Time-Area Tradeoff Improvement for Montgomery Multiplication Implementation

F. BERNARD

Université Paris 8

Cryptarchi 2007, Juin 19-22
Le Hameau de l'Etoile

Context

- Modular multiplication : main operation in asymmetric cryptography (Diffie-Hellmann, RSA, ECDSA)
- Different sizes of moduli :
  - Increasement of security level
  - Protocol used : ECDSA (192-256 bits), RSA (1024-2048 bits)
- Need of a “scalable” hardware for modular multiplication
  1. Design must fit any chip area
  2. Post-synthesis : computation with different sizes of moduli must be possible

Hardware constraints and specifications

- Hardware constraints
  - Technology (ASIC, FPGA)
  - Area (Gates number, % area)
  - Clock frequency
  - ...
- Specifications
  - Computational time
  - Scalability
  - Resistant against side-channel attacks
  - ...
- *Efficient implementation* = find the best Time/area tradeoff according to given constraints

Montgomery modular reduction

- Modular multiplication : $A \times B \mod N$
  - 1 multiplication, 1 division
  - Division is a costly operation
  - Alternative : *Modular multiplication without trial division*, P. Montgomery, 1985
- Montgomery modular reduction
  - $N = N_0 + N_1 r + \cdots + N_{n-1} r^{n-1}$ in radix representation
  - $R = r^n$ prime with N
  - Pre-computation of $N' = -N^{-1} \mod R$
  - Division by N replaced by simple right-shifts

High-radix Montgomery algorithm

Algorithm used ($r \geq 4$) : MM_nS

Inputs : $A, B \leq 2N - 1$

Output : $\begin{cases} S \equiv ABr^{-n-1} \bmod N \\ S \leq 2N - 1 \end{cases}$

$S \leftarrow a_0 B \leftarrow$ Multiplication-addition

For i from 1 to n do

$m_i = s_0 N' \bmod r \leftarrow$ Low-part Multiplication

$S = a_i B + \frac{m_i N + S}{r} \leftarrow$ 2 Multiplications-additions

$m_{n+1} = s_0 N' \bmod r \leftarrow$ Low-part Multiplication

$S = \frac{m_{n+1} N + S}{r} \leftarrow$ Multiplication-addition

Return S

Basic operations

- Low-part multiplication : $m_i = s_0 N' \bmod r$
- Multiplication-addition : $T \leftarrow \frac{m_i N + S}{r}, \quad S \leftarrow a_i B + T$
  - Loops

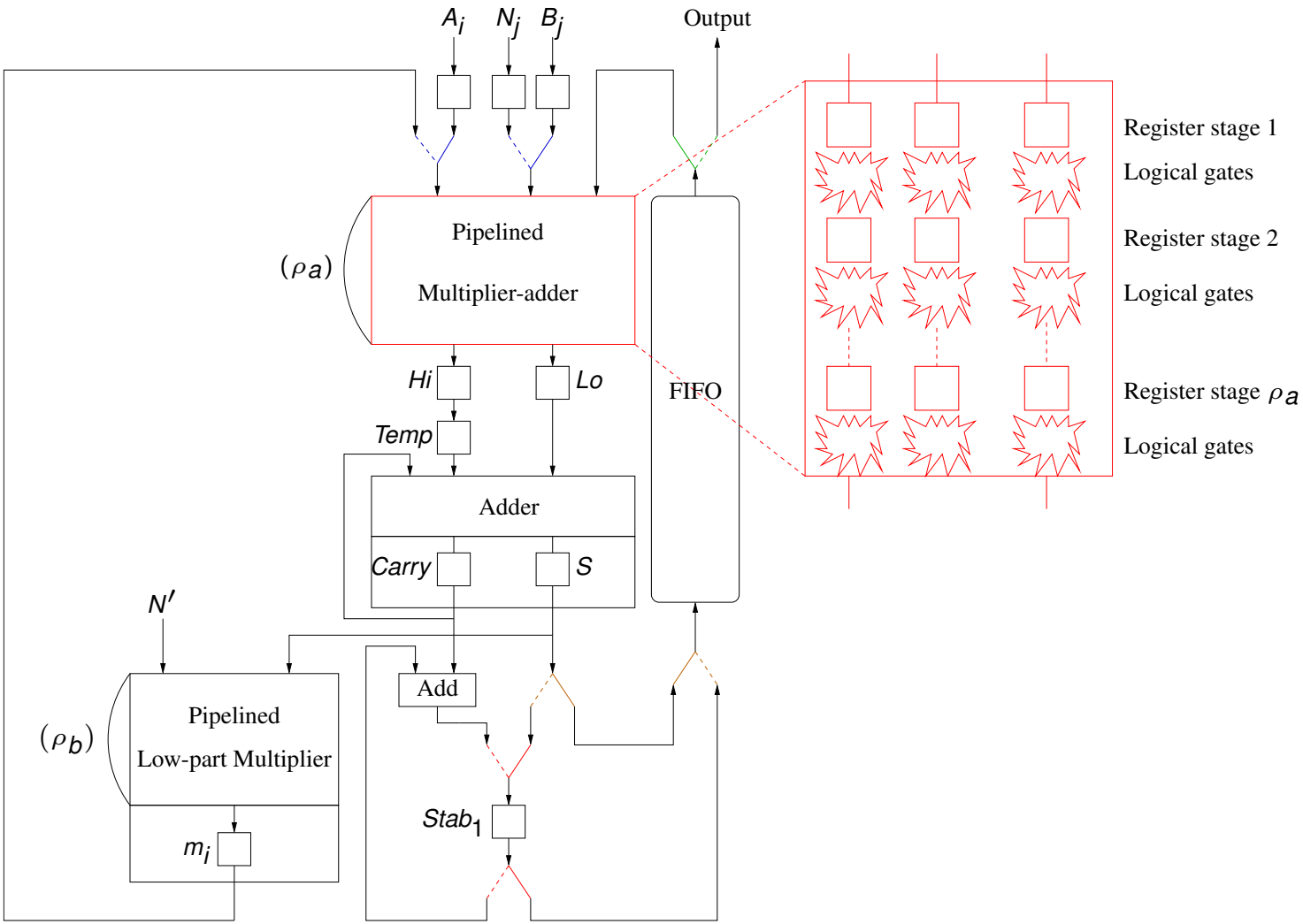
Classical multiplication
$c = 0$
For j from 0 to n do
  $P \leftarrow a_i b_j + t_j + c$
  $s_j \leftarrow Lo(P)$
  $c \leftarrow Hi(P)$
$s_{n+1} = c$

and

Right Shifted multiplication
$c = 0$
For j from 0 to n do
  $P \leftarrow m_i N_j + s_j + c$
  $t_{j-1} \leftarrow Lo(P)$
  $c \leftarrow Hi(P)$
$t_n = s_{n+1} + c$

- 2 basic operations :
  - $m_i = s_0 N' \bmod r \longrightarrow$ Low-part multiplier
  - $P = xy + z + c \longrightarrow$ Multiplier-adder + adder

Design


A_i
N_j
B_j
Output
Pipelined
Multiplier-adder
(ρa)
Hi
Lo
Temp
FIFO
Adder
Carry
S
N'
Add
Pipelined
Low-part Multiplier
(ρb)
m_i
$Stab_1$
Register stage 1
Logical gates
Register stage 2
Logical gates
Register stage ρa
Logical gates

Practical utilisation

- $R = r^{n+1}$
- Algorithmic Montgomery representation :

$$MM_nS(A, R^2) \Leftrightarrow \begin{cases} \widetilde{A} \equiv AR \mod N \\ \widetilde{S} \leq 2N - 1 \end{cases}$$

- Stable by Montgomery multiplication : no intermediate conversion
- Final output ($\widetilde{S}$) in Montgomery representation and $\widetilde{S} \leq 2N - 1$

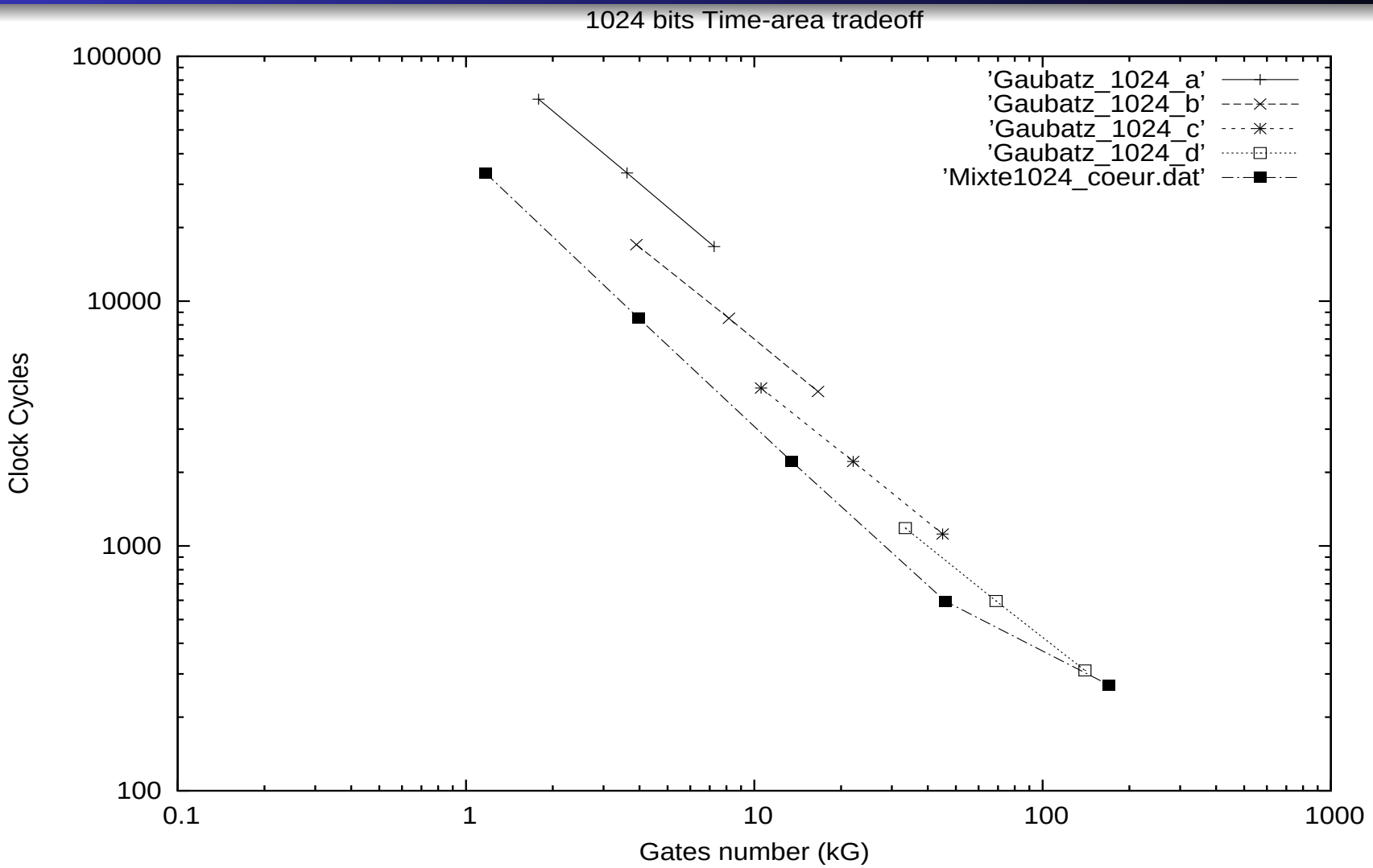
Property

If $\widetilde{S} \neq N$, then $MM_nS(1, \widetilde{S})$ convert $\widetilde{S}$ in its classical representation S with $0 \leq S \leq N - 1$, without any condition on the modulus

Results (Target : ASIC, $O, 35\mu$ SMARTCARD)

- Area : $\approx 50kG$ (64 bits wordsize) (FIFO not taken into account)
- Time : $T(n) = (n+1)(2n+3+n_{lat}+n'_{lat})$ cycles
  - $|N| = 512$ bits : $n = 8$ and $T(n) = 180$ cycles
  - $|N| = 1024$ bits : $n = 16$ and $T(n) = 595$ cycles
- Scalable Hardware :
  1. Wordsize from 2 bits to ≥ 128 bits
  2. Maximal size of the FIFO memory

Time-area tradeoff



Minimizing the different kinds of basic operations ?

- 1 basic operation = 1 basic processing unit
- Cell={basic processing units}
- Only one cell is used in our hardware design
- Minimizing the different kinds of basic processing units = reducing hardware area

Removing the Low-part multiplication

- Thomas Blum, 1999

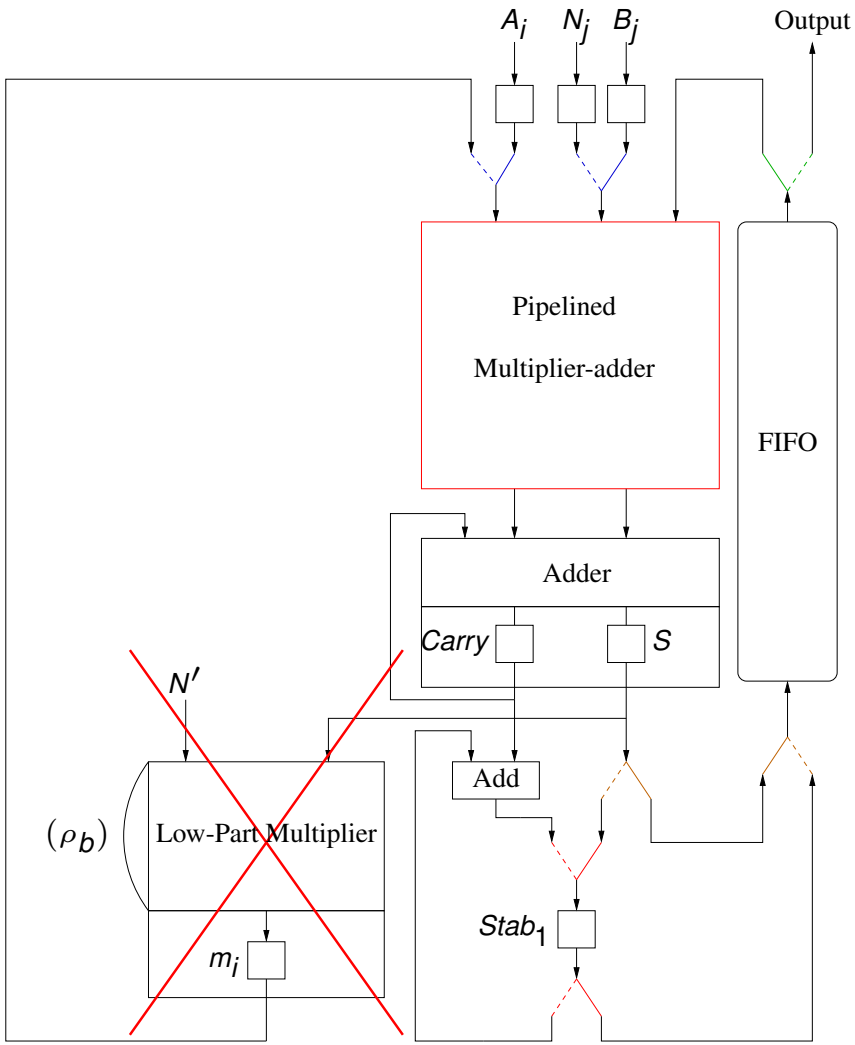
  "The Low-part multiplication can be removed."

- How ?
  - $m_i = S_0 N' \mod r$ $(N' = -N^{-1} \mod r)$
  - Idea : replace modulus N by $\widetilde{N} = N' \times N$
  - m_i such that $m_i \widetilde{N} + S$ is divisible by r

$$\begin{aligned} m_i = -s_0 \widetilde{N} \mod r &= -s_0 (N' \times N) \mod r \\ &= s_0 \mod r \end{aligned}$$

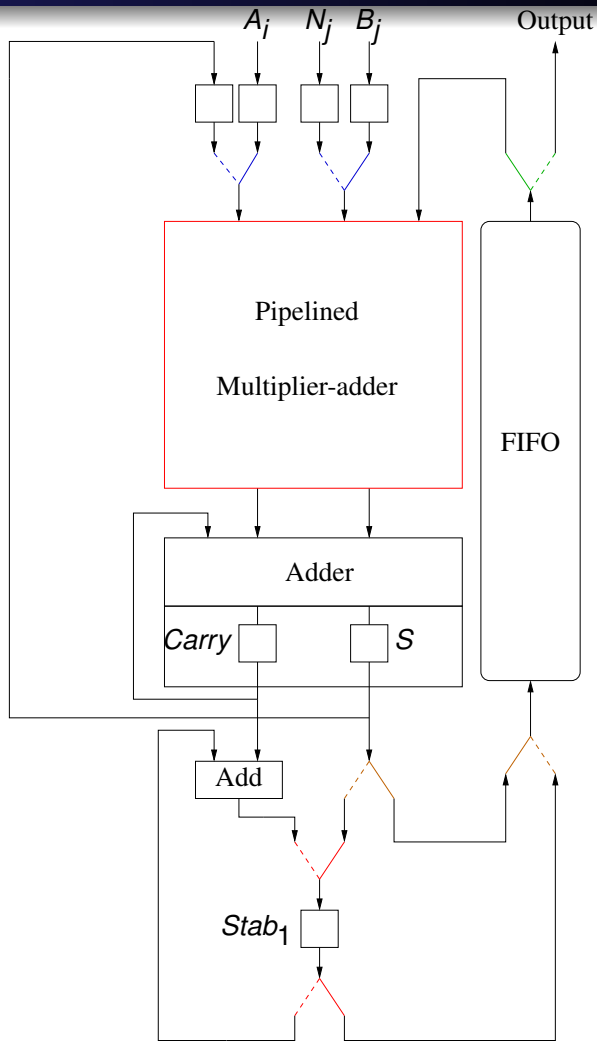
- m_i determination = read the Least Significant Digit of the output at iteration $i - 1$
- Low-part multiplier is no longer needed

Hardware simplification



Hardware simplification

- Hardware saving : $\approx 30\%$
- Doesn't depend on modulus size



Additional computational time cost

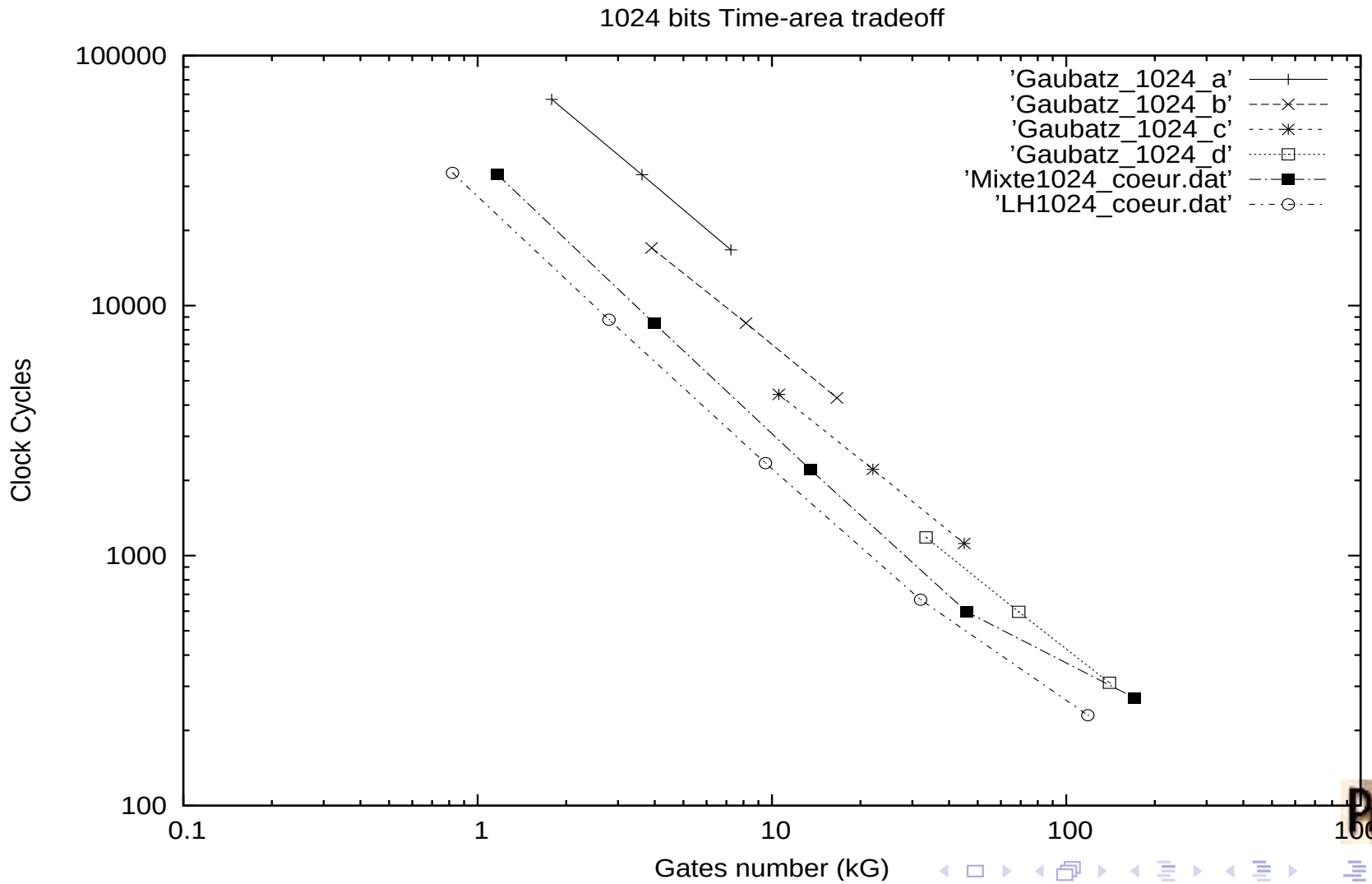
- New modulus used : $\widetilde{N} = N' \times N$
- Size : $n+1$ digits in radix r representation
- $\widetilde{T}(n) = (n+2)(2n+5+\widetilde{n}_{lat}+\widetilde{n}'_{lat})$
- Additional computational time cost : $4n+7$ cycles and a relative additional cost of :

$$+\frac{4n+7}{(n+1)(2n+3)}\%$$

- Exemples :

Size (bits)	512	1024	2048
Additional cost (%)	16.17	11.93	6.10

Time-area tradeoff



Final output : the problem

- Output $\widetilde{S}$ such that :
  - $\widetilde{S} \equiv ABr^{-n-2} \mod \widetilde{N}$
  - $\widetilde{S} < 2\widetilde{N}$
- $MM_{n+1}\widetilde{S}(1, \widetilde{S}) \leq \widetilde{N} < rN$
- Problem : result is r times greater than the expected result...

Software solution

- Only one correction step for a long modular exponentiation process
- Correction step might be done in software (division)
- Good solution if there is a software division implementation...

Hardware solution

- Call to the old algorithm $MM_{n+1}S$ with modulus N and not with $\widetilde{N}$

Property

$$If\ \widetilde{S} \neq N \mod N\ then\ MM_{n+1}S(1, \widetilde{S}) < N$$

- need of a low part multiplication ?
- Yes, but this operation will be performed by the multiplier-adder

Conclusion

- Strategy : 1 basic operation = 1 basic processing unit
- Only one cell used, with large datapath
- Reducing number of different kinds of operation = strongly reduces hardware area
- New time-area tradeoff is better even if computational time increases
- Final correction step is possible in hardware