

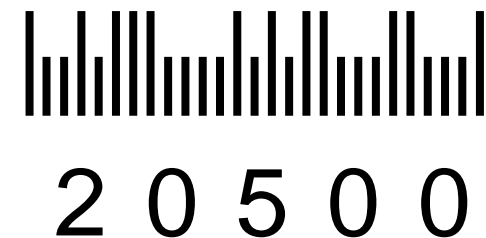
M-of-N Code Decomposition for Indicating Combinational Logic

Will Toms
School of Computer Science
University of Manchester

M-of-N Codes

- ⌚ Each code word has a fixed size (n) and weight (m)
- ⌚ Small codes concatenated together to form large (reduced) codes
- ⌚ Code words are *unordered*
 - ⌚ no code-word contained in any other

Digit	Code
0	llll
1	llll
2	llll
3	llll
4	llll
5	llll
6	llll
7	llll
8	llll
9	llll



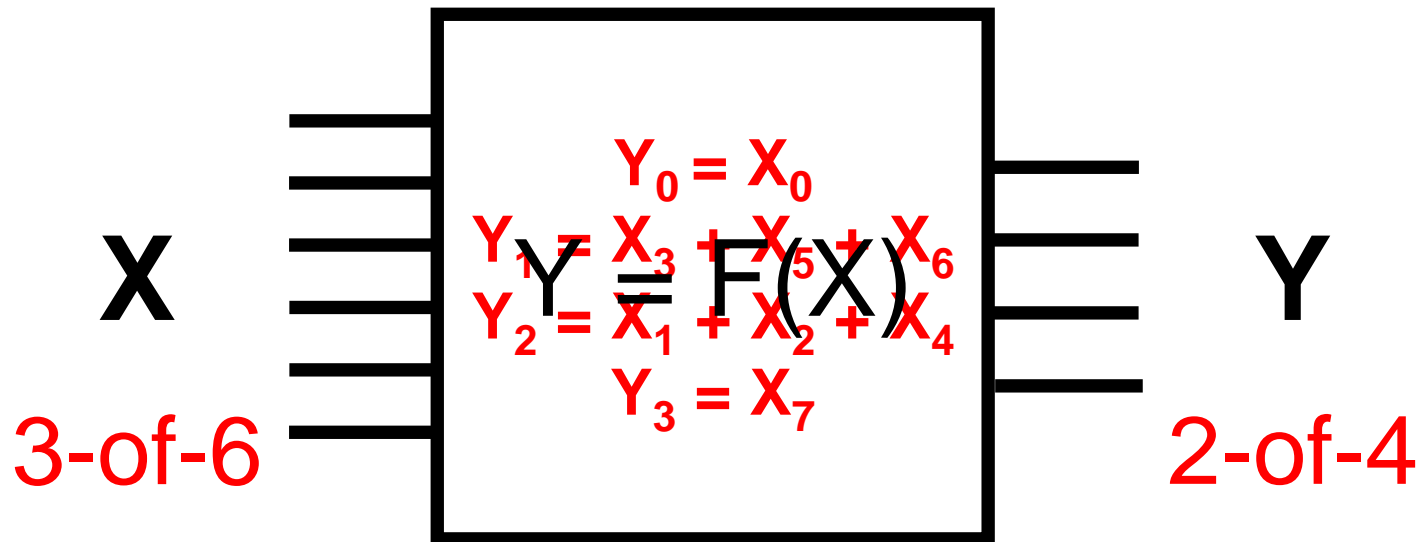
M-of-N Codes

- ⌚ In order to preserve code values all wires return-to-zero between data transmissions
- ⌚ Most commonly used codes for combinational-logic 1-of-2 (dual-rail) and 1-of-4
- ⌚ Codes with $m > 2$ lower energy and fewer wires
- ⌚ This paper presents a method to construct implementations for any m-of-n encoded function block

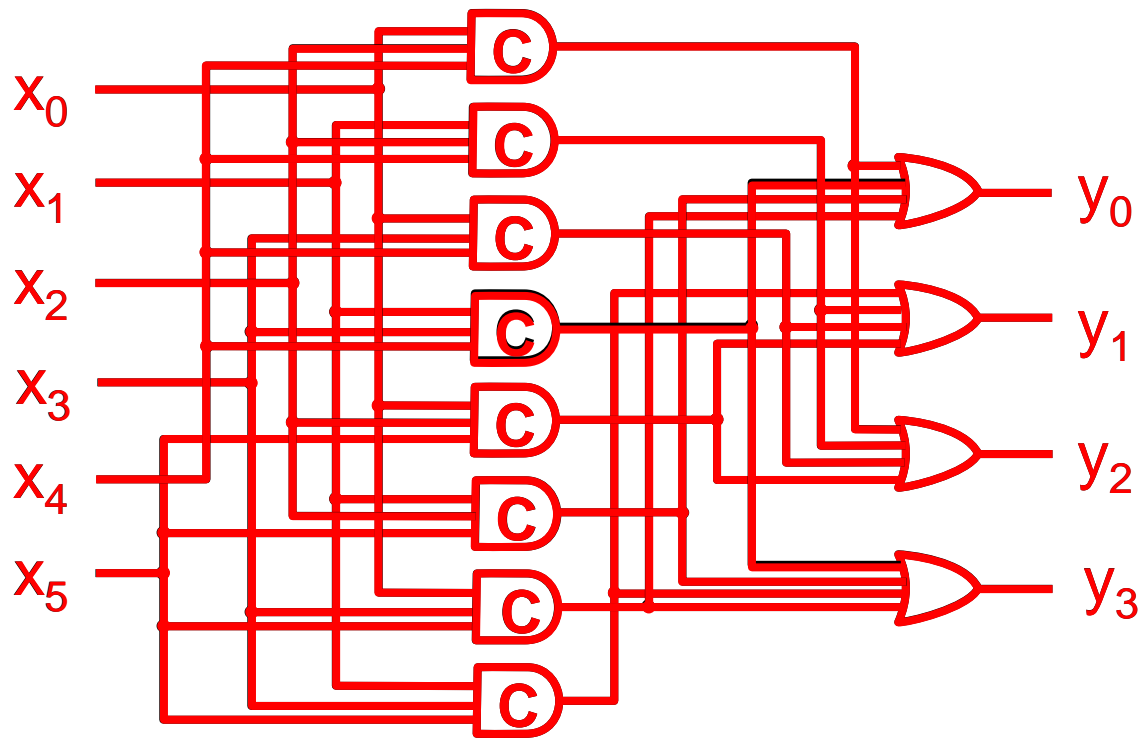
Combinational Logic

- ⌚ Combinational Logic maps one m-of-n code to another

Dual-Rail Full Adder

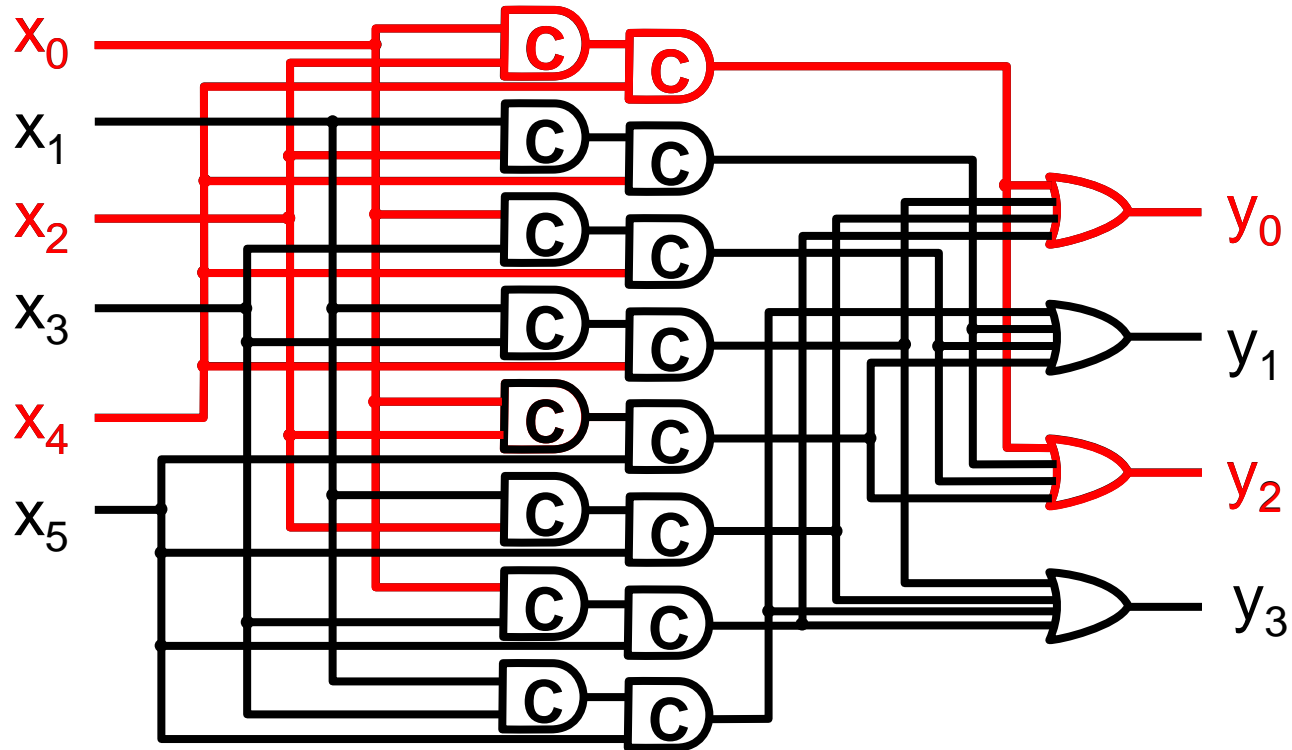


Combinational Logic



	X_0	X_1	X_2	X_3	X_4	X_5	y_0	y_1	y_2	y_3	
X_0	1	0	1	0	1	0	1	0	1	0	Y_0
X_1	0	1	1	0	1	0	0	1	1	0	Y_2
X_2	1	0	0	1	1	0	0	1	1	0	Y_2
X_3	0	1	0	1	1	0	1	0	0	1	Y_1
X_4	1	0	1	0	0	1	0	1	1	0	Y_2
X_5	0	1	1	0	0	1	1	0	0	1	Y_1
X_6	1	0	0	1	0	1	1	0	0	1	Y_1
X_7	0	1	0	1	0	1	0	1	0	1	Y_3

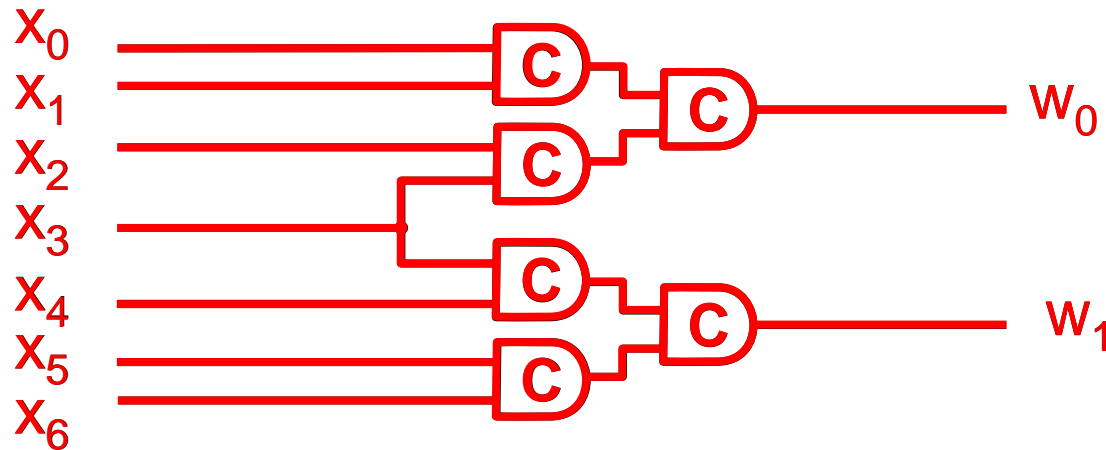
Combinational Logic



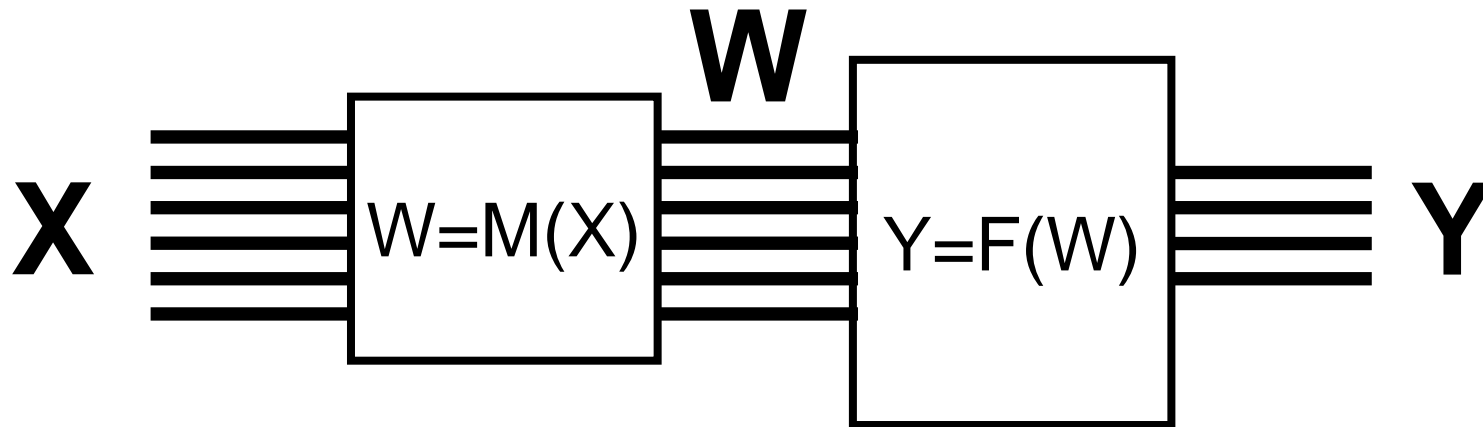
🕒 Each code-word must be implemented in one C-element

Decomposition

- ⌚ If code-words share at most 1 input C-elements can be decomposed arbitrarily



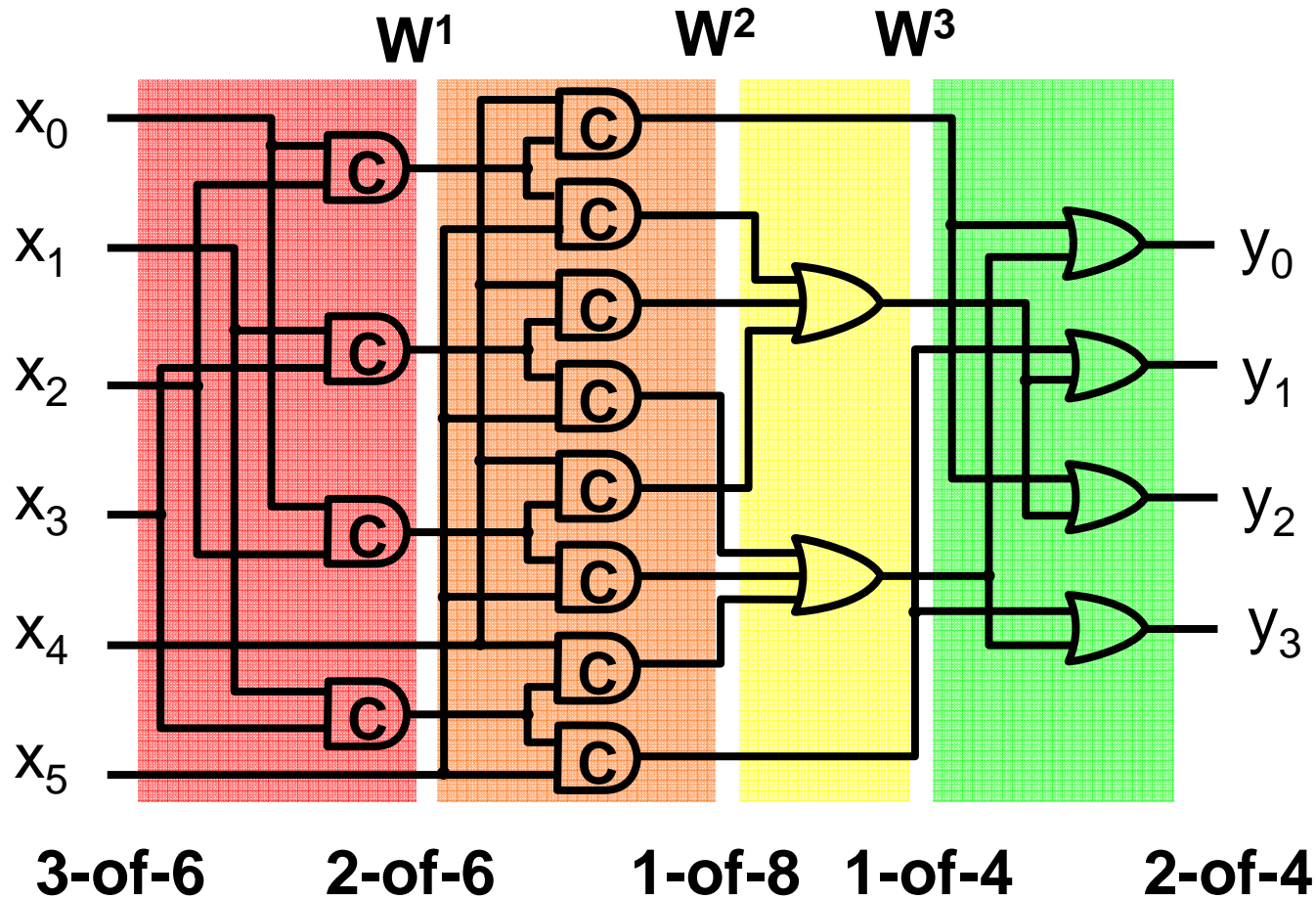
Re-encoding Input Codes



- ⌚ Create new encoding W that can be decomposed
- ⌚ Remap original functions to new encoding

8

Re-encoding Input Codes

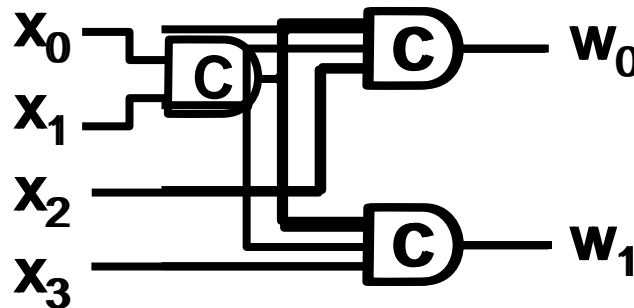


Re-encoding Input Codes

- ⌚ Can always construct 1-hot encoding for a code
 - ⌚ DIMS
- ⌚ With concatenated 1-of- n codes can combine code-groups to form implementable codes
 - ⌚ Fant's product terms
- ⌚ If using m -of- n codes have to decompose code groups

Decomposing M-of-N Codes

- ⌚ To reduce the number of shared variables in code-words
replace common variables (*cubes*) with new variables
 - ⌚ Each code-word contains a *differential* not in the other

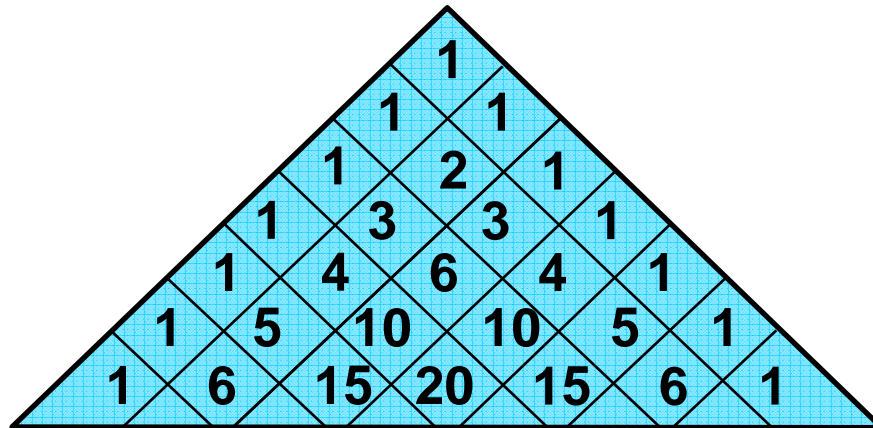


- ⌚ Need to find a set of cubes that reduces the sharing
between code-words

Binomial Co-efficients

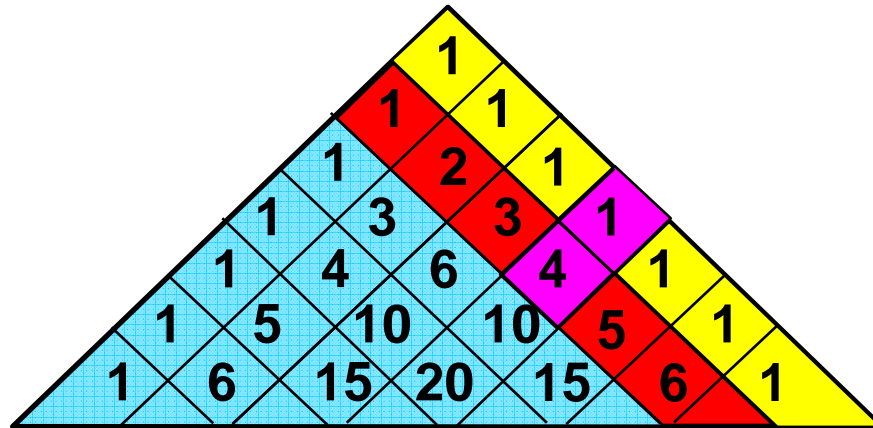
⌚ The number of code-words in an m-of-n code is:

$$C_m^n = \frac{n!}{m!(n-m)!}$$

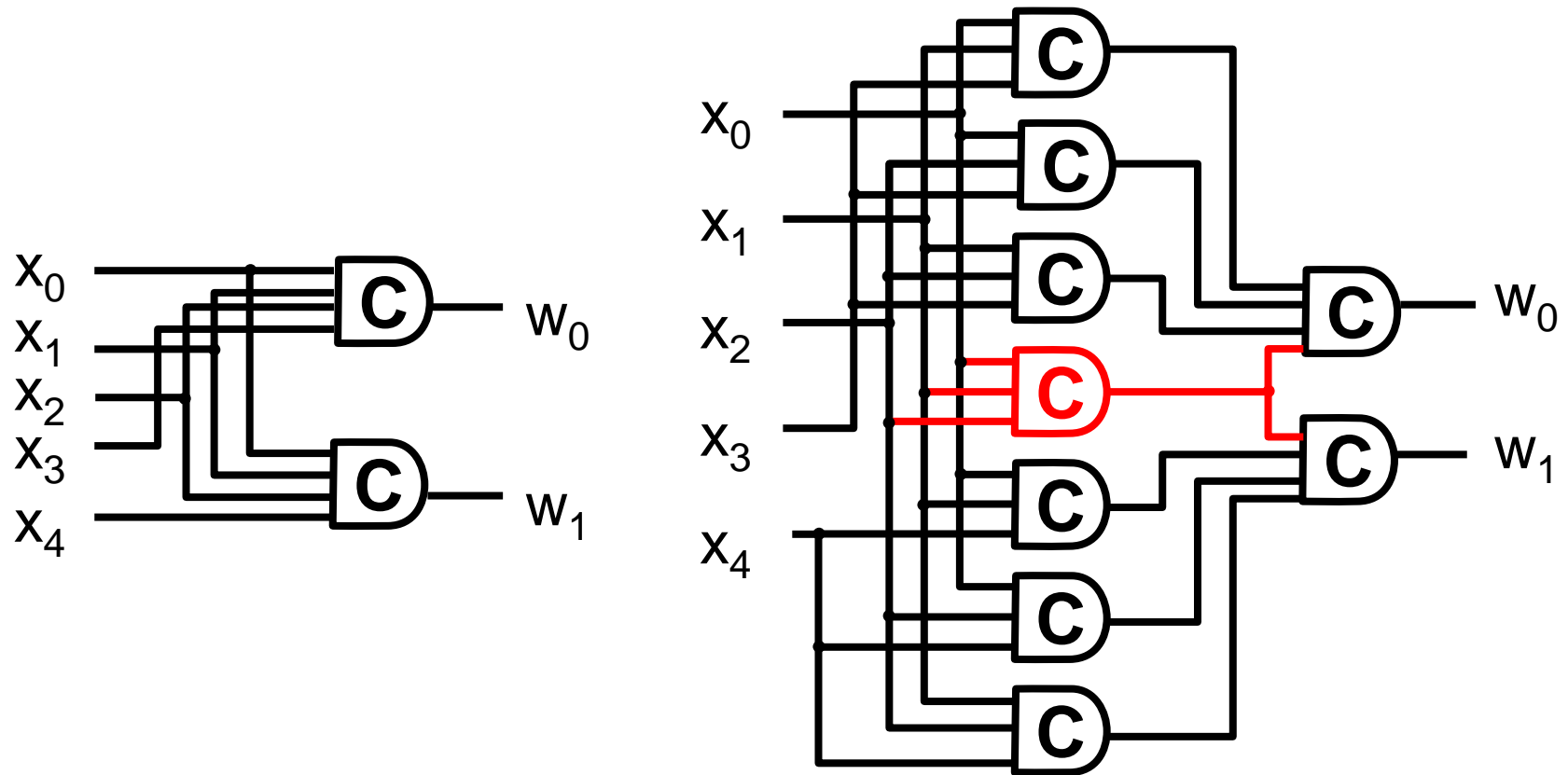


Decomposing M-of-N Codes

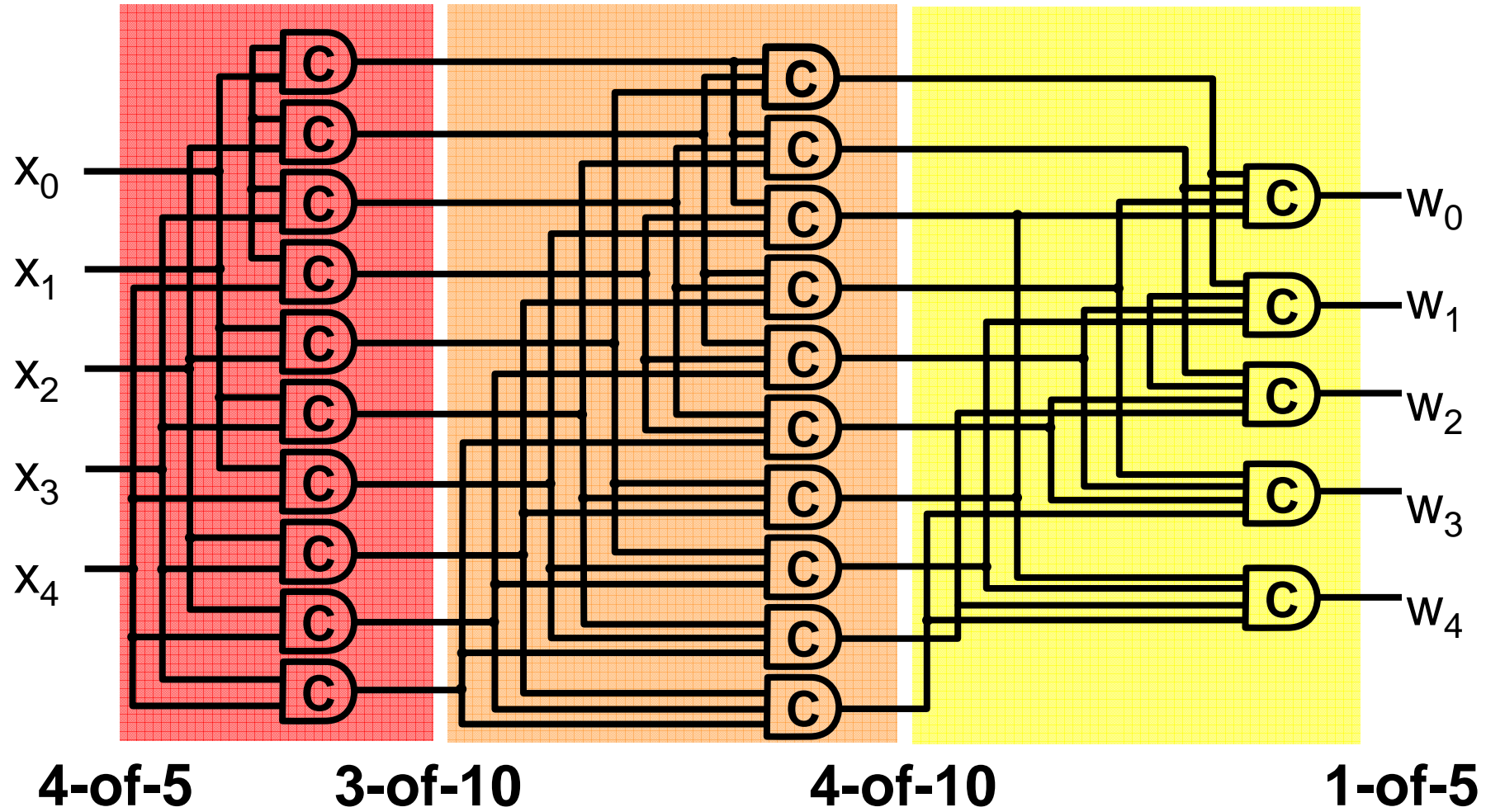
- ⌚ Any m -of- n code can be implemented by constructing W from the set of $m-1$ width sub-cubes of all code words.
- ⌚ Each code word has m sub-cubes of width $m-1$
- ⌚ Each Code words can share at most $m-1$ variables
- ⌚ No code-word can share more than 1 sub-cube



Decomposing M-of-N Codes



Decomposing M-of-N Codes

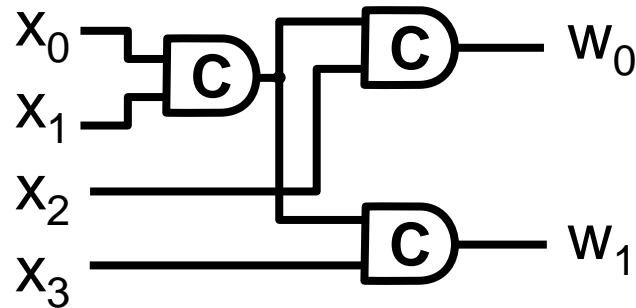


Reduced M-of-N Codes

- ⌚ Can implement any m -of- n encoded function-block
 - ⌚ Expensive
- ⌚ If input is not a full m -of- n code the size of the new encoding can be reduced
- ⌚ *Want to select a subset of cubes that form an unordered code*

Reduced M-of-N Codes

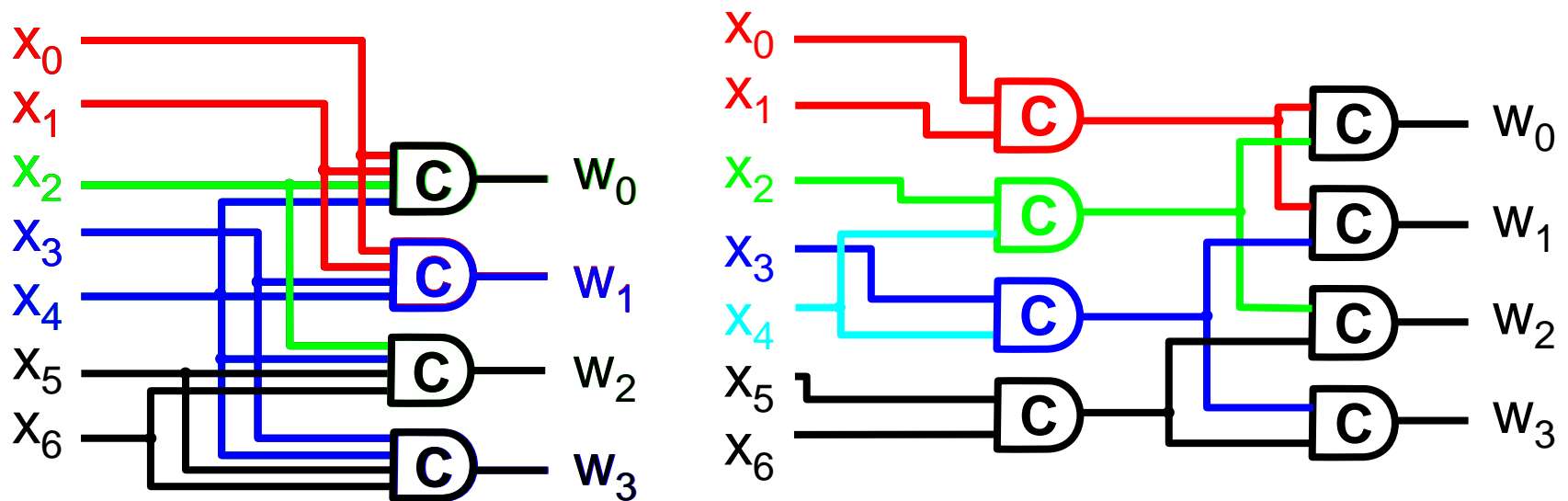
- ⌚ Need to use common cubes to reduce shared literals



- ⌚ ***For each common cube must include two further cubes as differentials***

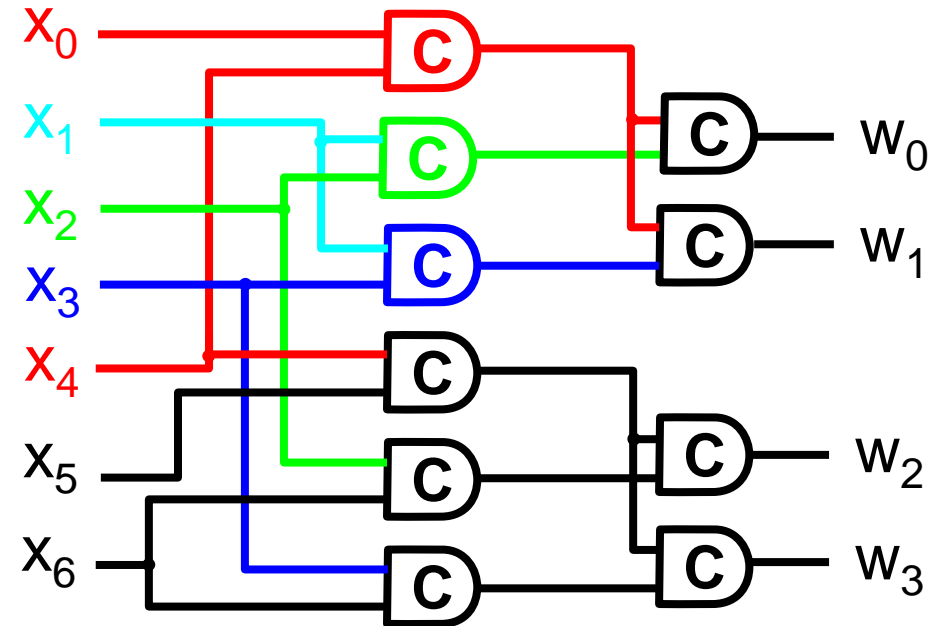
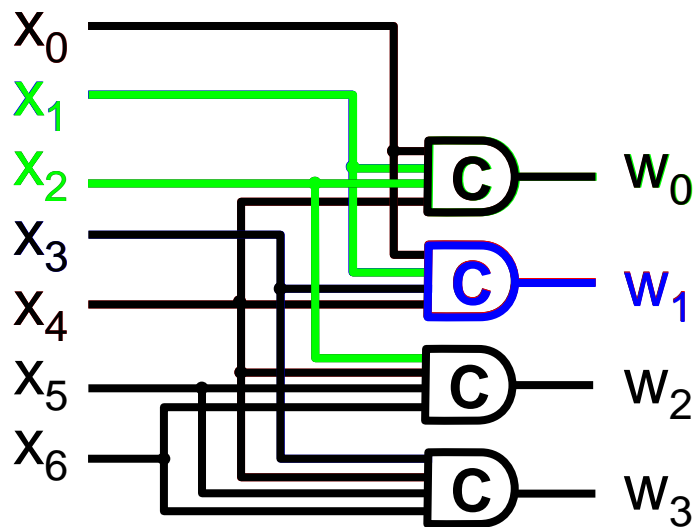
Reduced M-of-N Codes

- ⌚ **Can reduce the number of cubes by selecting common cubes whose differentials are also shared**



Reduced M-of-N Codes

- ⌚ **Can reduce the number of cubes by selecting common cubes whose differentials are also shared**



Kernel Extraction

- ⌚ Kernels are cube-free cores of expressions

$$f = abcg + abch$$

$$k(f) = g + h$$

- ⌚ Two expressions share a *multiple-cube divisor* if their kernels share two or more cubes:

$$g = defg + defh$$

$$k(g) = g + h$$

Co-kernel Matrix

	a_0	a_1	a_2	a_3	b_0	b_1	a_0b_0	a_1b_0	a_2b_0	a_3b_0	a_0b_1	a_1b_1	a_2b_1	a_3b_1	a_0a_1	a_0a_2	a_1a_2	a_0a_3	a_1a_3	a_2a_3	
a_0							0	1	3		6	7	9								
a_1							0		2	4	6		8	10							
a_2							1	2		5	7	8		11							
a_3							3	4	5		9	10	11								
b_0															0	1	2	3	4	5	
b_1															6	7	8	9	10	11	
a_0b_0			0	1	3																
a_1b_0		0		2	4																
a_2b_0		1	2		5																
a_3b_0		3	4	5																	
a_0b_1			6	7	9																
a_1b_1		6		8	10																
a_2b_1		7	8		11																
a_3b_1		9	10	11																	
a_0a_1					0	6															
a_0a_2					1	7															
a_1a_2					2	8															
a_0a_3					3	9															
a_1a_3					4	10															
a_2a_3					5	11															

Intersecting Rectangles

🕒 Distributed:

	b_0	b_1	a_0a_1	a_0a_2	a_1a_2	a_0a_3	a_1a_3	a_2a_3
b_0			0	1	2	3	4	5
b_1			6	7	8	9	10	11
a_0a_1	0	6						
a_0a_2	1	7						
a_1a_2	2	8						
a_0a_3	3	9						
a_1a_3	4	10						
a_2a_3	5	11						

🕒 Transposed rectangle contains differentials

Intersecting Rectangles

🕒 Combined:

	a_0b_0	a_1b_0	a_2b_0	a_3b_0	a_0b_1	a_1b_1	a_2b_1	a_3b_1
a_0		0	1	3		6	7	9
a_1	0		2	4	6		8	10
a_2	1	2		5	7	8		11
a_3	3	4	5		9	10	11	

🕒 Differentials contained within rectangle

🕒 Decomposes code group

Reduced Divisor Sets

- ⌚ Cost functions can evaluate the cost of various encodings
- ⌚ A few checks are necessary to determine whether encodings are unordered
- ⌚ Potential encodings can be determined based on structure of function block as well as input encoding

Results

Cluster	Inputs	Input Encoding	Outputs	Output Encoding	DIMS	Decomp	Divisor Sets
261	6	1of2	5	1of2	1014	216	7
	6	1of4	5	1of4	566	147	4
	6	2of7	5	2of7	567	489	2
	6	3of6	5	3of6	760	657	3
	6	4of8	5	3of7	633	843	3
156	8	1of2	5	1of2	4598	294	12
	8	1of4	5	1of4	2550	206	8
	8	2of7	5	2of7	2551	266	5
	8	3of6	5	3of6	3576	584	6
	8	4of11	5	3of7	2553	2573	3
174	9	1of2	5	1of2	9718	340	13
	9	1of4	5	1of4	5622	227	8
	9	2of7	5	2of7	5623	426	10
	9	3of6	5	3of6	7672	1430	5
	9	5of12	5	3of7	7672	1430	5
106	10	1of2	6	1of2	22516	1584	12
	10	1of4	6	1of4	11252	666	5
	10	2of7	6	2of7	11253	6921	4
	10	3of6	6	3of6	15350	1625	7
	10	6of13	6	4of8	18424	7610	5
102	11	1of2	8	1of2	55280	2920	14
	11	1of4	8	1of4	28656	1342	9
	11	2of7	8	2of7	28658	2212	6
	11	3of6	8	3of6	40948	3550	22
	11	6of14	8	4of11	28661	35009	5

Summary

- ⌚ Defined Conditions for implementing indicating combinational logic
- ⌚ Presented a method that can implement **any** m-of-n encoded function-block by re-encoding inputs
- ⌚ Developed efficient algorithms to enumerate and quantify potential encodings