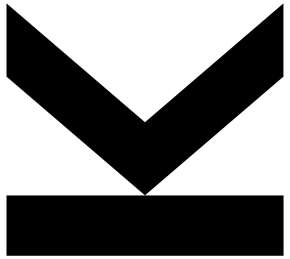
EXPLOITING CODING TECHNIQUES FOR LOGIC SYNTHESIS OF REVERSIBLE CIRCUITS

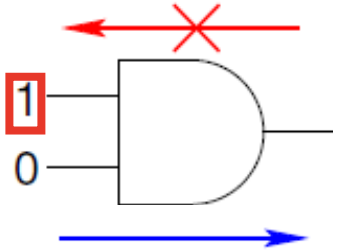


Alwin Zulehner, Robert Wille
Johannes Kepler University Linz, Austria

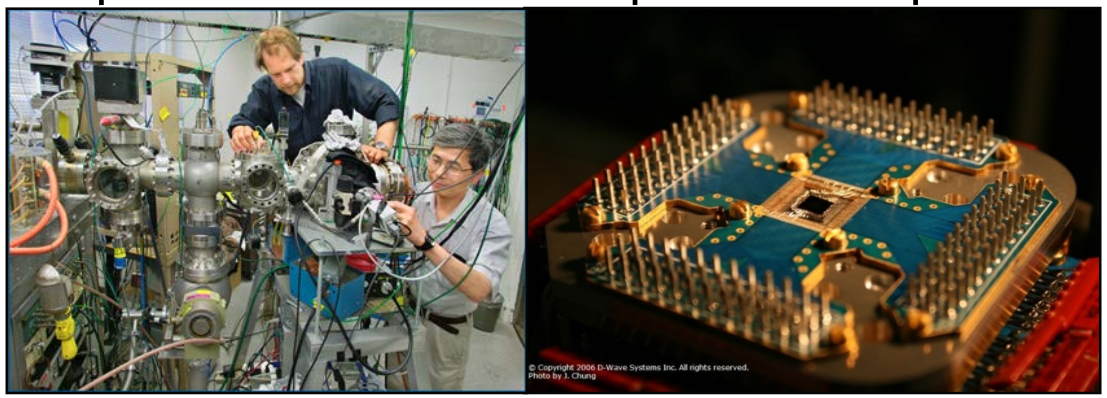
alwin.zulehner@jku.at robert.wille@jku.at

MOTIVATION: REVERSIBLE COMPUTATION

- Perform computations from inputs to outputs and vice versa
  - Not possible in conventional logic



- Required for Boolean components of quantum circuits



FUNCTIONAL SYNTHESIS FLOW

$f: B^n \rightarrow B^m$

Embedding: make output patterns distinguishable

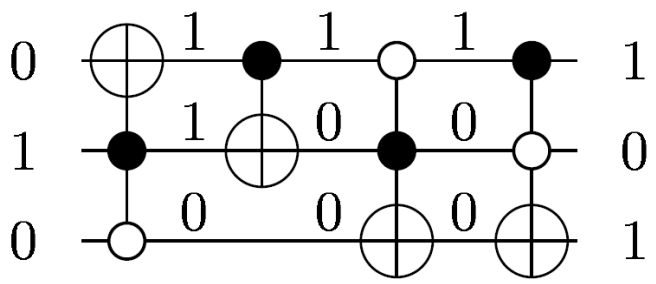
$f: B^k \rightarrow B^k$

Synthesis
Idea: $f \circ f^{-1} = I$

Reversible Circuit
(cascade of reversibe gates)

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	z	x'	y'	z'
0	**0**	**0**	**0**	0	0
0	**0**	**1**	**0**	0	1
0	**1**	**0**	**0**	1	0
0	**1**	**1**	**1**	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1



BOTTLENECK: THE EMBEDDING PROCESS

- Make output pattern distinguishable
  - Add $\log_2 \mu(p_1) = 2$ garbage outputs

x_3	x_2	x_1	x'_3	x'_2	x'_1
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	0	1

→

a_2	a_1	x_3	x_2	x_1	x'_3	x'_2	x'_1	g_2	g_1
0	0	**0**	**0**	**0**	**0**	**1**	**0**	0	0
0	0	**0**	**0**	**1**	**0**	**1**	**0**	0	1
0	0	**0**	**1**	**0**	**1**	**0**	**0**	0	0
0	0	**0**	**1**	**1**	**1**	**0**	**0**	0	1
0	0	**1**	**0**	**0**	**0**	**1**	**1**	0	0
0	0	**1**	**0**	**1**	**0**	**1**	**0**	1	0
0	0	**1**	**1**	**0**	**0**	**1**	**0**	1	1
0	0	**1**	**1**	**1**	**0**	**0**	**1**	0	0
		⋮					⋮		

i	p_i	$\mu(p_i)$
1	010	4
2	100	2
3	001	1
4	011	1

- Drawbacks:
  - More variables ➔ more complex synthesis
  - No degree of freedom in synthesis

KEY OBSERVATION AND IDEA

- Not all patterns require all $\log_2 \mu(p_1) = 2$ garbage outputs

- Use variable-length encoding
  - Frequent patterns: short code and many garbage outputs
  - Infrequent patterns: longer code but fewer garbage outptus

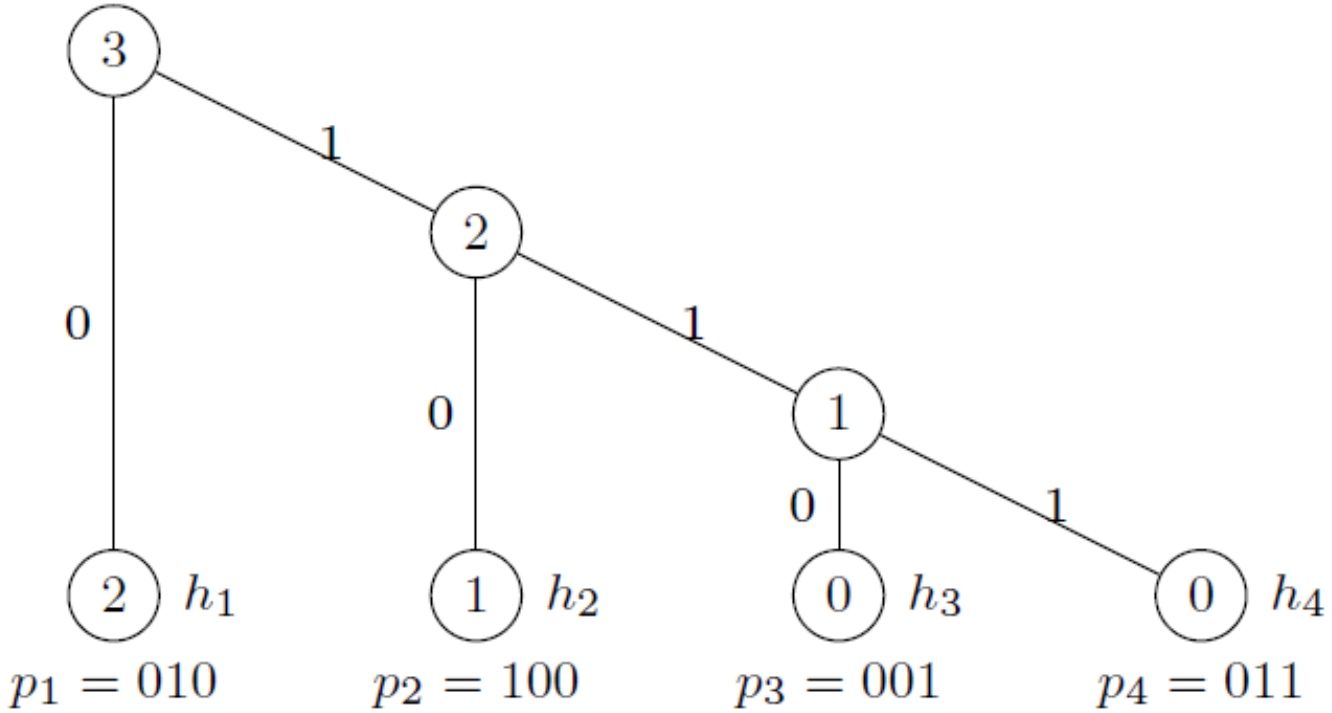
i	p_i	$\mu(p_i)$	$c(p_i)$
1	010	4	0 - -
2	100	2	1 0 -
3	001	1	1 1 0
4	011	1	1 1 1

- Synthesis with fewer variables and degree of freedom

x_3	x_2	x_1	x'_3	x'_2	x'_1
0	0	0	0	-	-
0	0	1	0	-	-
0	1	0	1	0	-
0	1	1	1	0	-
1	0	0	1	1	1
1	0	1	0	-	-
1	1	0	0	-	-
1	1	1	1	1	0

- **Note:** A decoder is required

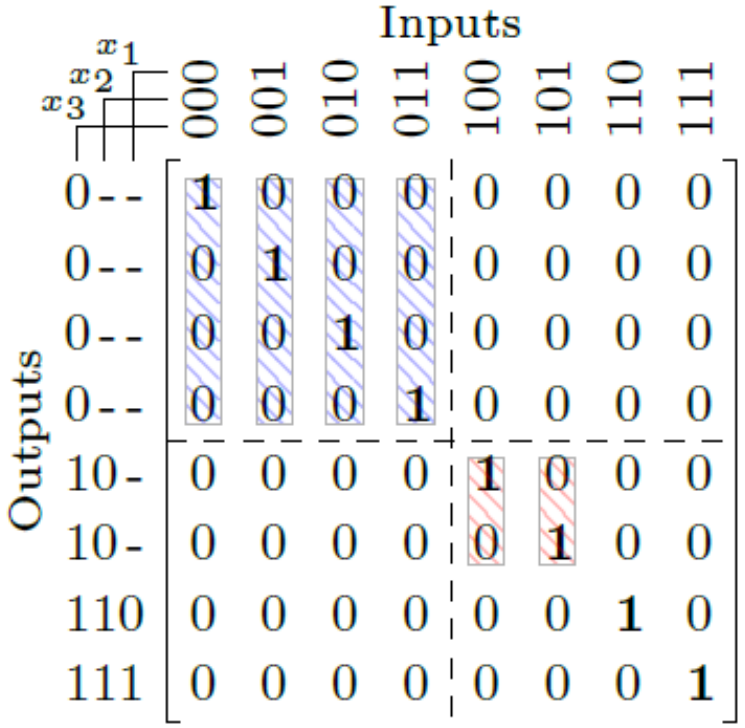
VARIABLE-LENGTH ENCODING: HUFFMAN CODE



i	p_i	$\mu(p_i)$	$c(p_i)$
1	010	4	0 - -
2	100	2	1 0 -
3	001	1	1 1 0
4	011	1	1 1 1

x_3	x_2	x_1	x'_3	x'_2	x'_1
0	0	0	0	-	-
0	0	1	0	-	-
0	1	0	1	0	-
0	1	1	1	0	-
1	0	0	1	1	1
1	0	1	0	-	-
1	1	0	0	-	-
1	1	1	1	1	0

SYNTHESIS OF THE ENCODED FUNCTION



Outputs \ Inputs ($x_1 x_2 x_3$)	000	001	010	011	100	101	110	111
0--	**1**	0	0	0	0	0	0	0
0--	0	**1**	0	0	0	0	0	0
0--	0	0	**1**	0	0	0	0	0
0--	0	0	0	**1**	0	0	0	0
10-	0	0	0	0	**1**	0	0	0
10-	0	0	0	0	0	**1**	0	0
110	0	0	0	0	0	0	**1**	0
111	0	0	0	0	0	0	0	**1**

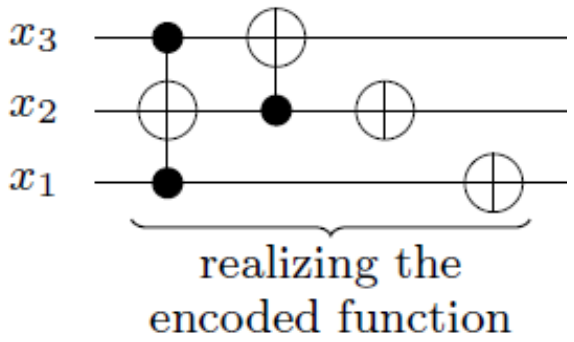
- Use a permutation matrix
  - Model degree of freedom
- Transform to identity variable-wise
  - Swap columns
- Exploit degree of freedom
  - Fewer control lines required

x_3	x_2	x_1	x'_3	x'_2	x'_1
0	0	0	0	-	-
0	0	1	0	-	-
0	1	0	1	0	-
0	1	1	1	0	-
1	0	0	1	1	1
1	0	1	0	-	-
1	1	0	0	-	-
1	1	1	1	1	0

DECODE THE FUNCTION

i	p_i	$\mu(p_i)$	$c(p_i)$
1	010	4	0 - -
2	100	2	1 0 -
3	001	1	1 1 0
4	011	1	1 1 1

- Circuit has $\log_2 \mu(p_1) = 2$ garbage outputs
- Easy for majority of the decoder
- Use synthesis for remaining outputs



COMPARISON TO THE STATE OF THE ART

- Comparison to the state of the art
  - Symbolic TBS
  - QMDD-based synthesis
- Coded function is more compact
- Much more scalable
  - Magnitudes fewer runtime
- Magnitudes fewer cost
  - 66.3% and 92.6% on average

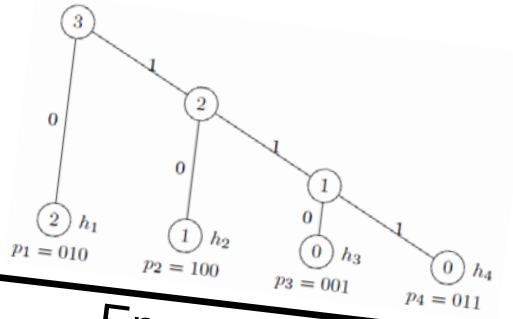
				TBS		QMDD		Proposed		
Name	n	m	l	t	T-depth	t	T-depth	l_c	t	T-depth
9symml	9	1	10	2.02	99 381	0.10	196 764	10	0.07	7 320
dk27	9	9	15	3.86	123 276	0.89	2 409 495	**10**	0.17	48 405
x2	10	7	16	25.10	391 404	1.98	4 516 011	**11**	0.13	21 075
alu3	10	8	14	19.75	337 281	2.08	3 368 610	**11**	2.51	533 685
dk17	10	11	19	56.66	492 033	17.52	37 365 105	**11**	0.94	258 510
apla	10	12	22	199.15	604 542	41.97	77 151 615	**11**	1.00	87 336
co14	14	1	15	TO	–	0.04	26 544	15	0.01	3 360
alu4	14	8	19	TO	–	331.85	324 374 364	**15**	70.39	11 027 733
cu	14	11	25	TO	–	TO	–	**15**	0.63	76 311
table3	14	14	28	TO	–	TO	–	**15**	6.93	463 260
s1488	14	25	38	TO	–	TO	–	**15**	197.74	9 553 668
in0	15	11	25	TO	–	TO	–	**16**	81.27	11 725 497
cm163a	16	13	25	TO	–	TO	–	**17**	708.99	80 405 748
pdc	16	40	55	TO	–	TO	–	**17**	3004.29	10 401 426
spla	16	46	61	TO	–	TO	–	**17**	2488.81	13 852 266
table5	17	15	32	TO	–	TO	–	**18**	77.55	10 065 483
mux	21	1	22	TO	–	TO	–	22	0.48	7 056
cordic	23	2	25	TO	–	TO	–	**24**	1028.91	17 630 250
e64	65	65	129	TO	–	TO	–	**65**	4.84	95 202

CONCLUSIONS

x_3	x_2	x_1	x'_3	x'_2	x'_1
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	0	1

Embedding

Encoding



a_2	a_1	x_3	x_2	x_1	x'_3	x'_2	x'_1	g_2	g_1
0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	1	0	0	1
0	0	0	1	0	1	0	0	0	0
0	0	0	1	1	1	0	0	0	1
0	0	1	0	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0	1	1
0	0	1	1	1	0	0	1	0	0
		⋮				⋮			

Synthesis

Exploiting coding techinques:
Better scalability and significantly smaller circuits

x_3	x_2	x_1	x'_3	x'_2	x'_1
0	0	0	0	-	-
0	0	1	0	-	-
0	1	0	1	0	-
0	1	1	1	0	-
1	0	0	1	1	1
1	0	1	0	-	-
1	1	0	0	-	-
1	1	1	1	1	0

Synthesis