



DEPARTMENT OF COMPUTER SCIENCE  
UNIVERSITY OF COPENHAGEN



DIKU

# Parallel Optimization of a Reversible (Quantum) Ripple-Carry Adder

Michael Kirkedal Thomsen

Holger Bock Axelsen

Department of Computer Science, DIKU,  
University of Copenhagen

UC 2008

# Overview

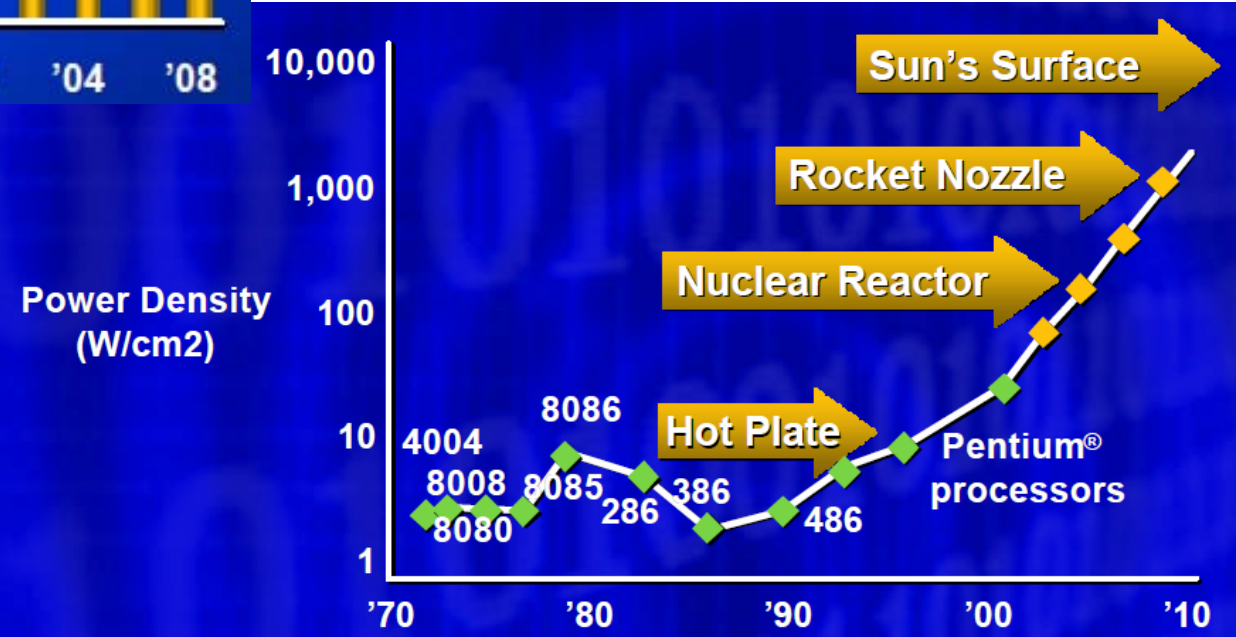


- The end of Moore's Law!
- Reversible logic, gates and circuits
- Binary adders
- Parallelization of addition
- The ripple-block carry adder



# The End of Moore's Law!

## Power Too High



[Gelsinger, Intel '01]

# Reversible Computing



The erasure of information leads to dissipation of heat.

[Landauer '61]

Erasure of information is not necessary for computation.

[Bennett '73]

Reversible computing is computing without any information loss.

It is possible to make universal reversible logic gates.

[Toffoli, Fredkin '80-'82]



# Conventional Logic

- Computation today depends on logic operations that **destroy** information.

Example: AND gate

In		Out
A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1



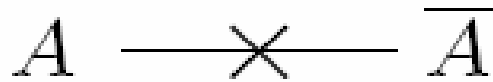
# Reversible Logic Gates

Solution: Avoid information loss.

1. The number of input lines is equal to the number of output lines (written  $n \times n$ ).
2. Its Boolean function  $B^n \rightarrow B^n$  is bijective.



# Not Gate

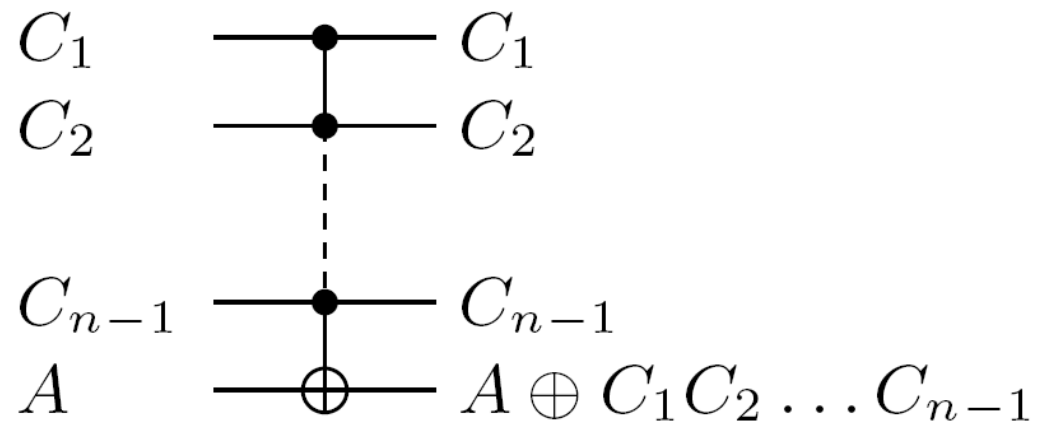


In	Out
A	$\neg A$
0	1
1	0

- **Simplest** reversible gate
- Only classical logic gate useable in reversible circuits



# n-bit Controlled-Not Gate



[Toffoli '80]

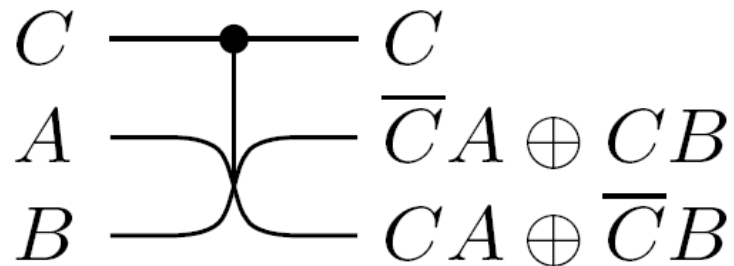
- 3-bit: Toffoli gate
- 2-bit: Feynman gate

In			Out		
$C_1$	$C_2$	$A$	$C_1$	$C_2$	$R_A$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0





# Controlled-Swap (Fredkin) Gate



[Fredkin, Toffoli '82]

- Can be generalized to a n-bit controlled-swap gate

In			Out		
C	A	B	C	$R_A$	$R_B$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1



# Diagram Shorthand

Shorthand:

Negated controls are shown with an open circle



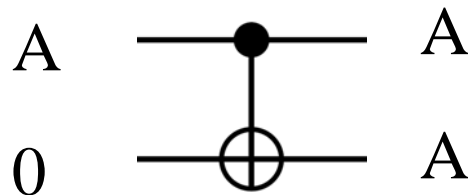


# Reversible Circuits

Important restriction:

- Fan-out is **not** permitted – it is an irreversible construction

Simulate fan-out through Feynman-gate






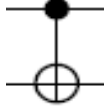
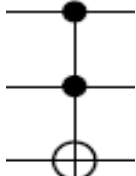

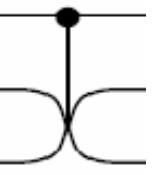
# Reversible Circuit Implementations

- **Semi conductor (MOS)**
  - Complementary (dual-line) pass-transistor logic (CPL)  
[De Vos et al.]
  - Split-phase charge-recovery logic  
[Vieri, Frank]
  - Y-branch Switching  
[Palm et al., Forsberg]
- **Quantum computing**
  - Quantum optical, Solid state, Nuclear magnetic resonance systems  
[Cirac, Duan, Zoller, ect.]



# Cost Metrics in CPL

- Transistor cost ~ “Space”
- Circuit delay ~ “Time”
- **Garbage bit**
  - Non-constant output that is not part of the desired result
- **Ancilla bit**
  - Bit that is constant at both input and output
- **Ancillae** is preferable to **garbage** due to reuse

Gate	Trans.	Delay
	0	0
	8	1
	16	1
	$8(n-1)$	1
	16	1

[De Vos, Rentergem '03]



# Addition

$$2 + 3 \rightarrow 5$$

$$1 + 4 \rightarrow 5$$

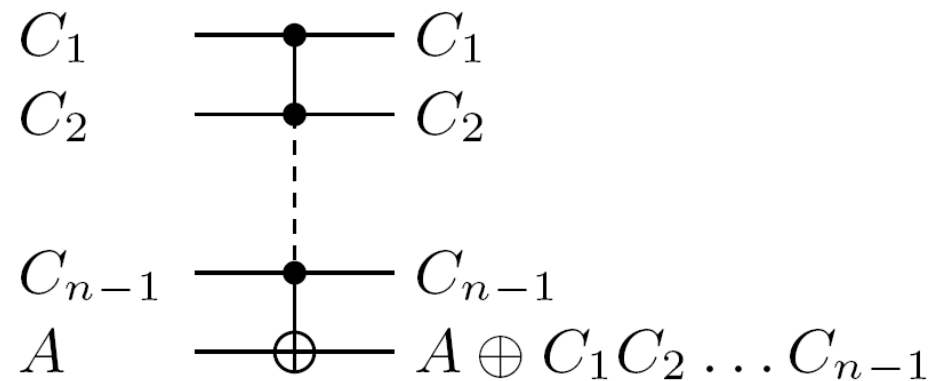
$$\underline{5 \rightarrow ? + ?}$$

- Addition is **ir**reversible
- We need a reversible way to add numbers



# Reversible Updates

- Updating a value **B** with a value that does *not* depend on **B**



*Example: The n-bit controlled-not gate*

- Addition:

$$(A, B) \mapsto (A, B + A \bmod 2^n)$$



# Binary Full-Adder

- Conventional binary addition is **not** reversible
- We need a new adder construction

$$S_i = C_i \oplus A_i \oplus B_i$$
$$C_{i+1} = C_i(A_i \oplus B_i) \oplus A_i B_i .$$

In			Out		
$A_i$	$B_i$	$C_i$	$A_i$	$S_i$	$C_{i+1}$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	1	1





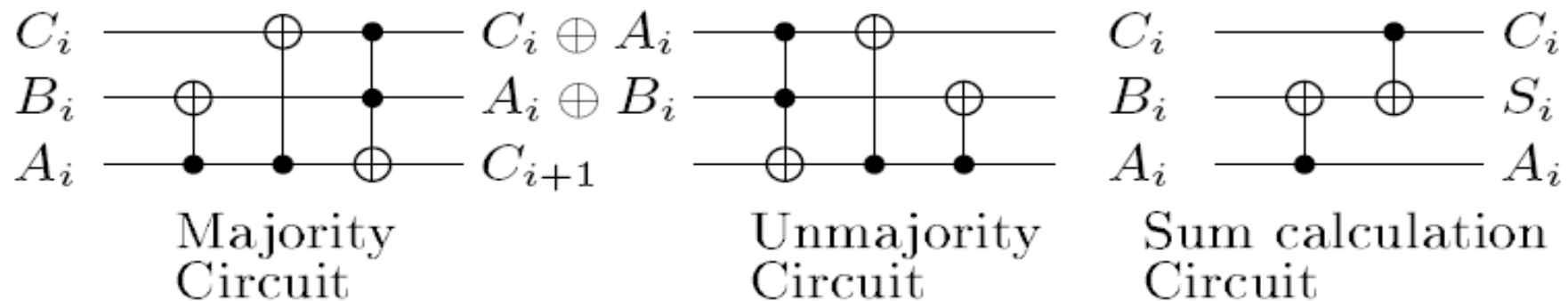
# CDKM Adder Circuit

$$S_i = C_i \oplus A_i \oplus B_i$$

$$C_{i+1} = C_i(A_i \oplus B_i) \oplus A_i B_i .$$

- Sum and carry can be calculated **independently**
- Sum is **not needed** for the sum of next bits

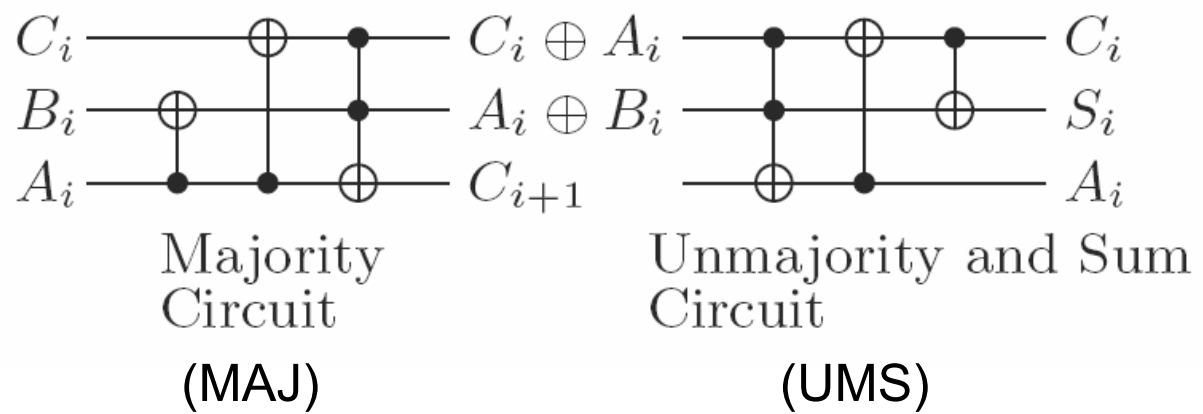
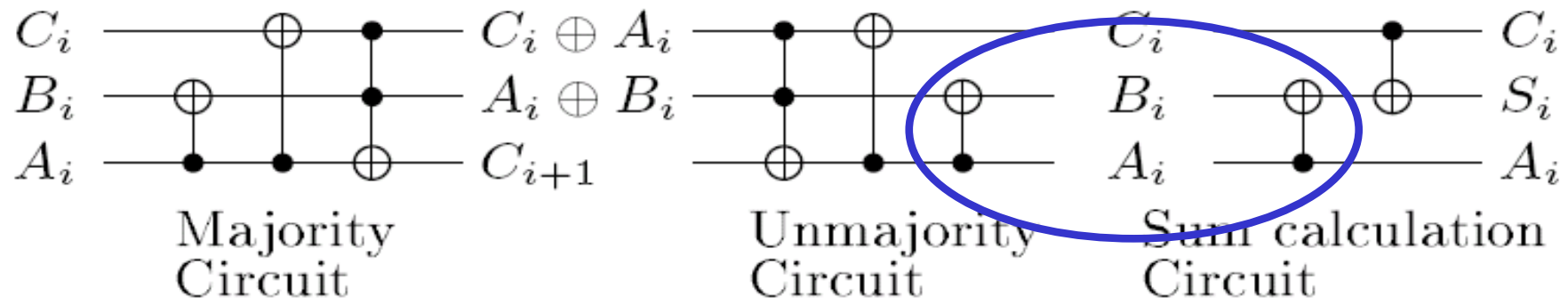
[Vedral, Barenco, Ekert '96]



[Cuccaro, Draper, Kutin, Moulton '05]

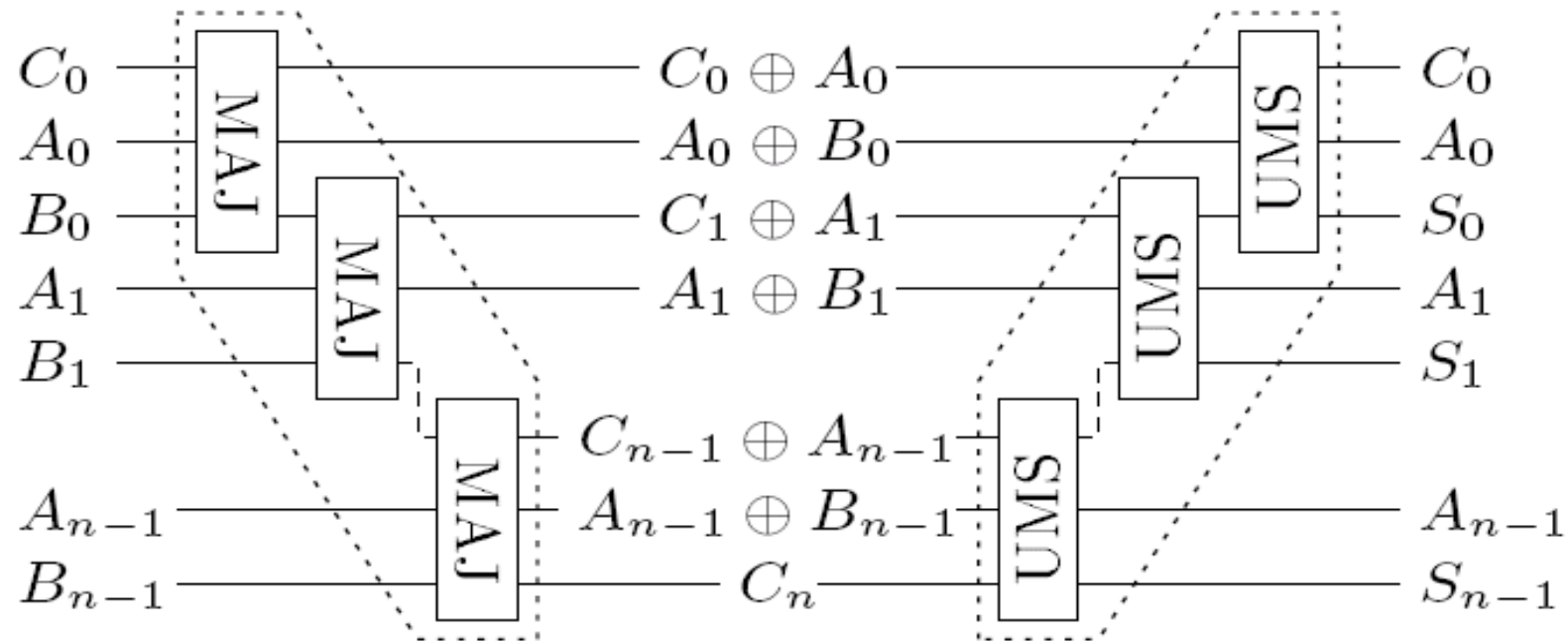


# CDKM Adder Circuit





# Ripple-Carry CDKM Adder Circuit

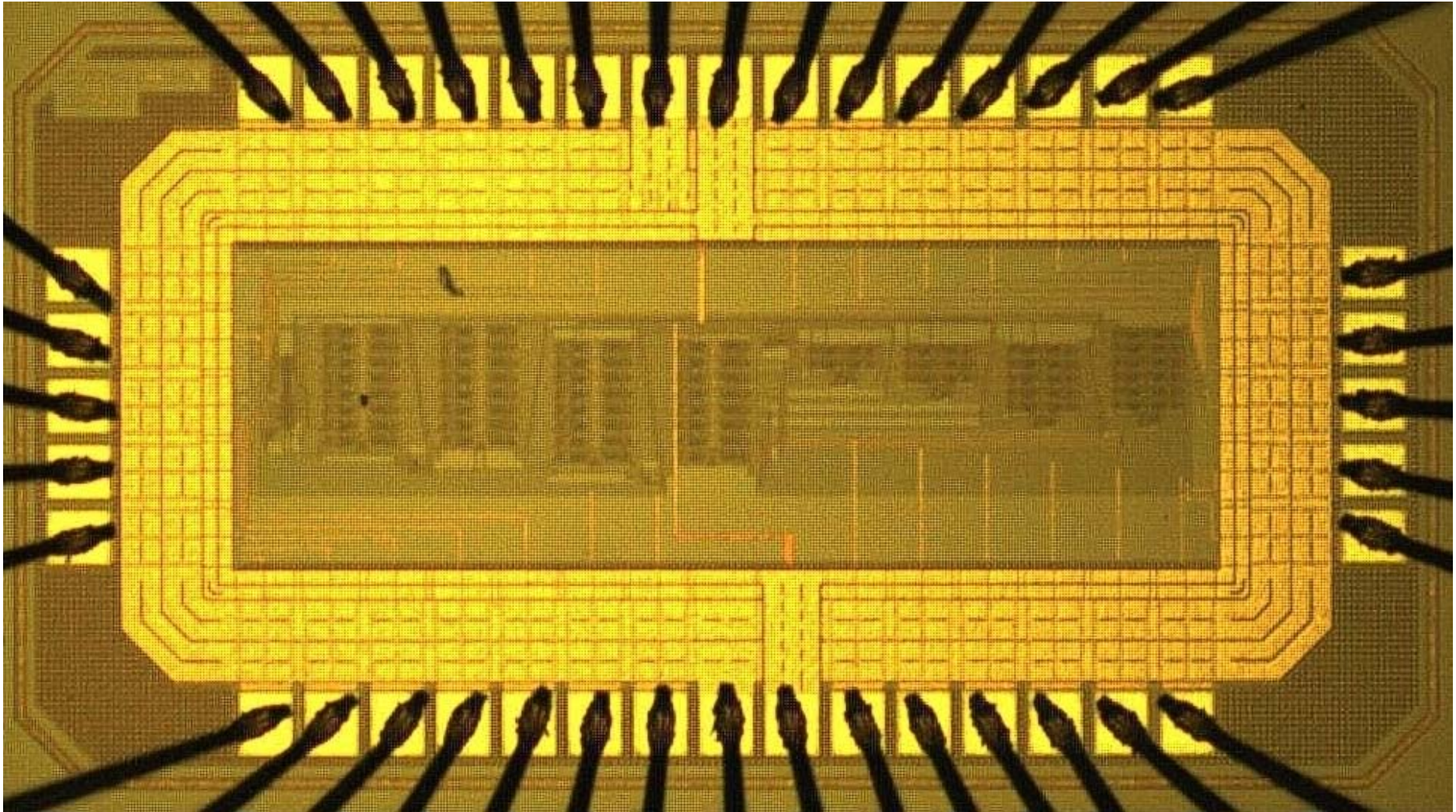


[Cuccaro, Draper, Kutin, Moulton '05]

Transistors	$O(n)$
Delay	$O(n)$
Ancillae	$O(1)$



# CDKM Adder Circuits in CPL



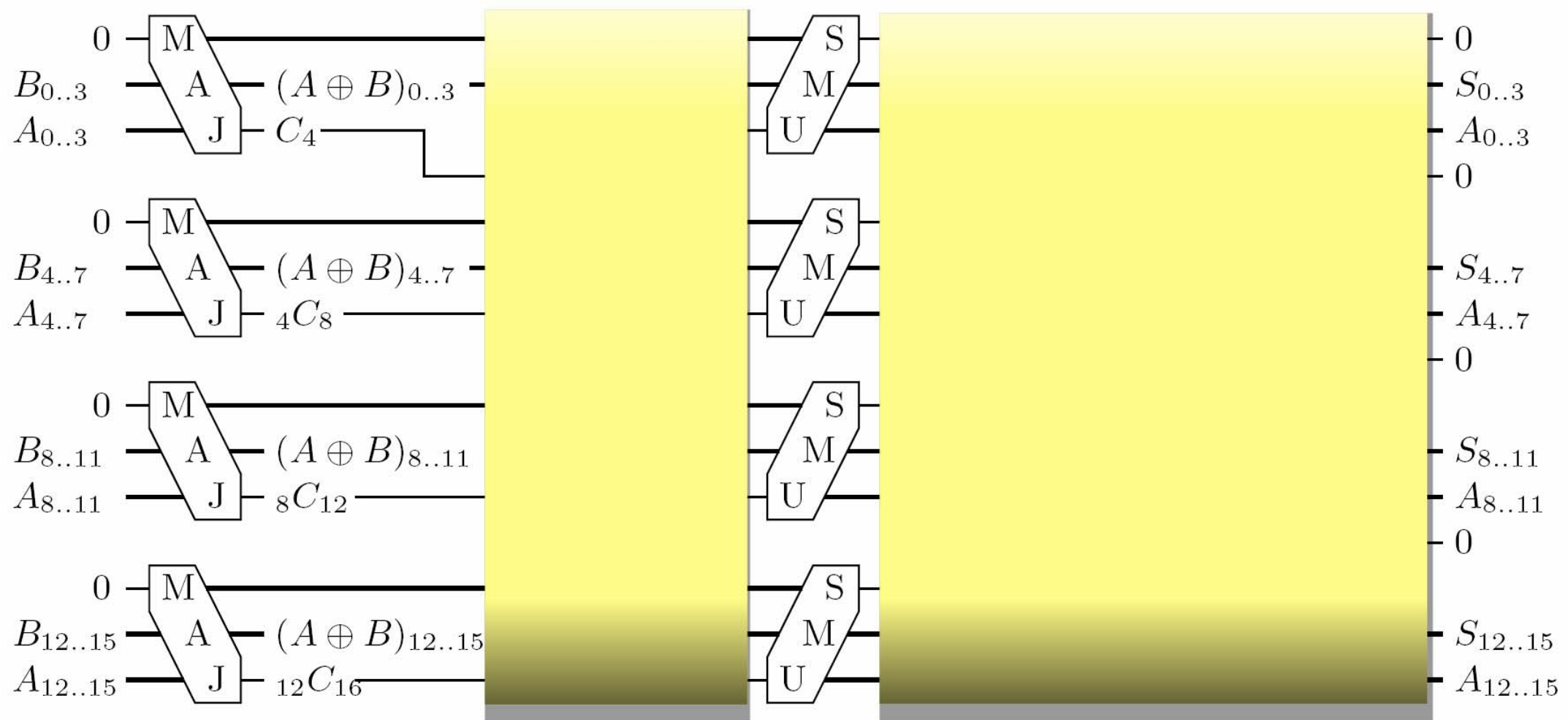


# Optimization using Parallelization

- Ripple-carry adders are **small** but **slow**
  - **Small**: Few transistors
  - **Slow**: Large delay
- Optimization schemes exist
  - Carry-lookahead, carry-save, conditional-sum...
  - These are **not** reversible
- Goal:  
Faster reversible adder using parallelization



# The Parallelization Idea





# Carry Correction

**Lemma 1 (Carry correction).** *Let  $A$  and  $B$  be  $n$ -bit binary numbers, then for all  $i$  and  $j$  such that  $0 \leq i < j \leq n$  it holds that*

$$C_j = \underbrace{(C_i(A_i \oplus B_i)(A_{i+1} \oplus B_{i+1}) \cdots (A_{j-1} \oplus B_{j-1}))}_{\text{green underline}} \oplus \underbrace{i C_j}_{\text{blue underline}}.$$

$C_{in}$	A	B	${}^n C_{out}$	$C_{out}$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



# Carry Correction

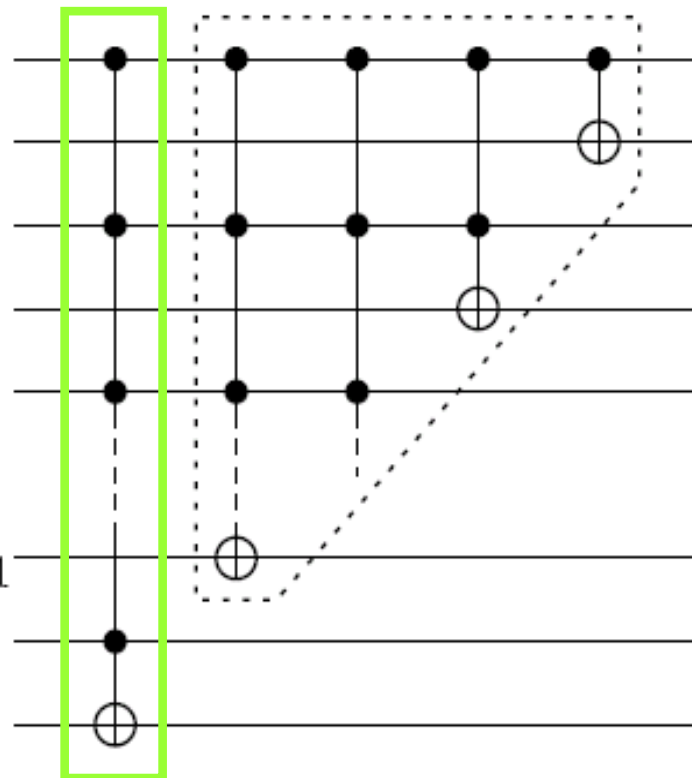
**Lemma 1 (Carry correction).** *Let  $A$  and  $B$  be  $n$ -bit binary numbers, then for all  $i$  and  $j$  such that  $0 \leq i < j \leq n$  it holds that*

$$C_j = \underbrace{(C_i(A_i \oplus B_i)(A_{i+1} \oplus B_{i+1}) \cdots (A_{j-1} \oplus B_{j-1}))}_{\text{CC}} \oplus \underbrace{iC_j}_{\text{MAJ}}$$

Output from MAJ

$C_i$   
 $A_i$   
 $A_i \oplus B_i$   
 $A_{i+1} \oplus iC_{i+1}$   
 $A_{i+1} \oplus B_{i+1}$

CC



Input to UMS

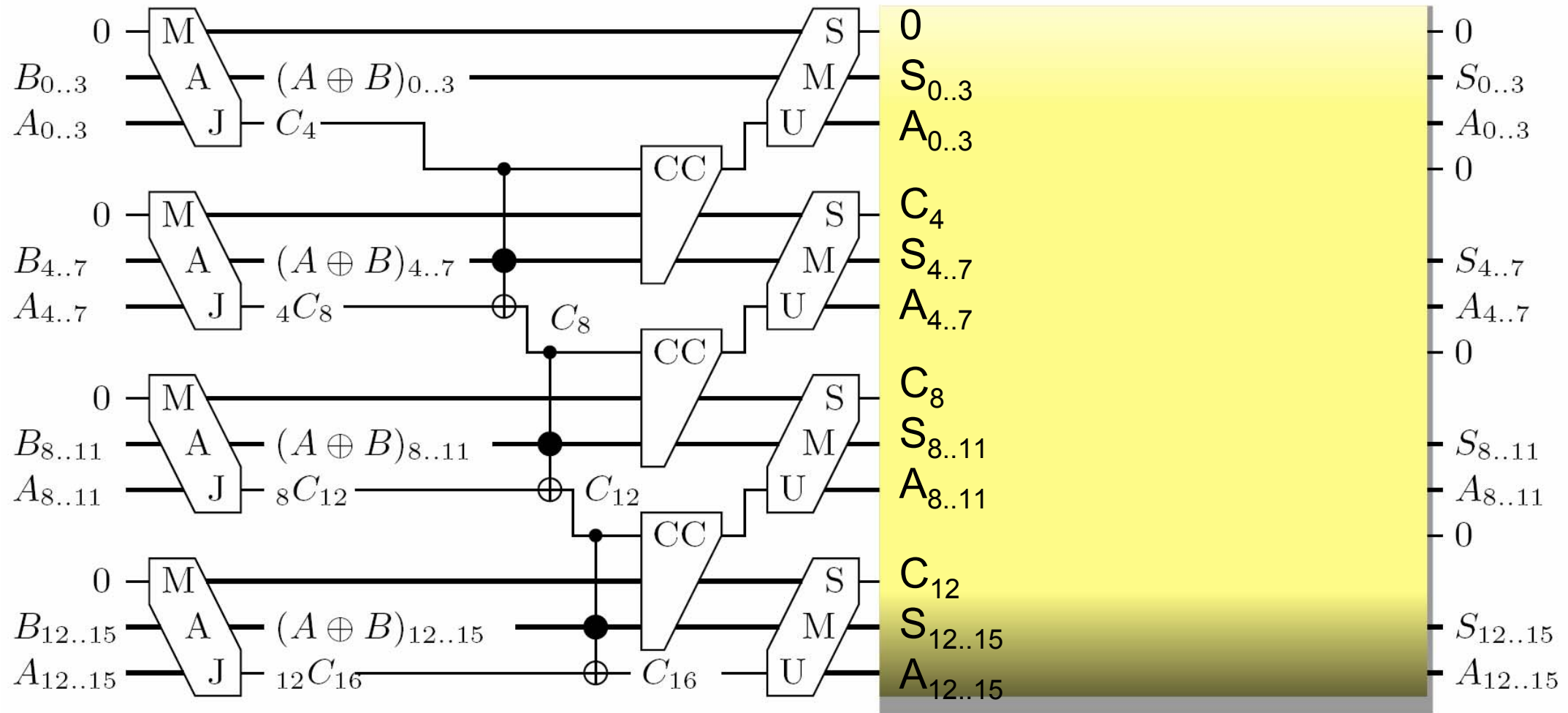
$C_i$   
 $A_i \oplus C_i$   
 $A_i \oplus B_i$   
 $A_{i+1} \oplus C_{i+1}$   
 $A_{i+1} \oplus B_{i+1}$   
  
 $A_{i+k-1} \oplus C_{i+k-1}$   
 $A_{i+k-1} \oplus B_{i+k-1}$   
 $C_{i+k}$

$A_{i+k-1} \oplus iC_{i+k-1}$   
 $A_{i+k-1} \oplus B_{i+k-1}$   
 $iC_{i+k}$





# Carry Correction



• Too many carries



# Uncomputation of Carries

**Lemma 2 (Carry-sum dependency).** *Let  $A, B, S$  be  $k$ -bit (non-negative) binary numbers  $\in \{0 \dots 2^k - 1\}$  and  $C_0, C_k$  be single bits (truth values) as described above. Then*

$$C_k = \underline{C_0(A = S)} \oplus \underline{(S < A)}.$$

$C_{in}$	A	B	S	$C_{out}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



# Uncomputation of Carries

**Lemma 2 (Carry-sum dependency).** *Let  $A, B, S$  be  $k$ -bit (non-negative) binary numbers  $\in \{0 \dots 2^k - 1\}$  and  $C_0, C_k$  be single bits (truth values) as described above. Then*

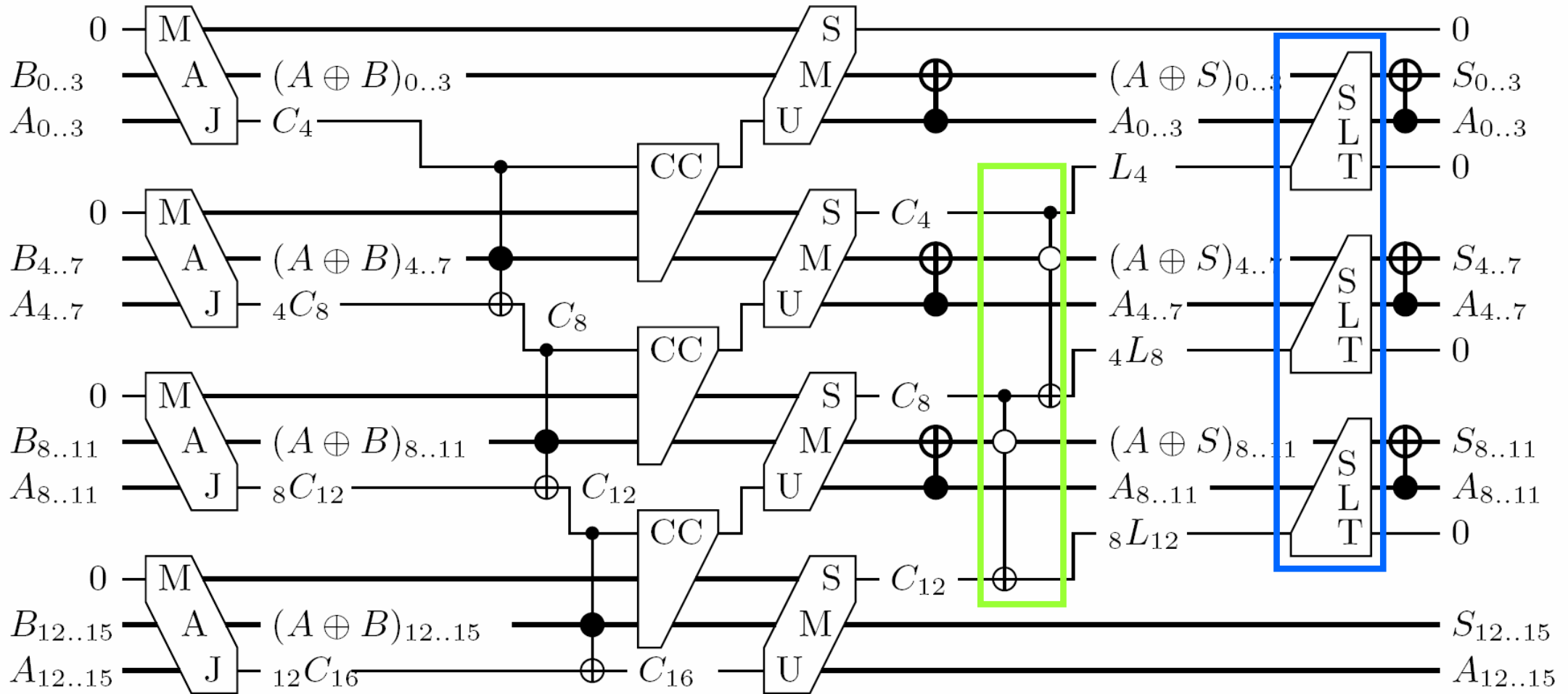
$$C_k = \underline{C_0(A = S)} \oplus \underline{(S < A)}.$$

**Corollary 1.** *Let  $A, B, S$  be  $n$ -bit numbers defined as in Lemma 2. For all  $i, j$ , where  $0 \leq i < j \leq n$ , the following recurrence holds.*

$$C_j = \underline{C_i(A_{i..j-1} = S_{i..j-1})} \oplus \underline{(S_{i..j-1} < A_{i..j-1})}$$



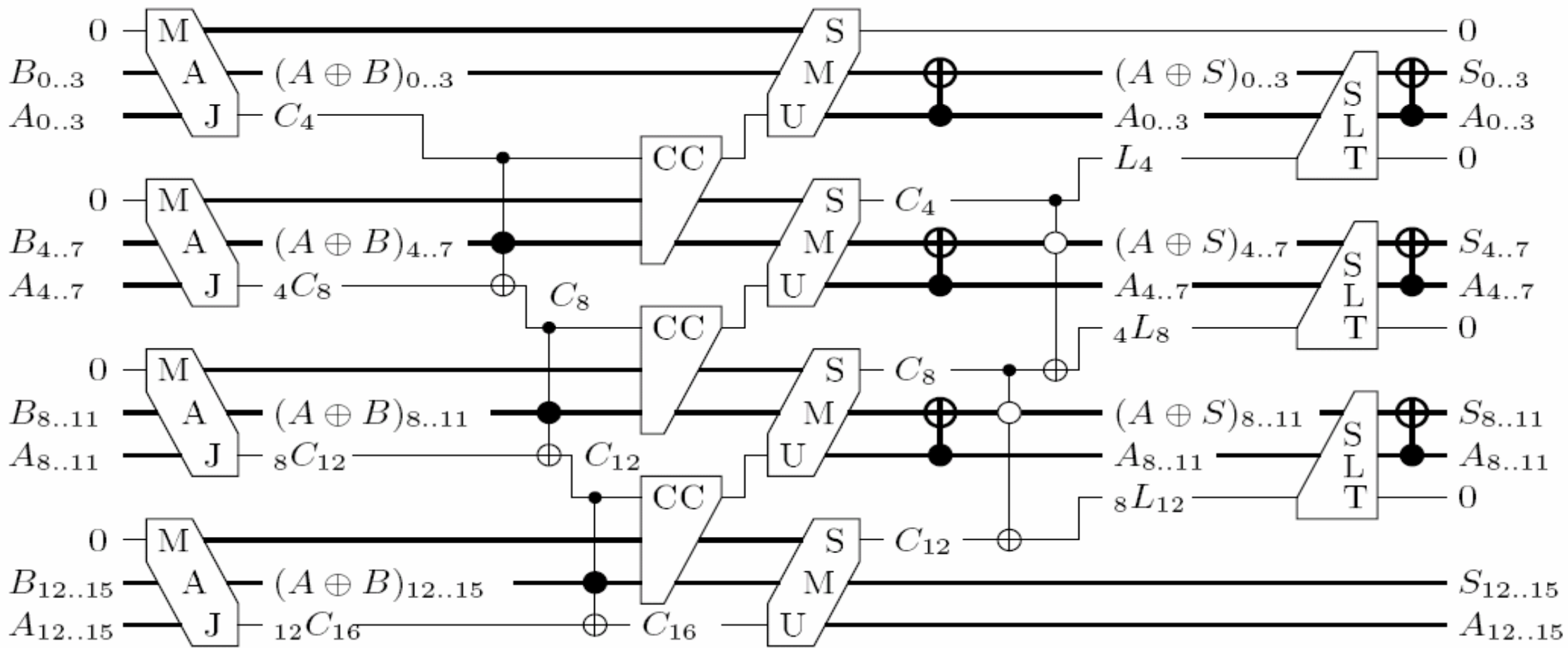
# Uncomputation of Carries



$$C_j = \underline{C_i(A_{i..j-1} = S_{i..j-1})} \oplus \underline{(S_{i..j-1} < A_{i..j-1})}$$



# Design of Parallelization





# Cost of Ripple-Block Carry Adder

*Delay in gates*

bits	$n$ -bit, $m$ -block ripple-block carry adder					CDKM-adder	
$n \setminus m$	2	4	8	16	32		
8	32	<b>22</b>	23			41	54%
16	60	36	<b>30</b>	39		81	37%
32	116	64	<b>44</b>	46	71	161	<b>27%</b>
64	228	120	72	<b>60</b>	78	321	19%
128	452	232	128	<b>88</b>	92	641	14%

*Transistor cost*

$n \setminus m$	2	4	8	16	32		
8	800	<b>912</b>	992			512	178%
16	1856	1984	<b>2000</b>	2080		1024	195%
32	4736	4704	<b>4352</b>	4176	4256	2048	<b>213%</b>
64	13568	12448	10400	<b>9088</b>	8528	4096	222%
128	43520	37152	27872	<b>21792</b>	18560	8192	266%



# Implementation Cost

	CDKM	RBCA
Transistors	$O(n)$	$O(\sqrt{n})$
Delay	$O(n)$	$O(n \sqrt{n})$
Ancillae	$O(1)$	$O(\sqrt{n})$
Garbage	0	0

- Results when RBCA is optimized wrt. delay



# “Energy” Comparison

Simple “space” \* “time” cost improvement:

$$\frac{CDKM_t(n) \cdot CDKM_d(n)}{RBCA_t(m, k) \cdot RBCA_d(m, k)}$$

Bits	CDKM		RBCA		Improve- ment
	Time	Space	Time	Space	
16	81	1024	30	2000	1.4
32	161	2048	40	4352	1.9
64	321	4096	60	9088	2.4



# Thank You!

