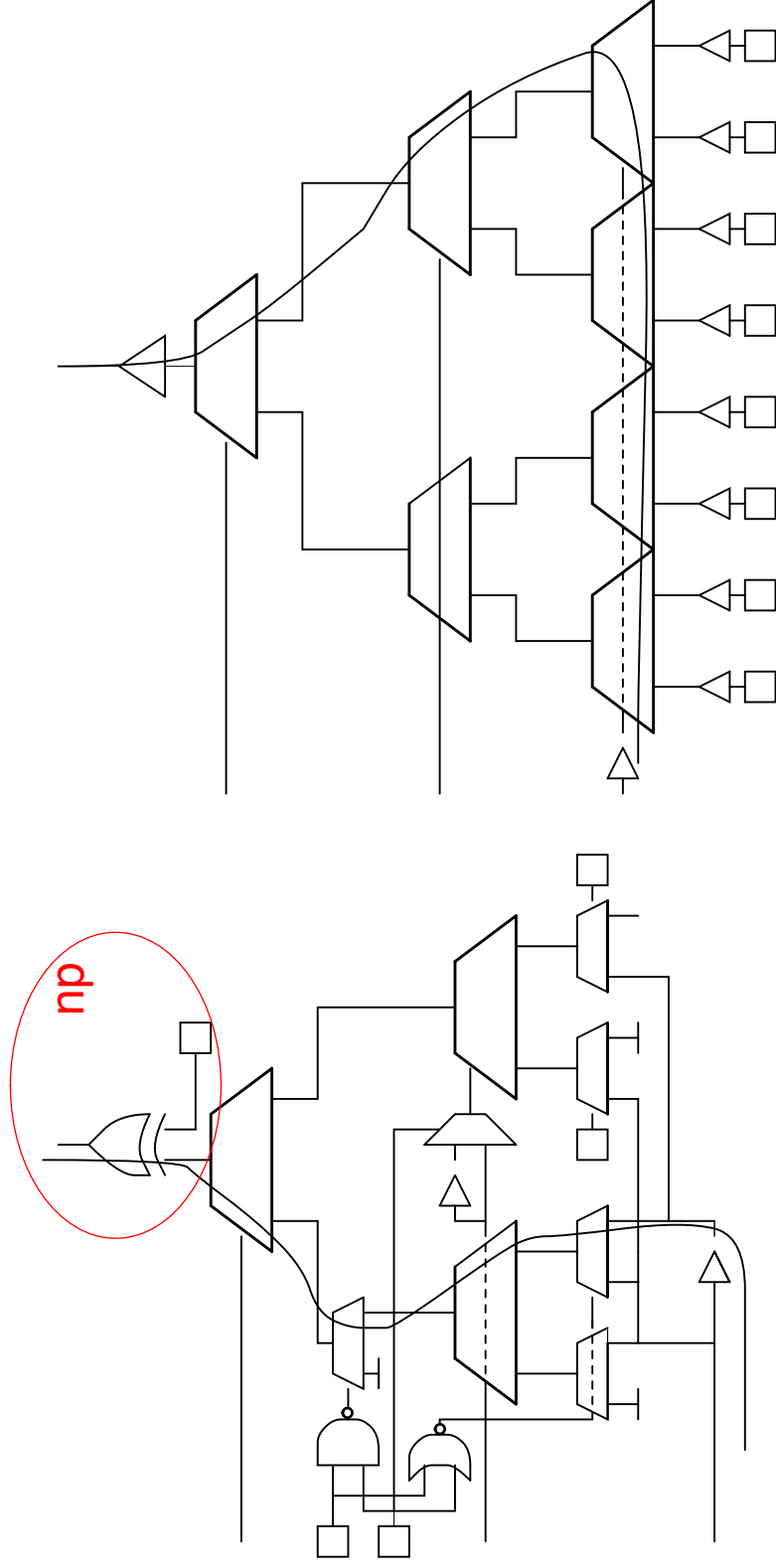
- Simple decoding logic:

$$S_1 = B_0 + B_1$$

$$S_2 = S_3 = B_0 \cdot B_1$$

$$S_4 = B_1$$

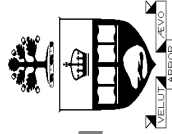
Implementation Comparison



	ULM.3	LUT.3
Memory	5 bits	8 bits
Transistors	70	78
Delay	1.31ns	1.38 ns

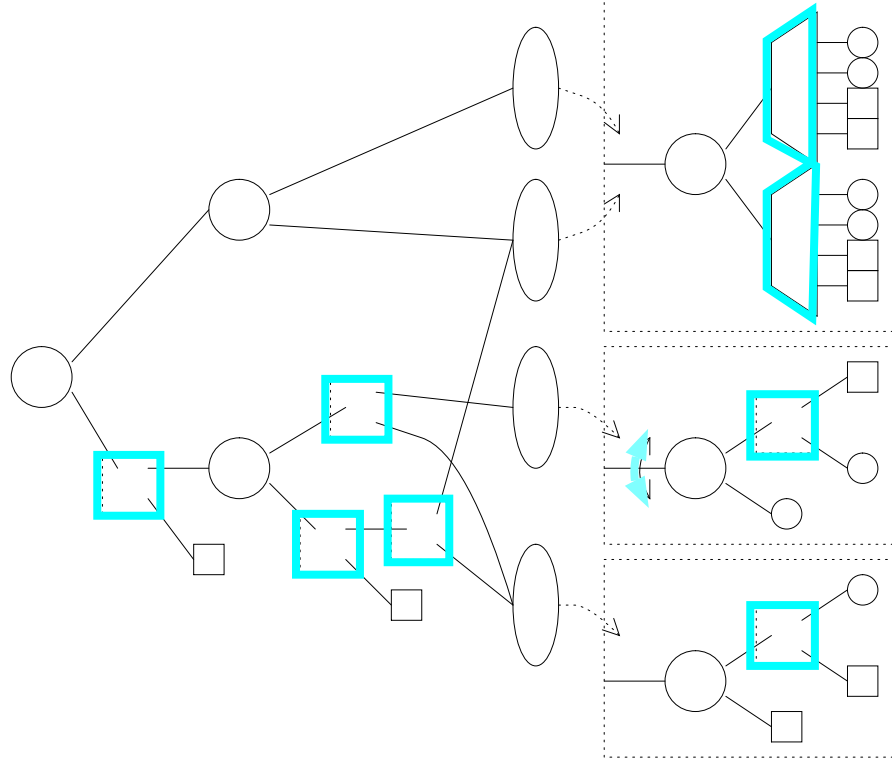
Larger Case: $n=4$

- Enumeration of classes
- Representative Realization: 208 BDDs
- Super BDD - optimize interconnection
- Minimize # switches in SBDD
- Optimize circuits by input encoding
 - Several alternative encodings



ULM.4 - LUT.4 Replacement

- Super BDD - SBDD.4



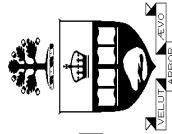
Input Encodings

- Optimal number of bits - logic too complex
- Tradeoff: area vs. number of bits
- Combine input encoding programs and decompositions

Encoding	Pre-encoding	Logic	Bits
None	none	none	16
4 Switches	2 gates	none	15
Bit Sharing	3-input decoder	10 gates	13
Dense Groups	groups decoder	126 literals	11
Group Compact	folding circuits	683 literals	9
Flat	none	719 literals	9

Technology Mapping and Applications

- **Technology Mapping**
 - Assume: np-equivalent ULM
 - Polarity disagreement - blocks need opposing polarities
 - [Lin, Sadowska 94] - 6.6% extra blocks needed for ULM.3
- **Functionally incomplete blocks**
 - Not interesting for $n=3$
 - Many possibilities - 201/208 functions with 13 bits and logic overhead of 2 gates
 - ULMs can help evaluate tradeoffs
- **Realistic architectures that can use ULMs**
 - Reconfigurable computing [Jones, Lewis]
 - Hard-wired blocks [Chung, Rose]



Conclusions

- Exploited permutation and inversion availability in FPGAs
- New kind of ULMS
 - Applicable in FPGAs
- LUT replacements that save programming bits
 - Comparable to LUTs (area, speed)
 - Require optimal or suboptimal number of programming bits
- Possible basis for new architectures

