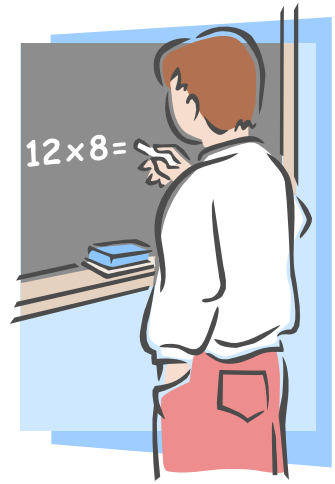
Fast Multi-Precision Multiplication for Public-Key Cryptography on Embedded Microprocessors

Michael Hutter and Erich Wenger

CHES 2011

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology

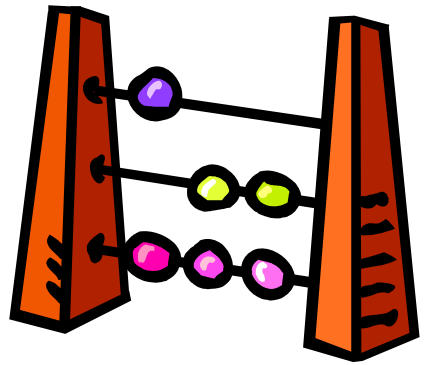
What is this talk about?



- New multiplication technique:
  - *Operand-Caching Multiplication*
- Idea: trade load against less store instructions by caching of operands
- Result: 10% improvement compared to related work on the ATmega128

Multi-Precision Multiplication

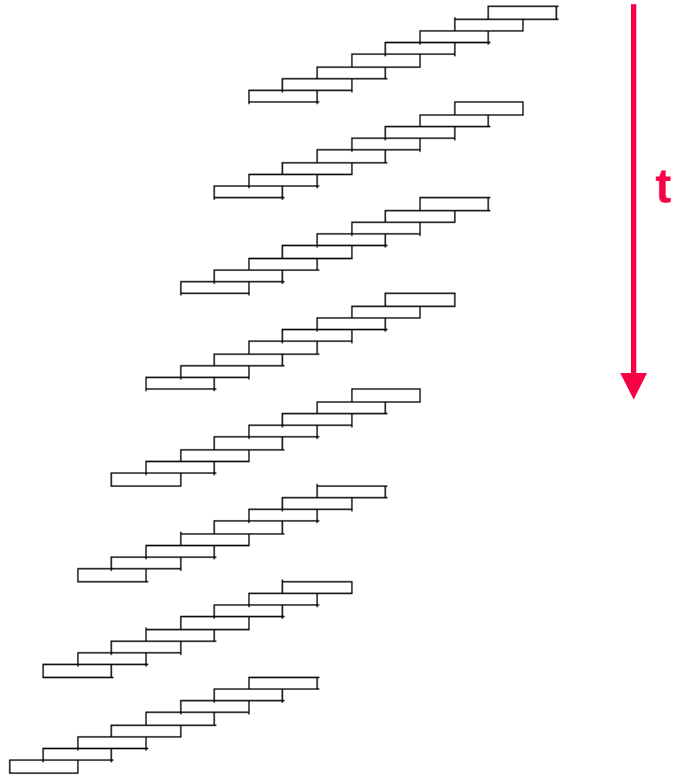
- Most important operation in PKC
- Applied in modern processors (8, 16, 32, 64 bits)



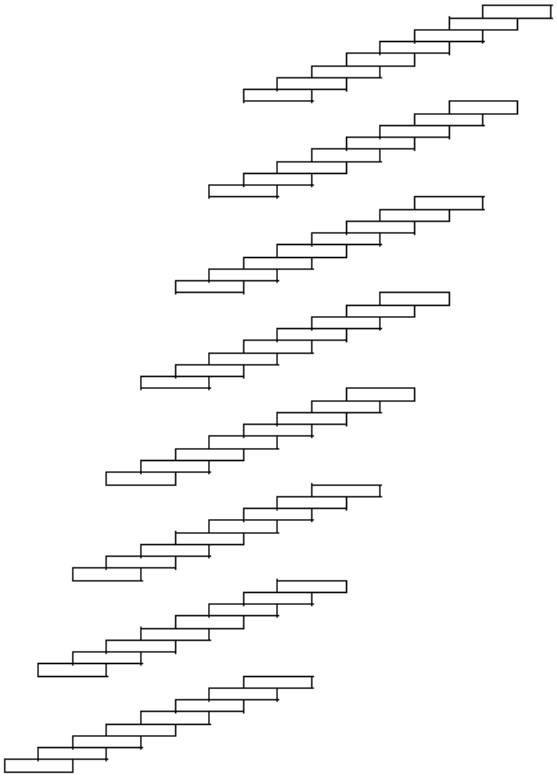
- Optimizations
  - Reduce expensive operations
  - Minimize number of load and/or store instructions

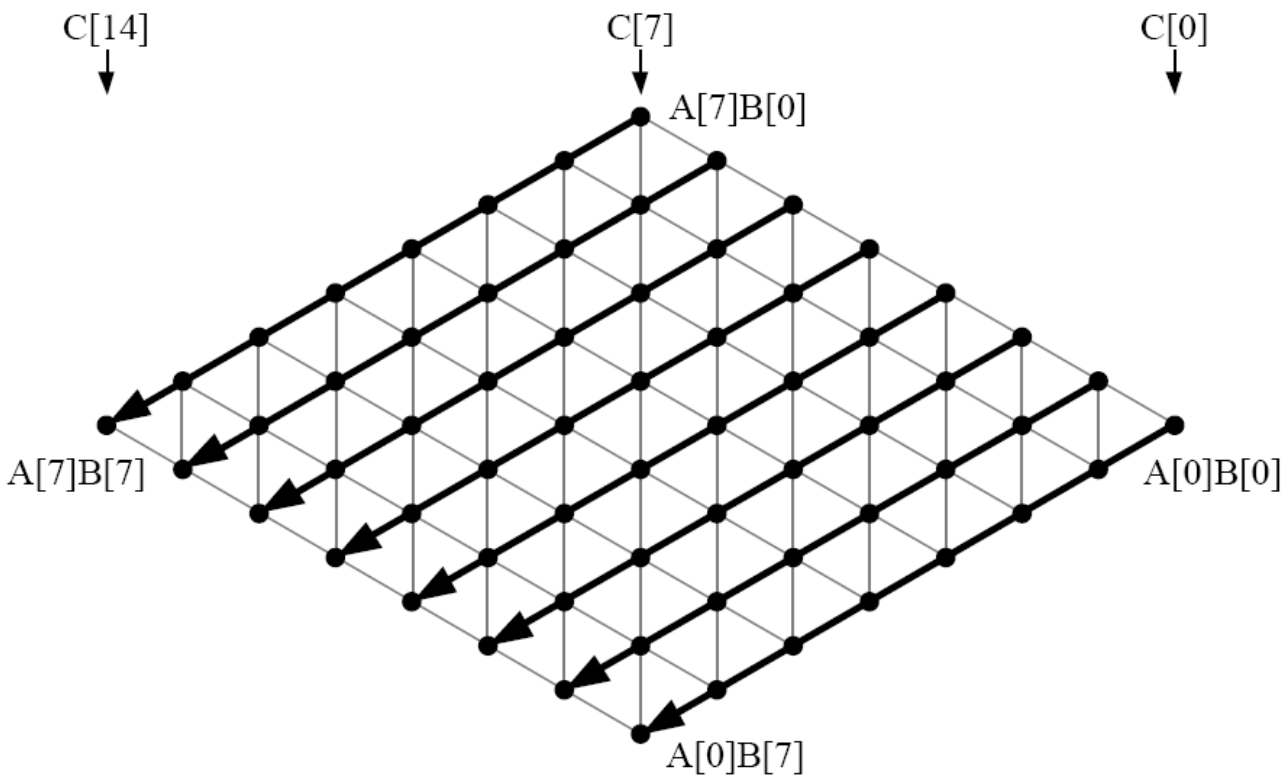
Operand Scanning

- "Schoolbook method"
  - a * b = c
- Row-wise processing
- 2 loops
- Example: n=8

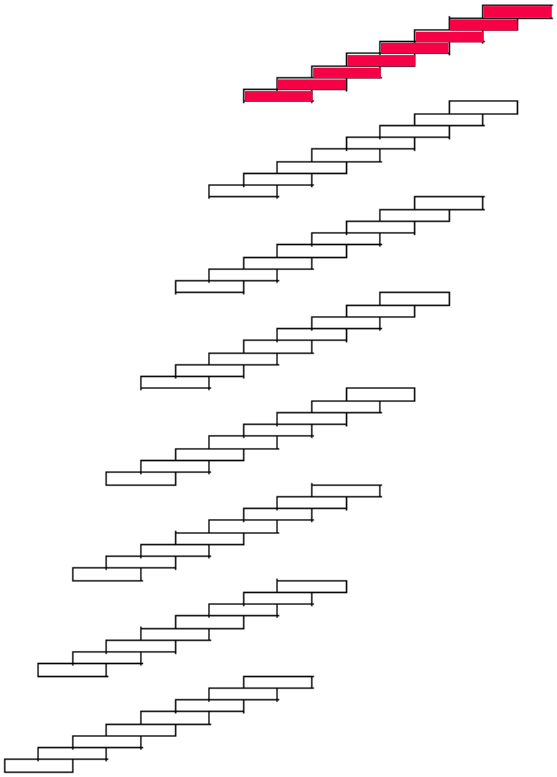


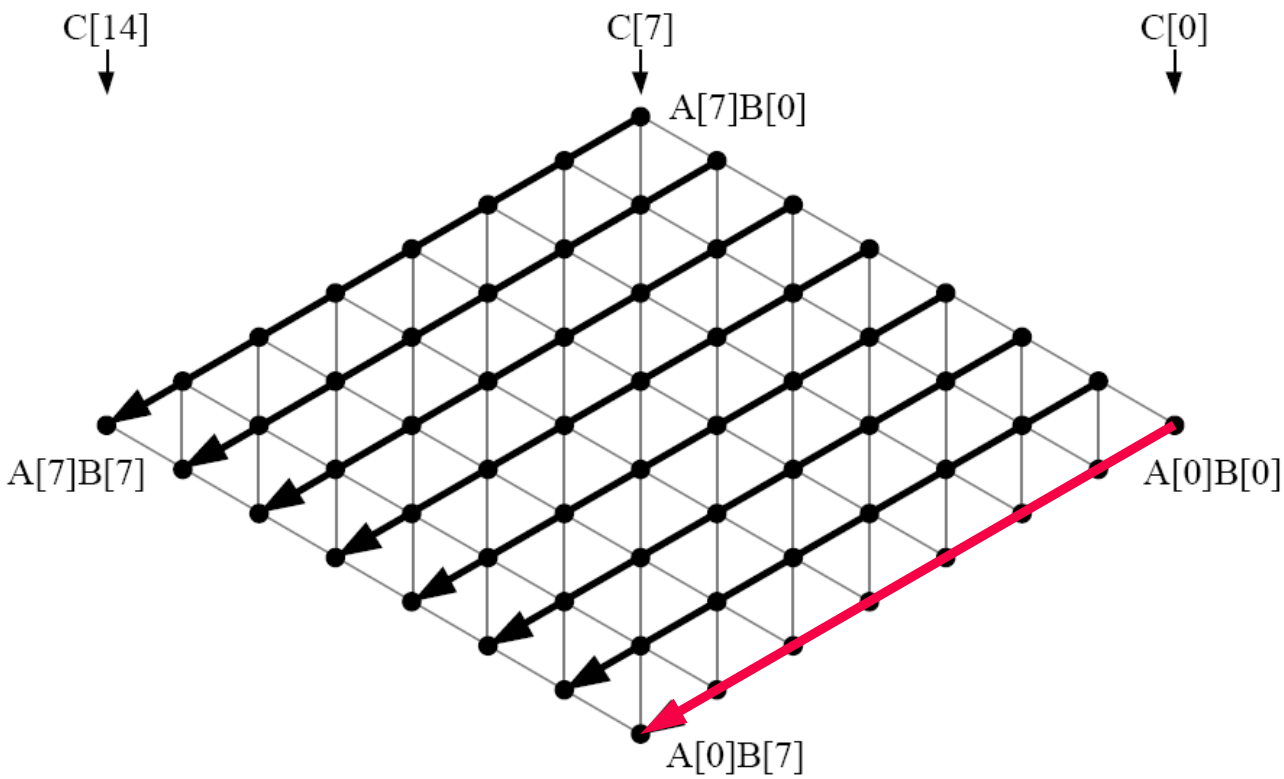
Operand Scanning



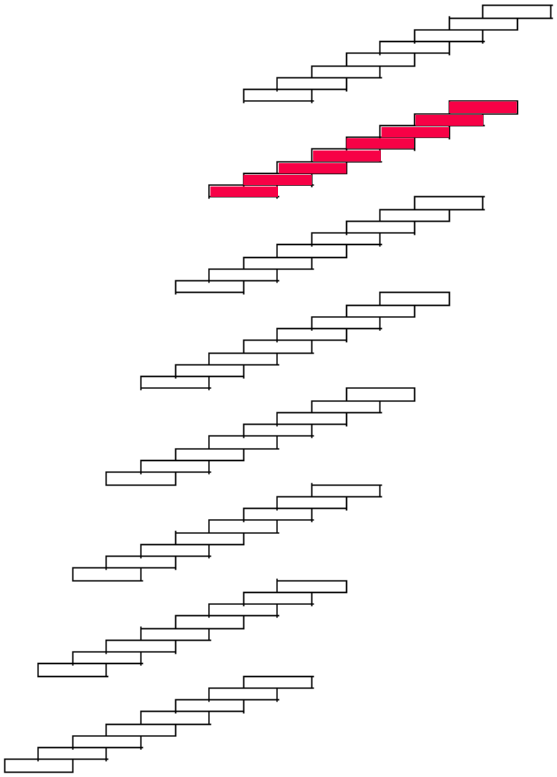


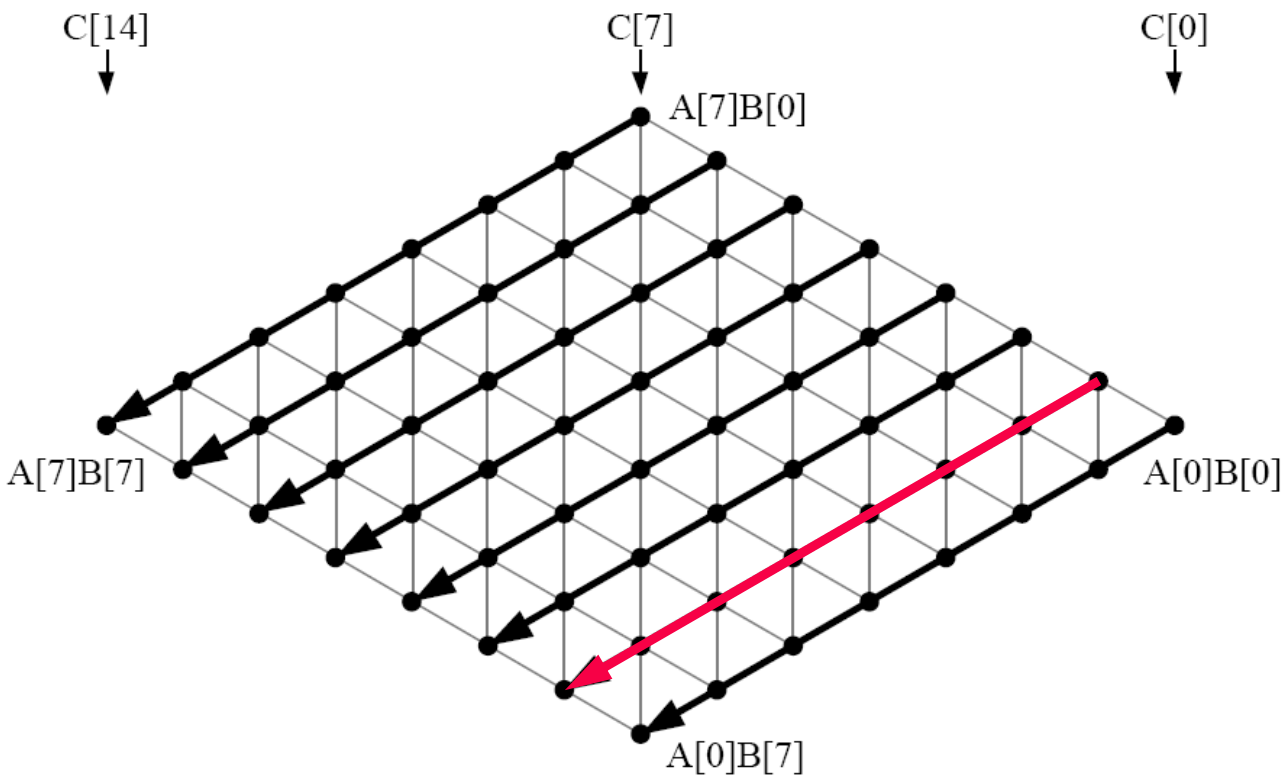
Operand Scanning



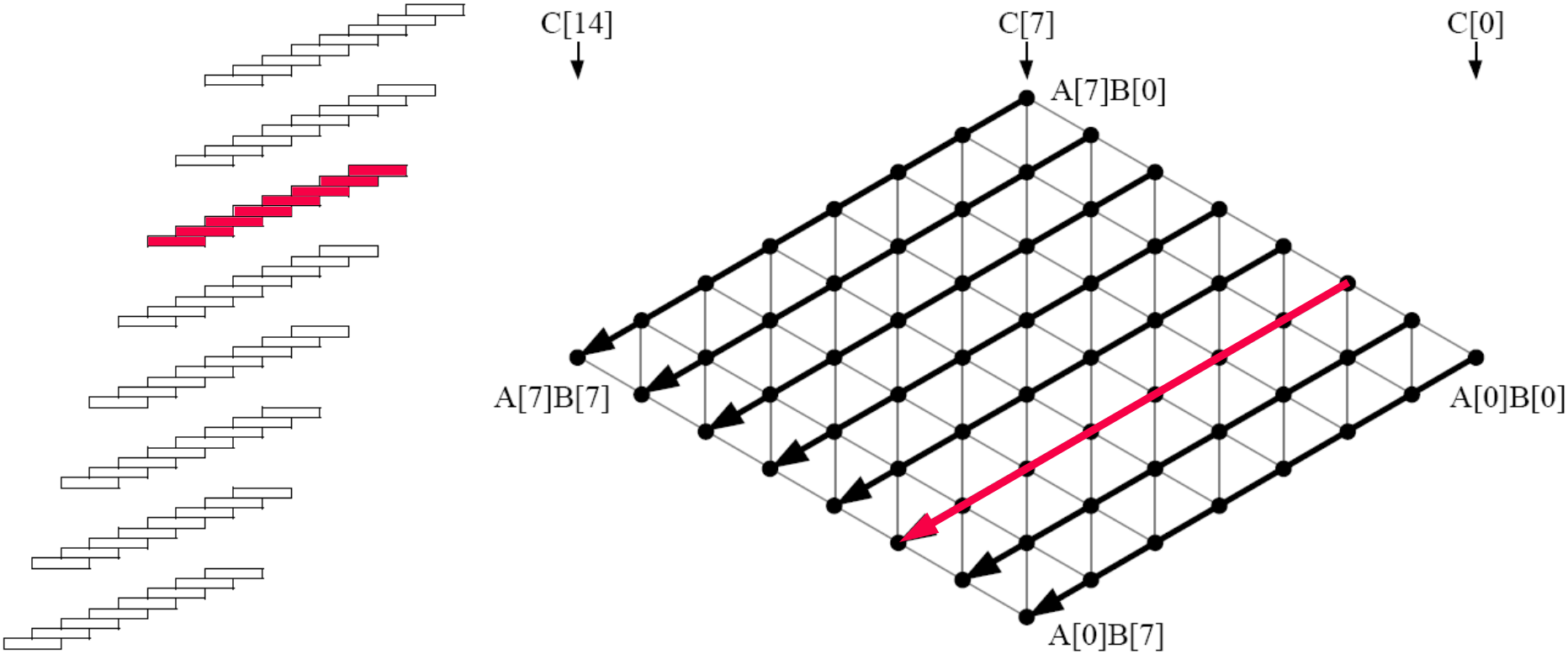


Operand Scanning

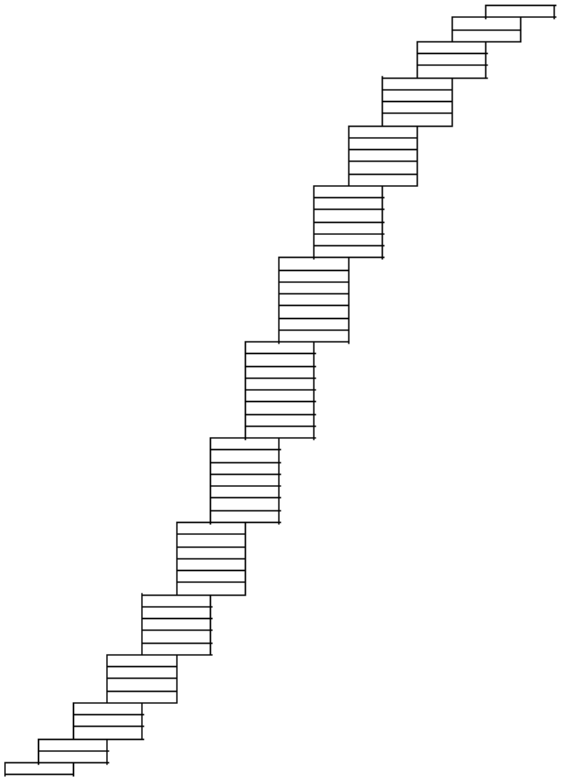


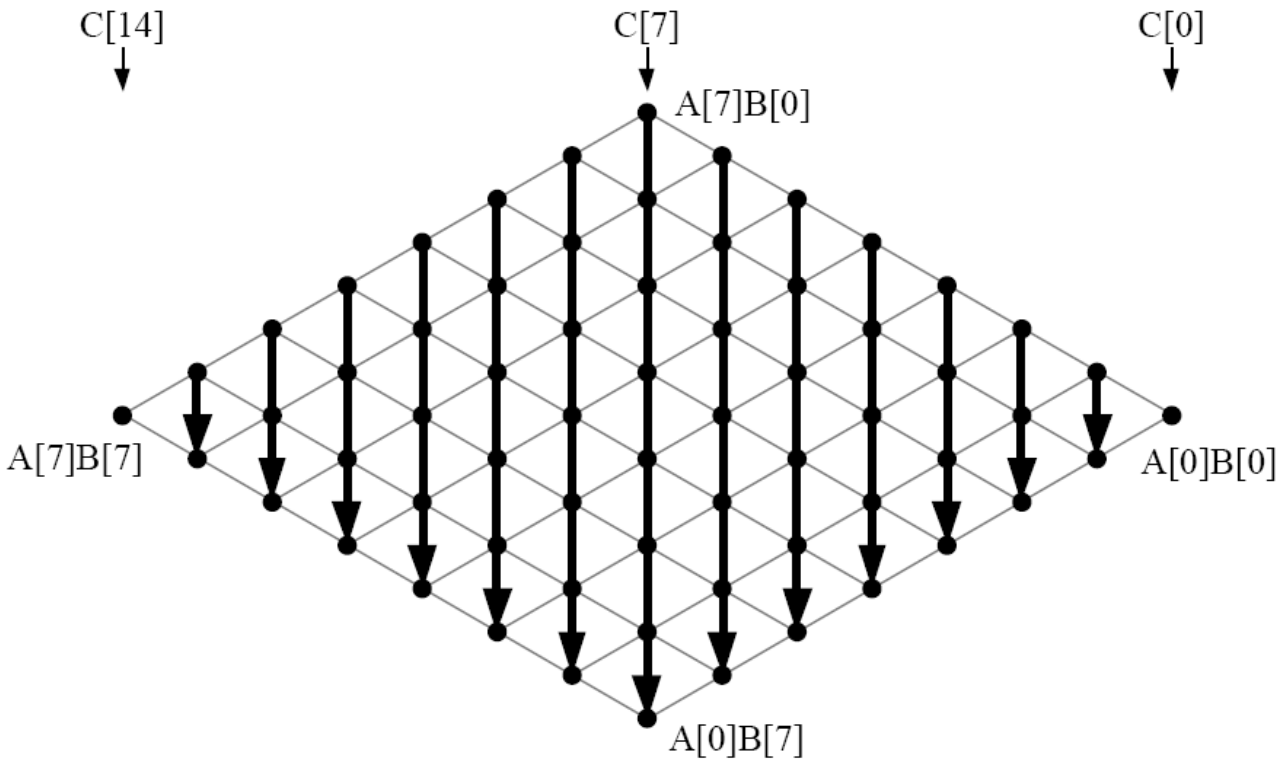


Operand Scanning

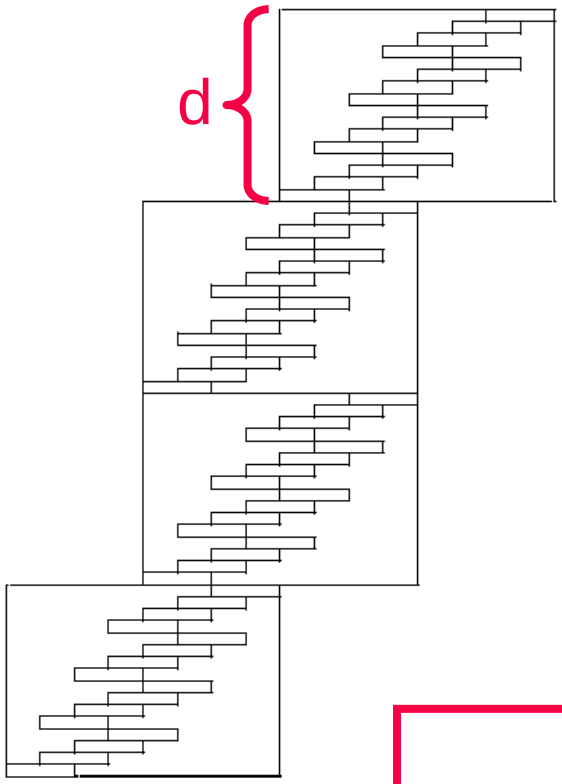


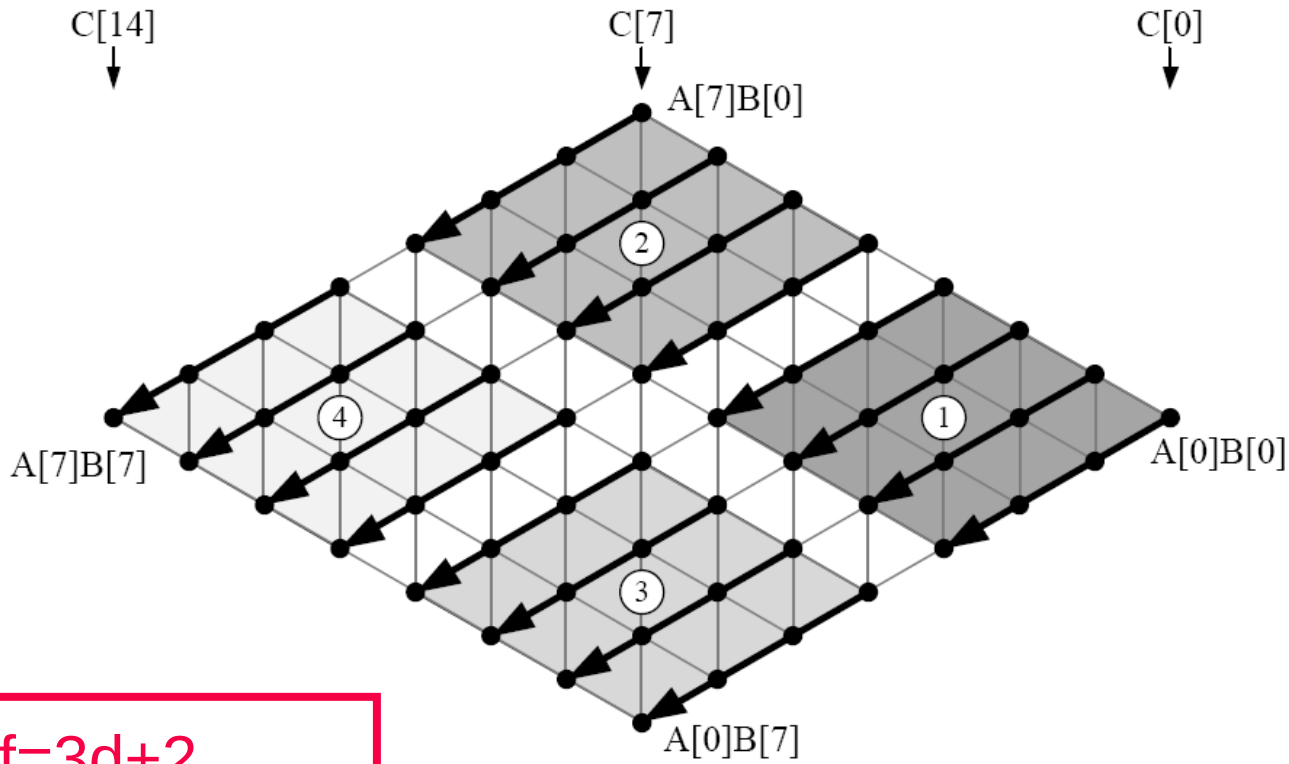
Product Scanning





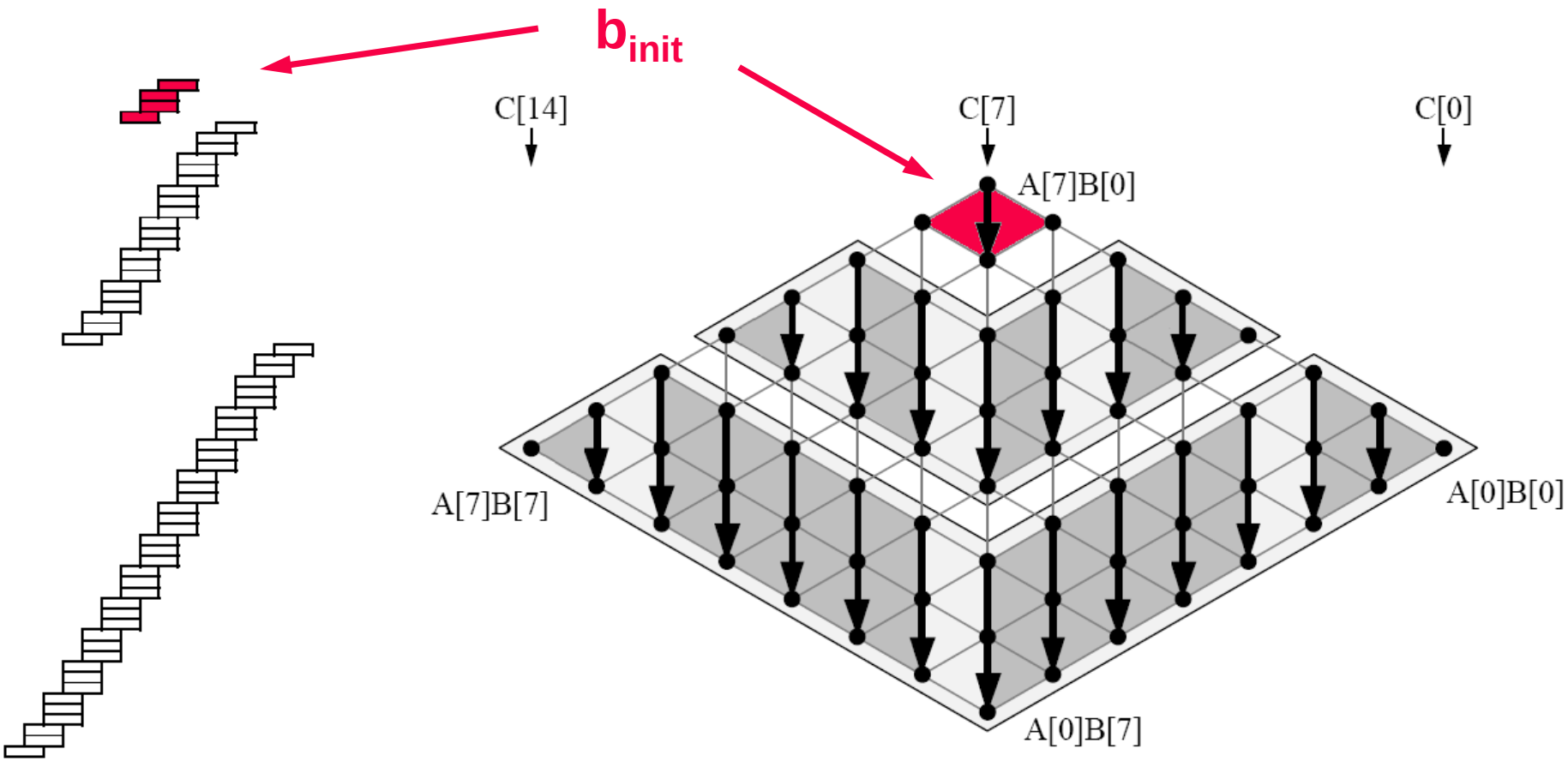
Hybrid Multiplication



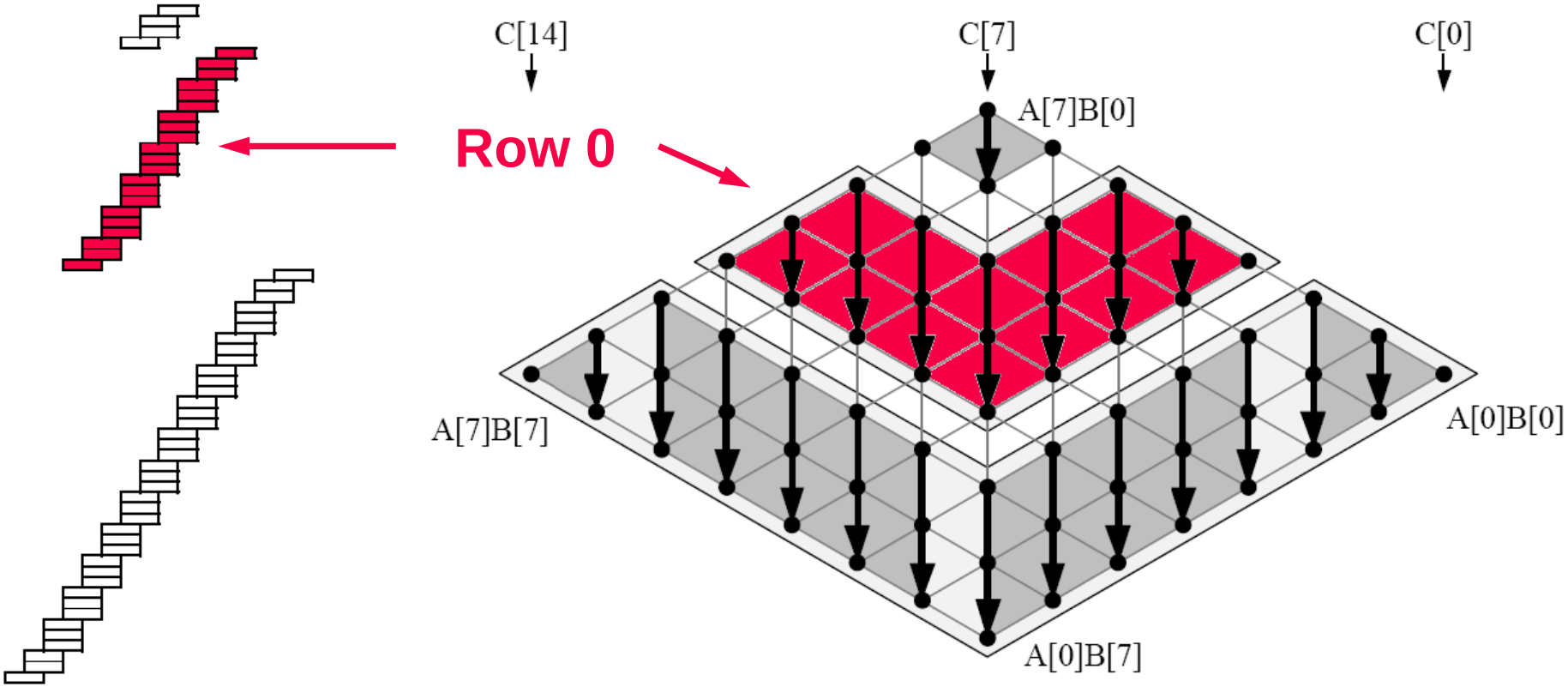


$$f=3d+2$$

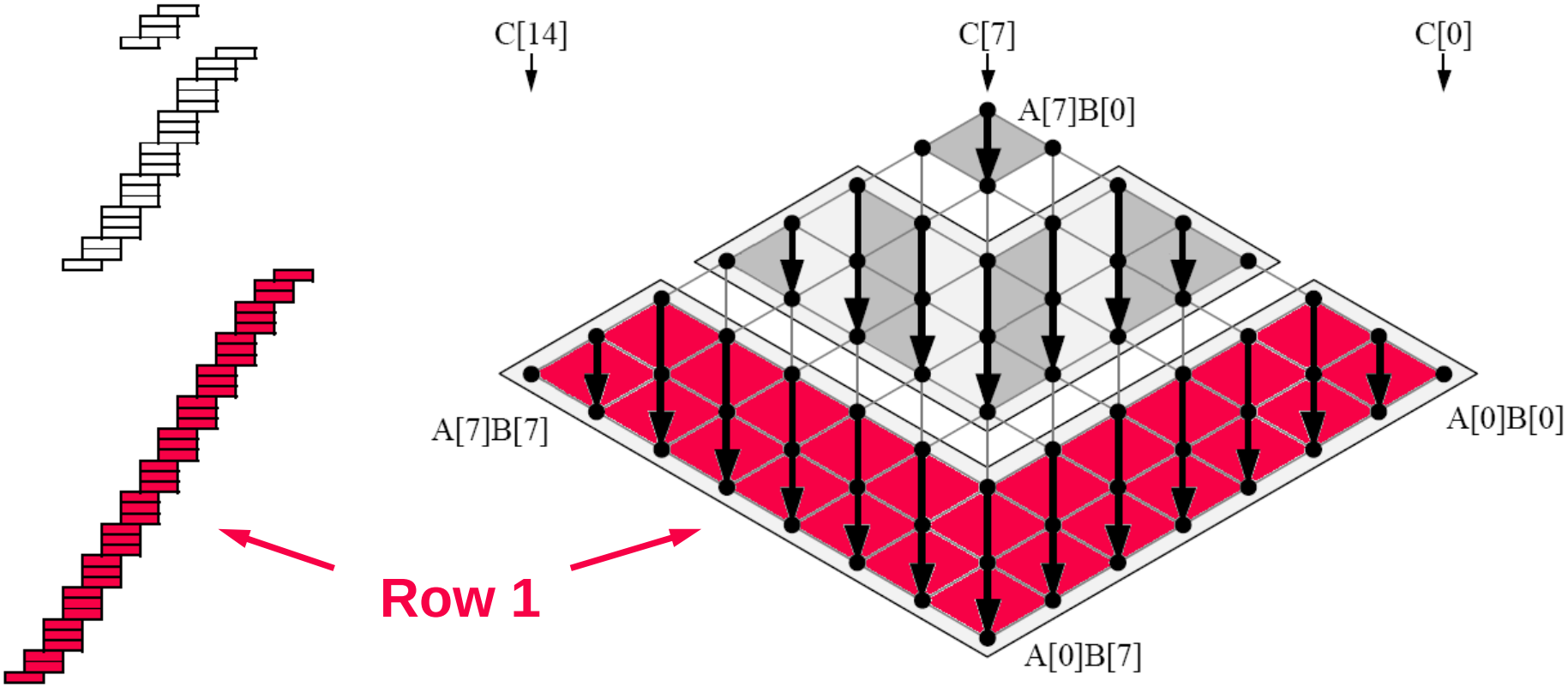
Operand-Caching Multiplication



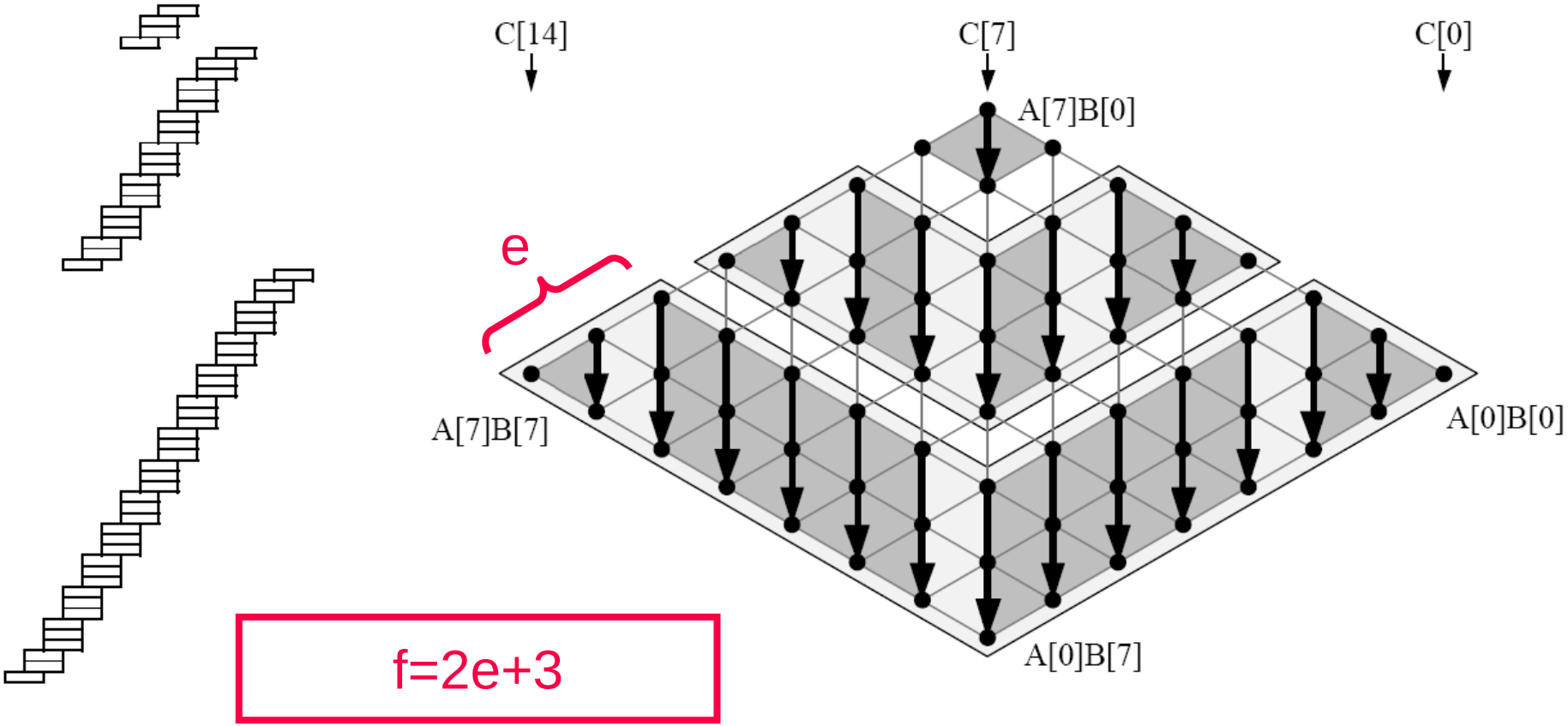
Operand-Caching Multiplication



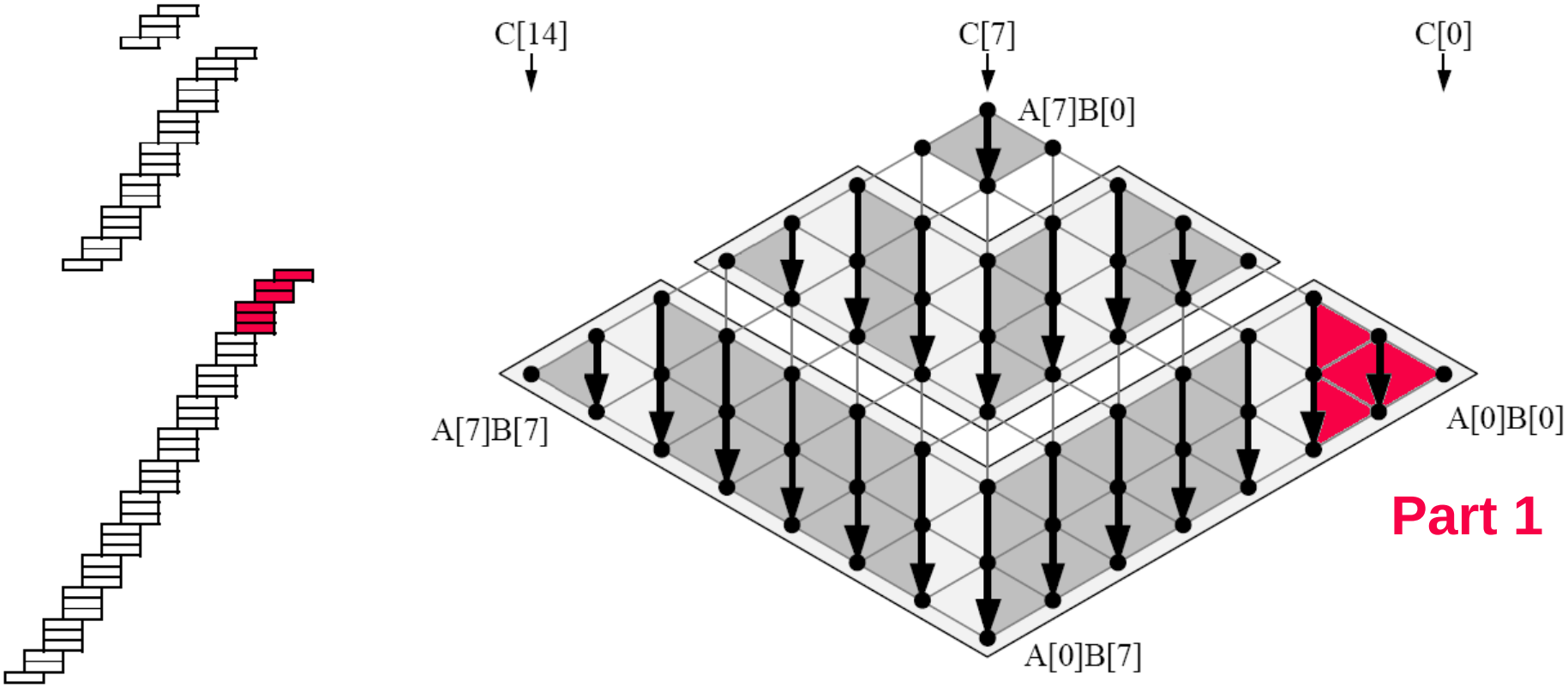
Operand-Caching Multiplication



Operand-Caching Multiplication

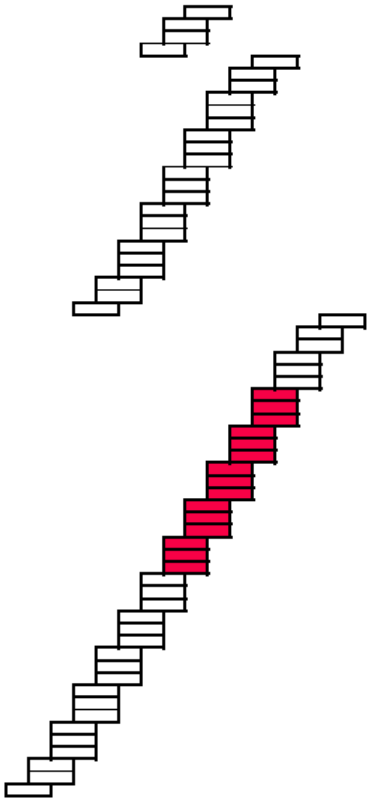


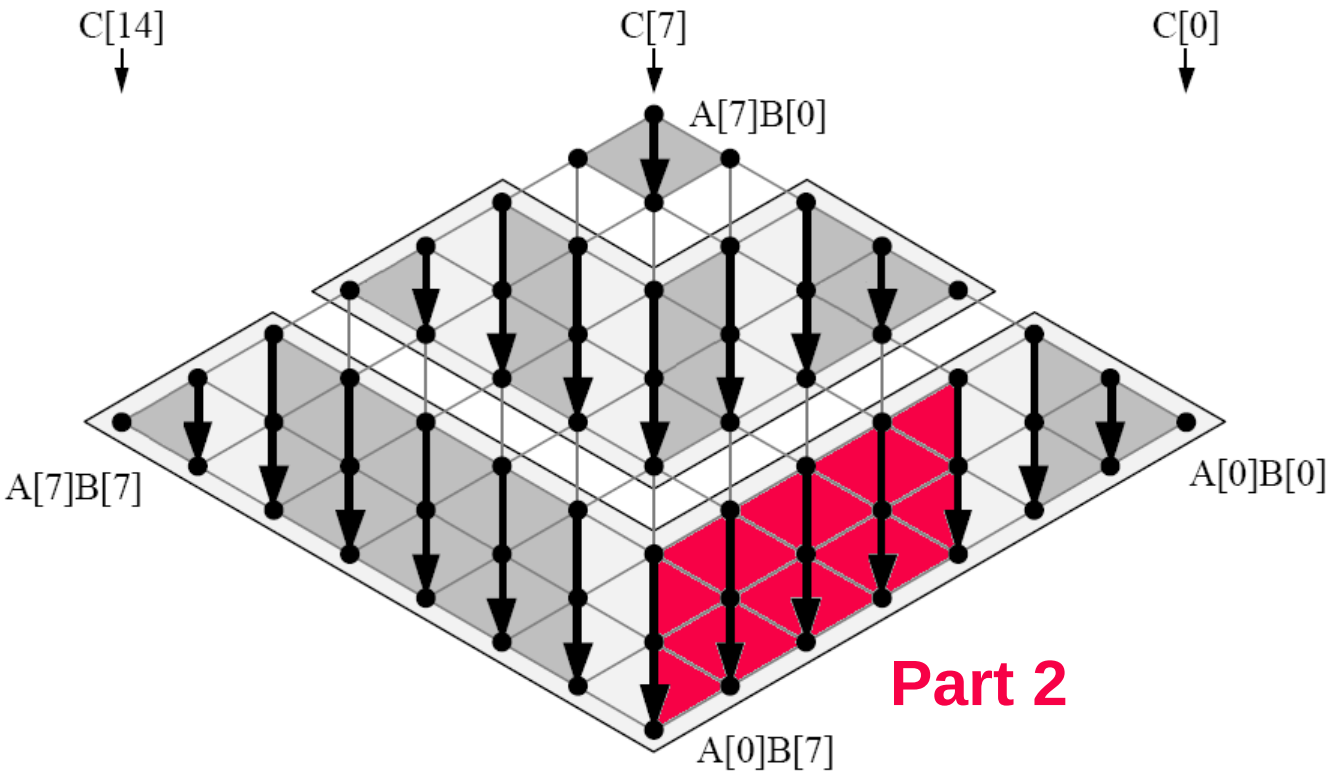
Operand-Caching Multiplication



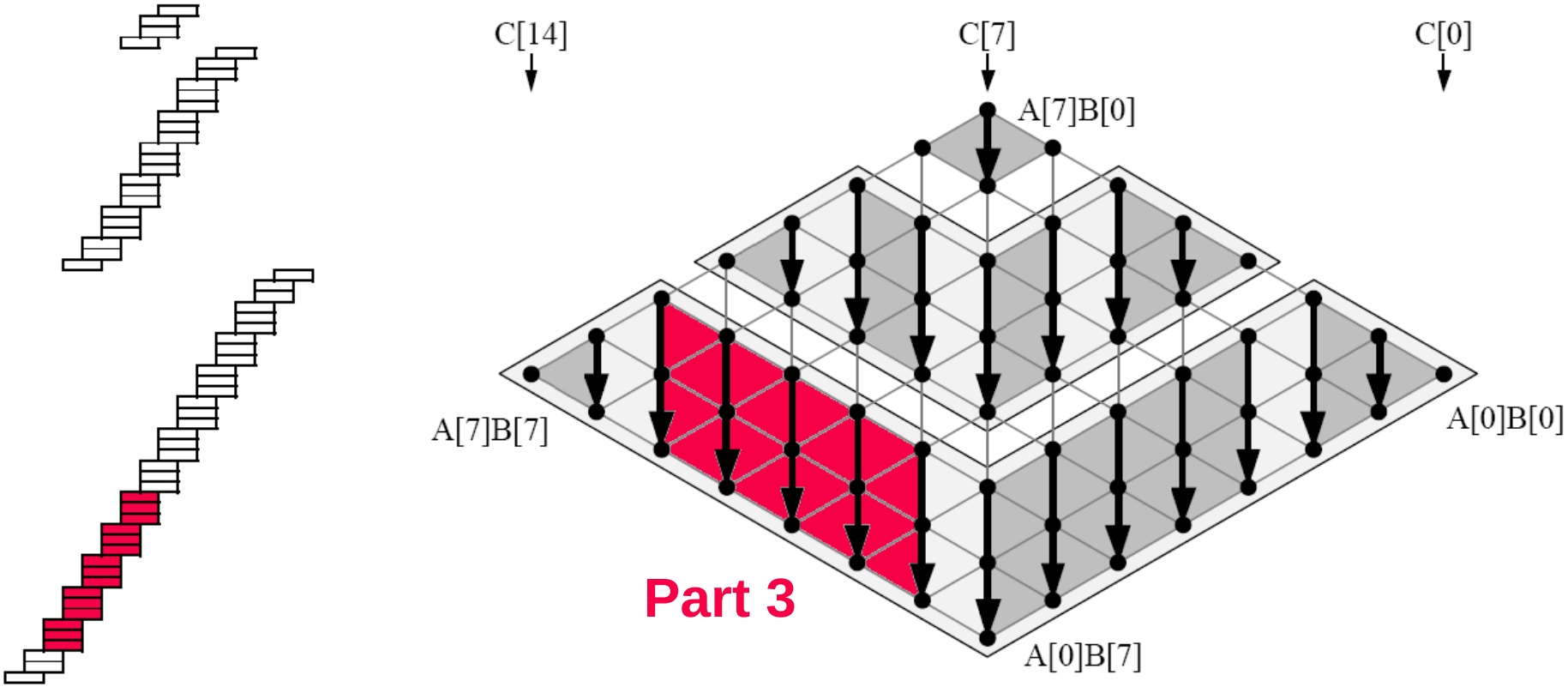
Part 1

Operand-Caching Multiplication

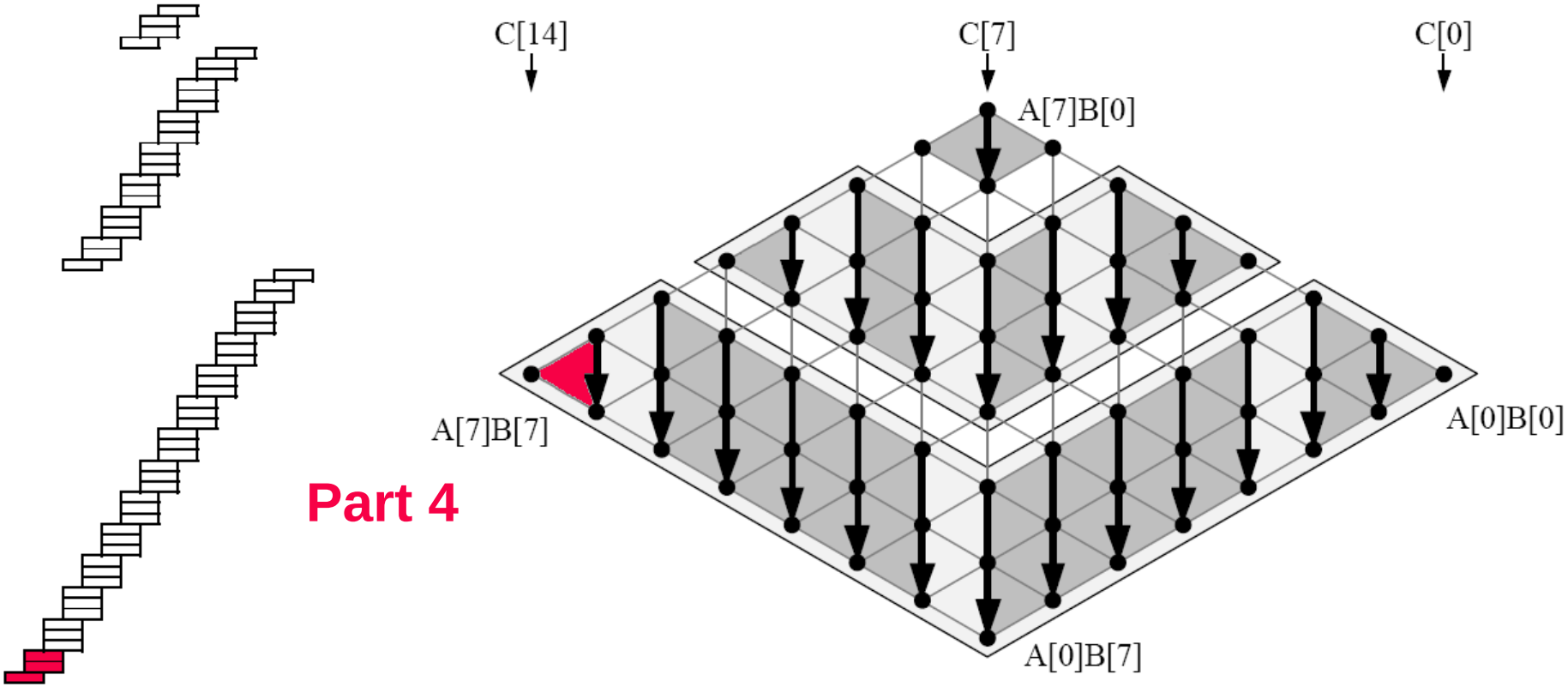




Operand-Caching Multiplication



Operand-Caching Multiplication



Part 4

Complexity

Method	Load Instructions	Store Instructions	Memory Instructions
Operand Scanning	$2n^2 + n$	$n^2 + n$	$3n^2 + 2n$
Product Scanning	$2n^2$	$2n$	$2n^2 + 2n$
Hybrid	$2\lceil n^2/d \rceil$	$2n$	$2\lceil n^2/d \rceil + 2n$
Operand Caching	$\mathbf{2n^2/e}$	$\mathbf{n^2/e + n}$	$\mathbf{3n^2/e + n}$

$$2e + 3 = 3d + 2 \implies e = \frac{3d - 1}{2} \quad \text{and} \quad e > d$$

Results

- 160-bit multiplication on the ATmega128
- Unrolled instructions

Method	Instruction						Clock Cycles
	LD	ST	MUL	ADD	MOVW	Others	
Operand Scanning	820	440	400	1,600	2	464	5,427
Product Scanning	800	40	400	1,200	2	159	3,957
Hybrid (d=4)	200	40	400	1,250	202	109	2,904
Operand Caching (e=10)	**80**	**60**	**400**	**1,240**	**2**	**68**	**2,395**

Comparison with Related Work

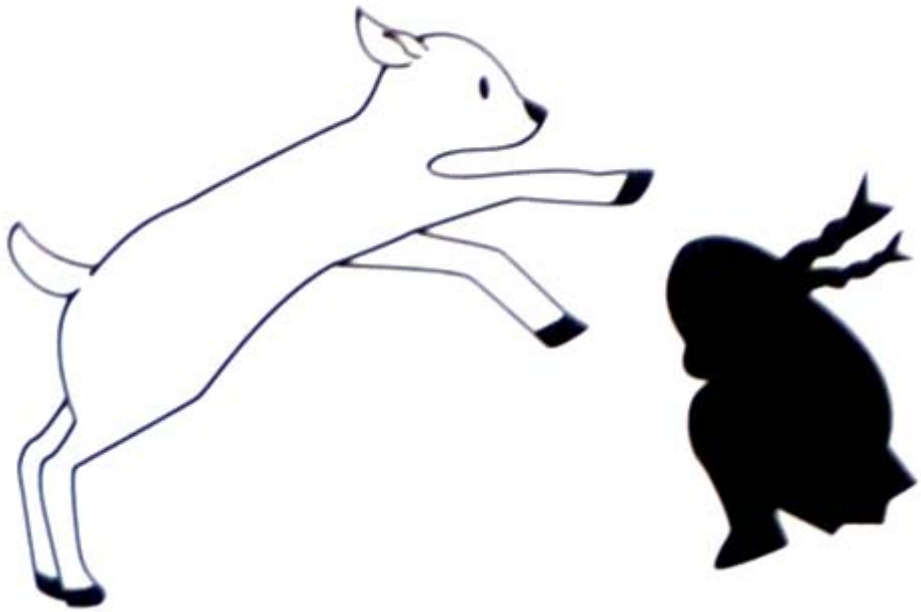
Method	Instruction						Clock Cycles
	LD	ST	MUL	ADD	MOVW	Others	
Hybrid							
Gura et al. (d=5)	167	40	400	1,360	355	197	3,106
Uhsadel et al. (d=5)	238	40	400	986	355	184	2,881
Scott et al. (d=4)	200	40	400	1,263	70	38	2,651
Liu et al. (d=4)	200	40	400	1,194	212	179	2,865
Operand Caching							
with looping (e=9)	**92**	**66**	**400**	**1,252**	**41**	**276**	**2,685**
unrolled (e=10)	**80**	**60**	**400**	**1,240**	**2**	**68**	**2,395**

- Note: Scott et al. unrolled the instructions

Let‘s summarize…

…it‘s faster…

Let‘s summarize…

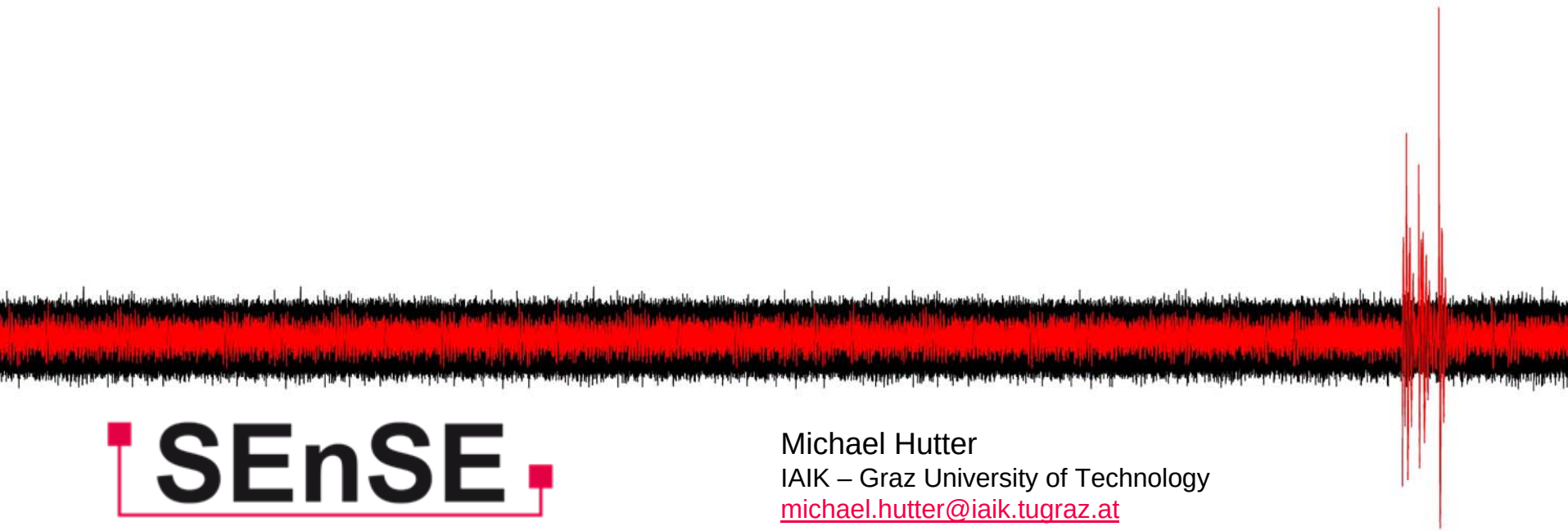


…it’s more energy efficient…

Let‘s summarize…

…it outperforms existing solutions!

Thank you!



SEnSE
Secure Entities for Smart Environments

Michael Hutter
IAIK – Graz University of Technology
michael.hutter@iaik.tugraz.at
www.iaik.tugraz.at

Recent Results

- Performance on the 32-bit ARM 7
- 192-bit multiplication
- 441 clock cycles needed

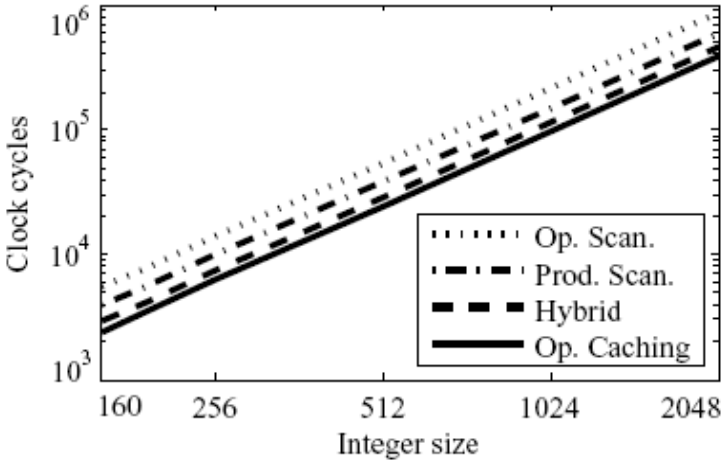
- 10% improvement compared to related work
  - Scott et al. reported 487 cycles

Memory-Access Complexity

Component	Load Instr.	Store Instr.	Total
b_{init}	$2(n-re)$	$2(n-re)$	$4(n-re)$
Part 1	$2e$	e	$3e$
Part 2	$2(n-e(p+1))$	$n-e(p+1)$	$3(n-e(p+1))$
Part 3	$2(n-e(p+1))$	$n-e(p+1)$	$3(n-e(p+1))$
Part 4	0	e	e

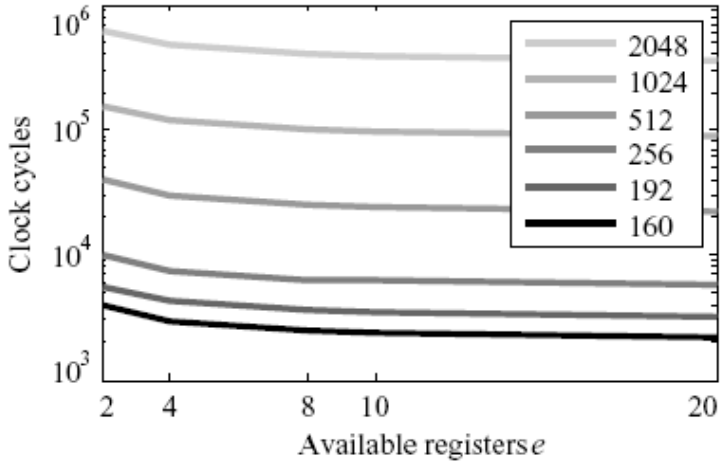
Performance for Larger Integers

Size [bit]	Op. Scan.	Prod. Scan.	Hybrid Method	Operand Caching
160	5,427	3,957	2,904	2,395
192	7,759	5,613	4,144	3,469
256	13,671	9,789	7,284	6,123
512	53,959	38,013	28,644	24,317
1,024	214,407	149,757	113,604	96,933
2,048	854,791	594,429	452,484	387,195



Available Registers

Size	e=2	e=4	e=8	e=10	e=20
160	3,915	2,965	2,513	2,395	2,205
192	5,611	4,255	3,577	3,469	3,207
256	9,915	7,531	6,339	6,123	5,671
512	39,291	29,915	25,227	24,317	22,451
1,024	156,411	119,227	100,635	96,933	89,529
2,048	624,123	476,027	401,979	387,195	357,581



160-bit Multiplication